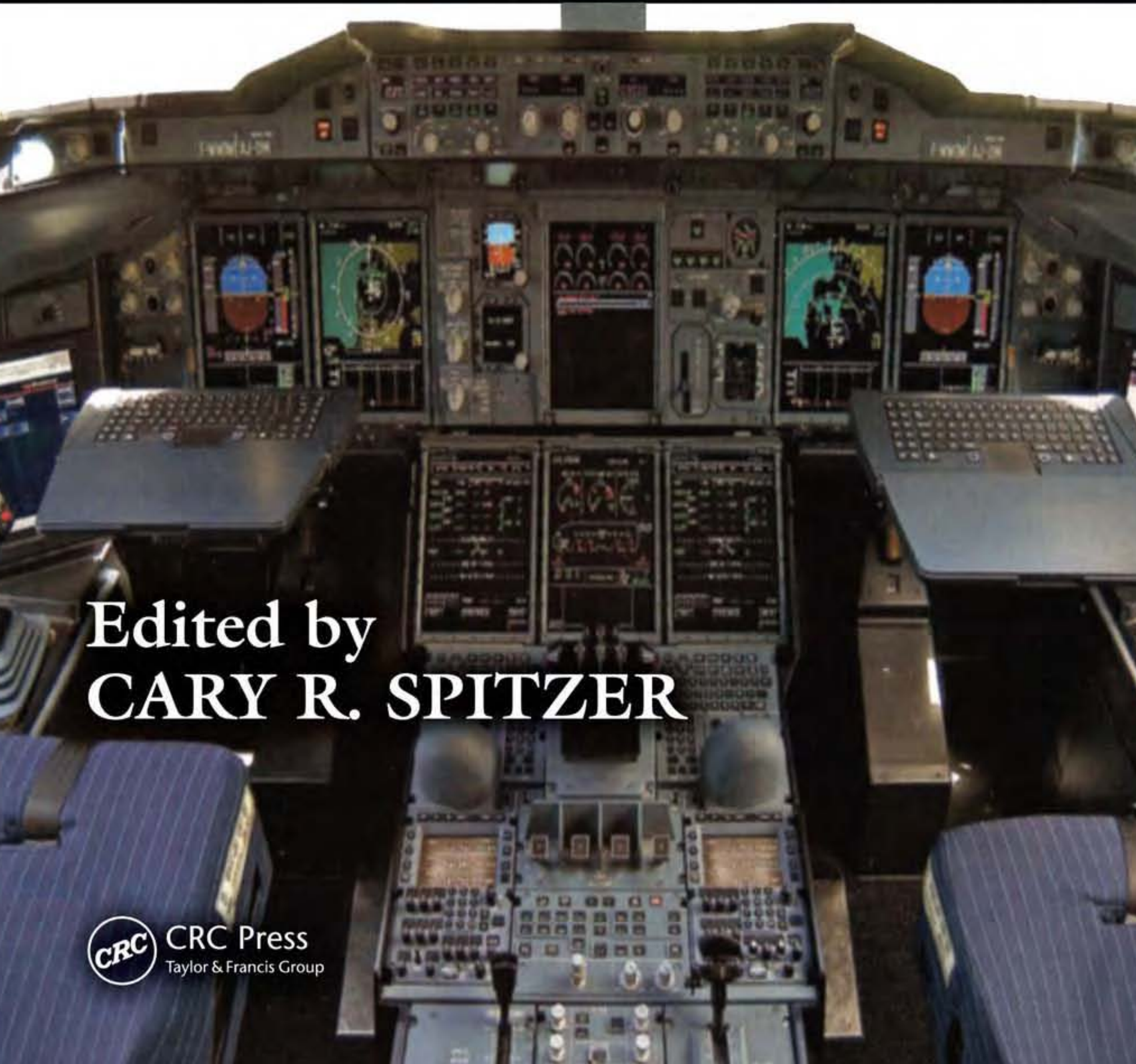
Digital Avionics Handbook

Second Edition

AVIONICS

DEVELOPMENT AND IMPLEMENTATION



Edited by

CARY R. SPITZER

CRC Press
Taylor & Francis Group

Digital Avionics Handbook
SECOND EDITION

AVIONICS
DEVELOPMENT AND IMPLEMENTATION

Digital Avionics Handbook
SECOND EDITION

AVIONICS
DEVELOPMENT AND IMPLEMENTATION

Edited by
CARY R. SPITZER
AvioniCon, Inc.
Williamsburg, Virginia, U.S.A.

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an informa business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

International Standard Book Number-10: 0-8493-8441-9 (Hardcover)
International Standard Book Number-13: 978-0-8493-8441-7 (Hardcover)

Library of Congress Cataloging-in-Publication Data

Spitzer, Cary R.
Avionics : development and implementation / Cary R. Spitzer.
p. cm. -- (Electrical engineering handbook ; 37)
Includes bibliographical references and index.
ISBN 0-8493-8441-9
1. Avionics. I. Title.

TL695.S7475 2006
629.135--dc22 2006050266

Preface

Avionics is the cornerstone of modern aircraft. More and more, vital functions on both military and civil aircraft involve electronic devices. After the cost of the airframe and the engines, avionics is the most expensive item on the aircraft, but well worth every cent of the price.

Many technologies emerged in the last decade that will be utilized in the new millennium. After proof of soundness in design through ground application, advanced microprocessors are finding their way onto aircraft to provide new capabilities that were unheard of a decade ago. The Global Positioning System has enabled satellite-based precise navigation and landing, and communication satellites are now capable of supporting aviation services. Thus, the aviation world is changing to satellite-based communications, navigation, and surveillance for air traffic management. Both the aircraft operator and the air traffic services provider are realizing significant benefits.

Familiar technologies in this book include data buses, one type of which has been in use for over 20 years, head mounted displays, and fly-by-wire flight controls. New bus and display concepts are emerging that may displace these veteran devices. An example is a retinal scanning display.

Other emerging technologies include speech interaction with the aircraft and synthetic vision. Speech interaction may soon enter commercial service on business aircraft as another way to perform some noncritical functions. Synthetic vision offers enormous potential for both military and civil aircraft for operations under reduced visibility conditions or in cases where it is difficult to install sufficient windows in an aircraft.

This book offers a comprehensive view of avionics, from the technology and elements of a system to examples of modern systems flying on the latest military and civil aircraft. The chapters have been written with the reader in mind by working practitioners in the field. This book was prepared for the working engineer and his or her boss and others who need the latest information on some aspect of avionics. It will not make one an expert in avionics, but it will provide the knowledge needed to approach a problem.

Editor

Cary R. Spitzer is a graduate of Virginia Tech and George Washington University. After service in the Air Force, he joined NASA Langley Research Center.

During the last half of his tenure at NASA he focused on avionics. He was the NASA manager of a joint NASA/Honeywell program that made the first satellite-guided automatic landing of a passenger transport aircraft in November 1990. In recognition of this accomplishment, he was nominated jointly by ARINC, ALPA, AOPA, ATA, NBAA, and RTCA for the 1991 Collier Trophy "for his pioneering work in proving the concept of GPS aided precision approaches." He led a project to define the experimental and operational requirements for a transport aircraft suitable for conducting flight experiments and to acquire such an aircraft. Today, that aircraft is the NASA Langley B-757 ARIES flight research platform.

Mr. Spitzer was the NASA representative to the Airlines Electronic Engineering Committee. In 1988 he received the Airlines Avionics Institute Chairman's Special Volare Award. He is only the second federal government employee so honored in over 30 years.

He has been active in the RTCA, including serving as chairman of Special Committee 200 Integrated Modular Avionics (IMA), chairman of the Airport Surface Operations Subgroup of Task Force 1 on Global Navigation Satellite System Transition and Implementation Strategy, and as Technical Program Chairman of the 1992 Technical Symposium. He was a member of the Technical Management Committee.

In 1993 Mr. Spitzer founded *A*vioni*C*on, an international avionics consulting firm that specializes in strategic planning, business development, technology analysis, and in-house training.

Mr. Spitzer is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) and an Associate Fellow of the American Institute of Aeronautics and Astronautics (AIAA). He received the AIAA 1994 Digital Avionics Award and an IEEE Centennial Medal and Millennium Medal. He is a Past President of the IEEE Aerospace and Electronic Systems Society. Since 1979, he has played a major role in the highly successful Digital Avionics Systems Conferences, including serving as General Chairman.

Mr. Spitzer presents one-week shortcourses on digital avionics systems and on satellite-based communication, navigation, and surveillance for air traffic management at the UCLA Extension Division. He has also lectured for the International Air Transport Association.

He is the author of *Digital Avionics Systems,* the first book in the field, published by McGraw-Hill, and Editor-in-Chief of *The Avionics Handbook,* published by CRC Press.

He and his wife, Laura, have a son, Danny.

His hobbies are working on old Ford products and kite flying.

Contributors

Ricky W. Butler
NASA Langley Research Center

Diganta Das
University of Maryland

Richard Hess
Honeywell

Ellis F. Hitt
Strategic Systems Solutions

Sally C. Johnson
NASA Langley Research Center

Joseph Lyvers
Xcelsi Group

G. Frank McCormick
Certification Services, Inc.

James N. Martin
The Aerospace Corporation

Jim Moore
Smiths Aerospace LLC

Michael J. Morgan
Honeywell

Dennis Mulcare
Retired

Michael G. Pecht
University of Maryland

Peter Potocki de Montalk
Airbus Industrie

Gordon R.A. Sandell
The Boeing Corporation

Cary R. Spitzer
AvioniCon

Jack Strauss
Xcelsi Group

Grant Stumpf
Spectral Systems Inc.

Nikhil Vichare
University of Maryland

Terry Venema
Xcelsi Group

Randy Walter
Smiths Aerospace LLC

Chris Watkins
Smiths Aerospace LLC

Ping Zhao
Medtronic

Table of Contents

Section I

Development

1 Processes for Engineering a System

James N. Martin
The Aerospace Corporation

1.1 Introduction

In April 1995, the G47 Systems Engineering Committee of the Electronic Industries Alliance (EIA) chartered a working group to convert the interim standard EIA/IS 632 into a full standard. This full standard was developed and released in December 1998 as ANSI/EIA-632-1998.

The interim standard (IS), EIA/IS 632, was titled "Systems Engineering." The full standard was expanded in scope to include *all the technical processes for engineering a system*. It is intended to be a higher-level abstraction of the activities and tasks found in the IS version plus those other technical activities and tasks deemed to be essential to the engineering of a system.

This chapter describes the elements of these processes and related key concepts. The intended purpose is to give the reader of the standard some background in its development and to help other standards activities in developing their own standard. There is a paper that describes the evolution from an interim standard to the full standard [Martin, 1998].

This standard is intended to be a "top tier" standard for the *processes essential to engineering a system*. It is expected that there will be second- and third-tier standards that define specific practices related to certain disciplines (e.g., systems engineering, electrical engineering, software engineering) and industry domains (e.g., aircraft, automotive, pharmaceutical, building, and highway construction).

It is important to understand several things that are *not* covered by this standard:

1. It does *not* define what "systems engineering" is;
2. It does *not* define what a "systems engineer" is supposed to do; and
3. It does *not* define what a "systems engineering organization" is supposed to do.

1.2 Structure of the Standard

The standard is organized as shown below:

Clause 1	*Scope*
Clause 2	*Normative references*
Clause 3	*Definitions and acronyms*
Clause 4	*Requirements*
Clause 5	*Application context*
Clause 6	*Application key concepts*
Annex A	*Glossary*
Annex B	*Enterprise-based life cycle*
Annex C	*Process task outcomes*
Annex D	*Planning documents*
Annex E	*System technical reviews*
Annex F	*Unprecedented and precedented development*
Annex G	*Requirement relationships*

1.3 Role of the EIA 632 Standard

Implementation of the requirements of EIA 632 are intended to be through establishment of enterprise policies and procedures that define the requirements for application and improvement of the adopted processes from the standard. This is illustrated in Figure 1.1.

1.4 Heritage of EIA 632

Figure 1.2 shows the relationship between EIA 632 and other standards on systems engineering. Some of the key software engineering standards are shown for comparison since there has been an intimate relationship between the development of both types of standards. There has been much activity recently in unifying the processes contained in each.

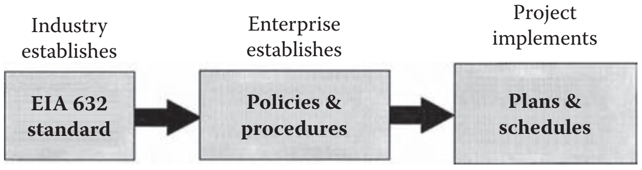


FIGURE 1.1 Role of the standard in relation to development projects. (Adapted from ANSI/EIA-632-1998. With permission.)

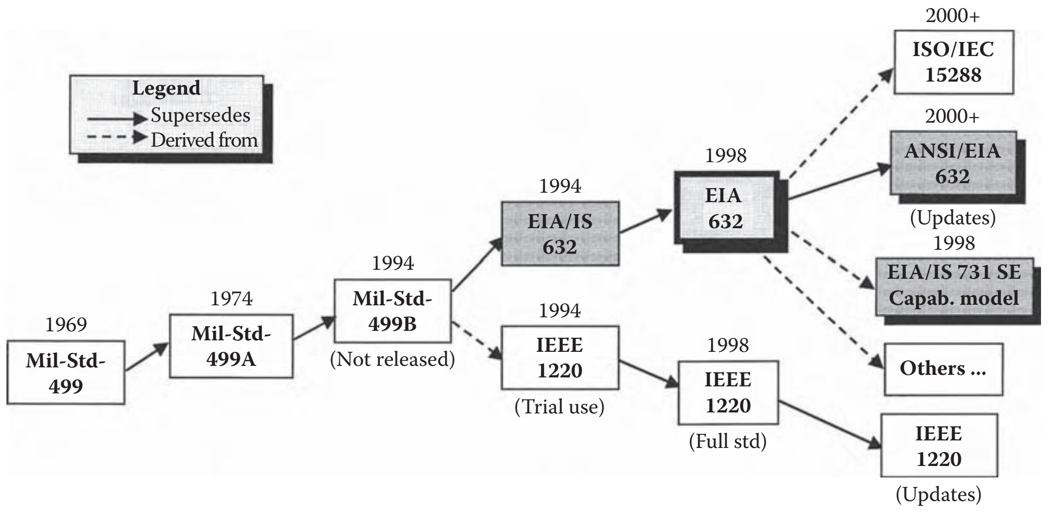


FIGURE 1.2 Heritage of systems engineering standards.

1.5 The Processes

Figure 1.3 shows the processes described in EIA 632 and their relationship to one another. Each enterprise will determine which of these processes are implemented by systems engineering personnel and how they are allocated to the organizational elements of the enterprise and its functional disciplines.

1.5.1 Process Hierarchy

The processes for engineering a system are grouped into the five categories as shown in Figure 1.4. This grouping was made for ease of organizing the standard and is not a required structure for process implementation. Traditional systems engineering is most often considered to include two of these processes: Requirements Definition and Systems Analysis. Often, Planning and Assessment are included in what is called systems engineering management.

1.5.2 Technical Management Processes

Technical Management provides oversight and control of the technical activities within a development project. The processes necessary to accomplish this are shown in Figure 1.5.

1.5.3 Acquisition and Supply Processes

Acquisition and Supply provides the mechanism for a project to supply its own products to a customer or higher-level project and to acquire the necessary products for its own product development activities. The processes necessary to accomplish this are shown in Figure 1.6.

1.5.4 System Design Processes

System Design provides the activities for a project to define the relevant requirements for its product development effort and to design solutions that meet these requirements. The processes necessary to accomplish this are shown in Figure 1.7.

1.5.5 Product Realization Processes

Product Realization provides the activities for a project to implement the product designs and to transition these products to their place of use. The processes necessary to accomplish this are shown in Figure 1.8.

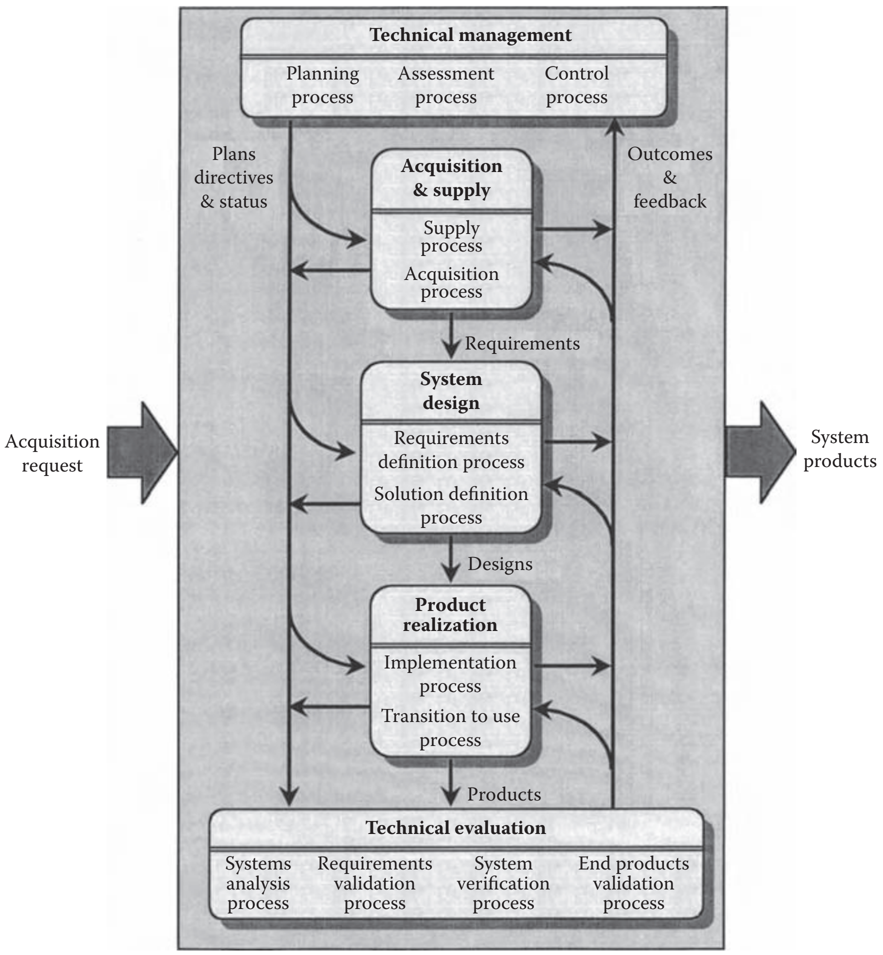


FIGURE 1.3 Top-level view of the processes for engineering a system. (From ANSI/EIA-632-1998. With permission.)

1.5.6 Technical Evaluation Processes

Technical Evaluation provides activities for a project to analyze the effectiveness of its proposed designs, validate the requirements and end products, and to verify that the system and its product meet the specified requirements. The processes necessary to accomplish this are shown in Figure 1.9.

1.6 Project Context

These "technical" processes fit into a larger context of a project (see Figure 1.10), and the project resides in some sort of enterprise, which in turn resides in an environment external to the enterprise. There are processes in the project and within the enterprise (but outside the project) that significantly affect the successful implementation of the technical processes.

1.7 Key Concepts

To understand the processes as described in this standard, it is essential to understand the distinct use of certain terms and the conceptual models that underlie each process. Some of the key terms are system,

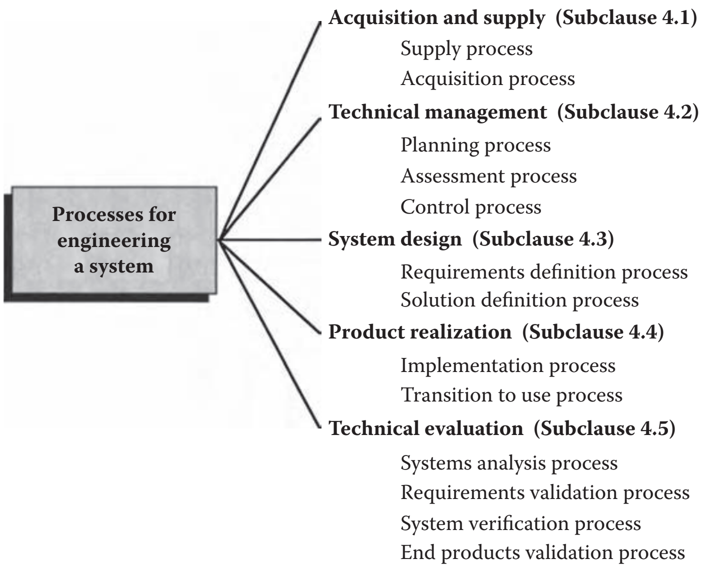

Processes for engineering a system
Acquisition and supply (Subclause 4.1)
Supply process
Acquisition process
Technical management (Subclause 4.2)
Planning process
Assessment process
Control process
System design (Subclause 4.3)
Requirements definition process
Solution definition process
Product realization (Subclause 4.4)
Implementation process
Transition to use process
Technical evaluation (Subclause 4.5)
Systems analysis process
Requirements validation process
System verification process
End products validation process

FIGURE 1.4 Hierarchical view of the processes for engineering a system. (From ANSI/EIA-632-1998. With permission.)

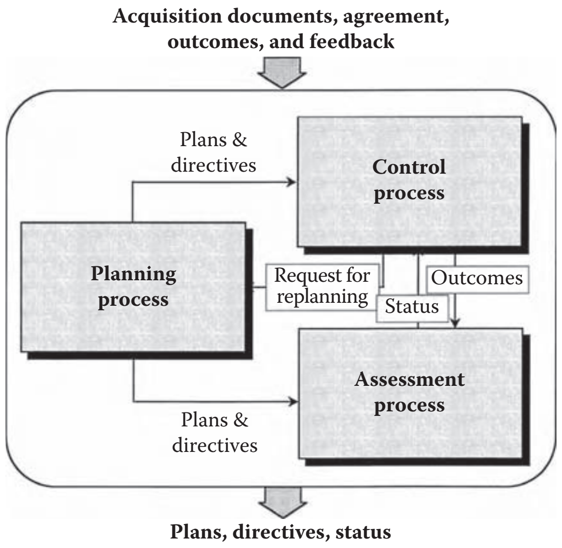


FIGURE 1.5 Technical Management Processes. (From ANSI/EIA-632-1998. With permission.)

product, verification, and validation. Some of the key concepts are building block, end products, associated processes, and development layers.

1.7.1 The System and Its Products

What is a system? The term "system" is commonly used to mean the set of hardware and software components that are developed and delivered to a customer. This standard uses this term in a broader sense in two aspects.

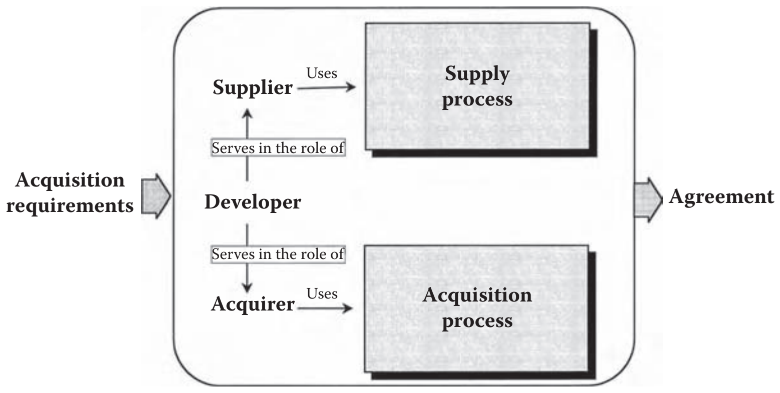


FIGURE 1.6 Acquisition and Supply Processes. (From ANSI/EIA-632-1998. With permission.)

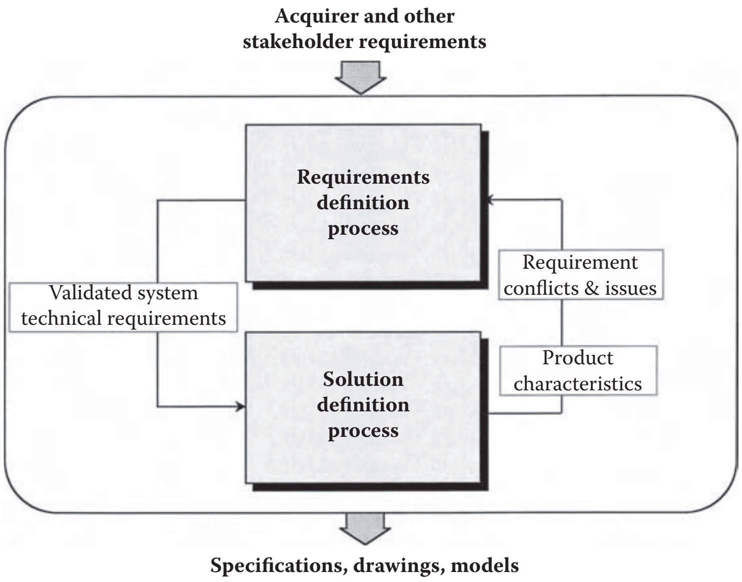

Acquirer and other stakeholder requirements
Requirements definition process
Validated system technical requirements
Requirement conflicts & issues
Product characteristics
Solution definition process
Specifications, drawings, models

FIGURE 1.7 System Design Processes. (From ANSI/EIA-632-1998. With permission.)

First, the system that needs to be developed consists of not only the "operations product" (that which is delivered to the customer and used by a user), but also the enabling products associated with that operations product. The operations product consists of one or more end products (so-called since these are the elements of the system that "end up" in the hands of the ultimate user). The associated processes are performed using enabling products that "enable" the end products to be put into service, kept in service, and retired from service.

Second, the end products that need to be developed often go beyond merely the hardware and software involved. There are also people, facilities, data, materials, services, and techniques. This is illustrated in Figure 1.12.

This is not intended to be an exhaustive list of the "basic" product types since these will vary depending on the particular business or technology domain. For example, in the television industry, "media" is

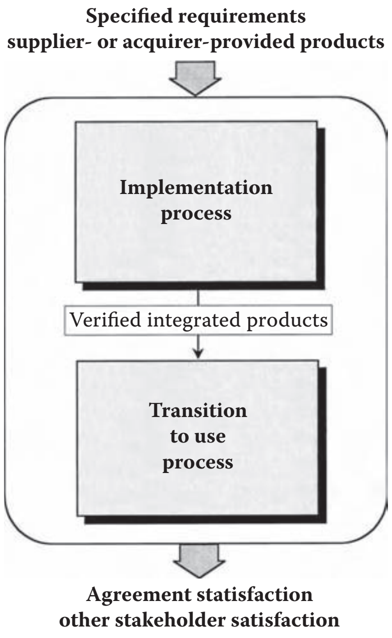

Specified requirements
supplier- or acquirer-provided products
Implementation process
Verified integrated products
Transition to use process
Agreement statisfaction
other stakeholder satisfaction

FIGURE 1.8 Product Realization Processes. (From ANSI/EIA-632-1998. With permission.)

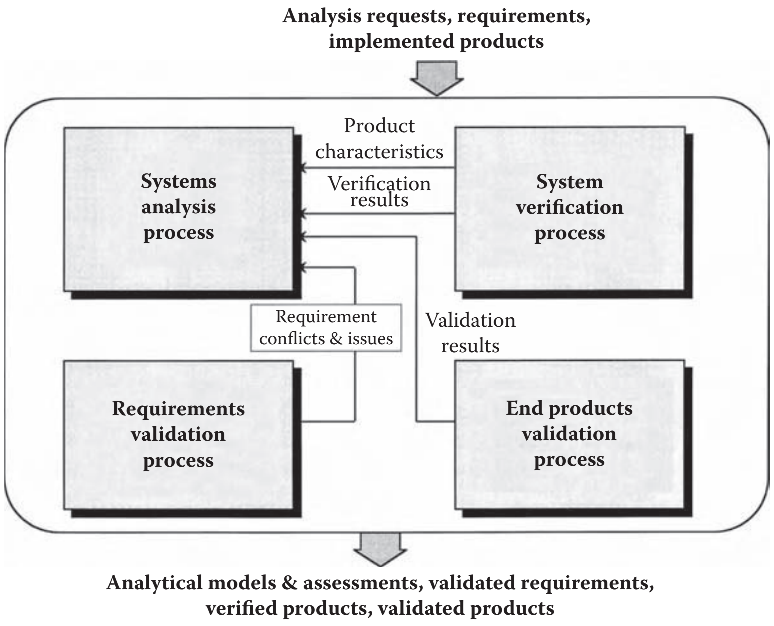

Analysis requests, requirements, implemented products
Systems analysis process
Product characteristics
Verification results
System verification process
Requirement conflicts & issues
Validation results
Requirements validation process
End products validation process
Analytical models & assessments, validated requirements, verified products, validated products

FIGURE 1.9 Technical Evaluation Processes. (From ANSI/EIA-632-1998. With permission.)

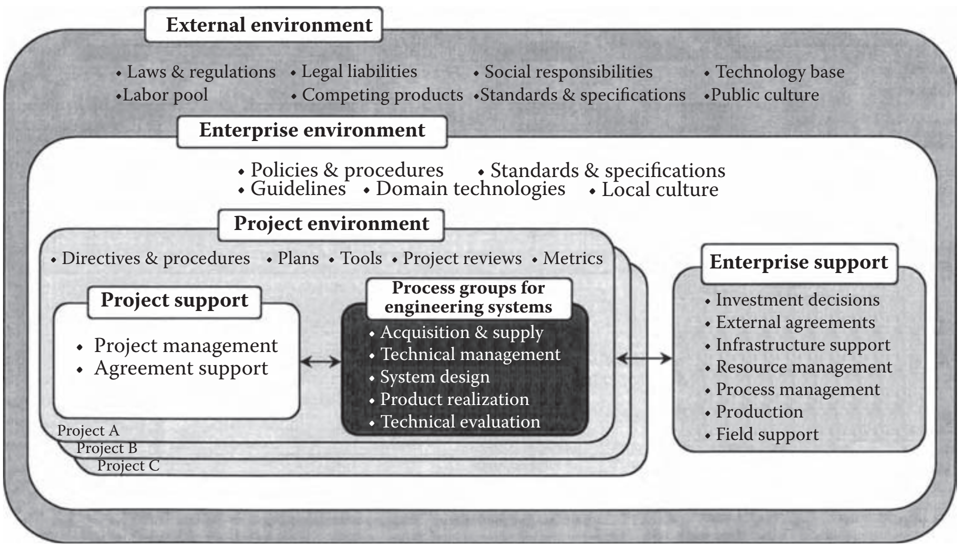


FIGURE 1.10 The technical processes in the context of a project and an enterprise. (From ANSI/EIA-632-1998. With permission.)

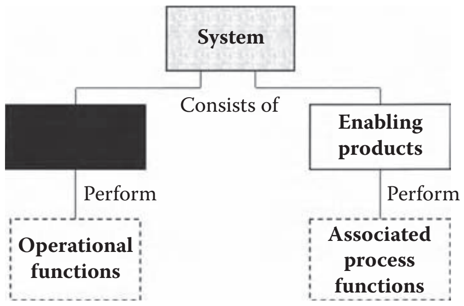


FIGURE 1.11 Constituents of the system. (From ANSI/EIA-632-1998. With permission.)

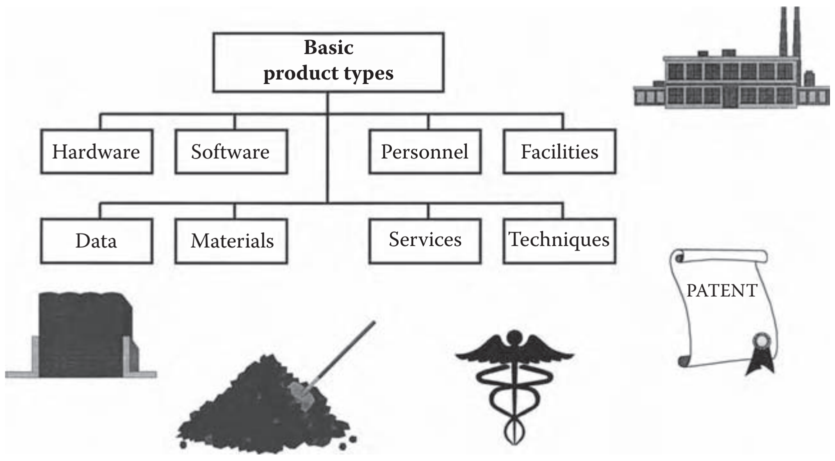


FIGURE 1.12 Different product types that make up a system.

certainly one of the system elements that constitute the overall system that is developed. "CBS News" might be considered the system, for example, with end products like:

Airwaves, worldwide web (media)
Cameras, monitors (hardware)
Schedule management tools, video compression algorithms (software)
Camera operators, news anchor (personnel)
Studio, broadcasting tower (facilities)
Script, program guide (data)
Pictures, stories (materials)
Airplane transportation, telephone (services)
Presentation method, editing procedures (techniques)

Note that any or all of these end products could be "off-the-shelf." But some of them may need to be conceived, designed, and implemented. Even if one of these items is truly off-the-shelf, it may still need some enabling product to allow effective use of that item. For example, even if you can use existing editing procedures, you may need to develop a training program to train the editors.

1.7.2 Building Block Framework

As we can see from the description above of the "CBS News" system, the nonhardware/software items may be crucial to successful realization of the whole system. You may also need to develop the associated processes along with the relevant enabling products. If we tie all these elements together, we can illustrate this using the so-called "building block" shown in Figure 1.13.

There are seven associated processes related to these sets of enabling products. These processes are used at various points in the life of a product (sometimes called "system life cycle elements"). Hence, the use of the building block concept is intended to help the developer in ensuring that the full life cycle of the end product is properly considered.

Note that each end product can consist of subsystems. Each of these subsystems may need its own development. The building block can be used at the various "development layers" of the total system. These development layers are illustrated in Figure 1.14.

1.7.3 Development of Enabling Products

As mentioned above, the enabling products may need to be developed also. For each associated process, there could be enabling products that either exist already or that need some degree of development. Figure 1.15 shows how enabling products related to the deployment process have their own building blocks.

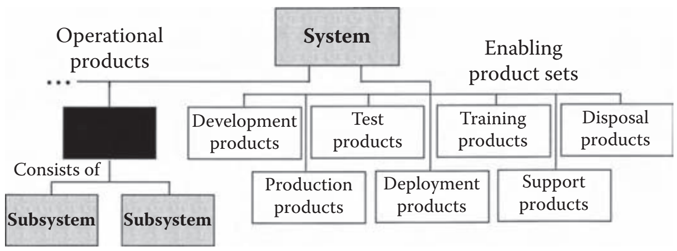


FIGURE 1.13 The building block concept. (From ANSI/EIA-632-1998. With permission.)

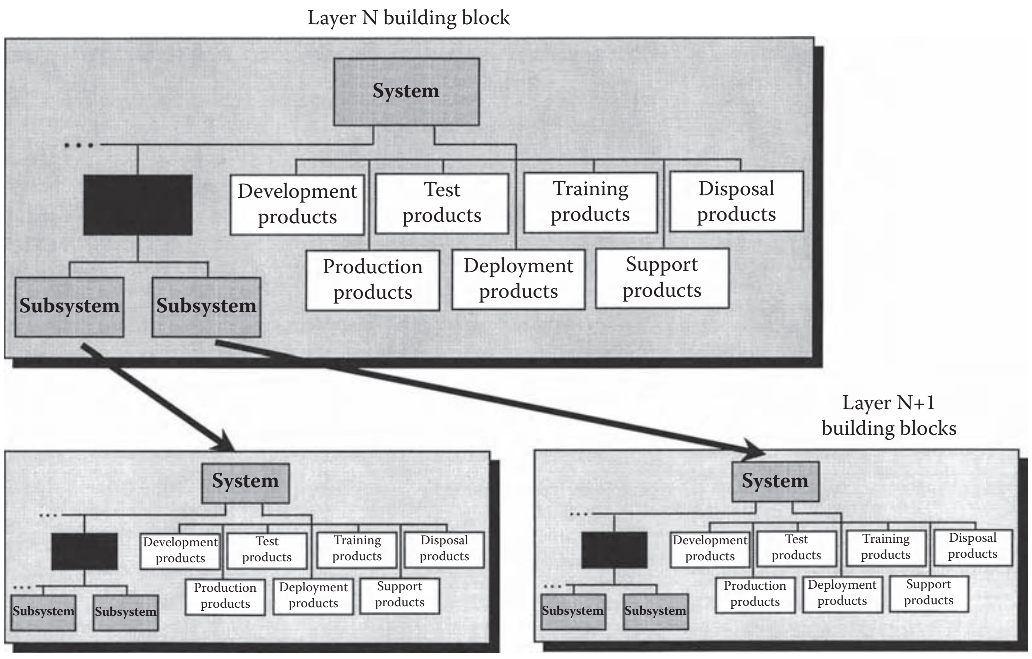
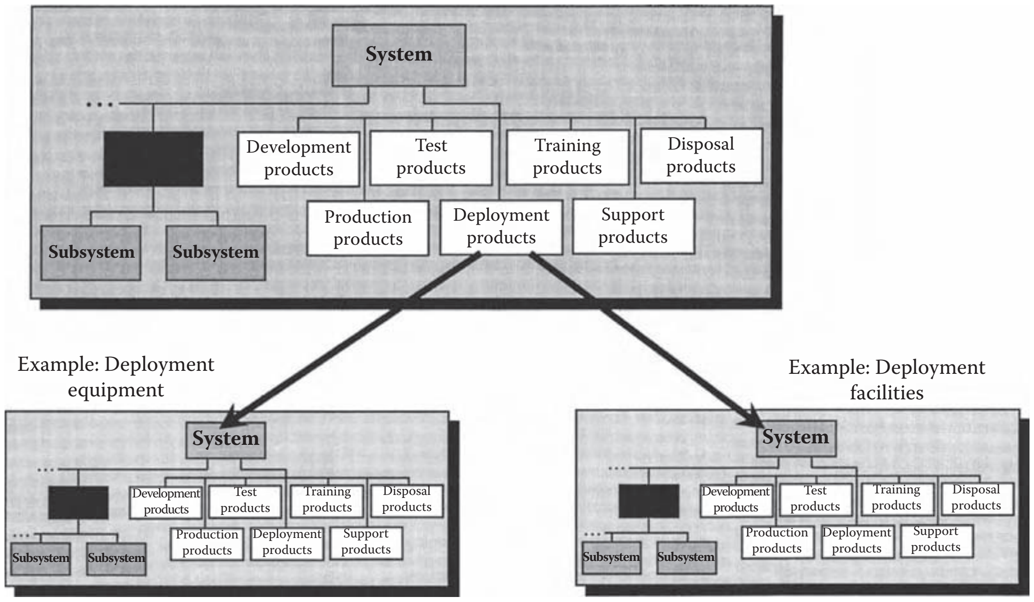


FIGURE 1.14 Development layers concept. (From ANSI/EIA-632-1998. With permission.)



Other deployment enabling products possibly needing development:

v Deployment procedures (e.g. installation manual)

v Deployment personnel (e.g. field technicians, installers)

v Deployment services (e.g. shipping & receiving, transportation)

FIGURE 1.15 Development of enabling products. (Adapted from ANSI/EIA-632-1998. With permission.)

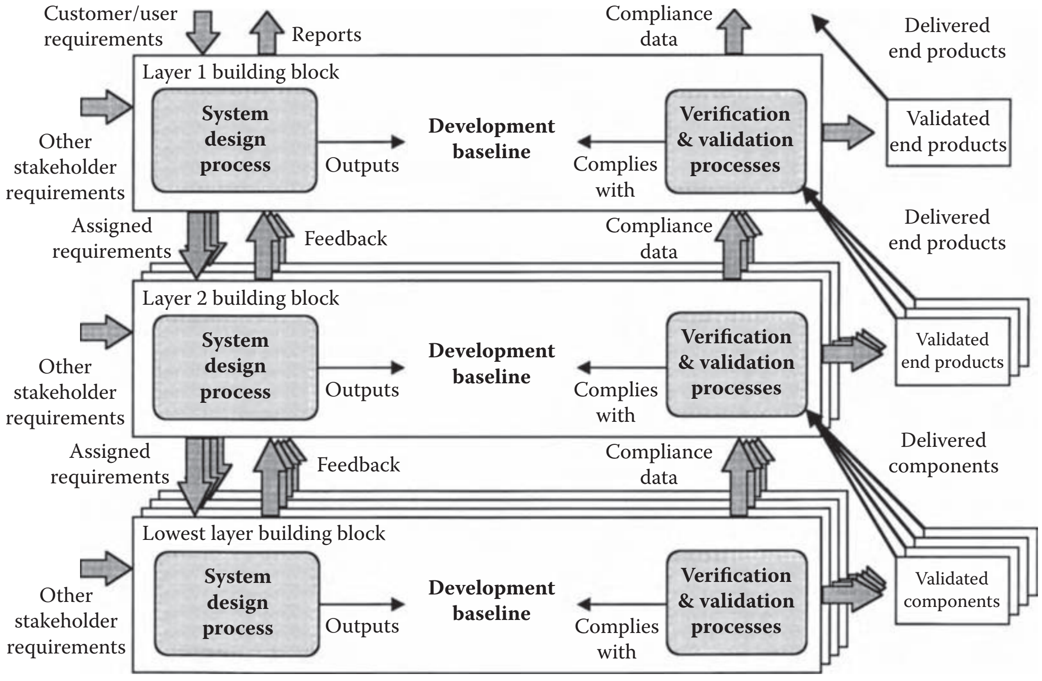


FIGURE 1.16 Development layers in a project context. (Adapted from ANSI/EIA-632-1998. With permission.)

1.7.4 Relationship between the Building Blocks and the Processes

The building blocks relevant to a particular system development can be "stacked" into a System Breakdown Structure (SBS). The System Design Process and Verification and Validation Processes have a special relationship to the SBS, as shown in Figure 1.16.

1.7.5 Hierarchy of Building Blocks

Typically, a project is developing a system of such a size that it will require more than one building block to complete the development. Figure 1.17 shows an example of how several building blocks would be related to each other within a project. Notice also how an "adjacent" Project A has building blocks that might have interfaces to the systems under development by Project B. There also may be building blocks above and below a project.

1.7.6 Requirements

Requirements are especially useful for engineering a system, especially when the system consists of several building blocks. Requirements act as a kind of "decoupling mechanism" between the building blocks. They are the "terms and conditions" between the layers of development. Requirements start out as vague, usually nontechnical stakeholder requirements and evolve into more technically precise and verifiable technical requirements. This is illustrated in Figure 1.18.

Lower-level requirements must be derived from higher-level requirements. This is usually accomplished through logical and physical analyses and design activities to determine the requirements that apply to the design solution. This is illustrated in Figure 1.19 which shows the different types of requirements dealt with by the processes described in EIA 632.

Requirements generally do not come from a particular customer. And even when they do, as is often the case in government-sponsored developments, they do not represent the full spectrum of the requirements that will eventually need to be defined for development of the products of a system.

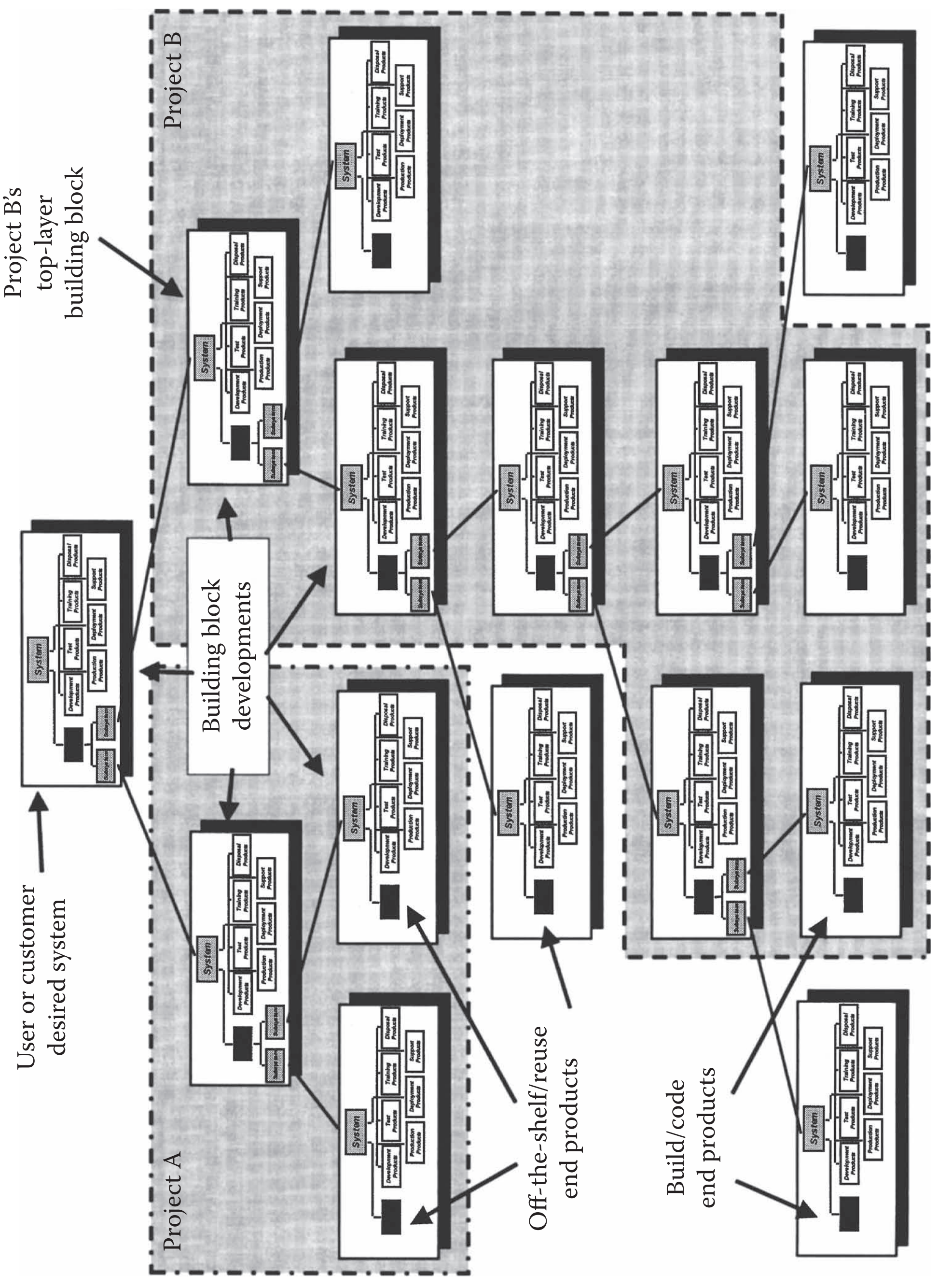


FIGURE 1.17 Building blocks in the context of a several projects. (From ANSI/EIA-632-1998. With permission.)

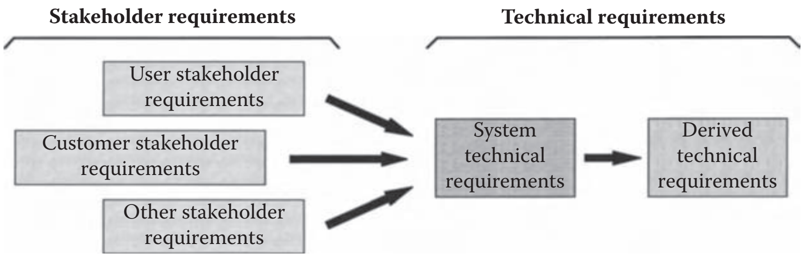

Stakeholder requirements
Technical requirements
User stakeholder requirements
Customer stakeholder requirements
Other stakeholder requirements
System technical requirements
Derived technical requirements

FIGURE 1.18 Evolution of requirements. (*Source*: ANSI/EIA-632-1998. With permission.)

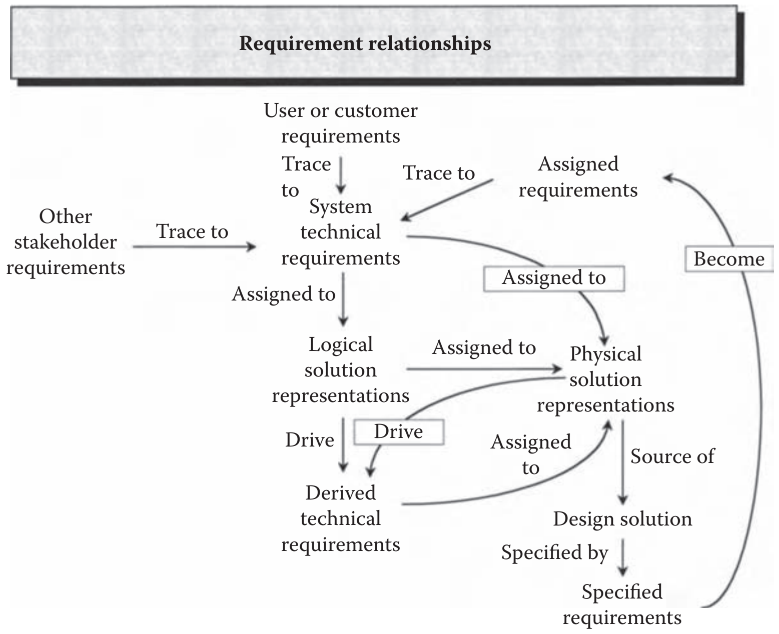

Requirement relationships
User or customer requirements
Trace to
System technical requirements
Trace to
Assigned requirements
Other stakeholder requirements
Trace to
Become
Assigned to
Assigned to
Logical solution representations
Assigned to
Physical solution representations
Drive
Drive
Assigned to
Source of
Derived technical requirements
Design solution
Specified by
Specified requirements

FIGURE 1.19 Types of requirements and their interrelationships. (From ANSI/EIA-632-1998. With permission.)

User Requirements. User requirements are often nontechnical in nature and usually conflict with each other. These need to be translated into "technical requirements" that are in the vernacular of the engineers on a project and are specific to the domain of the technology to be used in the system.

Customer Requirements. Customers will often translate their perceived user requirements into a set of requirements to be given to a development project. These also, like the user requirements, are often not technical enough and usually conflict with each other. Moreover, a particular project may have more than one customer for its products.

Stakeholder Requirements. In defining the technical requirements for the system, one must consider that there are other stakeholders for that system. For example, the manufacturing organization within your company may desire that you use their existing processes, tools, facilities, etc. There are constraints associated with these "enabling products." Hence, you must identify the stakeholder requirements that drive the development.

Requirements Decomposition. Once the system technical requirements are defined, further requirements are often "derived" by analyzing both the logical and physical aspects of the system. Some of these derived requirements, plus whatever system technical requirements are relevant, are "specified" for the system, its end products, and possibly subsystems of those end products.

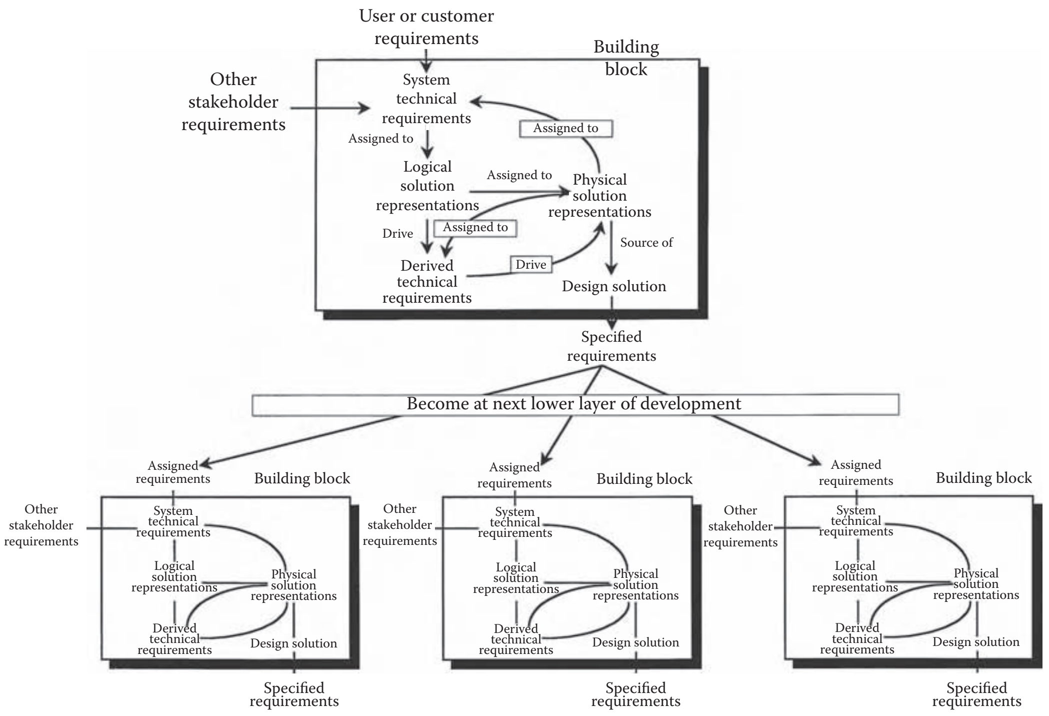

User or customer requirements
Building block
Other stakeholder requirements
System technical requirements
Assigned to
Assigned to
Logical solution representations
Assigned to
Physical solution representations
Drive
Assigned to
Source of
Derived technical requirements
Drive
Design solution
Specified requirements
Become at next lower layer of development
Assigned requirements
Building block
Other stakeholder requirements
System technical requirements
Logical solution representations
Physical solution representations
Derived technical requirements
Design solution
Specified requirements

FIGURE 1.20 Layers of requirements as they relate to the layers of development. (Adapted from ANSI/EIA-632-1998. With permission.)

Given these specified requirements for an item, the process starts all over again for items needing further development. In other words, another layer of building blocks may be required to develop an item far enough along such that the bottommost item can be procured from a supplier. This is illustrated in Figure 1.20.

1.7.7 Functional, Performance, and Interface Requirements

Another way to look at requirements is to categorize them as functional (what the product must accomplish), performance (how well the product must accomplish each function), and interface (under what conditions must the functions be accomplished). This is illustrated in Figure 1.21.

- **Functional requirements**
  >> What an item is to accomplish
    - Behavior of an item
    - An effect produced
    - Action or service to be performed
- **Performance requirements**
  >> How well an item is to accomplish a function
    - ... like how much, how often, how many, how few, . . .
- **Interface requirements**
  >> Conditions of interaction between items
    - ... could be functional, physical, logical, . . .

FIGURE 1.21 Types of requirements.

- **Verification**
  - >> Check compliance against specified requirements
  - >> "Have you done the job right?"
  - >> Two types
    - Product & process qualification
      - – Full compliance to specification
      - – Product requalification may be necessary when product is "redesigned"
      - – Process requalification may be necessary when process is "restarted"
    - Product acceptance
      - – Full compliance to key criteria
      - – Done on every unit or on sample basis
      - – Can be done prior to shipment or after installation

- **Validation**
  - >> Check satisfaction of stakeholders
  - >> "Have you done the right job?"
  - >> Two types
    - Requirements validation
      - – Check for "traceability"
      - – "Have we missed any requirements?"
      - – "Do we have extraneous requirements?"
    - Product validation
      - – Check if meeting "needs and expectations" of stakeholders

FIGURE 1.22 Distinction between verification and validation.

1.7.8 Verification and Validation

Verification and validation are both very important in engineering a system. Verification is the act of determining if a product meets its specified requirements. Validation is the act of determining if a product satisfies its stakeholders. This is illustrated in Figure 1.22.

Defining Terms

The following definitions are extracted from the standard.

Building block: A representation of the conceptual framework of a system that is used for organizing the requirements, work, and other information associated with the engineering of a system. An element in the structured decomposition of the system.

Enabling product: Item that provides the means for (a) getting an end product into service, (b) keeping it in service, or (c) ending its service.

End product: The portion of a system that performs the operational functions and is delivered to an acquirer.

Process: A set of interrelated tasks that, together, transform inputs into outputs.

Product: (1) An item that consists of one or more of the following: hardware, software, firmware, facilities, data, materials, personnel, services, techniques, and processes; (2) a constituent part of a system.

Requirement: Something that governs what, how well, and under what conditions a product will achieve a given purpose.

Stakeholder: An enterprise, organization, or individual having an interest or a stake in the outcome of the engineering of a system.

Subsystem: A grouping of items that perform a set of functions within a particular end product.

System: An aggregation of end products and enabling products to achieve a given purpose.

Validation: (1) Confirmation by examination and provision of objective evidence that the specific intended use of an end product (developed or purchased), or an aggregation of end products, is accomplished in an intended usage environment; (2) confirmation by examination that requirements (individually and as a set) are well formulated and are usable for intended use.

Verification: Confirmation by examination and provision of objective evidence that the specified requirements to which an end product is built, coded, or assembled have been fulfilled.

References

ANSI/EIA 632, *Processes for Engineering a System*. 1998.
EIA/IS 632, *Systems Engineering*. 1994.
Martin, J. N., "Evolution of EIA 632 from an Interim Standard to a Full Standard." Proc. INCOSE 1998 Int. Symp., 1998.

Further Information

The purpose of this chapter is to give a high-level overview of the processes for engineering a system described in ANSI/EIA 632. Further details can be found in the full publication of the standard or by contacting the EIA directly at 2500 Wilson Boulevard, Arlington, Virginia or at their Website (www.eia.org).

2 Digital Avionics Modeling and Simulation

Jack Strauss
Xcelsi Group

Joseph Lyvers
Xcelsi Group

Terry Venema
Xcelsi Group

Grant Stumpf
Spectral Systems Inc.

2.1 Introduction

In order to realize unprecedented but operationally essential levels of avionics system performance, reliability, supportability, and affordability commercial industry and the military will draw on advanced technologies, methods, and development techniques in virtually every area of aircraft design, development, and fabrication. Federated and integrated avionics architectures, hybrid systems architectures, and special purpose systems such as flight control systems, engine control systems, navigation systems, reconnaissance collection systems, electronic combat systems, weapons delivery systems, and communications systems all share certain characteristics that are germane to digital systems modeling and simulation. All of these classes of avionics systems are increasing in complexity of function and design and are making increased use of digital computer resources. Given these advancements, commercial and military designers of new avionics systems and of upgrades to existing systems must understand, incorporate, and make use of state-of-the-art methods, disciplines, and practices of digital systems design. This chapter presents fundamentals, best practices, and examples of digital avionics systems modeling and simulation.

2.2 Underlying Principles

The following comparison illuminates a fundamental principle of modeling and simulation. The results of most mathematical processes are either correct or incorrect, but modeling and simulation has a third possibility. The process can yield results that are correct but irrelevant (Strauss 1994). Given this perhaps startling but true realization of the potential results of modeling and simulation, it is important to understand the different perspectives that give rise to the motivation for modeling and simulation, the trade space for the development effort to include the users and systems requirements, and the technical underpinnings of the practice.

2.2.1 Historic Perspective

The past 35 years of aviation has seen extraordinary innovation in all aspects of complex systems design and manufacturing technology. Digital computing resources have been employed in all functional areas of avionics, including communication, navigation, flight controls, propulsion control, and all areas of military weapon systems. As analog, mechanical, and electrical systems have been replaced or enhanced with digital electronics there has been an increased demand for new, highly reliable, and secure digital computing techniques and for high-performance digital computing resources.

Special purpose data, signal, and display processors were commonly implemented in the late 1960s and early 1970s (Swangim et al., 1989). Special purpose devices gave way to programmable data, signal, and display processors in the early to mid 1980s. These devices were programmed at a low level; assembly language programming was common. The late 1980s and early 1990s have seen commercial and military avionics adopting the use of high-performance general-purpose computing devices programmed in high-order languages. The USAF F-22 fighter, for example, incorporates general-purpose, commercially available microprocessors programmed in Ada. The F-22 has an integrated avionics architecture with an operational flight program consisting of nearly one million lines of Ada code and onboard computing power on the order of 20 billion operations per second of signal processing and 500 million instructions per second (MIPS) of data processing. Additionally, there is increasing use of commercial off-the-shelf (COTS) products such as processor boards, storage devices, graphics displays, and fiber optic communications networks for many military and commercial avionics applications. The F-35 Joint Strike Fighter, currently in development, uses commercial processors programmed in the C language. Although total computer power is still in development, it will likely equal or exceed that of the F-22.

COTS product designers and avionics systems developers have made it a standard engineering practice to model commercial computer and communications products and digital avionics products and systems at all levels of design abstraction. As the complexity of electronics design dramatically has increased, so too has modeling and simulation technology in both functional complexity and implementation. Complex computer-aided design (CAD) software can be several hundred thousand source lines of code without taking into account the extensive libraries that may be purchased with these systems. These software products can require advanced engineering workstation computing resources with sophisticated file and storage structures and data management schemes. Workstations with multiple GHz processors with several hundred-gigabyte disk drives, connected with high-speed local area networks (LANs) are common. Additionally, special-purpose hardware environments, used to enhance and accelerate simulation and modeling, have increased in performance and complexity to supercomputing levels. Hardware emulators and rapid prototype equipment can reach near real-time system performance. At this point in history, the complexity and performance of modeling and simulation technology is every bit equal to the digital avionics products they are used to develop.

2.2.2 Economic Perspective

For commercial systems and product designers, time-to-market is a critical product development factor that has significant impact on the economic viability of any given product release. The first product to market generally recoups all of the nonrecurring engineering and development costs and commonly captures as much as half the total market. This is why technologies aimed at decreasing time-to-market remain important to all commercial developers. Early analysis shows the amount of development time saved as a result of digital system modeling and simulation (Donnelly, 1992). At a lower level, cost-sensitive design has two significant issues: the learning curve and packaging.

The learning curve is best described as an increase in productivity over time. For device manufacturing, this can be measured by a change (increase) in yield or the percentage of manufactured devices that survive testing. Whether it is a chip, a board, or a system, given sufficient volume, designs that have twice the yield will generally have half the cost (Hennessy and Patterson, 2003). The design reuse inherent in digital modeling and simulation directly enhances the learning curve. Packaging at the device, board, or system level has cost implications related to fundamental design correctness and system partitioning. A case study of performance modeling for system partitioning is presented later in this text.

For military systems designers, many of the same issues affecting commercial systems designers apply, especially as more commercial technologies are being incorporated as implementation components. Additionally, as will be shown (in Section 2.2.3), mission objectives and operational requirements are the correct point of entry for modern, top-down design paradigms. However, once a development effort has been initiated, the relative cost of error discovery (shown notionally in Figure 2.1) is more expensive at each successive level of system completeness (Portelli et al., 1989). Thus, a low-price solution that does not fully meet system requirements can turn into a high-cost problem later in the development cycle.

2.2.3 Design Perspective

Bell and Newell divided computer hardware into five levels of abstraction (Bell and Newell, 1973): processors-memory switches (system), instruction set (algorithm), register transfer, logic, and circuit. Hardware systems at any level can also be described by their behavior, structure, and physical implementation. Walker and Thomas combined these views and proposed a model of design representation and synthesis (Walker and Thomas, 1985) that includes the architecture level (system level), algorithmic level, functional block level (register transfer level), logic level, and circuit level. Each of these levels is defined in terms of their behavioral domain, structural domain, and physical domain. Behavioral design describes the behavior or function of the hardware using such notations as behavioral languages, register transfer languages, and logic equations. Structural design specifies functional blocks and components such as registers, gates, flip-flops, and their interconnections. Physical design describes the implementation of a

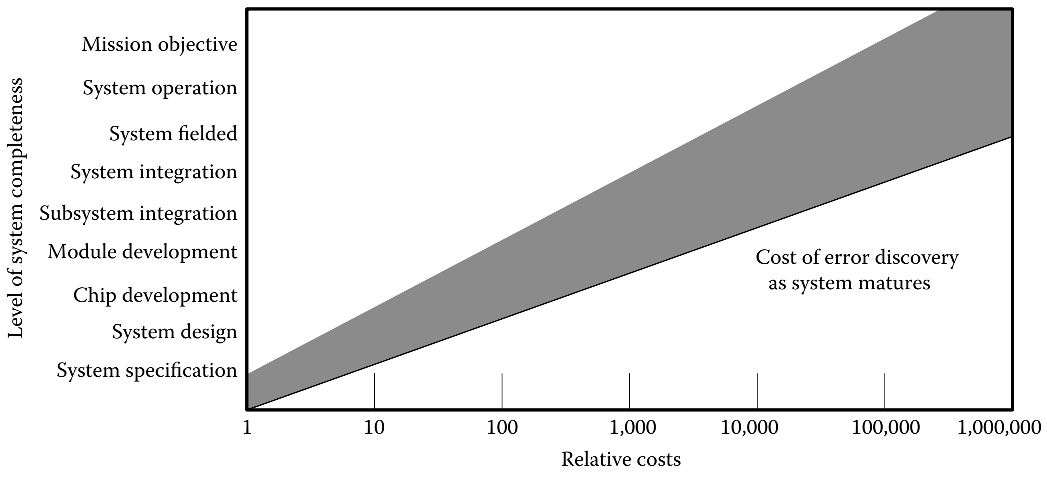


FIGURE 2.1 Relative cost of error discovery as design progresses.

design idea in a hardware platform, such as the floor plan of a circuit board and layout of transistors, standard cells, and macrocells in an integrated circuit (IC) chip.

Hierarchical design starts with high-level algorithmic or behavioral design. These high-level designs are converted (synthesis) to circuits in the physical domain. Various CAD tools are available for design entry and conversion. Digital modeling and simulation technologies and tools are directly incorporated into the process to assure correctness and completion of the design at each level and to validate the accuracy of the translation from one level to the next (Liu, 1992).

2.2.4 Market Perspective

It is generally assumed that there are two major market segments for avionics — the commercial avionics market (targeting airlines and general aviation) and the military avionics market. As stated earlier, there is great similarity in the technological forces at work on both military and commercial systems. There are, however, several fundamental differences in the product development cycle and business base that are important to consider because they impact the interpretation of the cost performance analysis for modeling and simulation. These differences are summarized in Table 2.1. Consider the impact of commercial versus military production volumes on a capital investment decision for modeling and simulation technology. The relative priority of this criterion is likely to differ for commercial products as compared to military systems.

2.2.5 Requirements in the Trade Space

Technology (commercial or military) without application, tactical, or doctrinal context is merely engineering curiosity. The development of an avionics suite or the implementation of an upgrade to an existing suite requires the careful balance of many intricate constraints. More than any other type of development, avionics has the most intricate interrelationships of constraints. The complex set of issues associated with volume, weight, power, cooling, capability, growth, reliability, and cost create some of the most complex engineering trades of any systems development effort. The risks associated with the introduction of new technologies and the development of enabling technologies create mitigation plans that have the characteristics of complete parallel developments. It is little wonder that avionics systems are becoming the most expensive portion of aircraft development.

To fully exploit the dollars available for avionics development it is necessary to invest a significant effort in an intimate understanding of the system requirements. Without knowledge of what the pilot needs to accomplish the mission, and how each portion of the system contributes to the satisfaction of that need, it is impossible to generate appropriate trades and understand the impacts on the engineering process. Indeed, often the mission needs are vague and performance requirements are not specified. This critical feature of the development is further complicated by the fact that pilots often are unaware or do not have the technical background to articulate the detailed technical features of the system or requirements set and cannot specifically identify critical technical system parameters and requirements.

TABLE 2.1 Market Factors Comparison for Commercial and Military Market Segments

Criterion	Commercial	Military
Financial basis	Market	Budget
Development focus	Product	Capability
Production volume	Medium–high	Low
System complexity	Medium–high	High
System design cycle	Short	Medium–long
Life cycle	Short–medium	Long–very long
Contractual concerns	Warranty, liability	Reliability, mortality

In the final analysis, the avionics suite is a tool used by the pilot to accomplish a task. The avionics are an extension of his senses and his capabilities. They provide orientation, perception, and function while he is attempting to complete an endlessly variable set of tasks. With this in mind, the first and most important step in the design and development of an avionics package is the development of the requirements. Modeling and simulation is well suited for this task.

2.2.6 Technical Underpinnings of the Practice

Allen defines modeling as the discipline for making predictions of system performance and capacity planning (Allen, 1994). He further categorizes techniques in terms of rules of thumb, back-of-the-envelope calculations, statistical forecasting, analytical queuing theory modeling, simulation modeling, and benchmarking. When applied intelligently, all methods have utility. Each has specific cost and schedule impacts. For nontrivial modeling and simulation, the areas of analytical queuing theory modeling, simulation modeling, and benchmarking have the greatest information content, while rules of thumb often hold the most wisdom. For quantitative estimates, at the system component level, back of the envelope calculations are appropriate.

Analytical queuing theory seeks to represent the system algorithmically. The fundamental algorithm set is the M/M/1 queuing system (Klienrock, 1975), which is an open system (implying an infinite stream of incoming customers or work) with one server (that which handles customers or does the work) that provides exponentially distributed service. Mathematically, the probability that the provided service will require not more than t time units is given by:

$$P[S \leqq t] = 1 - e^{-t/s}$$

where S is the average service time.

For the M/M/1 queuing system, the interarrival time, that is, the time between successive arrivals, also has an exponential distribution. Mathematically, this implies that if τ describes the interarrival time, then:

$$\mathrm{P}[\tau \leqq t] = 1 - e^{-\lambda\tau}$$

where λ is the average arrival rate.

Therefore, with the two parameters, the average arrival rate λ and the average service time *S*, we completely define the M/M/1 model.

Simulation modeling is defined as driving the model of a system with suitable inputs and observing the corresponding outputs (Bratley et al., 1987). The basic steps include construction of the model, development of the inputs to the model, measurement of the interactions within the model, formation of the statistics of the measured transactions, analysis of the statistics, and validation of the model. Simulation has the advantage of allowing more detailed modeling and analysis than analytical queuing theory, but has the disadvantage of requiring more resources in terms of development effort and computer resources to run.

Benchmarking is the process of running specialized test software on actual hardware. It has the advantage of great fidelity in that the execution time associated with a specific benchmark is not an approximation, but the actual execution time. This process is extremely useful for comparing the results of running the same benchmark on several different machines. There are two primary disadvantages: it requires actual hardware, which is difficult if the hardware is developmental, and unless your application is the benchmark, it may not represent accurately the workload of your specific system in operation.

2.2.7 Summary Comments

Historically, there have been dramatic increases in the complexity and performance of digital avionics systems. This trend is not likely to change. This increase in complexity has required many new tools and processes to be developed in support of avionics product design, development, and manufacture.

The commercial and military avionics marketplaces differ in many ways. Decisions made quantitatively must be interpreted in accordance with each marketplace. However, both commercial and military markets have economic forces that have driven the need for shorter development cycles. A shorter development cycle generally stresses the capabilities of design technology; thus, both markets have similar digital system design process challenges.

The most general design cycle proceeds from a concept through a design phase to a prototype test and integration phase, ending finally in release to production. The end date (be it a commercial product introduction or a military system deployment) generally does not move to the left on the schedule. Designs often take longer than scheduled due to complexity and coordination issues. The time between prototype release to fabrication and release to production, which should be spent in test and debug, gets squeezed. Ideally, it is desirable to lengthen the test and debug phase without compromising either design time or the production release date. Modeling and simulation does this by allowing the testing of designs before they are fabricated, when changes are easier, faster, and cheaper to make.

The design process for any avionics development must begin with the development of the requirements. A full understanding of the requirements set is necessary because of the inherent constraints it places on the development and the clarity it provides as to the system's intended use.

There are several techniques available for the analysis and prediction of a system's performance. These techniques include Allen's modeling techniques (Section 2.2.6): rules of thumb, back-of-the-envelope calculations, statistical forecasting, analytical queuing theory modeling, simulation modeling, and benchmarking. Each technique has advantages and disadvantages with respect to fidelity and cost of resources to perform. When taken together and applied appropriately, they form the rigor base for the best practices in digital avionics systems modeling and simulation.

2.3 Best Practices

In order for the results of any modeling and simulation effort to be valid, attention must be paid to the fundamental principle that any model may return irrelevant, but very correct, results. A corrective to this dilemma is a thorough understanding of the customer's requirements and the flow-down through system implementation with a completely traceable pedigree. Automated tool sets to enable such development still do not exist, although progress is being made toward the generation of fully compliant traceable requirements. The first step in crafting an understandable, predictable system is ensuring the proper requirements are being implemented correctly. Market forces and system engineering practices are encouraging the diffusion and maturation of appropriate tool sets that will enable avionics system designers to accomplish these goals. The following sections will summarize the best practices in requirements generation and system modeling simulation.

2.3.1 Requirements Engineering

Establishing the broad parameters of the system performance is critical to the design phase beginning with the generation of requirements. The process of developing the requirements initiates a step-wise refinement of the system, which culminates in the articulation of quantifiable mission effectiveness metrics that flow down to the implementation subsystems. Attention paid to the early modeling of the system will often yield unexpected results that feed back to the highest level of requirement understanding. Refining and narrowing the engineering trade space with high-level understanding gained by requirement modeling processes prevents costly missteps and programmatic embarrassments, failures, or even termination.

Begin at the very beginning. The first step is the generation of final (end-state) user requirements. The constraints of cost, schedule, and technical considerations need not be considered initially so as to allow the formation of a maximally effective system solution to implement what the user truly desires. Simulation of the initial system then allows for a benchmark to be used in the subsequent evaluation of necessary compromises, which are driven by programmatic constraints. Performance metrics then reflect the deviations from the maximum represented by the benchmark. Likewise, if the original baseline was suboptimal, performance under certain perturbations of system functionality would likely spawn improvements relative to the benchmark. Conditional-use exercises of the system model — which normally occur in the successive refinement and rationalization of requirements with the usual cost, schedule, and technical constraints — assist in the formation of early, deep understanding of the system performance.

Once the initial requirement set has been established, whether or not it is acknowledged to be optimal or suboptimal, the associated system performance predictions may be evaluated within the trade space available to the designer. The effects of the system requirements are evaluated by a design-of-experiments approach to the system mission scenarios that are used to stimulate the rudimentary system model. A thorough understanding of the impacts of the requirements on customer constraints is critical to a balanced system design.

It hardly needs to be stated that the system model and simulation used in assessing the utility of the various trade-off design approaches during the process of refining the requirements needs to be validated so as to be a reliable indicator of system performance. Many models have been thoroughly tested to be trusted as reliable. As the expense of avionics systems escalates, the value and trust placed in the models escalates; for this reason, the validation and verification of modeling and simulation continues to be a subject of much interest within government channels. Tremendous efforts will continue to be made in validation of the models.

Digital avionics simulation is an integral part of the avionics design of the modern cockpit. Whereas once it was possible to present crude nonreal-time or elementary functioning controls and display systems to aid in the evaluation of human interaction with the system, it is no longer an option to treat the system as anything other than an extension of the human interacting with his environment. The crucial nature of the data presented to the pilot must be understood in the context of the reaction time-dependent decision response expected. Simulation prior to rendering the system as a physical representation is the acceptable way to build confidence in the end-to-end effectiveness and performance of the entire closed-loop system, which includes the human response.

For this reason, extensive Man-in-the-Loop (MITL) simulations have been built to evaluate proposed systems. Whether the scope of the MITL simulations is limited to partial or full-scale mission simulations, extensive use of the resultant observations may be proffered as corroboration of the validity of the requirement set. Without validation derived from MITL, the likelihood that the requirements will be correct but irrelevant only increase. Validation of the requirement set then is the starting point toward a balanced design. Confidence in the utility of the resultant system must be established prior to development.

If, however, certain aspects of the proposed system are amenable to an abstraction of human interaction, then the MITL may be itself stimulated with an appropriately faithful synthetic stimulus. Understanding the limitations of the simulated MITL results *a priori* then tempers the extrapolation to the expected validation of requirement set. Research continues in the performance modeling of pilots in extremely high-demand task environments. Refined understanding will result in a more thorough system model of human interaction with complex systems, which will enable a higher level of trust in the results derived from simulated MITL models.

Finally, requirements validation engenders confidence in the end-to-end system performance prior to commitment to development and fielding of new systems. The MITL simulation folds together the abstracted levels of system performance in the chain from receiver sensor sensitivity to effective stimulation and response of the human operator. The questions modeling and simulation may answer prior to commitment of scarce resources are two-fold. First, does the proposed system perform as expected? Second, does the system satisfy the capability the user needs? Analysis of requirements then is the fundamental initiative in answering those questions.

2.3.2 Top-Down System Simulation

Top-Down System Simulation (TDSS) begins the process of refining the design with the aim of reducing the risk that the system will not perform properly. If the system can be designed with confidence that what is being built will perform correctly, with all of the subsystems integrating seamlessly and the interfaces mating properly, then the overall risk of false starts is minimized. Obvious benefits include eliminating hardware and software redesign tasks with cost and schedule implications, preventing false starts in integration, and promoting early resolution of requirements ambiguities. Management visibility into the design and development process of complex systems is heightened as well. The costs and resources incurred in committing to the implementation of a total or partial TDSS will often reap benefits far outweighing any short-term negative (incorrectly perceived) programmatic impact.

Early design practices were a step-wise refinement of specifications, beginning with the topmost requirement specification and followed by subsequent partitioning of the system into component parts until the lowest level of decomposition was achieved. Interfaces were captured; agreements were struck. Individual subsystem design processes within the respective disciplines refined the approaches. Not until the system integration phase were differences in assumptions and interpretations discovered, with resolution often recovered only at the expense of the schedule. Modeling and simulation of the system offers a way to avoid the design pitfalls.

The key to using TDSS as a risk reduction tool is to validate the design requirements, the functional decomposition into subsystems, and the data sets and their associated timing relationships. TDSS provides a visualization and virtual realization of the system prior to committing hardware and software resources to implement a potentially fundamentally flawed design, which avoids expensive reworking of the system to accommodate and resolve problems discovered late in the system integration phase.

An example of this methodology is the USAF Advanced Tactical Fighter (ATF) Demonstration and Validation (DEM-VAL) development effort. The interoperability of designs of five critical interfaces was tested through simulation. Of the five interfaces involved, testing revealed over 200 problems, both with the designs and with the specifications on which they were based. These problems would have resulted in many iterations of hardware fabrication during the integration phase, and several of them would probably not have been detected until the system was fielded. The Air Force concluded that the application of a TDSS methodology to the DEM-VAL program alone resulted in a savings of approximately $200 million, 25 times the cost of the initiative itself.

2.3.3 TDSS Plan

At the outset of a program to implement a design, all the relevant parties must commit to a TDSS with clearly articulated goals and timeframes for achieving those goals, and each of the stakeholders must agree to the allocation of resources to accomplish the goals. TDSS is indeed a virtual implementation performed in parallel with and ahead of the desired system. To be effective, it must precede the design by an appropriate time phase. TDSS performed exactly in phase or slightly behind the real system will not allow problems to be resolved without cost and schedule implications.

To be an effective representation, TDSS needs to be planned with enough detail and granularity to mirror the final system. Detailed planning of the implementation steps and coordination of the interfaces between the segments of the system are critical to the validity of the simulation. Assignment of responsibilities to the various Integrated Product Teams, along with performance targets, will ensure that the same assumptions, design ambiguity resolutions, and approaches are used by the teams in the implementation of the final system. For example, the System Architecture Team defines the subsystem portioning and functionality to implement the requirements of the system. Subsequently, it hands off to the Subsystem “A” Development Team a self-consistent requirements specification with defined interfaces and timing constraints. The Subsystem “A” Development Team, in turn, develops its own domain-specific model that then produces outputs corresponding to the input stimuli it receives from other subsystems. The net result is again a virtual system that simulates the performance of the ultimate system in all important domains. Problems with interfaces and timing are resolved early, prior to commitment to the actual hardware and software realizations of the system.

Given the importance of TDSS to the success of the system development, it is paramount that the design team approaches the implementation of the TDSS itself with the same rigor it uses with the system design planning. If it is to be a faithful representation of the system, the TDSS cannot be done "ad hoc"; adequate resources with sufficient scope and visibility must be committed to gain a valid realization of the requirements.

Exit criteria for each phase in the TDSS development must exist and be adhered to rigorously. Rushing through the simulation in an effort to "get through it" will nullify the utility of performing the simulation in the first place. The simulation needs to produce acceptable results prior to committing hardware designs to fabrication or software to the implementation phase. At each phase in the system design, whether preliminary or critical design, suitable reviews of the results of the simulation must be conducted and approved as entrance criteria to the next step. The consequences of slighting the results of the simulation are simply postponing discovery to a more expensive recovery stage later in the design process. Attention to virtual design assurance that the TDSS brings will conserve resources, the schedule, and the budget.

Planning the TDSS includes defining the data sets and formats passed between each of the teams involved in the system design. Prior to initiating the design, all parties must agree to the adequacy and accuracy of the data sets. To minimize unique definitions and usage, individual data sets ought to be common to all related teams sharing common domains, thus maximizing utility and consistent interpretations of the data. Reducing opportunities to interpret data differently will minimize errors driven by the translation process. Simple examples of errors are legion; one is the common definition and usage of "units."

Lastly, successively refining the model is akin to the successive reduction of the design from the abstract to the physical realization of hardware and software. It is critical to keep focused on the intent of the simulated system as a virtual representation of the real system as it transitions from concept through to integration and ultimately fielding.

2.3.4 TDSS Process

The process for TDSS follows the structure of the normal system development cycle with decomposition of the system functionality proceeding from higher levels of abstraction successively to lower levels of implementation. The development of TDSS begins with the system level definition modeling and simulation. Appropriate validation of the system model is performed to assure a thorough and complete representation of the system. Requirements are traced from the specification to the proposed implementation to assure completeness, rigor, and compliance. Shortfalls are shored up; over-design is eliminated; assumptions are clarified. Early in the program, the parties agree that the model faithfully represents the intended, specified system. Successive refinement of detail then takes place in the modeling domain, as it would in the real system. Each lower level of design detail is modeled with domain specific knowledge again to represent the performance of each component. Performance encompasses all the relevant information pertaining to the component itself: data transformation, accuracy, precision, timeliness of resultants, and adherence to other constraints, as well as statistical measures of allowable performance variance. Figure 2.2 illustrates the typical system model development flow. At each review, appropriate scrutiny is applied to the development of the model. At System Requirements Review, the planning for developing the model is reviewed to ascertain whether the proposed model is adequately complex to represent the system. At System Design Review, the maturity of the model is reviewed to assess whether there is sufficient detail to provide the benchmark for the system so as to be a useful tool in allowing critiques of alternative design approaches that various potential subcontractors and suppliers may propose.

Detailed design follows the modeled system portioning into subsystem, function, modules, and components. Machine-executable representations of the system are developed to refine the system further. The system architects review and approve the detailed models and verify that they indeed perform the

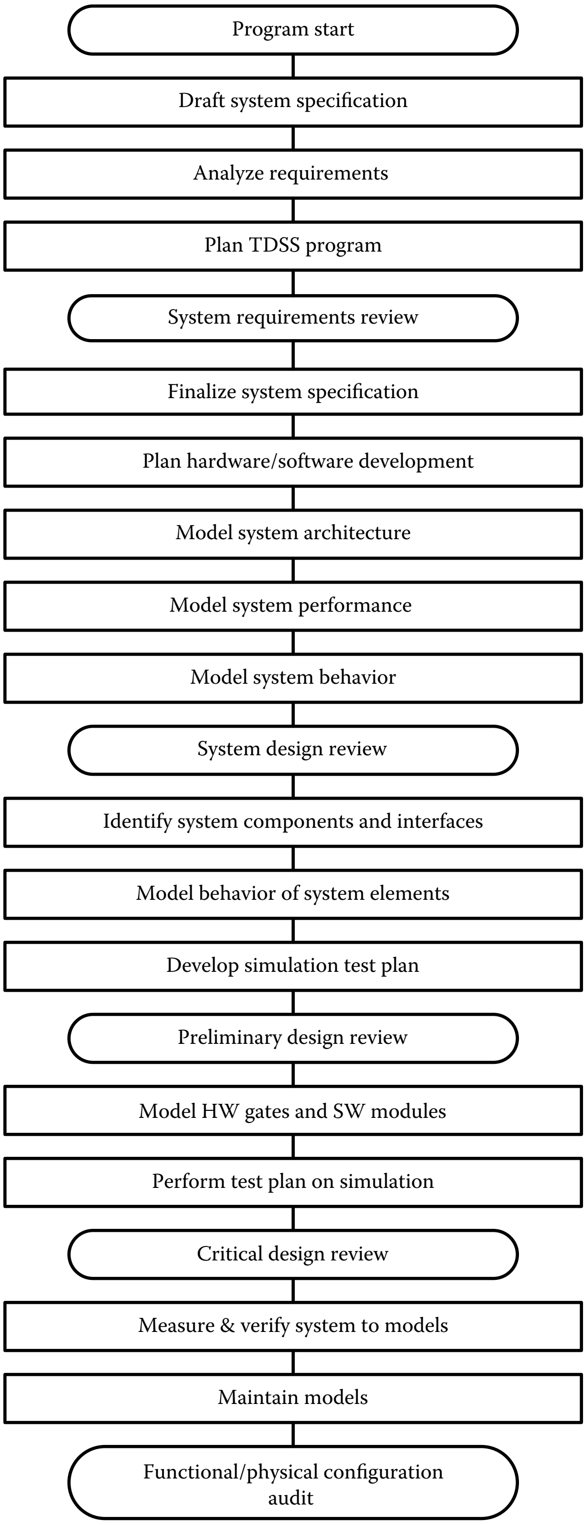

Program start
Draft system specification
Analyze requirements
Plan TDSS program
System requirements review
Finalize system specification
Plan hardware/software development
Model system architecture
Model system performance
Model system behavior
System design review
Identify system components and interfaces
Model behavior of system elements
Develop simulation test plan
Preliminary design review
Model HW gates and SW modules
Perform test plan on simulation
Critical design review
Measure & verify system to models
Maintain models
Functional/physical configuration audit

FIGURE 2.2 TDSS development flow.

expected partitioned functionality. Ensuring the correctness of the mid- and low-level models enables the system to be accurately and validly integrated in a synthetic sense prior to actually building the real hardware and software. Participating in the reviews, building the models, testing the modeled behavior, comparing results, and solving the discrepancies between differing team interpretations all contribute to the collective building of understanding within the entire implementation team. Confidence in the resulting system increases as does trust in the validity of the model.

Finally, the models are built to reflect the lowest level of hardware and software functionality. Designs are tested against the models, and deviations are repaired so as to yield the same results as the models predict. Remembering that the models are the standard by which the hardware and software is measured ensures that, once implemented, the lowest levels of the system perform as expected, and successively higher levels will likewise integrate properly. The TDSS models must be subjected to configuration control with the same rigor as the system hardware and software. Not doing so will allow the modeling effort to diverge.

By utilizing TDSS in flowing the system model to the lowest level, complete with executable models with common data sets implemented in understood, repeatable mechanizations will resolve the ambiguous interpretations arising between teams. By far the majority of system integration problems are due to those differing interpretations of ambiguous specifications. Giving the system designers a chance to perform multiple tests on the simulated system modeled top to bottom in virtual mode is a powerful tool. The system architects may then predict with confidence how the system will behave in a variety of conditions before the system is built. Demonstrating the system via modeling and simulation is the key to predictable and controlled system design. Discovering, understanding, solving, and testing the model interface differences goes a long way toward reducing the final system integration errors where the costs and schedule disruption associated with fixing problems is very high. These two results — confidence in the correctness of the system and verifiable integrated system behavior — justify the expense and resources required to create and maintain the TDSS.

2.4 Performance Modeling for System Partitioning: Case Study

The following case study describes a practical application of modeling and simulation used both to articulate and answer system-level questions regarding data latency so that system functional partitioning may be accomplished. Figure 2.3 represents a highly integrated avionics architecture. The system is intended to be fabricated utilizing COTS components to the greatest extent possible and is generalized to include connectivity to existing legacy subsystems.

2.4.1 System Description

This architecture includes the following functional components:

- Sensor Front Ends representing multiple sensors that receive signals and accomplish analog signal processing
- Embedded Computing Assets consisting of computing assets, closely coupled to the sensor front ends, which perform low-latency control processing, digital signal processing, and prepare data for transmission
- Asynchronous Transfer Mode (ATM) Dataflow Network serving as the data transfer medium on which data are transferred between Embedded Computing Assets, Core Computing Assets, Operator Workstations, and the LAN Gateway
- Core Computing Assets, including computing and memory devices used to perform application programs and services such as database and maintenance functions
- Operator Workstations serving as the man-machine interface

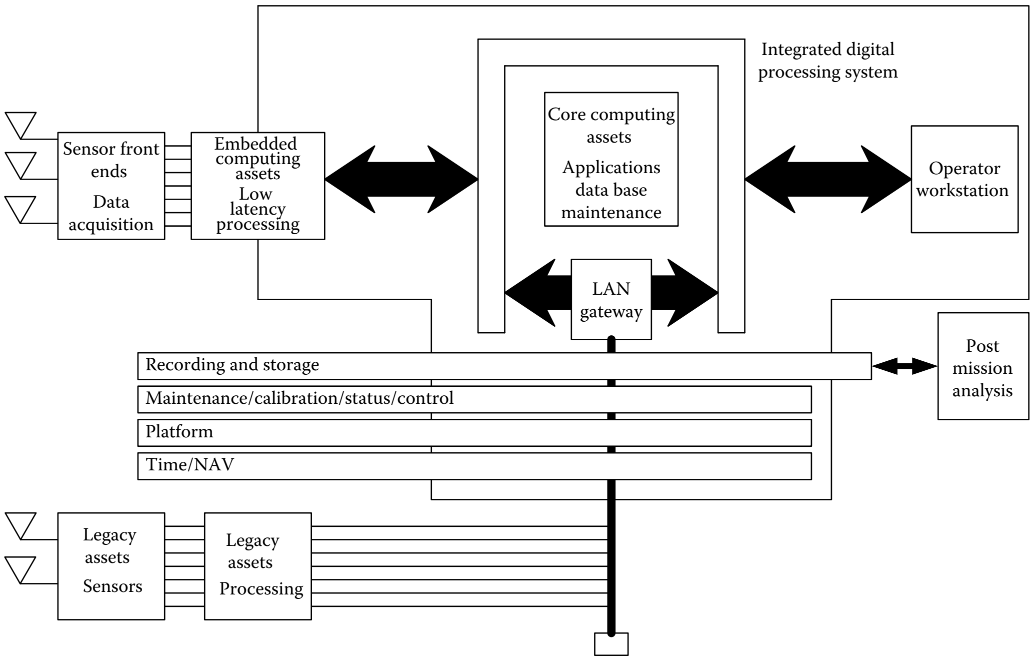


FIGURE 2.3 A generalized integrated avionics architecture.

- Recording and storage assets, maintenance, calibration, and control assets, as well as platform-specific and time/navigation assets
- LAN Gateway serving as the interface to Legacy Systems and networks

Given this architecture, the system designer must determine where to allocate software processes that operate on incoming sensor data. The designer must determine which processes are mission critical, which are low-latency processes, and which processes are non-low-latency processes that could be allocated to application software. Can the designer allocate all processes to the Core Computing Assets and use the core as a server? What processes, if any, should remain in computing assets local to the incoming sensor data? Is there any incoming sensor data that have to be processed within a time constraint for a successful mission? None of those questions can be answered until a key question is first addressed: Given that a data packet is stored in the RAM of the Embedded Computing Assets, what is the latency associated with that packet's transfer from Embedded Computing Asset RAM, across the ATM Dataflow Network, to Core Computing Asset RAM, and back to Embedded Asset RAM traversing the same path?

The example Embedded and Core Computing Assets each consist of a dual VME 64 backplane configuration, real-time Concurrent Maxion board set (including four 200-MHz R4400 CPU cards [XPU] with 128 MB of RAM per CPU, an input-output (I/O) card [XIO] conforming to VME 64 standards, and a crosspoint switching architecture capable of 1.2 GB/s peak), and two Fore Systems ATM to VME adapter cards supporting OC-3 155 MB/s SONET fiber (one of these cards is used for redundant purposes only). The Embedded Computing Assets also include MIL-STD-1553, IRIG B, and MXI cards. The Core Assets also include MIL-STD-1553, IRIG B, and SCSI-compatible VME cards. The ATM Dataflow Network (DFN) consists of a Fore Systems ASX-200BX 16-port switch capable of switching up to 1.2 GB/s data across a nonblocking switch fabric. This switch is used to connect the ATM adapter cards using OC-3 155 MB/s SONET fiber. The system is decomposed into the component block diagram shown in Figure 2.4, which was used as the basis for a performance model.

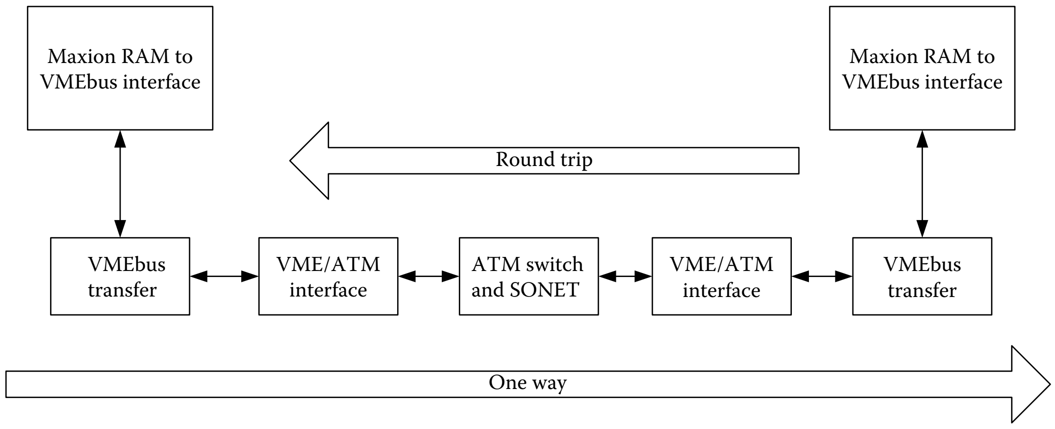


FIGURE 2.4 Component block diagram.

2.4.2 Model Development

Modeling and simulation of the system is accomplished using a commercial computer-aided engineering (CAE) tool (the OPNET modeling tool from OPNET Technologies Inc.), an event-driven simulation tool particularly well suited for computer and network modeling. The tool utilizes hierarchical domains defined as Network, Process, and State domains. A Network domain model for the system can be built using the Component Block Diagram shown in Figure 2.4. The Network domain of OPNET allows the specification of nodes and selection and modeling of data transfer pipes (OC-3 SONET, Ethernet, etc.). The Process domain allows the modeler to break down each network node into a configuration of queues, processors, transmitters, receivers, and packet generators. The State domain of the tool allows each processor and queue to be further decomposed into states, with definable transitions between states. Each state of a particular process is defined using C-based programming and simulation kernel calls to simulate the behavior of the process when it arrives at that state. To this end, any protocol behavior can be modeled with the proper combination of states and state behavior definitions. The tool also allows the modeler to probe the simulation at any point in the model to take measurements (throughput, queue capacities, delays, etc.) and analyze the results of these probes while the simulation is running. The system can be broken down into modeled components, as in the examples shown in Figure 2.5.

The model for the system is set up to transmit packets (1024 bytes) from RAM located in the Embedded Computing Assets, across the ATM DFN, to the RAM located in the Core Computing Assets, and then back to the RAM of the Embedded Computing Assets via the same path. During the simulation, the computing assets and ATM DFN are loaded at various levels with other data (a known characteristic of the real system is that other data with specific size and frequency rates will also be using the ATM DFN and computer resources at various points throughout the system).

Insertion of these data at various points throughout the model is achieved using the modeling tool's generator modules, which allows the model to be data-loaded at various points in a manner consistent with how the real system would be data loaded during normal operation. For this system the known system characteristics and protocols explicitly modeled in this example include, but are not limited to, the following:

- I/O read and write delays to/from the Core Computing Assets
- CPU wait cycles due to caching operations
- Crosspoint architecture arbitration
- TCP/IP protocol overhead and buffering operations
- VMEbus protocol, including service interrupt and acknowledgment and delays associated with the various addressing modes of VME operation
- ATM segmentation and reassembly delays

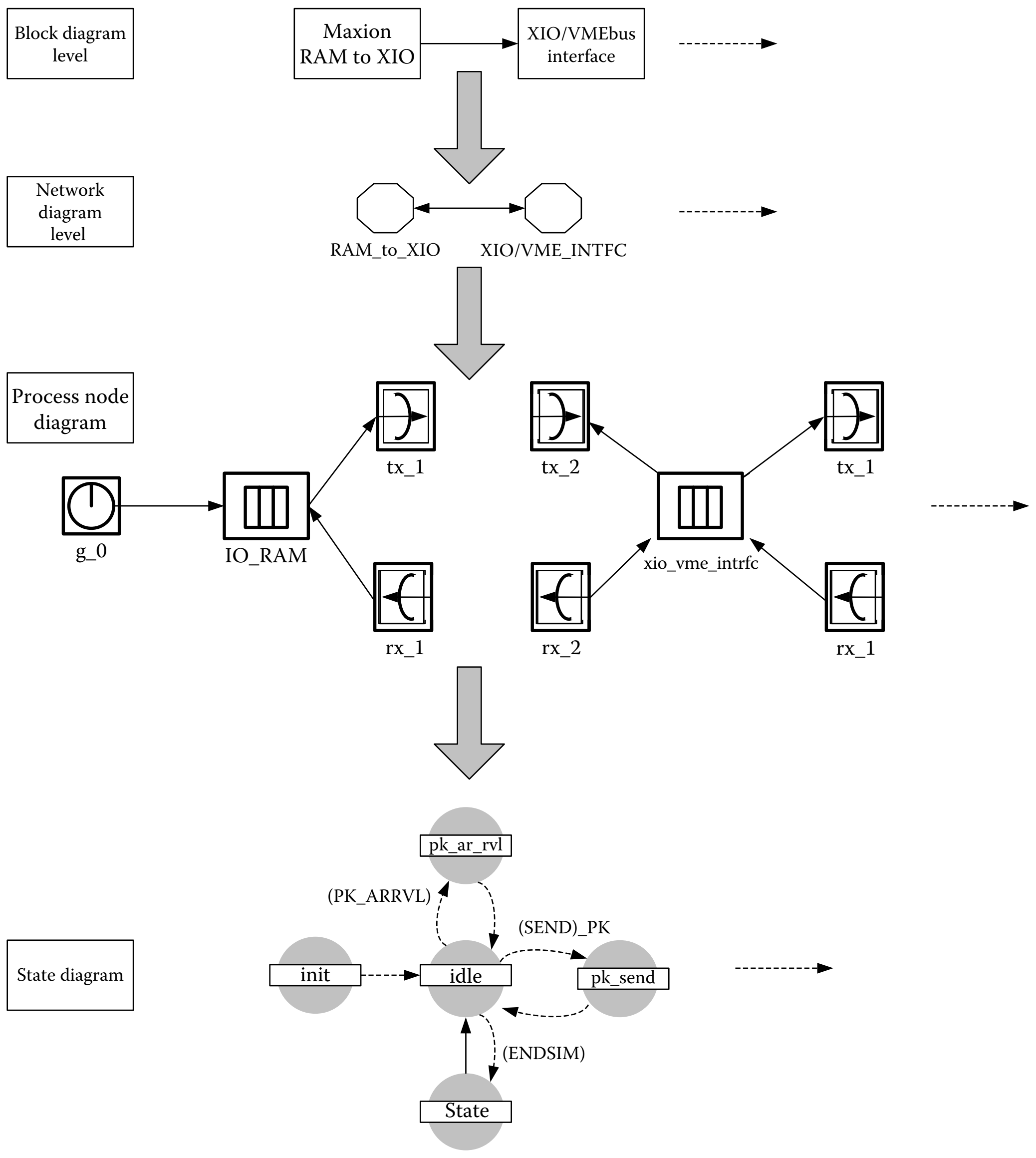


FIGURE 2.5 Modeling decomposition process includes component to network to process to state diagram development.

- ATM call setup delays
- ATM fabric switching
- Output port buffer delays

Also included in the model are known data sources that are used to simulate other traffic on the ATM Dataflow Network. A file of System Loading Parameters is also developed made up of a collection of all the modeling parameters available. These include, but are not strictly limited to, VMEBus addressing mode, TCP socket buffer thresholds, ATM switching delay parameters, SONET data transport overhead, proprietary communications layer overhead, and many others. The simulation was run ten times with the System Loading Parameters held constant. The average latency over the ten simulation runs was plotted as a function of those parameters. The parameters were changed, and another ten simulations

System load parameters	RAM to RAM latency
0.05	0.001444
0.10	0.010342
0.15	0.015953
0.20	0.027908
0.25	0.034309
0.30	0.039111
0.35	0.041234
0.40	0.043421
0.45	0.053901
0.50	0.059921
0.55	0.062344
0.60	0.063495
0.65	0.067401
0.70	0.075105
0.75	0.091120
0.80	0.099101
0.85	0.115230
0.90	0.136987
0.95	0.142335
1.00	0.150428

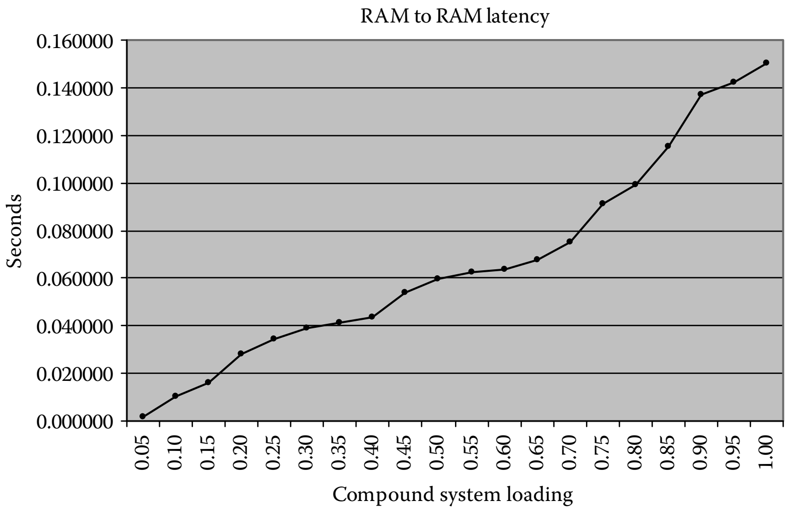


FIGURE 2.6 Case study simulation results.

were run at that particular level of system loading. System Loading Parameters were varied from 0.05 to 1.0 in increments of 0.05.

2.4.3 Modeling Results

The modeling simulation results are shown in Figure 2.6. Each point on the plot represents ten simulation runs. During each simulation run, the average round trip latency for the target data packets is measured. In a lightly loaded system, according to the simulation results, the target data could be expected to traverse the round trip path presented earlier in an average time of 0.001444 s. In a heavily loaded system, again using the simulation results, the data packets experience a latency of up to 0.150140 s. This is the maximum result just prior to 1.0 loading. Further, the shape of the data provides insight into system performance such that a potential "knee-in-the-curve" can be seen at 0.70 loading. These results are provided to the system designer as performance-based partitioning data and are used to allocate system functions between the Embedded Computing Assets and the Core Computing Assets.

2.4.4 Summary

The actual modeling and simulation effort that formed the basis for this case study was performed as part of a functional allocation process for an airborne system. The results allowed for system partitioning around a 150-ms boundary. By determining the latency early in the design phase, the design could be optimized to make best use of the available processing assets and avoid costly reallocations later in the system development.

This example further demonstrates several advantages to modeling and simulation of a system versus "vendor data sheet" calculations. First, vendors often report "best case" or "burst mode" performance characteristics. Modeling and simulation allows parameters of the system to be varied such that "heavily loaded" or "degraded performance" modes of the system may be investigated. Another distinct advantage is that the model is able to generate synchronous data, asynchronous data, and data based on probability distributions. Data latencies in the model are based on modeled event-triggered protocol disciplines rather than answers derived on a calculator or spreadsheet. Not every aspect of a real system can be modeled completely in every tool, but the known characteristics of the system that relate to performance form a quality model for performance estimation in a tool as used here.

2.5 Research Issues and Summary

As mentioned, not every aspect of a real system can be modeled completely in every tool (nor is it desired), but the known characteristics of the system that relate to performance form a quality model for performance estimation and capacity planning. Clearly market forces will continue to drive the quantity, quality, completeness, and rate of change of system engineering environments. As a practical matter, better integration and traceability between requirements automation systems and system/software engineering environments are called for. Improved traceability between proposed system concepts, their driving requirements, the resultant technical configuration(s), and their cost and schedule impacts will be a powerful result of this integration.

The requirement to integrate the trade space across technical, program, financial, and market boundaries is likely to continue and remain incomplete. To paraphrase Brooks, there is more common ground than the technical community will admit, and there are differences that most managers do not understand (Brooks, 1995). Automation alone will not reduce the complexities of this effort until there is a common, multidisciplinary, quantitative definition of cost versus price in system performance trades. As to the automation issue, the size, content, and format issues of current and legacy technical, financial, programmatic data bases will continue to grow and diverge until the stewards of college curricula produce graduates who solve more problems than they create (Parnas, 1989). These new practitioners must develop future information systems and engineering environments that encompass the disciplines, languages, and methods critical to improving the practice.

Defining Terms

ATM:	Asynchronous Transfer Mode
CAD:	Computer Automated Design
CAE:	Computer-Aided Engineering
COTS:	Commercial-off-the-shelf (products)
MIPS:	Millions of instructions per second, or, misleading indicator of processor speed
MITL:	Man-in-the-loop
OC"n":	Optical Carrier Level n (i.e., OC-3, a SONET specification)
RAM:	Random Access Memory
SONET:	Synchronous Optical Network
TDSS:	Top Down System Simulation

References

Allen, A.O., *Computer Performance Analysis with Mathematica,* Academic Press, New York. 1994.
Bell, C.G. and Newell, A., *Computer Structures: Readings and Examples*, McGraw-Hill, New York, 1973.
Bratley, P., Fox, B., and Schrage, L., *A Guide to Simulation*, 2nd ed., Springer-Verlag, New York, 1987.
Brooks J.R. and Fredrick P., *The Mythical Man Month*, Addison-Wesley, Reading, MA, 1995.
Donnelly, C.F., Evaluating the IOBIDS Specification Using Gate-Level System Simulation, in *Proc. IEEE Natl. Aerosp. Electron. Conf.*, 1992, p. 748.
Hennessy, J.L. and Patterson, D.A., *Computer Architecture: A Quantitative Approach,* 3rd ed., Morgan Kaufmann, San Francisco, 2003.
Kleinrock, L., *Theory, Queueing Systems*, vol. 1, John Wiley & Sons, New York, 1975.
Liu, H.-H., Software issues in hardware development, in *Computer Engineering Handbook*, McGraw-Hill, New York, 1992.
Parnas, D.L., Education for computing professionals, *Tech. Rep.* 89-247, 1989.
Portelli, W., Oseth, T., and Strauss, J.L., Demonstration of avionics module exchangeability via simulation (DAMES) program overview, in *Proc. IEEE Natl. Aerosp. Electron. Conf.,* 1989, pp. 660.

Strauss, J.L., The third possibility, in *Modeling and Simulation of Embedded Systems*, Proc. Embedded Computing Inst., 1994, pp. 160.

Swangim, J., Strauss, J.L., et al., Challenges of tomorrow — The future of secure avionics, in *Proc. IEEE Natl. Aerosp. Electron. Conf.*, 1989, pp. 580.

Walker, R.A. and Thomas, D.E., A model of design representation and synthesis, in *Proc. 22nd Design Automation Conf.*, 1985, pp. 453–459.

Further Information

Arnold Allen's text, *Computer Performance Analysis with Mathematica* (Academic Press, 1994), is an excellent and wonderfully readable introduction to computing systems performance modeling. It is out of print; if you can find a copy, buy it.

Hennessy and Patterson's third edition of their textbook *Computer Architecture: A Quantitative Approach* (Morgan Kaufmann, 2003) remains the definitive text in undergraduate and graduate-level computer engineering. The text is replete with "fallacies and pitfalls" sections that form boundaries around which modeling and simulation is useful.

For Performance Modeling and Capacity Planning, the following organizations provide information through periodicals and specialized publications:

- The Computer Measurement Group (CMG) (www.cmg.org)
- Association for Computing Machinery (www.acm.org)

For Computer-Aided Engineering, the following Web sites provide information on vendors of major tools and environments:

- www.opnet.com
- www.cadence.com
- www-306.ibm.com/software/rational/uml/
- www.mentor.com
- www-cdr.stanford.edu/SHARE/DesignNet.html

3 Formal Methods

Sally C. Johnson
NASA Langley Research Center

Ricky W. Butler
NASA Langley Research Center

3.1 Introduction

With each new generation of aircraft, the requirements for digital avionics systems become increasingly complex, and their development and validation consumes an ever-increasing percentage of the total development cost of an aircraft. The introduction of life-critical avionics, where failure of the computer hardware or software can lead to loss of life, brings new challenges to avionics validation. The FAA recommends that catastrophic failures of the aircraft be "so unlikely that they are not anticipated to occur during the entire operational life of all airplanes of one type" and suggests probabilities of failure on the order of 10^{-9} per flight hour [FAA, 1988].

There are two major reliability factors to be addressed in the design of ultrareliable avionics: hardware component failures and design errors. Physical component failures can be handled by using redundancy and voting. This chapter addresses the problem of design errors. Design errors are errors introduced in the development phase rather than the operational phase. These may include errors in the specification of the system, discrepancies between the specification and the design, and errors made in implementing the design in hardware or software. The discussion in this chapter centers mainly around software design errors; however, the increasing use of complex, custom-designed hardware instead of off-the-shelf components that have stood the test of time makes this discussion equally relevant for hardware.

The problem with software is that the complexity exceeds our ability to have intellectual control over it. Our intuition and experience is with continuous systems, but software exhibits discontinuous behavior. We are forced to separately reason about or test millions of sequences of discrete state transitions. Testing of software sufficient to assure ultrahigh reliability has been shown to be infeasible for systems of realistic complexity [Butler and Finelli, 1993], and fault tolerance strategies cannot be relied upon because of the demonstrated lack of independence between design errors in multiple versions of software [Knight and Leveson, 1986]. Since the reliability of ultrareliable software cannot be quantified, life-critical avionics software must be developed in a manner that concentrates on producing a correct design and implementation rather than on quantifying reliability after a product is built. This chapter describes how rigorous analysis employing formal methods can be applied to the software development process. While not yet standard practice in industry,* formal methods is included as an alternate means of compliance in the DO178B standard for avionics software development.

3.2 Fundamentals of Formal Methods

Formal methods is the use of formal mathematical reasoning or logic in the design and analysis of computer hardware and software. Mathematical logic serves the computer system designer in the same way that calculus serves the designer of continuous systems—as a notation for describing systems and as an analytical tool for calculating and predicting the behavior of systems.

Formal logic provides rules for constructing arguments that are sound because of their form and independent of their meaning, and consequently can be manipulated with a computer program. Formal logic provides rules for manipulating formulas in such a manner that, given valid premises, only valid conclusions are deducible. The manipulations are called a proof. If the premises are true statements about the world, then the soundness theorems of logic guarantee that the conclusion is also a true statement about the world. Furthermore, assumptions about the world are made explicit and are separated from the rules of deduction.

Formal methods can be roughly divided into two basic components: specification and verification. Each of these is discussed below.

3.2.1 Formal Specification

Formal specification is the use of notations derived from formal logic to define (1) the requirements that the system is to achieve, (2) a design to accomplish those requirements, and (3) the assumptions about the world in which a system will operate. The requirements explicitly define the functionality required from the system as well as enumerating any specific behaviors that the system must meet, such as safety properties.

A design specification is a description of the system itself. In practice, a set of hierarchical specifications is often produced during the design process ranging from a high-level, abstract representation of the system to a detailed implementation specification, as shown in Figure 3.1. The highest-level specification might describe only the basic requirements or functionality of the system in a state-transition format. The lowest-level specification would detail the algorithms used to implement the functionality. The hierarchical specification process guides the design and analysis of a system in an orderly manner, facilitating better understanding of the basic abstract functionality of the system as well as unambiguously clarifying the implementation decisions. Additionally, the resulting specifications serve as useful system documentation for later analysis and modification.

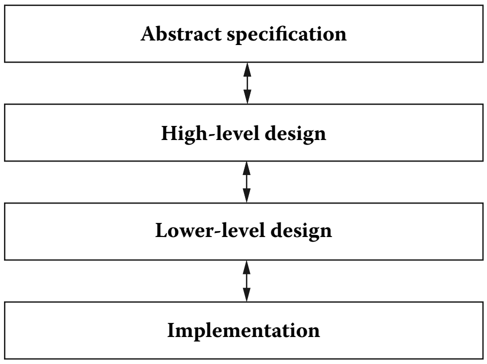


FIGURE 3.1 Hierarchy of formal specifications.

*However, Intel, Motorola, and AMD are using formal methods in the development of microchips.

An avionics system can be thought of as a complex interface between the pilot and the aircraft. Thus, the specification of the avionics system relies on a set of assumptions about the behaviors of the pilot and the response of the aircraft. Likewise, the specification of a modular subsystem of an integrated avionics system will rely on a set of assumptions about the behaviors and interfaces of the other subsystems. The unambiguous specification of interfaces between subsystems also prevents the problem of developers of different subsystems interpreting the requirements differently and arriving at the system integration phase of software development with incompatible subsystems.

3.2.2 Formal Verification

Formal verification is the use of proof methods from formal logic to (1) analyze a specification for certain forms of consistency and completeness, (2) prove that the design will satisfy the requirements, given the assumptions, and (3) prove that a more detailed design implements a more abstract one. The formal verifications may simply be paper-and-pencil proofs of correctness; however, the use of a mechanical theorem prover to ensure that all of the proofs are valid lends significantly more assurance and credibility to the process. The use of a mechanical prover forces the development of an argument for an ultimate skeptic who must be shown every detail.

In principle, formal methods can accomplish the equivalent of exhaustive testing, if applied to the complete specification hierarchy from requirements to implementation. However, such a complete verification is rarely done in practice because it is difficult and time-consuming. A more pragmatic strategy is to concentrate the application of formal methods on the most critical portions of a system. Parts of the system design that are particularly complex or difficult to comprehend can also be good candidates for more rigorous analysis. Although a complete formal verification of a large, complex system is impractical at this time, a great increase in confidence in the system can be obtained by the use of formal methods at key locations in the system.

There are several publicly available theorem-proving toolkits. These tools automate some of the tedious steps associated with formal methods, such as *typechecking* of specifications and conducting the simplest of proofs automatically. However, these tools must be used by persons skilled in mathematical logic to perform rigorous proofs of complex systems.

3.2.3 Limitations

For many reasons, formal methods do not provide an absolute guarantee of perfection, even if checked by an automated theorem prover. First, formal methods cannot guarantee that the top-level specification is what was intended. Second, formal methods cannot guarantee that the mathematical model of the physical world, such as aircraft control characteristics, is accurate. The mathematical model is merely a simplified representation of the physical world. Third, the formal verification process often is only applied to part of the system, usually the critical components. Finally, there may be errors in the formal verification tools themselves.* Nevertheless, formal methods provide a significant capability for discovering/removing errors in large portions of the design space.

3.3 Example Application

The techniques of formal specification and verification of an avionics subsystem will be demonstrated on a very simplified example of a mode control panel. An informal, English-language specification of the mode control panel, representative of what software developers typically encounter in practice, will be presented. The process of clarifying and formalizing the English specification into a formal specification, often referred to as requirements capture, will then be illustrated.

*We do not believe this to be a significant source of undetected errors in formal specifications or verifications.

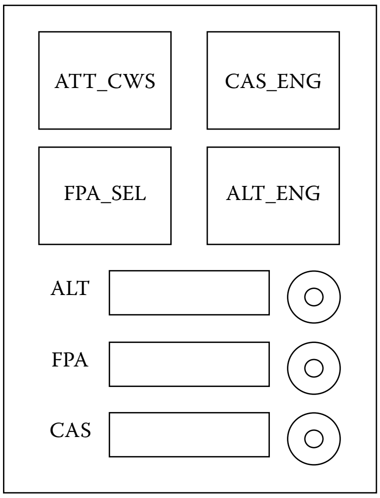


FIGURE 3.2 Mode control panel.

3.3.1 English Specification of the Example System

This section presents the informal, English-language specification of the example system. The English specification is annotated with section numbers to facilitate references back to the specification in later sections.

1. *The mode control panel contains four buttons for selecting modes and three displays for dialing in or displaying values, as shown in Figure 3.2. The system supports the following four modes:*

   attitude control wheel steering (`att_cws`),
   flight path angle selected (`fpa_sel`),
   altitude engage (`alt_eng`), *and*
   calibrated air speed (`cas_eng`).

   Only one of the first three modes can be engaged at any time. However, the `cas_eng` *mode can be engaged at the same time as any of the other modes. The pilot engages a mode by pressing the corresponding button on the panel. One of the three modes,* `att_cws`, `fpa_sel`, *or* `alt_eng`, *should be engaged at all times. Engaging any of the first three modes will automatically cause the other two to be disengaged since only one of these three modes can be engaged at a time.*
2. *There are three displays on the panel: air speed, flight path angle, and altitude. The displays usually show the current values for the air speed, flight path angle, and altitude of the aircraft. However, the pilot can enter a new value into a display by dialing in the value using the knob next to the display. This is the target or "preselected" value that the pilot wishes the aircraft to attain. For example, if the pilot wishes to climb to 25,000 feet, he will dial 250 into the altitude display window* and then press the* `alt_eng` *button to engage the altitude mode. Once the target value is achieved or the mode is disengaged, the display reverts to showing the "current" value.***
3. *If the pilot dials in an altitude that is more than 1200 feet above the current altitude and then presses the* `alt_eng` *button, the altitude mode will not directly engage. Instead, the altitude engage mode will change to "armed" and the flight path angle select mode is engaged. The pilot must then dial in*

*Altitude is expressed in terms of "flight level," which is measured in increments of 100 ft.

**In a real mode control panel, the buttons would be lit with various colored lights to indicate which modes are currently selected and whether "preselected" or "current" values are being displayed. Because of space limitations, the display lights will not be included in the example specification.

a flight path angle for the flight-control system to follow until the aircraft attains the desired altitude. The flight path angle select mode will remain engaged until the aircraft is within 1200 feet of the desired altitude, then the altitude engage mode is automatically engaged.

4. *The calibrated air speed and the flight path angle values need not be preselected before the corresponding modes are engaged — the current values displayed will be used. The pilot can dial in a different value after the mode is engaged. However, the altitude must be preselected before the altitude engage button is pressed. Otherwise the command is ignored.*
5. *The calibrated air speed and flight path angle buttons toggle on and off every time they are pressed. For example, if the calibrated air speed button is pressed while the system is already in calibrated air speed mode, that mode will be disengaged. However, if the attitude control wheel steering button is pressed while the attitude control wheel steering mode is already engaged, the button is ignored. Likewise, pressing the altitude engage button while the system is already in altitude engage mode has no effect.*

Because of space limitations, only the mode control panel interface itself will be modeled in this example. The specification will only include a simple set of commands the pilot can enter plus the functionality needed to support mode switching and displays. The actual commands that would be transmitted to the flight-control computer to maintain modes, etc. are not modeled.

3.3.2 Formally Specifying the Example System

In this section, we will describe the process of formally specifying the mode control panel described in English in the previous section. Quotes cited from the English specification will be annotated with the corresponding section number in parentheses. The goal is to completely describe the system requirements in a mathematical notation, yet not overly constrain the implementation.

This system collects inputs from the pilot and maintains the set of modes that are currently active. Thus, it is appropriate to model the system as a state machine. The state of the machine will include the set of modes that are active, and the pilot inputs will be modeled as events that transition the system from the current state (S_c) to a new state (S_n):

$$S_c \longrightarrow S_n$$

The arrow represents a transition function, `nextstate`, which is a function of the current state and an event, say ev:

$$S_n = \texttt{nextstate}(S_c,\ \texttt{ev})\ .$$

The goal of the formal specification process is to provide an unambiguous elaboration of the nextstate function. This definition must be complete; i.e., it must provide a next state for all possible events and all possible states. Thus, the first step is to elaborate all possible events and the content of the state of the machine. The system will be specified using the PVS specification language [Owre et al., 1993].

3.3.2.1 Events

The pilot interacts with the mode control panel by pressing the mode buttons and by dialing preselected values into the display. The pilot actions of pressing one of the four buttons will be named as follows: `press_att_cws`, `press_cas_eng`, `press_alt_eng`, and `press_fpa_sel`. The actions of dialing a value into a display will be named as follows: `input_alt`, `input_fpa`, and `input_cas`. The behavior of the mode control panel also depends upon the following inputs it receives from sensors: `alt_reached`, `fpa_reached`, and `alt_gets_near`. In PVS, the set of events are specified as follows:

```
events: TYPE = {press_att_cws, press_cas_eng, press_fpa_sel,
                press_alt_eng, input_alt, input_fpa, input_cas,
                alt_gets_near, alt_reached, fpa_reached}
```

3.3.2.2 State Description

The *state* of a system is a collection of attributes that represents the system's operation. In the example system, the set of active modes would certainly be a part of the system state. Also, the values in the displays that show altitude, flight path angle, and air speed, and the readings from the airplane sensors would be included in the state. Formalization of the system state description entails determining which attributes are necessary to fully describe the system's operation, then choosing a suitable formalism for representing those attributes.

One possible approach to describing the system state for the example is to use a set to delineate which modes are active. For example, {att_cws, cas_eng} would represent the state of the system where both att_cws and cas_eng are engaged but alt_eng and fpa_sel are not engaged. Also, the alt_eng mode has the additional capability of being armed. Thus, a better approach to describing the example system state is to associate with each mode one of the following values: off, armed, or engaged. In PVS, a type or domain can be defined with these values:

```
mode_status: TYPE = {off, armed, engaged}
```

The state descriptor is as follows:*

```
[# % RECORD
att_cws: mode_status,
cas_eng: mode_status,
fpa_sel: mode_status,
alt_eng: mode_status,
#] % END
```

For example, the record [att_cws=engaged, cas_eng=engaged, fpa_sel=off, alt_eng=off] would indicate a system where both att_cws and cas_eng are engaged, fpa_sel is off and alt_eng is off. However, there is still a problem with the state descriptor; in the example system only alt_eng can be armed. Thus, more restrictive domains are needed for the modes other than alt_eng. This can be accomplished by defining the following sub-types of mode_status:

```
off_eng: TYPE={mode: mode_status | mode = off OR
                                   mode = engaged}
```

The type off_eng has two values: off and engaged. The state descriptor is thus corrected to become:

```
[# % RECORD
att_cws: off_eng,
cas_eng: off_eng,
fpa_sel: off_eng,
alt_eng: mode_status
#] % END
```

The mode panel also maintains the state of the displays. To simplify the example, the actual values in the displays will not be represented. Instead, the state descriptor will only keep track of whether the value is a preselected value or the actual value read from a sensor. Thus, the following type is added to the formalism:

```
disp_status: TYPE = {pre_selected, current}
```

and three additional fields are added to the state descriptor:

*In the PVS language, a record definition begins with [# and ends with #]. The % denotes that the remainder of the line contains a comment and is ignored.

```
    alt_disp: disp_status,
    fpa_disp: disp_status, and
    cas_disp: disp_status.
```

The behavior of the mode control panel does not depend upon the actual value of "altitude" but rather on the relationship between the actual value and the preselected value in the display. The following "values" of altitude are sufficient to model this behavior:

```
altitude_vals: TYPE = {away, near_pre_selected,
                       at_pre_selected},
```

and the following is added to the state descriptor:

```
altitude: altitude_vals.
```

The final state descriptor is

```
states: TYPE = [# % RECORD
    att_cws: off_eng,
    cas_eng: off_eng,
    fpa_sel: off_eng,
    alt_eng: mode_status,
    alt_disp: disp_status,
    fpa_disp: disp_status,
    cas_disp: disp_status,
    altitude: altitude_vals
    #] END
```

3.3.2.3 Formal Specification of Nextstate Function

Once the state descriptor is defined, the next step is to define a function to describe the system's operation in terms of state transitions. The `nextstate` function can be defined in terms of ten subfunctions, one for each event, as follows:

```
event: VAR events
st: VAR states =
nextstate(st,event): states
   CASES event OF
     press_att_cws: tran_att_cws(st),
     press_cas_eng: tran_cas_eng(st),
     press_fpa_sel: tran_fpa_sel(st),
     press_alt_eng: tran_alt_eng(st),
     input_alt    : tran_input_alt(st),
     input_fpa    : tran_input_fpa(st),
     input_cas    : tran_input_cas(st),
     alt_gets_near: tran_alt_gets_near(st),
     alt_reached  : tran_alt_reached(st),
     fpa_reached  : tran_fpa_reached(st)
   ENDCASES
```

The CASES statement is equivalent to an IF-THEN-ELSIF-ELSE construct. For example, if the event is `press_fpa_sel then nextstate(st,event) = tran_fpa_sel(st).` The step is to define each of these subfunctions.

3.3.2.4 Specifying the att_cws Mode

The tran_att_cws function describes what happens to the system when the pilot presses the att_cws button. This must be specified in a manner that covers all possible states of the system. According to the English specification, the action of pressing this button attempts to engage this mode if it is off. Changing the att_cws field to engaged is specified as follows:

```
st WITH [att_cws := engaged].
```

The WITH statement is used to alter a record in PVS. This expression produces a new record that is identical to the original record st in every field except att_cws. Of course, this is not all that happens in the system. The English specification also states that, "*Only one of the* [att_cws, fpa_sel, *or* cas_eng] *modes can be engaged at any time*" (1). Thus, the other modes must become something other than engaged. It is assumed that this means they are turned off. This would be indicated as:

```
st WITH [att_cws:= engaged, fpa_sel:= off, alt_eng:= off].
```

The English specification also states that when a mode is disengaged, "*…the display reverts to showing the "current value:"* (2):

```
st WITH [att_cws := engaged, fpa_sel := off, alt_eng := off,
         alt_disp := current, fpa_disp := current].
```

The English specification also says that, "*…if the attitude control wheel steering button is pressed while the attitude control wheel steering mode is already engaged, the button is ignored*" (5). Thus, the full definition is

```
tran_att_cws(st): states =
  IF att_cws(st) = off THEN
       st WITH [att_cws := engaged, fpa_set := off,
                alt_eng := off, alt_disp := current,
                fpa_disp := current]
  ELSE st %% IGNORE: state is not altered at all
  ENDIF
```

The formal specification has elaborated exactly what the new state will be. The English specification does not address what happens to a preselected alt_eng display or preselected fpa_sel display when the att_cws button is pressed. However, the formal specification does explicitly indicate what will happen: these modes are turned off. The process of developing a complete mathematical specification has uncovered this ambiguity in the English specification. If the displays should not be changed when the corresponding mode is off, then the following specification would be appropriate:

```
tran_att_cws(st): states =
  IF att_cws(st) = off THEN
    st WITH [att_cws := engaged,
             fpa_sel := off,
             alt_eng := off,
      alt_disp := IF alt_eng(st) = off THEN alt_disp(st)
                  ELSE current ENDIF,
      fpa_disp := IF fpa_sel(st) = off THEN fpa_disp(st)
                  ELSE current ENDIF]
  ELSE st %% IGNORE : state is not altered at all
  ENDIF
```

We realize that this situation will arise in several other cases as well. In particular, whenever a mode becomes engaged, should any other preselected displays be changed to current or should they remain preselected? We decide to consult the system designers. They agree that the displays should be returned to current and suggest that the following be added to the English specification:

6. *Whenever a mode other than* `cas_eng` *is engaged, all other preselected displays should be returned to* `current`*.*

3.3.2.5 Specifying the `cas_eng` Mode

The `tran_cas_eng` function describes what happens to the system when the pilot presses the `cas_eng` button. This is the easiest mode to specify because its behavior is completely independent of the other modes. Pressing the `cas_eng` button merely toggles this mode on and off. The complete function is

```
tran_cas_eng(st): states =
       IF cas_eng(st) = off THEN
         st WITH [cas_eng := engaged]
       ELSE st WITH [cas_eng := off, cas_disp := current]
       ENDIF
```

This specification states that if the `cas_eng` mode is currently off, pressing the button will engage the mode. If the mode is engaged, pressing the button will turn the mode off. Thus, the button acts like a switch.

3.3.2.6 Specifying the `fpa_sel` Mode

The `tran_fpa_sel` function describes the behavior of the system when the `fpa_sel` button is pressed. The English specification states that this mode "*need not be preselected*" (4). Thus, whether the mode is off or preselected, the outcome is the same:

```
IF fpa_sel(st) = off THEN
    st WITH [fpa_sel := engaged, att_cws := off,
                alt_eng := off, alt_disp := current]
```

Note that not only is the `fpa_sel` mode engaged, but `att_cws` and `alt_eng` are turned off as well. This was included because the English specification states that, "*Engaging any of the first three modes will automatically cause the other two to be disengaged*" (1). Also note that this specification indicates that `alt_disp` is set to "current." The English specification states that, "*Once the target value is achieved or the mode is disengaged, the display reverts to showing the 'current' value*" (2). Thus, the altitude display must be specified to return to "current" status. If the `alt_eng` mode was not currently active, the `WITH` update does not actually change the value, but merely updates that attribute to the value it already holds.

Since PVS requires that functions be completely defined, we must also cover the case where `fpa_sel` is already engaged. We consult the English specification and find, "*The calibrated air speed and flight path angle buttons toggle on and off every time they are pressed.*" We interpret this to mean that if the `fpa_sel` button is pressed while the mode is engaged, the mode will be turned off. This is specified as follows:

```
st WITH [fpa_sel := off, fpa_disp := current]
```

Because the mode is disengaged, the corresponding display is returned to current. We realize that we also must cover the situation where the `alt_eng` mode is armed and the `fpa_sel` is engaged. In fact, Section (3) of the English specification indicates that this will occur when one presses the `alt_eng` button and the airplane is far away from the preselected altitude. However, Section (3) does not tell us whether the disengagement of `fpa_sel` will also disengage the armed `alt_eng` mode. We decide to consult the system designers. They inform us that pressing the `fpa_sel` button should turn off both the `fpa_sel` and `alt_eng` mode in this situation. Thus, we modify the state update statement as follows:

```
st WITH [fpa_sel := off,  alt_eng := off,
             fpa_disp := current, alt_disp := current]
```

The complete specification is thus:

```
tran_fpa_sel(st): states =
    IF fpa_sel(st) = off THEN
      st WITH [fpa_sel := engaged, att_cws  := off,
                   alt_eng := off, alt_disp := current]
    ELSE st WITH [fpa_sel := off,  alt_eng := off,
                   fpa_disp := current, alt_disp := current]
    ENDIF
```

The perspicacious reader may have noticed that there is a mistake in this formal specification. The rest of us will discover it when a proof is attempted using a theorem prover in the later section entitled, "Formal Verification of the Example System."

3.3.2.7 Specifying the `alt_eng` Mode

The `alt_eng` mode is used to capture a specified altitude and hold it. This is clearly the most difficult of the four to specify since it has a complicated interaction with the `fpa_sel` mode.

The English specification states that, "*The altitude must be preselected before the altitude engage button is pressed*" (4). This is interpreted to mean that the command is simply ignored if it has not been preselected. Consequently, the specification of `tran_alt_eng` begins:

```
tran_alt_eng(st): states =
   IF alt_disp(st) = pre_selected THEN
       ...
   ELSE st % IGNORE
   ENDIF
```

This specifies that the system state will change as a result of pressing the `alt_eng` button only if the `alt_disp` is preselected.

We must now proceed to specify the behavior when the `IF` expression is true. The English specification indicates that if the aircraft is more than 1200 ft from the target altitude, this request will be put on hold (the mode is said to be armed) and the `fpa_sel` mode will be engaged instead. The English specification also says that, "*The pilot must then dial in a flight path angle at this point*" (3). The question arises whether the `fpa_sel` engagement should be delayed until this is done. Another part of the English specification offers a clue, "*The calibrated air speed and flight path angle values need not be preselected before the corresponding modes are engaged*" (4). Although this specifically addresses the case of pressing the `fpa_sel` button and not the situation where the `alt_eng` button indirectly turns this mode on, we suspect that the behavior is the same. Nevertheless, we decide to check with the system designers to make sure. The system designers explain that this is the correct interpretation and that this is the reason the mode is called "flight path angle select" rather than "flight path angle engage."

The behavior must be specified for the two situations: when the airplane is near the target and when it is not. There are several ways to specify this behavior. One way is for the state specification to contain the current altitude in addition to the target altitude. This could be included in the state vector as two numbers:

```
target_altitude: number
actual_altitude: number
```

The first number contains the value dialed in and the second the value last measured by a sensor. The specification would then contain:

```
IF abs(target_altitude − actual_altitude) > 1200 THEN
```

where abs is the absolute value function. If the behavior of the mode control panel were dependent upon the target and actual altitudes in a multitude of ways, this would probably be the proper approach. However, in the example system the behavior is only dependent upon the relation of the two values to each other. Therefore, another way to specify this behavior is by abstracting away the details of the particular values and only storing information about their relative values in the state descriptor record. In particular, the altitude field of the state record can take on one of the following three values:

away	the preselected value is $>$ 1200 feet away
near_pre_selected	the preselected value is $<=$ 1200 feet away
at_pre_selected	the preselected value is $=$ the actual altitude

The two different situations can then be distinguished as follows:

```
IF altitude(st) = away THEN
```

When the value is not away, the alt_eng mode is immediately engaged. This is specified as follows:

```
st WITH [alt_eng := engaged, att_cws := off,
         fpa_sel := off, fpa_disp := current
```

Note that not only is the alt_eng mode engaged, but this specification indicates that several other modes are affected just as in the tran_fpa_sel subfunction.

Now the behavior of the system must be specified for the other case, when the aircraft is away from the target altitude. In this case fpa_sel is engaged and alt_eng is armed:

```
ELSE
 st WITH [fpa_sel := engaged, att_cws := off,
           alt_eng := armed]
```

As before, the att_cws mode is also turned off.

So far we have not considered whether the behavior of the system should be different if the alt_eng mode is already armed or engaged. The English specification states that, "*Pressing the altitude engage button while the system is already in altitude engage mode has no effect*" (5). However, there is no information about what will happen if the mode is armed. Once again, the system designers are consulted, and we are told that the mode control panel should ignore the request in this case as well. The complete specification of tran_alt_eng becomes:

```
tran_alt_eng(st): states =
   IF alt_eng(st) = off AND alt_disp(st) = pre_selected THEN
        IF altitude(st) /= away THEN %% ENGAGED
             st WITH [att_cws := off, fpa_sel := off,
                      alt_eng := engaged, fpa_disp := current]
        ELSE st WITH [att_cws := off, fpa_sel := engaged,
                      alt_eng := armed] %% ARMED
        ENDIF
   ELSE st %% IGNORE request
   ENDIF
```

Note that the last ELSE takes care of both the armed and engaged cases.

3.3.2.8 Input to Displays

The next three events that can occur in the system are input_alt, input_fpa, and input_cas. These occur when the pilot dials a value into one of the displays. The input_alt event corresponds to the subfunction of nextstate named tran_input_alt. The obvious thing to do is to set the appropriate field as follows:

```
st WITH [alt_disp := pre_selected]
```

This is certainly appropriate when alt_eng is off. However, we must carefully consider the two cases: (1) when the alt_eng mode is armed, and (2) when it is engaged. In this case, the pilot is changing the target value after the alt_eng button has been pressed. The English specification required that the alt_eng mode be preselected before it could become engaged, but did not rule out the possibility that the pilot could change the target value once it was armed or engaged. We consult the system designers once again. They inform us that entering a new target altitude value should return the alt_eng mode to off and the pilot must press the alt_eng button again to reengage the mode. We add the following to the English specification:

7. *If the pilot dials in a new altitude while the alt_eng button is already engaged or armed, then the alt_eng mode is disengaged and the att_cws mode is engaged.*

The reason given by the system designers was that they didn't want the altitude dial to be able to automatically trigger a new active engagement altitude. They believed it was safer to force the pilot to press the alt_eng button again to change the target altitude.

Thus, the specification of tran_input_alt is

```
tran_input_alt(st): states =
  IF alt_eng(st) = off THEN
     st WITH [alt_disp := pre_selected]
  ELSE % alt_eng(st) = armed OR alt_eng(st) = engaged THEN
     st WITH [alt_eng := off, alt_disp := pre_selected,
              att_cws := engaged,
              fpa_sel := off, fpa_disp := current]
  ELSE st
  ENDIF
```

The other input event functions are similar:

```
tran_input_fpa(st): states =
  IF fpa_sel(st)  = off THEN st WITH [fpa_disp :  = pre_selected]
  ELSE st ENDIF
tran_input_cas(st): states =
  IF cas_eng(st) = off THEN st WITH [cas_disp := pre_selected]
  ELSE st ENDIF
```

3.3.2.9 Other Actions

There are other events that are not initiated by the pilot but that still affect the mode control panel; in particular, changes in the sensor input values. As described previously, rather than including the specific values of the altitude sensor, the state descriptor only records which of the following is true of the preselected altitude value: away, near_pre_selected or at_pre_selected. Events must be defined that correspond to significant changes in the altitude so as to affect the value of this field in the state. Three such events affect the behavior of the panel:

alt_gets_near	the altitude is now near the preselected value
alt_reached	the altitude reaches the preselected value
alt_gets_away	the altitude is no longer near the preselected value

The transition subfunction associated with the first event must consider the case where the *alt_eng* mode is armed because the English specification states that, "*The flight path angle select mode will remain* *engaged until the aircraft is within 1200 feet of the desired altitude, then the altitude engage mode is automatically engaged*" (3). Thus we have:

```
tran_alt_gets_near(st): states =
     IF alt_eng(st) = armed THEN
          st WITH [altitude := near_pre_selected,
                   alt_eng := engaged, fpa_sel  := off]
   ELSE st WITH [altitude := near_pre_selected]
   ENDIF
```

The subfunction associated with the second event is similar because we cannot rule out the possibility that the event `alt_reached` may occur without `alt_gets_near` occurring first:

```
tran_alt_reached(st): states =
     IF alt_eng(st) = armed THEN
          st WITH [altitude := at_pre_selected,
                   alt_disp := current, alt_eng  := engaged,
                   fpa_sel := off]
   ELSE st WITH [altitude := at_pre_selected,
                 alt_disp := current]
   ENDIF
```

Note that in this case, the `alt_disp` field is returned to current because the English specification states,"*Once the target value is achieved or the mode is disengaged, the display reverts to showing the 'current' value*" (2).

However, the third event is problematic in some situations. If the `alt_eng` mode is engaged, is it even possible for this event to occur? The flight-control system is actively holding the altitude of the airplane at the preselected value. Thus, unless there is some major external event such as a windshear phenomenon, this should never occur. Of course, a real system should be able to accommodate such unexpected events. However, to shorten this example, it will be assumed that such an event is impossible. The situation where the `alt_eng` mode is not engaged or armed would not be difficult to model, but also would not be particularly interesting to demonstrate.

There are three possible events corresponding to each of the other displays. These are all straightforward and have no impact on the behavior of the panel other than changing the status of the corresponding display. For example, the event of the airplane reaching the preselected flight path angle is `fpa_reached`. The specification of the corresponding subfunction `tran_fpa_reached` is:

```
tran_fpa_reached(st): states =
     st WITH [fpa_disp  := current]
```

3.3.2.10 Initial State

The formal specification must include a description of the state of the system when the mode control panel is first powered on. One way to do this would be to define a particular constant, say `st0`, that represents the initial state:

```
st0: states = (# att_cws  := engaged, cas_eng   := off,
                 fpa_sel  := off, alt_eng   := off,
                 alt_disp   := current, fpa_disp   := current,
                 cas_disp   := current, altitude   := away #)
```

Alternatively, one could define a predicate (i.e., a function that returns true or false) that indicates when a state is equivalent to the initial state:

```
is_initial(st) : bool =
    att_cws(st) = engaged AND cas_eng(st)  = off AND
    fpa_sel(st) = off AND alt_eng(st)  = off AND
    alt_disp(st) = current AND fpa_disp(st)  = current AND
    cas_disp(st) = current
```

Note that this predicate does not specify that the altitude field must have the value "away." Thus, this predicate defines an equivalence class of states, not all identical, in which the system could be initially. This is the more realistic way to specify the initial state since it does not designate any particular altitude value.

3.3.3 Formal Verification of the Example System

The formal specification of the mode control panel is complete. But how does the system developer know that the specification is correct? Unlike the English specification, the formal specification is known to be detailed and precise. But it could be unambiguously wrong. Since this is a requirements specification, there is no higher-level specification against which to prove this one. Therefore, ultimately the developer must rely on human inspection to insure that the formal specification is "what was intended." Nevertheless, the specification can be analyzed in a formal way. In particular, the developer can postulate properties that he believes should be true about the system and attempt to prove that the formal specification satisfies these properties. This process serves to reinforce the belief that the specification is what was intended. If the specification cannot be proven to meet the desired properties, the problem in the specification must be found or the property must be modified until the proof can be completed. In either case, the developer's understanding of and confidence in the system is increased.

In the English specification of the mode control panel, there were several statements made that characterize the overall behavior of the system. For example, *"One of the three modes [`att_cws`, `fpa_sel`, or `alt_eng`] should be engaged at all times"* (1). This statement can be formalized, and it can be proven that no matter what sequence of events occurs, this will remain true of the system. Properties such as this are often called system *invariants*. This particular property is formalized as follows:

```
att_cws(st) = engaged
OR fpa_sel(st) = engaged
OR alt_eng(st) = engaged
```

Another system invariant can be derived from the English specification: *"Only one of the first three modes [`att_cws`, `fpa_sel`, `alt_eng`] can be engaged at any time"* (1). This can be specified in several ways. One possible way is as follows:

```
(alt_eng(st) /= engaged OR fpa_sel(st) /= engaged) AND
(att_cws(st) = engaged IMPLIES
     alt_eng(st) /= engaged AND fpa_sel(st) /= engaged)
```

Finally, it would be prudent to insure that whenever `alt_eng` is armed that `fpa_sel` is engaged:

```
(alt_eng(st) = armed IMPLIES fpa_sel(st) = engaged).
```

All three of these properties can be captured in one *predicate* (i.e., a function that is true or false) as follows:

```
valid_state(st): bool =
       (att_cws(st) = engaged OR fpa_sel(st) = engaged
                             OR alt_eng(st) = engaged)
       AND (alt_eng(st) /= engaged OR fpa_sel(st) /= engaged)
       AND (att_cws(st) = engaged IMPLIES
       alt_eng(st) /= engaged AND fpa_sel(st) /= engaged)
       AND  (alt_eng(st)= armed IMPLIES
              fpa_sel(st)= engaged)
```

The next step is to prove that this is always true of the system. One way to do this is to prove that the initial state of the system is valid and that if the system is in a valid state before an event then it is in a valid state after an event, no matter what event occurs. In other words, we must prove the following two theorems:

```
initial_valid: THEOREM is_initial(st) IMPLIES valid_state(st)
nextstate_valid: THEOREM valid_state(st) IMPLIES
                              valid_state(nextstate(st,event))
```

These two theorems effectively prove by induction that the system can never enter a state that is not valid.* Both of these theorems are proved by the single PVS command, GRIND. The PVS system replays the proofs in 17.8 s on a 450 MHz PC (Linux) with 384 MB of memory.

As mentioned earlier, the specification of fpa_sel contains an error. On the attempt to prove the nextstate_valid theorem on the erroneous version of fpa_sel described earlier, the prover stops with the following sequent:

```
   nextstate_valid :
[21] fpa_sel(st!1)=engaged
[22] press_fpa_sel?(event!1)
  u-------
[1] att_cws(st!1)=engaged
[2] alt_eng(st!1)=engaged
[3] press_att_cws?(event!1)
[4] press_alt_eng?(event!1)
```

The basic idea of a sequent is that one must prove that one of the statements after the |------- is provable from the statements before it. In other words, one must prove:

```
[−1] AND [−2] ===> [1] OR [2] OR [3] OR [4]
```

We see that formulas [3] and [4] are impossible, because press_fpa_sel?(event!1) tells us that event!1 = press_fpa_sel and not press_att_cws or press_alt_eng. Thus, we must establish [1] or [2]. However, this is impossible. There is nothing in this sequent to require that att_cws(st!1) =engaged or that alt_eng(st!1)=engaged. Thus, it is obvious at this point that something is wrong with the specification or the proof. It is clear that the difficulty surrounds the case when the event press_fpa_sel occurs, so we examine tran_fpa_sel more closely. We realize that the specification should have set att_cws to engaged as well as turning off the fpa_sel mode and alt_eng mode:

```
tran_fpa_sel(st): states =
  IF fpa_sel(st)=off THEN
      st WITH [fpa_sel := engaged, att_cws := off,
            alt_eng := off, alt_disp := current]
  ELSE st WITH [fpa_sel := off,  alt_eng := off,
```

*In order for this strategy to work, the invariant property (i.e., valid_state) must be sufficiently strong for the induction to go through. If it is too weak, the property may not be provable in this manner even though it is true. This problem can be overcome by either strengthening the invariant (i.e., adding some other terms) or by decomposing the problem using the concept of reachable states. Using the latter approach, one first establishes that a predicate "reachable (st)" delineates all of the reachable states. Then one proves that all reachable states are valid, i.e., reachable (st) ⇒ valid_state(st).

```
                        att_cws := engaged,
                        fpa_disp := current, alt_disp := current]
    ENDIF
```

This modification is necessary because otherwise the system could end up in a state where no mode was currently active. After making the correction, the proof succeeds.

Thus we see that formal verification can be used to establish global properties of a system and to detect errors in the specifications.

3.3.4 Alternative Methods of Specifying Requirements

Many systems can be specified using the state-machine method illustrated in this chapter. However, as the state-machine becomes complex, the specification of the state transition functions can become exceedingly complex. Therefore, many different approaches have been developed to elaborate this machine. Some of the more widely known are decision tables, decision trees, state-transition diagrams, state-transition matrices, Statecharts, Superstate, and R-nets [Davis, 1988].

Although these methods effectively accomplish the same thing—the delineation of the state machine —they vary greatly in their input format. Some use graphical notation, some use tables, and others use language constructs. Aerospace industries have typically used the table-oriented methods because they are considered the most readable when the specification requires large numbers of pages. Although there is insufficient space to discuss any particular method in this chapter, expression of a part of the mode control panel using a table-oriented notation will be illustrated briefly.

Consider the following table describing the `att_cws` mode:

`att_cws`	Event	New Mode
`off`	`press_att_cws`	`engaged`
	`press_fpa_sel WHEN fpa_sel /= off`	`engaged`
	`input_alt WHEN alt_eng /= off`	`engaged`
	`all others`	`off`
`engaged`	`press_att_cws`	`engaged`
	`press_alt_eng WHEN alt_eng = off AND alt_disp = pre_selected`	`off`
	`press_fpa_sel WHEN fps_sel = off`	`off`
	`All others`	`engaged`

The first column lists all possible states of `att_cws` prior to the occurrence of an event. The second column lists all possible events, and the third column lists the new state of the `att_cws` mode after the occurrence of the event. Note also that some events may include other conditions. This table could be specified in PVS as follows:

```
att_cws_off: AXIOM att_cws(st) = off IMPLIES
att_cws(nextstate(event, st)) =
          IF (event = press_att_cws) OR
             (event = press_fpa_sel AND fpa_sel(st) /= off) OR
             (event = input_alt AND alt_eng(st) /= off) THEN engaged
          ELSE off
          ENDIF
att_cws_eng: AXIOM att_cws(st) = engaged IMPLIES
       att_cws(nextstate(event,st))=
         IF (event = press_alt_eng AND alt_eng(st) = off
                       AND alt_disp(st) = pre_selected) OR
             (event = press_fpa_sel AND fps_sel(st) = off) THEN off
```

```
        ELSE engaged
        ENDIF
```

This approach requires that nextstate be defined in pieces (axiomatically) rather than definitionally. If this approach is used, it is necessary to analyze the formal specification to make sure that `nextstate` is completely defined. In particular, it must be shown that the function's behavior is defined for all possible values of its arguments and that there are no duplications. This must be performed manually if the tables are not formalized. The formalization can be done using a general-purpose theorem prover such as PVS or using a special-purpose analysis tool such as Tablewise* [Hoover and Chen, 1994].

When the specification can be elaborated in a finite-state machine, there are additional analysis methods available that are quite powerful and fully automatic. These are usually referred to as model-checking techniques. Some of the more widely known tools are SMV [Burch et al., 1992] and Murphi [Burch and Dill, 1994]. These require that the state-space of the machine be finite. Our example specification has a finite state space. However, if the values of chosen and measured altitude had not been abstracted away, the state-space would have been infinite.

3.4 Some Additional Observations

The preceding discussion illustrates the process that one goes through in translating an English specification into a formal one. Although the example system was contrived to demonstrate this feature, the process demonstrated is typical for realistic systems, and the English specification for the example is actually more complete than most because the example system is small and simple.

The formal specification process forces one to clearly elaborate the behavior of a system in detail. Whereas the English specification must be examined in multiple places and interpreted to make a judgment about the desired system's behavior, the formal specification completely defines the behavior. Thus, the requirements capture process includes making choices about how to interpret the informal specification. Traditional software development practices force the developer to make these interpretation choices (consciously or unconsciously) during the process of creating the design or implementation. Many of the choices are hidden implicitly in the implementation without being carefully thought out or verified by the system designers, and the interpretations and clarifications are seldom faithfully recorded in the requirements document. On the other hand, the formal methods process exposes these ambiguities early in the design process and forces early and clear decisions, which are fully documented in the formal specification.

A more detailed version of this paper:

R. W. Butler, An Introduction to Requirements Capture Using PVS: Specification of a Simple Autopilot, NASA TM-110255, May 1996, p. 33.

is available at `http://techreports.larc.nasa.gov/ltrs/ltrs.html.` Recent work has looked at using formal methods to detect and eliminate mode confusion in flight guidance systems [Miller and Potts, 1999; Butler et al., 1998].

Defining Terms

Invariant: A property of a system that always remains true throughout all operational modes.

Mode: A particular operational condition of a system. The mode control panel controls switching between operational conditions of the flight-control system.

Predicate: A function that returns true or false.

State: A particular operational condition of a system. A common method of representing the operational functioning of a system is by enumerating all of the possible system states and transitions between them. This is referred to as a state-transition diagram or finite-state machine representation.

*The Tablewise tool was previously named Tbell.

Typechecking: Verification of consistency of data types in a specification. The detailed use of data types to differentiate between various kinds of objects, when supported by automated typechecking, can make a specification more readable, maintainable, and reliable.

References

Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., and Hwang, L.J., 1992. Symbolic Model Checking: 10^{20} States and Beyond, *Inf. Comput.* 98(2):142–170.

Burch, J.R. and Dill, David L., 1994. Automatic Verification of Pipelined Microprocessor Control, *Computer-Aided Verification, CAV'94,* Stanford, CA, pp. 68–80, June.

Butler, R.W. and Finelli, G.B. 1993. The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software, *IEEE Trans. Software Eng,* 19(1):3–12.

Butler, R. W., Miller, S. P., Potts, J. N., and Carreno, V. A, 1998. A Formal Methods Approach to the Analysis of Mode Confusion, *17th Digital Avionics Syst. Conf.,* Bellevue, WA, October 31–November 6.

Davis, A.M., 1988. A Comparison of Techniques for the Specification of External System Behavior., *CACM*, 31(9):1098–1115.

FAA, 1988. System Design and Analysis, Advisory Circular AC 25.1309-1A, U.S. Department of Transportation, Washington, D.C., June.

Hoover, D.N. and Chen, Z., 1994. Tbell: A Mathematical Tool for Analyzing Decision Tables, NASA Contractor Rep. 195027, November.

Knight, J.C. and Leveson, N.G., 1991. An Experimental Comparison of Software Fault-Tolerance and Fault Elimination, *IEEE Trans. Software Eng.,* SE-12(1):96–109.

Miller, S.P. and Potts, J. N., 1999. Detecting Mode Confusion Through Formal Modeling and Analysis, NASA/CR-1999-208971, January.

Owre, S., Shankar, N., and Rushby, J.M., 1993. The PVS Specification Language (Beta Release), Computer Science Laboratory, SRI International, Menlo Park, CA, 1993.

Further Information

A good introduction to the fundamentals of mathematical logic is presented in *Mathematical Logic* by Joseph R. Schoenfield.

The application of formal methods to digital avionics systems is discussed in detail in *Formal Methods and Digital Systems Validation for Airborne Systems,* NASA Contractor Report 4551, by John Rushby. Rushby's report was written for certifiers of avionics systems and gives useful insights into how formal methods can be effectively applied to the development of ultra-reliable avionics systems.

Two NASA Guidebooks on formal methods: "Formal Methods Specification and Verification Guidebook for Software and Computer Systems, Volume I: Planning and Technology Insertion" [NASA/TP-98-208193], 1998, and "Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems, Volume II: A Practitioner's Companion" [NASA-GB-001-97], 1997 are available on the Web at `http://eis.jpl.nasa.gov/quality/Formal_Methods/`.

The complete specification and proofs for the mode control panel example described in this chapter can be obtained at `http://shemesh.larc.nasa.gov/fm/ftp/larc/mode-control-ex/`. Several other formal methods application examples and papers can also be obtained from that site.

4 Electronic Hardware Reliability

Nikhil Vichare
CALCE, University of Maryland

Ping Zhao
Medtronic

Diganta Das
CALCE, University of Maryland

Michael G. Pecht
CALCE, University of Maryland

4.1 Introduction

Reliability is the ability of a product to perform as intended (i.e., without failure and within specified performance limits) for a specified time in its life cycle application environment. To achieve product reliability over the complete life cycle demands an approach that consists of a set of tasks, each requiring total engineering and management commitment and enforcement. These tasks impact electronic hardware reliability through the selection of materials, structural geometries and design tolerances, manufacturing processes and tolerances, assembly techniques, shipping and handling methods, operational conditions, and maintenance and maintainability guidelines [1]. The tasks are as follows:

1. *Define realistic product requirements and constraints for certain useful lifetimes based on target markets.* This includes defining the functionality, physical attributes, performance, life cycle environment, useful life (with warranties), life cycle cost, testing and qualification methods, sched-

ules, and end-of-life requirements for the product. The manufacturer and the customer must mutually define the product requirements in light of both the customer's needs and the manufacturer's capability to meet those needs.

2. *Define the product life cycle environment* by specifying all expected phases for a product from manufacture to end of life, including assembly, storage, handling, shipping, and operating and nonoperating conditions of the product. Identify significant life cycle loads (individual and combination) for each phase and quantify the loads (typical range and variability) a product is expected to experience.
3. *Select the parts required for the product* by using a well-defined assessment procedure to ensure that reliability issues are addressed before and during design. The process enables maximizing the profit and minimizing the time to profit from the product, provides product differentiation, effectively uses the global supply chain, and ensures effective assessment, mitigation, and management of life cycle risks in using the part.
4. *Use a systematic methodology to identify all potential failure modes, mechanisms, and sites by which the product can be expected to fail.* Develop an understanding of the relationships between product requirements and the physical characteristics of the product (and their variation in the production process), the interactions of product materials with loads (stresses at application conditions), and their influence on product failure susceptibility with respect to the use conditions. This involves finding the failure mechanisms and the reliability models to quantitatively evaluate failure susceptibility. Failure mechanisms need to be prioritized to determine the environmental and operational loads that need to be accounted for in the design or to be controlled during operation.
5. *Design the product based on knowledge of the expected life cycle conditions* by using the selected materials and processes. Set appropriate specification limits, considering various stress limits and margins, derating, redundancy, and protective architecture.
6. *Qualify the product design, manufacturing, and assembly processes.* Qualification tests should be conducted to verify the reliability of the product in the expected life cycle conditions. Qualification tests should provide an understanding of the influence of process variations on product reliability. The goal of this step is to provide a physics-of-failure basis for design decisions, with an assessment of all possible failure mechanisms for the product.
7. *Identify, measure, and optimize the key processes in manufacturing and assembly required to make the part.* Monitor and control the manufacturing and assembly processes addressed in the design to reduce defects. Develop screens and tests to assess statistical process control of manufacturing and assembly steps.
8. *Manage the life cycle usage of the product* using closed loop management procedures. This includes inspection, testing, and maintenance procedures.

4.2 Product Requirements and Constraints

A product's requirements and constraints are defined in terms of customer demands and the company's core competencies, culture, and goals. There are various reasons to justify the creation of a product, such as to fill a perceived market need, to open new markets, to remain competitive in a key market, to maintain market share and customer confidence, to fill a need of specific strategic customers, to demonstrate experience with a new technology or methodology, to improve maintainability of an existing product, and to reduce the life cycle costs of an existing product.

From the perspective of requirements and constraints, products can be classified into three types: multicustomer products, single-customer products, and custom products with variability of the targeted customer population. To make reliable products, there should be a joint effort between the supplier and the customer. According to IEEE 1332 [2], there are three reliability objectives to achieve in the stage of defining products' requirements and constraints:

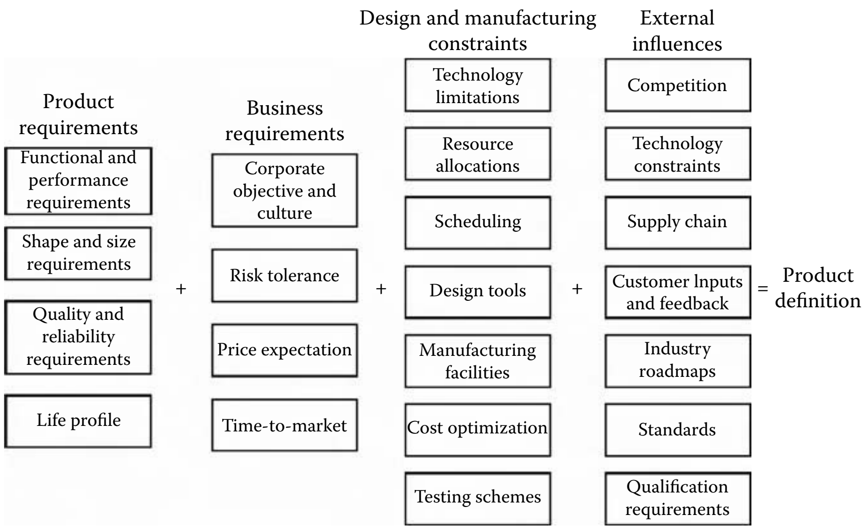


FIGURE 4.1 Constraints in the product definition process.

- The supplier, working with the customer, shall determine and understand the customer's requirements and product needs, so that a comprehensive design specification can be generated.
- The supplier shall structure and follow a series of engineering activities so that the resulting product satisfies the customer's requirements and product needs with regard to product reliability.
- The supplier shall include activities that assure the customer that the reliability requirements and product needs have been satisfied.

If the product is for direct sale to end users, marketing usually takes the lead in defining the product's requirements and constraints through interaction with the customer's marketplace, examination of product sales figures, and analysis of the competition. Alternatively, if the product is a subsystem that fits within a larger product, the requirements and constraints are determined by the customer's product into which the subsystem fits. The product definition process involves multiple influences and considerations. Figure 4.1 shows a diagram of constraints in the product definition process.

Two prevalent risks in requirements and constraints definition are the inclusion of irrelevant requirements and the omission of relevant requirements. The inclusion of irrelevant requirements can involve unnecessary design and testing time as well as money. Irrelevant or erroneous requirements result from two sources: requirements created by persons who do not understand the constraints and opportunities implicit in the product definition, and inclusion of requirements for historical reasons. The omission of critical requirements may cause the product not to be functional or may significantly reduce the effectiveness and the expected market size of the product.

The initial requirements are formulated into a requirements document, where they are prioritized. The requirements document should be approved by engineers, management, customers, and other involved parties. Usually, the specific people involved in the approval will vary with the organization and the product. For example, for human-safety-critical products, legal representatives should attend the approval to identify legal considerations with respect to the implementation of the parts selection and management team's directives. The results of capturing product requirements and constraints allow the design team to choose parts and develop products that conform to company objectives.

Once a set of requirements has been completed, the product engineering function creates a response to the requirements in the form of a specification, which states the requirements that must be met; the schedule for meeting the requirements; the identification of those who will perform the work; and the identification of potential risks. Differences in the requirements document and the preliminary specification become the topic of trade-off analyses.

Once product requirements are defined and the design process begins, there should be constant comparison between the product's requirements and the actual product design. As the product's design becomes increasingly detailed, it becomes increasingly more important to track the product's characteristics (size, weight, performance, functionality, reliability, and cost) in relation to the original product requirements. The rationale for making changes should be documented. The completeness with which requirements tracking is performed can significantly reduce future product redesign costs. Planned redesigns or design refreshes through technology monitoring and use of roadmaps ensure that a company is able to market new products or redesigned versions of old products in a timely, effective manner to retain their customer base and ensure continued profits.

4.3 The Product Life Cycle Environment

The life cycle environment of the product plays a significant role in determining product requirements and making reliability assessments. It influences decisions regarding product design and development, parts selection and management, qualification, product safety, warranty, and support.

The phases in a product's life cycle include manufacturing and assembly, testing, rework, storage, transportation and handling, operation (modes of operation, on-off cycles), repair and maintenance, and disposal. During each phase of its life cycle, the product experiences various environmental and usage loads. The life cycle loads can be thermal (steady-state temperature, temperature ranges, temperature cycles, temperature gradients), mechanical (pressure levels, pressure gradients, vibrations, shock loads, acoustic levels), chemical (aggressive or inert environments, humidity levels, contamination), physical (radiation, electromagnetic interference, altitude), environmental (ozone, pollution, fuel spills), or operational loading conditions (stresses caused by power, power surge, heat dissipation, current, voltage spikes). These loads, either individually or in various combinations, may influence the reliability of the product. The extent and rate of product degradation depend upon the nature, magnitude, and duration of exposure to such loads.

Defining and characterizing the life cycle loads is often the most uncertain input in the overall reliability planning process. The challenge is further exacerbated because products designed for the same environmental conditions can experience completely different application conditions, including the application length, the number of applications, the product utilization or nonutilization profile, and maintenance or servicing conditions. For example, typically all desktop computers are designed for office environments. However, the operational profile of each unit may be completely different depending on user behavior. Some users may shut down the computer every time after it is used; others may shut down only once at the end of the day; still others may keep their computers powered on all the time. Thus, the temperature profile experienced by each product, and hence its degradation due to thermal loads, would be different. Four methods used to estimate the life cycle loads on a product are discussed in the following subsections.

4.3.1 Market Studies and Standards

Market surveys and standards generated independently by agencies* provide a coarse estimate of the actual environmental loads expected in the field. The environmental profiles available from these sources are typically classified per industry type, such as military, consumer, telecommunications, automotive, and commercial avionics. Often the data includes worst case and example use conditions. The data available are derived most often from a similar kind of environment to provide an estimate of the actual environmental loads that the targeted equipment will experience. Care should be taken while using this data as an absolute estimate of environmental loads. The use of these data sources can become inevitable due to time and cost constraints during the design phase.

*For example, IPC SM-785 specifies the use and extreme temperature conditions for electronic products categorized under different industry sectors such as telecommunication, commercial, and military.

4.3.2 Field Trial, Service, and Failure Records

Field trial records, which estimate the environmental loads encountered by previous or prototype equipment, are also sometimes used to get estimates on environmental profiles. The data available are for shorter durations and are extrapolated to get an estimate of actual environmental conditions. Service records and failure records usually document the causes of unscheduled maintenance and the nature of failure, possibly due to certain environmental or usage conditions. These data give an idea of the critical conditions but should not be used as a basis for developing the entire life cycle profile. They should be used only to accommodate the extreme or special conditions the equipment might encounter.

4.3.3 Similarity Analysis

Similarity analysis is a technique for estimating environmental loads when sufficient field histories for similar products are available. The characteristic differences in design and application for the two products need to be reviewed before using data on existing products for proposed designs. Changes and discrepancies in the conditions of the two products should be critically analyzed to ensure the applicability of available loading data for the new product. For example, electronics inside a washing machine in a commercial laundry are expected to experience a wider distribution of loads and use conditions (due to several users) and higher usage rates compared with a home washing machine. These differences should be considered during similarity analysis.

4.3.4 *In Situ* Monitoring

Environmental and usage loads experienced by the product in its life cycle can be monitored *in situ*. This data is often collected using sensors, either mounted externally or integrated with the product and supported by telemetry systems. Devices such as health and usage monitoring systems (HUMS) are popular in aircraft and helicopters for *in situ* monitoring of usage and environmental loads. Load distributions should be developed from data obtained by monitoring products used by different customers, ideally from various geographical locations where the product is used. The data should be collected over a sufficient period to provide an accurate estimate of the loads and their variation over time. Designers must ensure that the data is not biased, even if the users are aware of the monitoring process. *In situ* monitoring has the potential to provide the most accurate account of load history for use in health (condition) assessment and design of future products.

4.4 Parts Selection and Management

A parts selection and management methodology helps a company to make risk-informed decisions when purchasing and using electronic parts. The parts selection and management process is usually not carried out by a single individual, but by a multidisciplinary team. The parts management team, as a whole, is responsible for the following [3]:

- Assigning parts selection and management responsibilities to groups within the company
- Establishing communication channels within and outside the company
- Managing information flow within the team and to departments outside the team
- Identifying process and assessment criteria and acceptability levels
- Applying the parts selection and management methodology to the candidate part
- Identifying potential supplier intervention procedures, authorizing such action when required, and ensuring the associated effectiveness
- Monitoring periodically and making improvements to the methodology continuously

The methodology in Figure 4.2 describes this approach to parts selection and management. The overall purpose is to help organizations to maximize profits, provide product differentiation, effectively utilize the global supply chain, and assess, mitigate, and manage the life cycle risks in selecting and using parts.

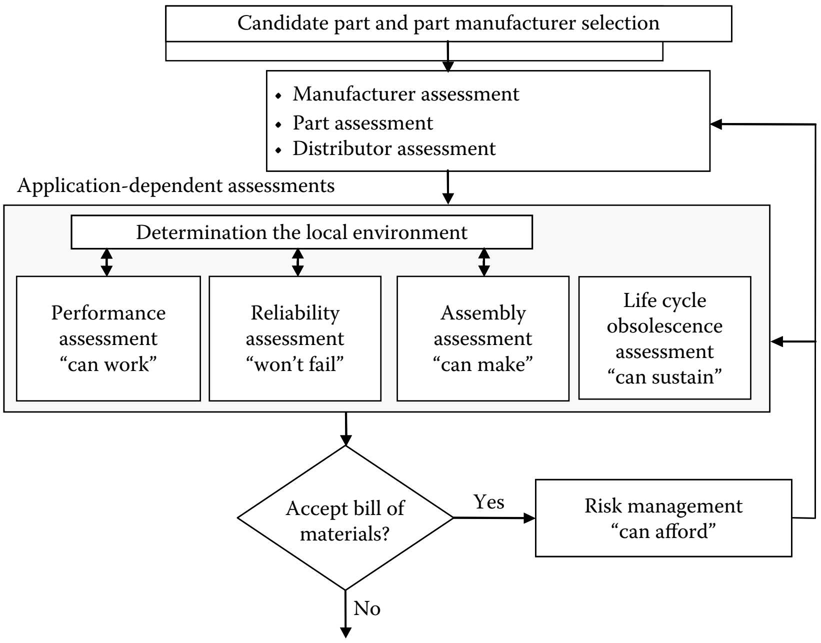

Candidate part and part manufacturer selection
• Manufacturer assessment
• Part assessment
• Distributor assessment
Application-dependent assessments
Determination the local environment
Performance assessment "can work"
Reliability assessment "won't fail"
Assembly assessment "can make"
Life cycle obsolescence assessment "can sustain"
Accept bill of materials?
Yes
No
Risk management "can afford"

FIGURE 4.2 Parts selection and management methodology.

Several of these assessment methods directly impact product reliability, and those steps are described next. Several other steps, such as assembly assessment and life cycle obsolescence assessment, are also performed in this process. Although those steps impact overall product performance and profitability, their impact on reliability is not direct.

4.4.1 Manufacturer, Part, and Distributor Assessment

The manufacturer assessment step evaluates the part manufacturer's ability to produce parts with consistent quality, and the part assessment step gauges the candidate part's quality and consistency. The distributor assessment evaluates the distributor's ability to provide parts without affecting the initial quality and reliability and to provide certain specific services, such as part problem and change notifications. If the part satisfies the minimum acceptability criteria set by the equipment manufacturer for all three categories, the candidate part then moves to "application-dependent assessments," as shown in Figure 4.2. Appropriate implementation of this step helps improve reliability by reducing the risk of maverick part lots into a product build.

4.4.2 Performance Assessment

The goal of performance assessment is to evaluate the part's ability to meet the performance requirements (e.g., functional, mechanical, and electrical) of the product. In general, there are no distinct boundaries for parameters such as mechanical load, voltage, current, temperature, and power dissipation above which immediate failure will occur and below which a part will operate indefinitely [4]. However, there is often a minimum and a maximum limit beyond which the part will not function properly, and in general, those bounds are defined by the recommended operating conditions for the part. It is the responsibility of the parts selection and management team to establish that the electrical, mechanical, or functional performance of the part is suitable for the life cycle conditions of the particular product. If a product must be operated outside the manufacturer-specified operating conditions, then uprating [5] may have to be considered.

4.4.3 Reliability Assessment

Reliability assessment results provide information about the ability of a part to meet the required performance specifications in its life cycle application environment for a specified period of time. If the parametric and functional requirements of the system cannot be met within the required local environment, then the local environment may have to be modified, or a different part may have to be used. If part reliability is not ensured through the reliability assessment process, an alternate part or product redesign should be considered.

Reliability assessment is conducted through the use of reliability test data (conducted by the part manufacturer), virtual reliability assessment results, or accelerated test results. If the magnitude and duration of the life cycle conditions are less severe than those of the reliability tests, and if the test sample size and results are acceptable, the part reliability is acceptable. Otherwise, virtual reliability assessment should be considered. Virtual reliability is a simulation-based methodology used to identify the dominant failure mechanisms associated with the part under life cycle conditions to determine the acceleration factor for a given set of accelerated test parameters, and to determine the time-to-failures corresponding to the identified failure mechanisms. If virtual reliability proves insufficient to validate part reliability, accelerated testing should be performed on representative part lots at conditions that represent the application condition to determine the part reliability. This decision process is illustrated in the diagram in Figure 4.3.

4.4.4 Risk Management

After a part is accepted, resources must be applied to managing its life cycle, including supply chain management, obsolescence assessment, manufacturing and assembly feedback, manufacturer warranties management, and field failure and root-cause analysis. Including time, data, opportunity, and money, determine whether risks should be managed or not. The risks associated with including a part in the product fall into two categories:

- *Managed risks* — risks that the product development team chooses to proactively manage by creating a management plan and performing a prescribed monitoring regime of the part's field performance, manufacturer, and manufacturability
- *Unmanaged risks* — risks that the product development team chooses not to proactively manage

If risk management is considered necessary, a plan should be prepared that contains details about how the part is monitored (data collection) and how the results of the monitoring feed back into various

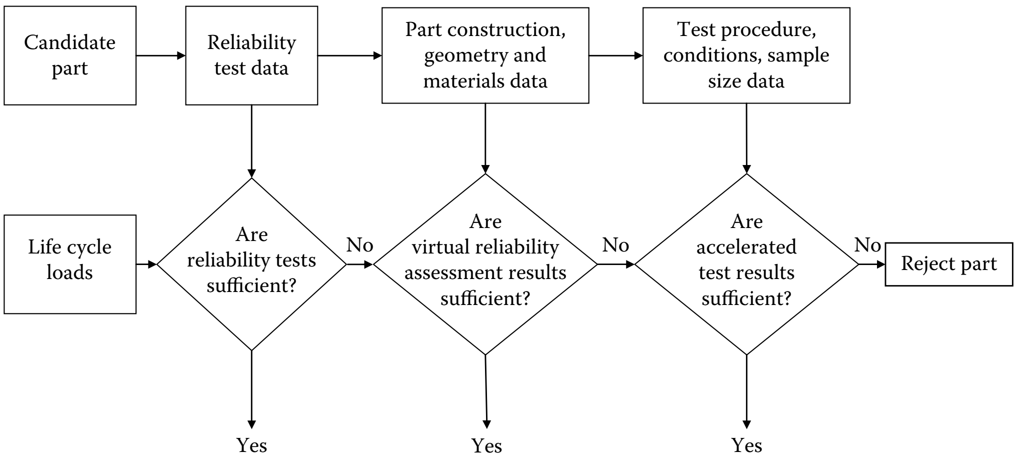

Candidate part
Reliability test data
Part construction, geometry and materials data
Test procedure, conditions, sample size data
Life cycle loads
Are reliability tests sufficient?
No
Are virtual reliability assessment results sufficient?
No
Are accelerated test results sufficient?
No
Reject part
Yes
Yes
Yes

FIGURE 4.3 Decision process for part reliability assessment.

parts selection and management processes. The feasibility, effort, and cost involved in management processes must be considered prior to the final decision to select the part. Feedback regarding the part's assembly performance, field performance, and sales history is essential to ascertain the risks.

4.5 Failure Modes, Mechanisms, and Effects Analysis

Electronic hardware is typically a combination of board, components, and interconnects, all with various failure mechanisms by which they can fail in the life cycle environment. A potential failure mode is the manner in which a failure can occur — that is, the ways in which the reviewed item fails to perform its intended design function or performs the function but fails to meet its objectives. Failure modes are closely related to the functional and performance requirements of the product. Failure mechanisms are the processes by which a specific combination of physical, electrical, chemical, and mechanical stresses induce failures. Investigation of the possible failure modes and mechanisms of the product aids in developing failure-free and reliable designs.

Catastrophic failures due to a single occurrence of a stress event when the intrinsic strength of the material is exceeded are termed overstress failures. Failure mechanisms due to monotonic accumulation of incremental damage beyond the endurance of the material are called wearout mechanisms [6]. When the damage exceeds the endurance limit of the component, failure will occur. Unanticipated large stress events can either cause an overstress (catastrophic) failure or shorten life by causing the accumulation of wearout damage. Examples of such stresses are accidental abuse and acts of God. On the other hand, in well-designed and high-quality hardware, stresses should cause only uniform accumulation of wearout damage; the threshold of damage required to cause eventual failure should not occur within the usage life of the product.

Electrical performance failures can be caused by individual components with improper electrical parameters, such as resistance, impedance, capacitance, or dielectric properties, or by inadequate shielding from electromagnetic interference (EMI) or particle radiation. Failure modes can manifest as reversible drifts in transient and steady-state responses, such as delay time, rise time, attenuation, signal-to-noise ratio, and crosstalk. Electrical failures common in electonic hardware include overstress mechanisms due to electrical overstress (EOS) and electrostatic discharge (ESD); examples of such failures in semiconductor components include dielectric breakdown, junction breakdown, hot electron injection, surface and bulk trapping, surface breakdown, and wearout mechanisms such as electromigration and stress-driven diffusive voiding.

Thermal performance failures can arise due to incorrect design of thermal paths in an electronic assembly. This includes incorrect conductivity and surface emissivity of individual components, as well as ill-designed convective and conductive paths for heat transfer. Thermal overstress failures are a result of heating a component beyond critical temperatures such as the glass-transition temperature, melting point, fictile point, or flash point. Some examples of thermal wearout failures are aging due to depolymerization, intermetallic growth, and interdiffusion. Failures due to inadequate thermal design may be manifested as components running too hot or too cold and causing operational parameters to drift beyond specifications, although the degradation is often reversible upon cooling. Such failures can be caused either by direct thermal loads or by electrical resistive loads, which in turn generate excessive localized thermal stresses. Adequate design checks require proper analysis for thermal stress and should include conductive, convective, and radiative heat paths.

Mechanical performance failures include those that may compromise the product performance without necessarily causing any irreversible material damage, such as abnormal elastic deformation in response to mechanical static loads, abnormal transient response (such as natural frequency or damping) to dynamic loads, and abnormal time-dependent reversible (anelastic) response, as well as failures that cause material damage, such as buckling, brittle or ductile fracture, interfacial separation, fatigue crack initiation and propagation, creep, and creep rupture. To take one example, excessive elastic deformations in slender structures in electronic packages can sometimes constitute functional failure due to overstress loads, such as excessive flexing of interconnection wires, package lids, or flex circuits in electronic devices, causing shorting or excessive crosstalk. However, when the load is removed, the deformations (and consequent functional abnormalities) disappear completely without any permanent damage.

Radiation failures — principally caused by uranium and thorium contaminants and secondary cosmic rays — can cause wearout, aging, embrittlement of materials, and overstress soft errors in electronic hardware, such as logic chips. Chemical failures occur in adverse chemical environments that result in corrosion, oxidation, or ionic surface dendritic growth. There may also be interactions between different types of stresses. For example, metal migration may be accelerated in the presence of chemical contaminants and composition gradients, and thermal loads can accelerate a failure mechanism due to a thermal expansion mismatch.

The design team must be aware of the possible failure mechanisms if they are to design hardware capable of withstanding loads without failing. Failure mechanisms and their related physical models are also important for planning tests and screens to audit nominal design and manufacturing specifications, as well as the level of defects introduced by excessive variability in manufacturing and material parameters.

Failure modes, mechanisms, and effects analysis (FMMEA) is a novel and systematic methodology to identify potential failure mechanisms and models for all potential failures modes and to prioritize failure mechanisms. FMMEA enhances the value of traditional methods such as FMEA (failure modes and effects analysis) and FMECA (failure modes effects and criticality analysis) by identifying high-priority failure mechanisms to create an action plan to mitigate their effects. The knowledge about the cause and consequences of mechanisms found through FMMEA helps in several efficient and cost-effective ways.

FMMEA is based on understanding the relationships between product requirements and the physical characteristics of the product (and their variation in the production process), the interactions of product materials with loads (stresses at application conditions), and their influence on product failure susceptibility with respect to the use conditions. This involves finding the failure mechanisms and the reliability models to quantitatively evaluate failure susceptibility. The FMMEA methodology is shown in Figure 4.4.

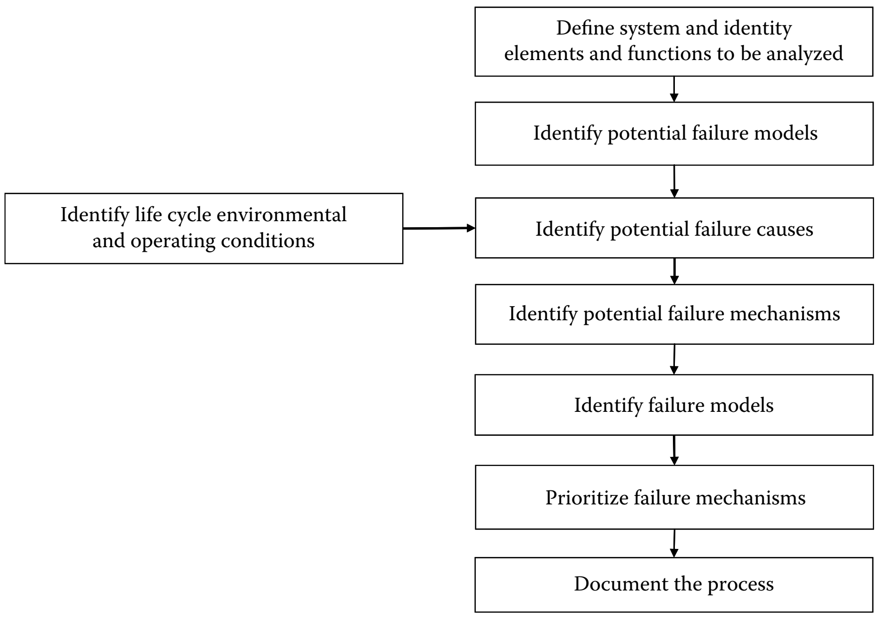

Define system and identity elements and functions to be analyzed
Identify potential failure models
Identify life cycle environmental and operating conditions
Identify potential failure causes
Identify potential failure mechanisms
Identify failure models
Prioritize failure mechanisms
Document the process

FIGURE 4.4 FMMEA methodology.

4.5.1 System Definition, Elements, and Functions

The FMMEA process begins by defining the system to be analyzed. A system is a composite of subsystems or levels that are integrated to achieve a specific objective. The system is divided into various subsystems and continues to the lowest possible level — the component or element. In a printed circuit board system, a location breakdown would include the package, plated through-hole (PTH), metallization, and the board itself.

4.5.2 Potential Failure Modes

For the elements that have been identified, all possible failure modes for each given element are listed. For example, in a solder joint the potential failure modes are open or intermittent change in resistance, which can hamper its functioning as an interconnect. In cases where information on possible failure modes is not available, potential failure modes may be identified using numerical stress analysis, accelerated tests to failure (e.g., HALT*), past experience, and engineering judgment [7]. A potential failure mode at one level may be the cause of a potential failure mode in a higher level system or subsystem or be the effect of one in a lower level component.

4.5.3 Potential Failure Causes

A failure cause is defined as the circumstances during design, manufacture, or use that lead to a failure mode [7]. For each failure mode, all possible ways a failure can result are listed. Failure causes are identified by finding the basic reason that may lead to a failure during design, manufacturing, storage, transportation or use. Knowledge of potential failure causes can help identify the failure mechanisms driving the failure modes for a given element. For example, in an automotive under-hood environment, the solder joint failure modes such as open and intermittent change in resistance can potentially be caused due to temperature cycling, random vibration, and shock impact.

4.5.4 Potential Failure Mechanisms

Failure mechanisms frequently occurring in electronics were discussed earlier in this section. Studies on electronic material failure mechanisms and the application of physics-based damage models to the design of reliable electronic products comprising all relevant wearout and overstress failures in electronics are available in literature [8][9].

4.5.5 Failure Models

Failure models use appropriate stress and damage analysis methods to evaluate susceptibility to failure. Failure susceptibility is evaluated by assessing the time-to-failure or likelihood of a failure for a given geometry, material construction, and environmental and operational condition set. Based on JEDEC standards, JEP122B (Failure Mechanisms and Models for Semiconductor Device) and JEP 148 (Reliability Qualification of Semiconductor Devices Based on Physics of Failure Risk and Opportunity Assessment), Table 4.1 provides a list of common failure mechanisms in electronics and associated models. Failure models may be limited by the availability and accuracy of models for quantifying the time to failure of the system. They may also be limited by the ability to combine the results of multiple failure models for a single failure site and the ability to combine results of the same model for multiple stress conditions [7]. If no failure models are available, the appropriate parameter(s) to monitor can be selected based on an empirical model developed from prior field failure data or models derived from accelerated testing.

*Highly accelerated life testing.

TABLE 4.1 Failure Mechanisms, Relevant Loads, and Models for Electronics

Failure Mechanism	Failure Sites	Relevant Loads	Sample Model
Fatigue	Die attach, wirebond/TAB, solder leads, bond pads, traces, vias/PTHs, interfaces	Δ T, T_{mean}, dT/dt, dwell time, Δ H, Δ V	Nonlinear power; law (Coffin-Manson)
Corrosion	Metallizations	M, Δ V, T	Eyring (Howard)
Electromigration	Metallizations	T, J	Eyring (Black)
Conductive Filament Formation	Between metallizations	M, ΛV	Power law (Rudra)
Stress-Driven Diffusion Voiding	Metal traces	s, T	Eyring (Okabayashi)
Time-Dependent Dielectric Breakdown	Dielectric layers	V, T	Arrhenius (Fowler-Nordheim)

Δ: Cyclic range V: Voltage; T: Temperature; s: Stress
Λ: Gradient M: Moisture; J: Current density; H: Humidity

4.5.6 Failure Mechanism Prioritization

Environmental and operating conditions are used for initial prioritization of all potential failure mechanisms. If the stress levels generated by certain operational and environmental conditions are nonexistent or negligible, the failure mechanisms that are exclusively dependent on those environmental and operating conditions are assigned a “low” risk level and are eliminated from further consideration. For all other failure mechanisms, the failure susceptibility is evaluated by conducting a stress analysis to determine if failure is precipitated under the given environmental and operating conditions. To provide a qualitative measure of the failure effect, each failure mechanism is assigned a severity rating. The failure effect is assessed first at the level being analyzed, then the next higher level, the subsystem level, and so on to the system level. The final step involves prioritizing the failure mechanisms into three risk levels: high, medium, and low.

4.5.7 Documentation

The FMMEA process facilitates data organization, distribution, and analysis for all the steps. In addition, FMMEA documentation also includes the actions considered and taken based on the FMMEA. For products already developed and manufactured, root-cause analysis may be conducted for the failures that occur during development, testing, and use, and corrective actions can be taken to avoid or to reduce the impacts of those failures. The history and lessons learned, contained within the documentation, provide a framework for future product FMMEA. It is also possible to maintain and update an FMMEA after the corrective actions to generate a new list of high-priority failure mechanisms.

4.6 Design Techniques

Once the parts, materials, processes, and stress conditions are identified, the objective is to design a product using parts and materials that have been sufficiently characterized in terms of performance over time when subjected to the manufacturing and application profile conditions. A reliable and cost-effective product can be designed only through a methodical design approach using physics-of-failure analysis, testing, and root-cause analysis.

Design guidelines based on physics-of-failure models can also be used to develop tests, screens, and derating factors. Tests based on physics-of-failure models can be designed to measure specific quantities, to detect the presence of unexpected flaws, and to detect manufacturing or maintenance problems. Screens can be designed to precipitate failures in the weak population while not cutting into the design life of the normal population. Derating or safety factors can be determined to lower the stresses for the dominant failure mechanisms.

4.6.1 Protective Architectures

In designs where safety is an issue, it is generally desirable to incorporate some means for preventing a part, structure, or interconnection from failing or from causing further damage when it fails. Fuses and circuit breakers are examples of elements used in electronic products to sense excessive current drain and to disconnect power from the concerned part. Fuses within circuits safeguard parts against voltage transients or excessive power dissipation and protect power supplies from shorted parts. As another example, thermostats can be used to sense critical-temperature limiting conditions and to shut down the product or a part of the system until the temperature returns to normal. In some products, self-checking circuitry can also be incorporated to sense abnormal conditions and to make adjustments to restore normal conditions or to activate switching means to compensate for the malfunction [4].

In some instances, it may be desirable to permit partial operation of the product after a part failure rather than permitting total product failure. By the same reasoning, degraded performance of a product after failure of a part is often preferable to complete stoppage. An example is the shutting down of a failed circuit whose function is to provide precise trimming adjustment within a deadband of another control product; acceptable performance may thus be achieved, perhaps under emergency conditions, with the deadband control product alone [4].

Sometimes, the physical removal of a part from a product can harm or cause failure in another part by removing load, drive, bias, or control. In such cases, the first part should be equipped with some form of interlock mechanism to shut down or otherwise protect the second part. The ultimate design, in addition to its ability to function after a failure, should be capable of sensing and adjusting for parametric drifts to avert failures.

In the use of protective techniques, the basic procedure is to take some form of action after an initial failure or malfunction to prevent additional or secondary failures. By reducing the number of failures, techniques such as enhancing product reliability can be considered, although they also affect availability and product effectiveness. Equally important considerations are the impacts of maintenance, repair, and part replacement. For example, if a fuse protecting a circuit is replaced, the following questions need to be answered: What is the impact when the product is reenergized? What protective architectures are appropriate for postrepair operations? What maintenance guidance must be documented and followed when fail-safe protective architectures have or have not been included?

4.6.2 Stress Margins

A properly designed product should be capable of operating satisfactorily with parts that drift or change with time and with changes in operating conditions such as temperature, humidity, pressure, and altitude as long as the parameters remain within their rated tolerances. Figure 4.5 provides a schematic of stress levels and margins for a product. The specification or tolerance value is given by the manufacturer to

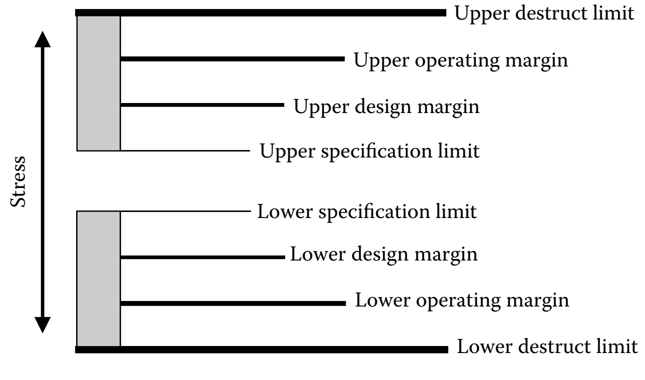


FIGURE 4.5 Stress limits and margins.

limit the conditions of customer use. The design margin is the value a product is designed to survive, and the operating margin is the expected value for a recoverable failure of a distribution of a product. The expected value for permanent (overstress) failure of a distribution of a product is called the destruct margin.

To guard against out-of-tolerance failures, the design team must consider the combined effects of tolerances on parts to be used in manufacturing, of subsequent changes due to the range of expected environmental conditions, of drifts due to aging over the period specified in the reliability requirement, and of tolerances in parts used in future repair or maintenance functions. Parts and structures should be designed to operate satisfactorily at the extremes of the parameter ranges, and allowable ranges must be included in the procurement or reprocurement specifications.

Statistical analysis and worst-case analysis are methods of dealing with part and structural parameter variations. In statistical analysis, a functional relationship is established between the output characteristics of the structure and the parameters of one or more of its parts. In worst-case analysis, the effect that a part has on product output is evaluated on the basis of end-of-life performance values or out-of-specification replacement parts.

To ensure that the parts used in a system remain within the stated margins, derating can be used. Derating is the practice of limiting thermal, electrical, and mechanical stresses to levels below the manufacturer's specified ratings to improve reliability. Derating allows added protection from system anomalies unforeseen by the designer (e.g., transient loads, electrical surge). For example, manufacturers of electronic hardware often specify limits for supply voltage, output current, power dissipation, junction temperature, and frequency. The equipment design team may decide to select an alternative component or make a design change that ensures that the operational condition for a particular parameter, such as temperature, is always below the rated level. These lower stresses are expected to extend useful operating life where the failure mechanisms under consideration are of wearout type. This practice is also expected to provide a safer operating condition by furnishing a margin of safety when the failure mechanisms are of the overstress type.

The term "derating" suggests a two-step process; first a "rated" stress value is determined from a part manufacturer's data-book, and then some reduced value is assigned. The margin of safety supposed to be provided by derating is the difference between the maximum allowable actual applied stress and the demonstrated limits of the part capabilities. The part capabilities as given by manufacturer specifications are taken as the demonstrated limits. The propensity of the system design team often inclines toward using conservative stresses at the expense of overall productivity. There are reasons to believe that the part manufacturers already provide safety margin when choosing the operating limits. When these values are derated by the users, effectively a second level of safety margin is added.

In order to be effective, derating criteria must target the right stress parameters to address modeling of the relevant failure mechanisms. Once the failure models for the critical failure mechanisms have been identified using the FMMEA process, the impact of derating on the effective reliability of the part for a given load can be determined. Instead of derating the stress rating values provided by the device manufacturers, the goal should be to determine the safe operating envelope for each part and subsystem and then operate within that envelope.

4.6.3 Redundancy

Redundancy exists when one or more of the parts of a system can fail and the system can still function with the parts that remain operational. Two common types of redundancy are active and standby. In active redundancy, all the parts are energized and operational during the operation of a system. In active redundancy, the parts will consume life at the same rate as the individual components.

In standby redundancy, some parts are not contributing to the operations of the system and are switched on only when there are failures in the active parts. The parts in standby ideally should last longer than the parts in active redundancies. There are three conceptual types of standby redundancy: cold, warm, and hot. In cold standby, the secondary part is shut down until needed. This lowers the amount of time that the part is active and does not consume any useful life; however, the transient stresses on the part during switching may be high. This transient stress can cause faster consumption of life during switching. In warm standby, the secondary part is usually active but is idling or unloaded. In hot standby, the secondary part forms an active parallel system. The life of the hot standby part is consumed at the same rate as active parts.

A design team often finds that redundancy is (1) the quickest way to improve product reliability if there is insufficient time to explore alternatives, or if the part is already designed; (2) the cheapest solution, if the cost of redundancy is economical in comparison with the cost of redesign; and/or (3) the only solution, if the reliability requirement is beyond the state of the art. However, it is often difficult to realize the benefits of redundancy (both active and standby) due to other reasons. Usually, these reasons include common mode failures, load sharing, and switching and standby failures.

Common mode failures are caused by phenomena that create dependencies between two or more redundant parts, which cause them to fail simultaneously. Common mode failures can be caused by many factors — for example, common electric connections, shared environmental stresses, and common maintenance problems. In system reliability analysis, common mode failures have the same effect as putting in an additional part in series with the parallel redundant configuration.

Load-sharing failures occur when the failure of one part increases the stress level of the other parts. This increased stress level can affect the life of the active parts. For redundant engines, motors, pumps, structures, and many other systems and devices in active parallel setup, the failure of one part may increase the load on the other parts and decrease their times to failure (or increase their hazard rates).

Several common assumptions are generally made regarding the switching and sensing of standby systems. We assume that switching is in one direction only, that switching devices respond only when directed to switch by the monitor, and that switching devices do not fail if not energized. Regarding standby, the general assumption is that standby nonoperating units cannot fail if not energized. When any of these idealizations are not met, switching and standby failures occur. Monitor or sensing failure includes both dynamic (failure to switch when active path fails) and static (switching when not required) failures.

Besides these limitations, the design team may find that the following disadvantages outweigh the benefits of redundancy implementation:

- Costly redundant sensors and switching devices make it too expensive.
- Limitations on size and weight are exceeded.
- It exceeds power limitations, particularly in active redundancy.
- It requires sensing and switching circuitry so complex as to offset the reliability advantage of redundancy.

4.6.4 Integrated Diagnostics and Prognostics

Design guidelines and techniques should involve strategies to assess the reliability of the product in its life cycle environment. A product's health is the extent of deviation or degradation from its expected normal (in terms of physical and performance) operating condition [10]. Knowledge of a product's health can be used for the detection and isolation of faults or failures (diagnostics) and for prediction of impending failure based on current conditions (prognostics). Thus, by determining the advent of failure based on actual life cycle conditions, procedures can be developed to mitigate, manage, and maintain the product.

Diagnostics and prognostics can be integrated into a product by (1) installing built-in fuses and canary structures that will fail faster than the actual product when subjected to life cycle conditions [11]; (2) sensing parameters that are precursors to failure, such as defects or performance degradation [12], (3) sensing the life cycle environmental and operational loads that influence the system's health, and (4) processing the measured data to estimate the life consumed [13] [14]. The life cycle data measured by such integrated monitors can be extremely useful in future product design and end-of-life decisions [15].

An example of integrated prognostics in electronic products is the self-monitoring analysis and reporting technology (SMART) currently employed in select computing equipment for hard disk drives (HDDs) [16]. HDD operating parameters, including flying height of the head, error counts, variations in spin time, temperature, and data transfer rates, are monitored to provide advance warning of failures. This is achieved through an interface between the computer's start-up program (BIOS) and the hard disk drive.

4.7 Qualification and Accelerated Testing

Qualification includes all activities that ensure that the nominal design and manufacturing specifications will meet or exceed the desired reliability targets. Qualification validates the ability of the nominal design and manufacturing specifications of the product to meet the customer's expectations and assesses the probability of survival of the product over its complete life cycle. The purpose of qualification is to define the acceptable range of variability for all critical product parameters affected by design and manufacturing, such as geometric dimensions, material properties, and operating environmental limits. Product attributes that are outside the acceptable ranges are termed defects, since they have the potential to compromise product reliability [17].

Qualification tests should be performed only during initial product development and immediately after any design or manufacturing changes in an existing product. A well-designed qualification procedure provides economic savings and quick turnaround during development of new products or products subject to manufacturing and process changes.

Investigating failure mechanisms and assessing the reliability of products where longevity is required may be a challenge, since a very long test period under actual operating conditions is necessary to obtain sufficient data to determine actual failure characteristics. The results from FMMEA should be used to guide this process. One approach to the problem of obtaining meaningful qualification data for high-reliability devices in shorter time periods is using methods such as virtual qualification and accelerated testing to achieve test-time compression.

4.7.1 Virtual Qualification

Virtual qualification is a process that requires significantly less time and money than accelerated testing to qualify a part for its life cycle environment. This simulation-based methodology is used to identify and rank the dominant failure mechanisms associated with the part under life cycle loads, to determine the acceleration factor for a given set of accelerated test parameters, and to determine the time-to-failure corresponding to the identified failure mechanisms.

Each failure model is comprised of a stress analysis model and a damage assessment model. The output is a ranking of different failure mechanisms based on the time to failure. The stress model captures the product architecture, while the damage model depends on a material's response to the applied stress. This process is therefore applicable to existing as well as new products. Virtual qualification can be used to optimize the product design in such a way that the minimum time to failure of any part of the product is greater than its desired life. Although the data obtained from virtual qualification cannot fully replace those obtained from physical tests, they can increase the efficiency of physical tests by indicating the potential failure modes and mechanisms that can be expected.

Ideally, a virtual qualification process will involve identification of quality suppliers, quality parts, physics-of-failure qualification, and a risk assessment and mitigation program. The process allows qualification to be readily incorporated into the design phase of product development, since it allows design, manufacturing, and testing to be conducted promptly and cost-effectively. It also allows consumers to qualify off-the-shelf components for use in specific environments without extensive physical tests. Since virtual qualification reduces emphasis on examining a physical sample, it is imperative that the manufacturing technology and quality assurance capability of the manufacturer be taken into account. If the data on which the virtual qualification is performed are inaccurate or unreliable, all results are suspect. In addition, if a reduced quantity of physical tests is performed in the interest of simply verifying virtual results, the operator needs to be confident that the group of parts selected is sufficient to represent the product. Further, it should be remembered that the accuracy of the results using virtual qualification depends on the accuracy of the inputs to the process — that is, the accuracy of the life cycle loads, the choice of the failure models used, the choice of the analysis domain (for example, 2D, pseudo-3D, full 3D), the constants in the failure model, the material properties, and so on. Hence, to obtain a reliable prediction, the variability in the inputs should be specified using distribution functions, and the validity of the failure models should be tested by conducting accelerated tests.

4.7.2 Accelerated Testing

Accelerated testing is based on the concept that a product will exhibit the same failure mechanism and mode in a short time under high-stress conditions as it would exhibit in a longer time under actual life cycle stress conditions. The purpose is to decrease the total time and cost required to obtain reliability information about the product under study. Usually it should be possible to quantitatively extrapolate from the accelerated environment to the usage environment with some reasonable degree of assurance.

Accelerated tests can be roughly divided into two categories: qualitative tests and quantitative tests. Qualitative tests, in the form of overstressing the products to obtain failure, are perhaps the oldest form of reliability testing. Those tests typically arise as a result of a single load condition, such as shock, temperature extremes, and electrical overstress. Qualitative tests usually yield failure mode information but do not reveal failure mechanism and time-to-failure information. Quantitative tests target wearout failure mechanisms, in which failures occur as a result of cumulative load conditions and time to failure is the major outcome of quantitative accelerated tests.

The easiest and most common form of accelerated life testing is continuous-use acceleration. For example, a washing machine is used for 10 hours per week in average. If it is operated without stop, the acceleration factor would be 16.8. However, this method is not applicable for high-usage products. Under such circumstances, accelerated testing is conducted to measure the performance of the test product at loads or stresses that are more severe than would normally be encountered in order to enhance the damage accumulation rate within a reduced time period. The goal of such testing is to accelerate time-dependent failure mechanisms and the damage accumulation rate to reduce the time to failure. Based on the data from those accelerated tests, life in normal use conditions can be extrapolated.

Accelerated testing begins by identifying all the possible overstress and wearout failure mechanisms. The load parameter that directly causes the time-dependent failure is selected as the acceleration parameter and is commonly called the accelerated load. Common accelerated loads include the following:

- Thermal loads, such as temperature, temperature cycling, and rates of temperature change
- Chemical loads, such as humidity, corrosives, acid, and salt
- Electrical loads, such as voltage or power
- Mechanical loads, such as vibration, mechanical load cycles, strain cycles, and shock and impulses

The accelerated environment may include a combination of these loads. Interpretation of results for combined loads requires a quantitative understanding of their relative interactions and the contribution of each load to the overall damage.

Failure due to a particular mechanism can be induced by several acceleration parameters. For example, corrosion can be accelerated by both temperature and humidity, and creep can be accelerated by both mechanical stress and temperature. Furthermore, a single accelerated stress can induce failure by several wearout mechanisms simultaneously. For example, temperature can accelerate wearout damage accumulation not only by electromigration, but also by corrosion, creep, and so on. Failure mechanisms that dominate under usual operating conditions may lose their dominance as the stress is elevated. Conversely, failure mechanisms that are dormant under normal-use conditions may contribute to device failure under accelerated conditions. Thus, accelerated tests require careful planning if they are to represent the actual usage environments and operating conditions without introducing extraneous failure mechanisms or nonrepresentative physical or material behavior. The degree of stress acceleration is usually controlled by an acceleration factor, defined as the ratio of the life of the product under normal use conditions to that under the accelerated condition. The acceleration factor should be tailored to the hardware in question and can be estimated from an acceleration transform (that is, a functional relationship between the accelerated stress and the life cycle stress) in terms of all the hardware parameters.

Once the failure mechanisms are identified, it is necessary to select the appropriate acceleration load; determine the test procedures and the stress levels; determine the test method, such as constant stress acceleration or step-stress acceleration; perform the tests; and interpret the test data, which includes extrapolating the accelerated test results to normal operating conditions. The test results provide failure information for improving the hardware through design or process changes.

4.8 Manufacturing Issues

Manufacturing and assembly processes can significantly impact the quality and reliability of hardware. Improper assembly and manufacturing techniques can introduce defects, flaws, and residual stresses that act as potential failure sites or stress raisers later in the life of the product. The effect of manufacturing variability on time to failure is depicted in Figure 4.6. A shift in the mean or an increase in the standard deviation of key geometric parameters during manufacturing can result in early failure due to a decrease in strength of the product. If these defects and stresses can be identified, the design analyst can proactively account for them during the design and development phase.

Auditing the merits of the manufacturing process involves two crucial steps. First, qualification procedures are required, as in design qualification, to ensure that manufacturing specifications do not compromise the long-term reliability of the hardware. Second, lot-to-lot screening is required to ensure that the variability of all manufacturing-related parameters is within specified tolerances [17][18]. In other words, screening ensures the quality of the product by identifying latent defects before they reach the field.

4.8.1 Process Qualification

Like design qualification, process qualification should be conducted at the prototype development phase. The intent at this step is to ensure that the nominal manufacturing specifications and tolerances produce acceptable reliability in the product. The process needs requalification when process parameters, materials, manufacturing specifications, or human factors change.

Process qualification tests can be the same set of accelerated wearout tests used in design qualification. As in design qualification, overstress tests may be used to qualify a product for anticipated field overstress loads. Overstress tests may also be used to ensure that manufacturing processes do not degrade the intrinsic material strength of the hardware beyond a specified limit. However, such tests should supplement, not replace, the accelerated wearout test program, unless explicit physics-based correlations are available between overstress test results and wearout field-failure data.

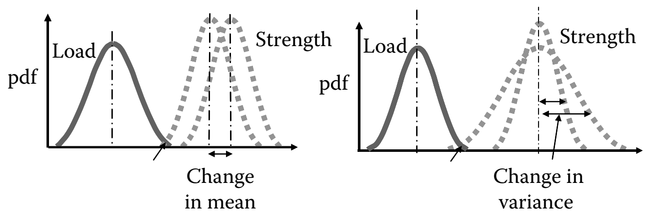


FIGURE 4.6 Influence of quality on failure probability.

4.8.2 Manufacturability

The control and rectification of manufacturing defects has typically been the concern of production and process-control engineers, not the design team. In the spirit and context of concurrent product development, however, hardware design teams must understand material limits, available processes, and manufacturing process capabilities to select materials and construct architectures that promote producibility and reduce the occurrence of defects, increasing yield and quality. Therefore, no specification is complete without a clear discussion of manufacturing defects and acceptability limits. The reliability engineer must have clear definitions of the threshold for acceptable quality and of what constitutes nonconformance. Nonconformance that compromises hardware performance and reliability is considered a defect, and failure mechanism models provide a convenient vehicle for developing such criteria. It is important for the reliability analyst to understand which deviations from specifications can compromise performance or reliability and which are benign and can be accepted.

A defect is any outcome of a process (manufacturing or assembly) that impairs or has the potential to impair the functionality of the product at any time. The defect may arise during a single process or may be the result of a sequence of processes. The yield of a process is the fraction of products that are acceptable for use in a subsequent manufacturing sequence or product life cycle. The cumulative yield of the process is approximately determined by multiplying the individual yields of each of the individual process steps. The source of defects is not always apparent, because defects resulting from a process can go undetected until the product reaches some downstream point in the process sequence, especially if screening is not employed.

It is often possible to simplify the manufacturing and assembly processes to reduce the probability of workmanship defects. As processes become more sophisticated, however, process monitoring and control are necessary to ensure a defect-free product. The bounds that specify whether the process is within tolerance limits, often referred to as the process window, are defined in terms of the independent variables to be controlled within the process and the effects of the process on the product or the dependent product variables. The goal is to understand the effect of each process variable on each product parameter to formulate control limits for the process — that is, the points on the variable scale where the defect rate begins to have a potential for causing failure. In defining the process window, the upper and lower limits of each process variable beyond which it will produce defects must be determined. Manufacturing processes must be contained in the process window by defect testing, analysis of the causes of defects, and elimination of defects by process control, such as by closed-loop corrective action systems. The establishment of an effective feedback path to report process-related defect data is critical. Once this is done and the process window is determined, the process window itself becomes a feedback system for the process operator.

Several process parameters may interact to produce a different defect than would have resulted from an individual parameter acting independently. This complex case may require that the interaction of various process parameters be evaluated in a matrix of experiments. In some cases, a defect cannot be detected until late in the process sequence. Thus, a defect can cause rejection, rework, or failure of the product after considerable value has been added to it. These cost items due to defects can reduce a return on investment by adding to hidden factory costs. All critical processes require special attention for defect elimination by process control.

4.8.3 Process Verification Testing

Process verification testing, often called screening, involves 100% auditing of all manufactured products to detect or precipitate defects. The aim of this step is to preempt potential quality problems before they reach the field. Thus, screening aids in reducing warranty returns and increases customer goodwill. In principle, screening should not be required for a well-controlled process; however, it is often used as a safety net.

Some products exhibit a multimodal probability density function for failures, as depicted in Figure 4.7, with peaks during the early period of their service life due to the use of faulty materials, poorly

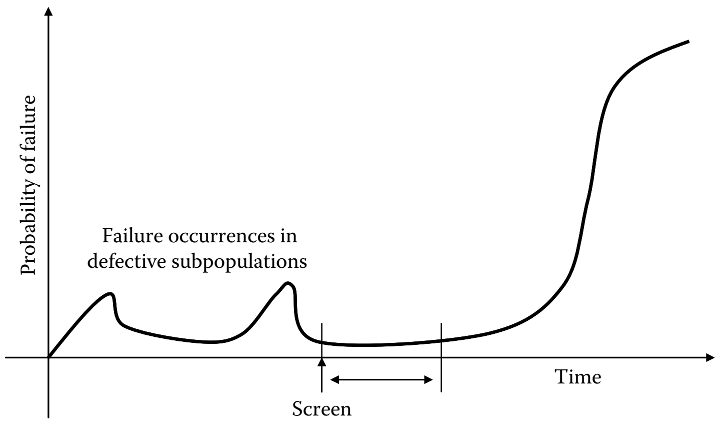


FIGURE 4.7 Candidate for screening due to wearout failure.

controlled manufacturing and assembly technologies, or mishandling. This type of early-life failure is often called infant mortality. Properly applied screening techniques can successfully detect or precipitate these failures, eliminating or reducing their occurrence in field use. Screening should only be considered for use during the early stages of production, if at all, and only when products are expected to exhibit infant mortality field failures. Screening will be ineffective and costly if there is only one main peak in the failure probability density function. Further, failures arising due to unanticipated events such as acts of God (lightning, earthquakes) may be impossible to screen cost-effectively.

Since screening is done on a 100% basis, it is important to develop screens that do not harm good components. The best screens, therefore, are nondestructive evaluation techniques, such as microscopic visual exams, x-rays, acoustic scans, nuclear magnetic resonance (NMR), electronic paramagnetic resonance (EPR), and so on. Stress screening involves the application of stresses, possibly above the rated operational limits. If stress screens are unavoidable, overstress tests are preferred over accelerated wearout tests, since the latter are more likely to consume some useful life of good components. If damage to good components is unavoidable during stress screening, then quantitative estimates of the screening damage, based on failure mechanism models, must be developed to allow the design team to account for this loss of usable life. The appropriate stress levels for screening must be tailored to the specific hardware. As in qualification testing, quantitative models of failure mechanisms can aid in determining screen parameters.

A stress screen need not necessarily simulate the field environment or even use the same failure mechanism as the one likely to be triggered by this defect in field conditions. Instead, a screen should exploit the most convenient and effective failure mechanism to stimulate the defects that can show up in the field as infant mortality. Obviously, this requires an awareness of the possible defects that may occur in the hardware and extensive familiarity with the associated failure mechanisms.

Unlike qualification testing, the effectiveness of screens is maximized when screening is conducted immediately after the operation believed to be responsible for introducing the defect. Qualification testing is conducted preferably on the finished product or as close to the final operation as possible; on the other hand, screening only at the final stage, when all operations have been completed, is less effective, since failure analysis, defect diagnostics, and troubleshooting are difficult and impair corrective actions. Further, if a defect is introduced early in the manufacturing process, subsequent value added through new materials and processes is wasted, which additionally burdens operating costs and reduces productivity.

Admittedly, there are also several disadvantages to such an approach. The cost of screening at every manufacturing station may be prohibitive, especially for small batch jobs. Further, components will experience repeated screening loads as they pass through several manufacturing steps, which increases the risk of accumulating wearout damage in good components due to screening stresses. To arrive at a screening matrix that addresses as many defects and failure mechanisms as feasible with each screen test an optimum situation must be sought through analysis of cost-effectiveness, risk, and the criticality of the defects. All defects must be traced back to the root cause of the variability.

Any commitment to stress screening must include the necessary funding and staff to determine the root cause and appropriate corrective actions for all failed units. The type of stress screening chosen should be derived from the design, manufacturing, and quality teams. Although a stress screen may be necessary during the early stages of production, stress screening carries substantial penalties in capital, operating expense, and cycle time, and its benefits diminish as a product approaches maturity. If almost all of the products fail in a properly designed screen test, the design is probably faulty. If many products fail, a revision of the manufacturing process is required. If the number of failures in a screen is small, the processes are likely to be within tolerances, and the observed faults may be beyond the resources of the design and production process.

4.9 Closed-Loop Monitoring

Product reliability needs to be ensured using a closed-loop process that provides feedback to design and manufacturing in each stage of the product life cycle, including after the product is shipped and is used in its application environment. Data obtained from periodic maintenance, inspection and testing, or health (condition) and usage monitoring methods can be used to perform timely maintenance for sustaining the product and preventing catastrophic failures. Figure 4.8 depicts the closed-loop process for managing the reliability of a product over the complete life cycle.

The objective of closed-loop monitoring is to analyze the failures occurring in testing and field conditions to identify the root cause of failure. The root cause is the most basic casual factor or factors that, if corrected or removed, will prevent recurrence of the situation. The purpose of determining the root cause or causes is to fix the problem at its most basic source so it does not occur again, even in other products, as opposed to merely fixing a failure symptom.

Root-cause analysis is a methodology designed to help (1) describe *what* happened during a particular occurrence; (2) determine *how* it happened; and (3) understand *why* it happened. Only when we are able to determine why an event or failure occurred will we be able to determine corrective measures over time. Root-cause analysis is different than troubleshooting, which is generally employed to eliminate a symptom in a given product, as opposed to finding a solution to the root cause to prevent this and other products from failing.

Correctly identified root causes during design and manufacturing, followed by appropriate actions to fix the design and processes, result in fewer field returns, major cost savings, and customer goodwill. Root-cause analysis of field failures can be more challenging if the conditions during and prior to failure

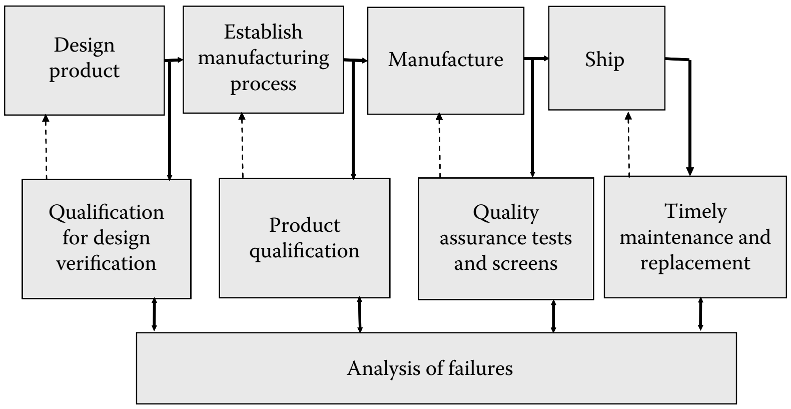

Design product
Establish manufacturing process
Manufacture
Ship
Qualification for design verification
Product qualification
Quality assurance tests and screens
Timely maintenance and replacement
Analysis of failures

FIGURE 4.8 Reliability management using a closed-loop process.

are unknown. The lessons learned from each failure analysis need to be documented, and appropriate actions need to be taken to update the design, manufacturing process, and maintenance actions or schedules.

4.10 Summary

Reliability is not a matter of chance or good fortune; rather, it is a rational consequence of conscious, systematic, rigorous efforts during the life cycle of the product. High product reliability can only be assured through robust product designs, capable processes that are known to be within tolerances, and qualified components and materials from vendors whose processes are also capable and within tolerances. Quantitative understanding and modeling of all relevant failure mechanisms provide a convenient vehicle for formulating effective design, process, and test specifications and tolerances.

The physics-of-failure approach is not only a tool to provide better and more effective designs, but it also helps develop cost-effective approaches for improving the entire approach to building electronic products. Proactive improvements can be implemented for defining more realistic performance requirements and environmental conditions, identifying and characterizing key material properties, developing new product architectures and technologies, developing more realistic and effective accelerated stress tests to audit reliability and quality, enhancing manufacturing-for-reliability through mechanistic process modeling, and characterization allowing proactive process optimization, increasing first-pass yields, and reducing hidden factory costs associated with inspection, rework, and scrap.

When utilized early in the concept stage of a product's development, reliability aids determination of feasibility and risk. In the design stage of product development, reliability analysis involves methods to enhance performance over time through the selection of materials, design of structures, design tolerances, manufacturing processes and tolerances, assembly techniques, shipping and handling methods, and maintenance and maintainability guidelines. Engineering concepts such as strength, fatigue, fracture, creep, tolerances, corrosion, and aging play a role in these design analyses. The use of physics-of-failure concepts coupled with mechanistic and probabilistic techniques are often required to understand the potential problems and trade-offs and to take corrective actions. The use of factors of safety and worst-case studies as part of the analysis is useful in determining stress screening and burn-in procedures, reliability growth, maintenance modifications, field testing procedures, and various logistics requirements.

Defining Terms

Accelerated testing: Tests conducted at stress levels that are more severe than the normal operating levels, in order to enhance the damage accumulation rate within a reduced time period.

Damage: The extent of a product's degradation or deviation from a defect-free state.

Derating: The practice of subjecting parts to lower electrical or mechanical stresses than they can withstand to increase the life expectancy of the part.

Failure mechanism: A process (such as creep, fatigue, or wear) through which a defect nucleates and grows as a function of stresses (such as thermal, mechanical, electromagnetic, or chemical loadings) ultimately resulting in the degradation or failure of a product.

Failure mode: Any physically observable change caused by a failure mechanism.

Integrity: A measure of the appropriateness of the tests conducted by the manufacturer and the part's ability to survive those tests.

Overstress failures: Catastrophic sudden failures due to a single occurrence of a stress event that exceeds the intrinsic strength of a material.

Product performance: The ability of a product to perform as required according to specifications.

Qualification: All activities that ensure that the nominal design and manufacturing specifications will meet or exceed the reliability goals.

Quality: A measure of a part's ability to meet the workmanship criteria of the manufacturer.

Reliability: The ability of a product to perform as intended (i.e., without failure and within specified performance limits) for a specified time, in its life cycle application environment.

Wearout failures: Failures due to accumulation of incremental damage, occurring when the accumulated damage exceeds the material endurance limit.

References

1. Pecht, M., *Integrated Circuit, Hybrid, and Multichip Module Package Design Guidelines — A Focus on Reliability*, John Wiley & Sons, New York, 1994.
2. IEEE Standard1332, IEEE Standard Reliability Program for the Development and Production of Electronic Systems and Equipment, 1998.
3. Jackson, M., Mathur, A., Pecht, M., and R. Kendall, Part manufacturer assessment process, *Qual. Reliability Eng. Int.*, 15:457–468, 1999.
4. Sage, A.P. and Rouse, W.B., *Handbook of Systems Engineering and Management*, John Wiley & Sons, New York, 1999.
5. Humphrey, D., Condra, L., Pendse, N., Das, D., Wilkinson, C., and Pecht, M., An avionics guide to uprating of electronic parts, *IEEE Trans. Components and Packag. Technolog.*, 23:595-599, 2000.
6. Tullmin, M., and Roberge, P.R., Corrosion of metallic materials, *IEEE Trans. Reliability*, 44:271–278, 1995.
7. IEEE 1413.1, Guide for selecting and using reliability predictions based on IEEE 1413, *IEEE Standard*, Feb. 2003.
8. JEP 122B, Failure mechanisms and models for semiconductor devices, *JEDEC Standard*, Aug. 2003.
9. JEP 148, *Reliability Qualification of Semiconductor Devices Based on Physics of Failure Risk and Opportunity Assessment*, Apr. 2004.
10. Vichare, N., Eveloy, V., Rodgers, P., Pecht, M., *In-situ* temperature measurement of a notebook computer - a case study in health and usage monitoring of electronics, *IEEE Trans. Device Materials Reliability*, 4(4): 658–663, 2004.
11. Mishra, S., and Pecht, M., *In-situ* sensors for product reliability monitoring, *Proc. SPIE*, 4755, 10–19, 2002.
12. Pecht, M., Dube, M., Natishan, M., and I. Knowles, An evaluation of built-in test, *IEEE Trans. Aerospace Electronic Systems*, 37(1):266–272.
13. Ramakrishnan, A. and Pecht, M., A life consumption monitoring methodology for electronic systems, *IEEE Trans. Components Packaging Technol.*, 26(3):625–634.
14. Mishra, S., Pecht, M., Smith, T., McNee, I., and Harris, R., Remaining life prediction of electronic products using life consumption monitoring approach, *Proc. Euro. Microelectron. Packag. Interconnection Symp.*, Cracow, Poland, 16–18 June 2002, pp. 136-142.
15. Vichare, N., Rodgers, P., Azarian, M. H., and Pecht, M., Application of health monitoring to product take-back decisions, Joint International Congress and Exhibition — Electronics Goes Green 2004+, Berlin, Germany, 6–8 Sept. 2004.
16. Self-Monitoring Analysis and Reporting Technology (SMART), PC Guide, http://www.pcguide.com/ref/hdd/perf/qual/featuresSMART-c.html, last accessed on 22 August 2004.
17. Pecht, M., Dasgupta, A., Evans, J.W., and Evans, J.Y., *Quality Conformance and Qualification of Microelectronic Packages and Interconnects,* John Wiley & Sons, New York, 1994.
18. Upadhyayula, K. and Dasgupta, A., Guidelines for Physics-of-failure Based Accelerated Stress Testing, *Proc. Annu. Reliability Maintainability Symp.,* New York, 1998, pp. 345–357.

5 Electromagnetic Environment

Richard Hess
Honeywell Aerospace

5.1 Introduction

The advent of digital electronic technology in electrical/electronic systems has enabled unprecedented expansion of aircraft system functionality and evolution of aircraft function automation. As a result, systems incorporating such technology are used more and more to implement aircraft functions, including Level A systems that affect the safe operation of the aircraft; however, such capability is not free. The electromagnetic environment (EME) is a form of energy that is the same type (electrical) used by electrical/electronic equipment to process and transfer information. As such, this environment represents a fundamental threat to the proper operation of systems that depend on such equipment. It is a common mode threat that can defeat fault-tolerant strategies reliant upon redundant electrical/electronic systems.

Electrical/electronic systems, characterized as Level A, provide functions that can affect the safe operation of an aircraft and depend upon information (i.e., guidance, control, etc.) processed by electronic equipment. Thus, the EME threat to such systems may translate to a threat to the airplane itself. The computers associated with modern aircraft guidance and control systems are susceptible to upset from lightning and sources that radiate radio frequencies (RF) at frequencies predominantly between 1 and several GHz and produce aircraft internal field strengths of 5 to 200 V/m or greater. Internal field strengths greater than 200 V/m are usually periodic pulses with pulsewidths less than 10 ms. Internal lightning-induced voltages and currents can range from approximately 50 V and 20 A to over 3000 V and 5000 A.

Electrical/electronic system susceptibility to such an environment has been the suspected cause of nuisance disconnects, hardovers, and upsets. Generally, this form of system upset occurs at significantly lower levels of electromagnetic (EM) field strength than that which could cause component failure, leaves no trace, and is usually nonrepeatable.

5.2 EME Energy Susceptibility

It is clear that the sources of EM threats to the electrical/electronic system, whether digital or analog, are numerous. Although both respond to the same threats, there are factors that can make the threat response to a momentary transient (especially intense transients like those that can be produced by lightning) far more serious in digital processing systems than in analog systems. For example, the information bandwidth, and therefore the upper noise response cutoff frequency, in analog devices is limited to, at most, a few MHz . In digital systems, it is often in excess of 100 MHz and continues to increase. This bandwidth difference, which is at least ten times more severe in digital systems, allows substantially more EM energy of many types (modulated, pulse, etc.) to be coupled into the digital system. Moreover, the bandwidths of analog circuits associated with autopilot and flight management systems are on the order of 50 Hz for servo loops and much less for other control loops (less than 1 Hz for outer loops). Thus, if the disturbance is short relative to significant system time constants, even though an analog circuit device possessing a large gain and a broad bandwidth may be momentarily upset by an electromagnetic transient, the circuit will recover to the proper state. It should be recognized that, to operate at high speeds, proper circuit card layout control and application of high-density devices is a must. When appropriate design tools (signal integrity, etc.) are applied, effective antenna loop areas of circuit card tracks become extremely small, and the interfaces to a circuit card track (transmission line) are matched. Older (1970s–1980s) technology with wirewrap backplanes and processors built with discrete logic devices spread over several circuit cards were orders of magnitude more susceptible. Unlike analog circuits, digital circuits and corresponding computational units, once upset, may not recover to the proper state and may require external intervention to resume normal operation. It should be recognized that (for a variety of reasons) large-gain bandwidth devices are and have been used in the digital computing platforms of aircraft systems. A typical discrete transistor can be upset with 10^{-5} J (2000 V at 100 μA for 50 μs). The energy to upset a "typical" integrated circuit can range from 10^{-9} J (20 V at 1 μA for 50 μs), to less than 10^{-10} J (5 V at less than 0.4 μA for 50 μs). As time goes on and processor semiconductor junction feature sizes get smaller and smaller, this problem becomes worse.

It should be noted that, in addition to upset, lightning-induced transients appearing at equipment interfaces can, because of the energy they possess, produce hard faults (i.e., damage circuit components) in interface circuits of either analog or digital equipment. For instance, mechanical, electromechanical, and electrohydraulic elements associated with conventional (not electronic primary flight controls with associate "smart" actuators) servo loops and control surface movement are either inherently immune or vastly more robust when exposed to EME energy effects than the electronic components in an electrical/electronic system.

Immunity of electronic components to damage is a consideration that occurs as part of the circuit design process. The circuit characteristic (immunity to damage) is influenced by a variety of factors:

1. Circuit impedances (resistance, inductance, capacitance), which may be distributed as well as lumped
2. The impedances around system component interconnecting loops along with the characteristic (surge) impedance of wiring interfacing with circuit components
3. Properties of the materials used in the construction of a component (e.g., thick-film/thin-film resistors)
4. Threat level (open circuit voltage/short circuit current), resulting in a corresponding stress on insulation, integrated circuit leads, PC board trace spacing, etc.
5. Semiconductor device nonlinearities (e.g., forward biased junctions, channel impedance, junction/gate breakdown)

Immunity to upset for analog processors is achieved through circuit design measures, and for digital processors it is achieved through architectural as well as circuit design measures.

5.2.1 Soft Faults

Digital circuit upset, also known in the avionics digital computer and information processing community as a "soft fault," is a condition known to occur even in relatively benign operating environments. Soft faults occur despite the substantial design measures (timing margins, transmission line interconnects, ground and power planes, clock enablers of digital circuits) to achieve a relatively high degree of integrity in digital processor operation.

In a normal operating environment, the occurrence of soft faults within digital processing systems is relatively infrequent and random. Such occasional upset events should be treated as probabilistic in nature and can be the result of the following:

- Coincidence of EME energy with clocked logic clock edges, etc.
- Occasional violation of a device's operational margin (resulting margin from the design, processing, and manufacturing elements of the production cycle)

From this perspective, the projected effect of a substantial increase in the severity of the electromagnetic environment will be an increased probability of a soft fault occurrence. That is, in reality, a soft fault may or may not occur at any particular point in time; but, on the average, soft faults will occur more frequently with the new environmental level.

Once developed, software is "burned into nonvolatile" memory (becomes "firmware"); the result will be a special-purpose real-time digital-electronic-technology data processing machine with the inherent potential for "soft faults." Because it is a hardware characteristic, this potential exists even when a substantial amount of attention is devoted to developing an "error-free" operating system and application programs (software) for the general purpose digital machine (computing platform, digital engine, etc.).

5.2.2 MTBUR/MTBF

In the past, service experience with digital systems installed on aircraft has indicated that the confirmed failure rates equal or exceed predicted values that were significantly better than previous generation analog equipment. However, the unscheduled removal rate remains about the same. In general, the disparity in Mean Time Between Unscheduled Removal (MTBUR) and the Mean Time Between Failure (MTBF) continues to be significant. The impact of this disparity on airline direct operating costs is illustrated in Figure 5.1.

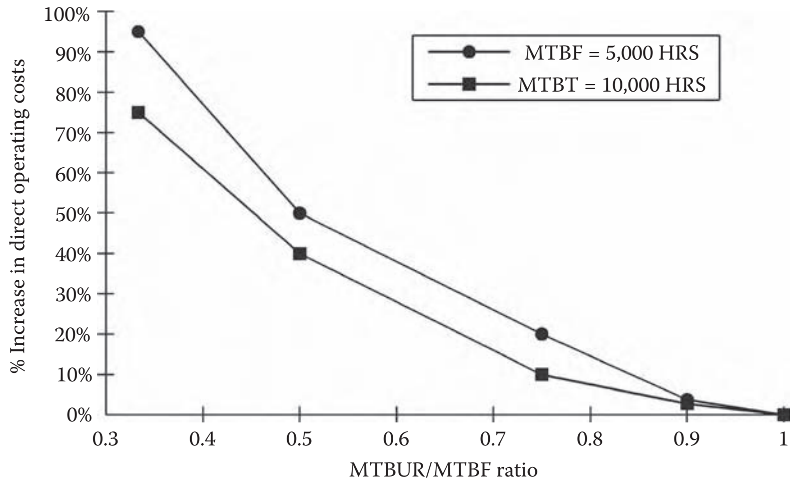


FIGURE 5.1 MTBUR/MTBF ratio impact of operating costs.

To the extent that soft faults contribute to the MTBUR/MTBF disparity, any reduction in soft fault occurrence and propagation could translate into reduction of this disparity.

5.3 Civil Airworthiness Authority Concerns

The following groups have identified the lightning and High-Intensity Radiated Field (HIRF) elements of the EME as a safety issue for aircraft functions provided by electrical/electronic systems:

- Federal Aviation Administration (FAA)
- Joint Aviation Authorities (JAA): Non-European Union
- European Aviation Safety Agency (EASA): European Union

The following factors, identified by the FAA and JAA, have led to this concern about lightning and HIRF effects:

- Increased reliance on electrical and electronic systems to perform functions that may be necessary for the continued safe flight and landing of the aircraft.
- Reduction of the operating power level of electronic devices that may be used in electrical and electronic systems, which may cause circuits to be more reactive to induced lightning and RF voltages and currents leading to malfunction or failure.
- Increased percentage of composite materials in aircraft construction. Because of their decreased conductivity, composite materials may result in less inherent shielding by the aircraft structure.
- Since current flowing in the lightning channel will be forced (during lightning attachment) into and through the aircraft structure without attenuation, decreased conductivity for aircraft structure materials can be particularly troubling for lightning.

The direct effects of lightning (dielectric puncture, blasting, melting, fuel ignition, etc.) have been recognized as flight hazards for decades, and in 1972 the Society of Automotive Engineers (SAE) formed the AE4 Special Task F (which later became AE4L and is now AE2) to address this issue. In the early 1980s, the FAA began developing policy relative to the effects of lightning on electrical/electronic systems (indirect effects), and AE4L supported the FAA and JAA by providing the technical basis for international standards (rules and regulations) and guidance material that, for aircraft type certification, would provide an acceptable means for demonstrating compliance to those rules and regulations. AE4L also supported RTCA Special Committee 135 (SC-135) to integrate lightning environment conditions and test procedures into airborne equipment standards (DO-160) and the European Organization for Civil Aviation Equipment (EUROCAE) standards counterpart (ED-14). In 1987, EUROCAE formed Working Group 31 to be the European counterpart of AE4L.

In 1986, the FAA and JAA identified High-Energy Radio Frequency (HERF) electromagnetic fields as an issue for aircraft electrical/electronic systems. Some time later, the term HERF was changed to its present designation, which is HIRF. Subsequent to the FAA identifying HIRF as a safety issue, SAE and EUROCAE formed Committee AE4R and Working Group 33, respectively, to support the FAA and JAA in much the same way as AE4L and lightning. In addition, unlike the case for lightning, RTCA SC-135 formed a HIRF working group (the corresponding European group was already part of EUROCAE/WG33) to integrate HIRF requirements into DO-160/ED-14.

In the interim between the absence and existence of a rule for lightning and HIRF, special conditions have been or are issued to applicants for aircraft-type certification (TC, STC, Amended Type Certificate [ATC]). What follows is the rationale for the special condition:

> These series of aircraft will have novel or unusual design features associated with the installation of new technology electrical and electronic systems, which perform critical or essential functions. The applicable airworthiness regulation does not contain adequate or appropriate safety standards for the protection of these systems from the effects of lightning and radio frequency energy. This notice contains the additional safety standards that the Administrator considers necessary to ensure

that critical and essential functions the new technology electrical and electronic systems perform are maintained when the airplane is exposed to lightning and RF energy.

Presently, the code of federal regulations associated with the FAA has been updated to include the "indirect effects" of lightning but not HIRF. In the time period between the absence and existence of a rule for HIRF, special conditions for HIRF are being issued to applicants for aircraft certification. However, the FAA has established the Aviation Rule-Making Advisory Committee, which in turn established the Electromagnetic Effects Harmonization Working Group (EEHWG) to develop the rule-making package for HIRF and for amendments to the lightning rules.

Portable electronic devices (PEDs) have not been identified for regulatory action, but in 1992 the FAA requested the RTCA to study the EME produced by PEDs. In response to an FAA request relative to PEDs, RTCA formed Special Committee 177 (SC-177) in 1992. In 1996, SC-177 issued a report titled "Portable Electronic Devices Carried Onboard Aircraft" (DO-233). Currently, control of PEDs and their associated EM emissions are handled by integrating some of the RTCA recommendations into airline policy regarding instructions given to passengers (prohibiting personal cellular phone use, asking passengers to turn off PEDs during taxi, take-off, and landing, etc.). Presently SC-202, established in 2003, is reexamining the PED issue but has not yet published any findings.

5.3.1 EME Compliance Demonstration for Electrical and Electronic Systems

FAA, JAA, and EASA FARs and JARs require compliance demonstration either explicitly or implicitly for the following EME elements:

- Lightning
- HIRF (FAA)
- HIRF (JAA/EASA)
- Electromagnetic Compatibility (EMC)

At the aircraft level, the emphasis should be on lightning and HIRF because most of the energy and system hazards arise from these threats. Their interaction with aircraft systems is global and also the most complex, requiring more effort to understand. Intrasystem electromagnetic emissions fall under the broad discipline of EMC. PEDs are a source of EM emissions that fall outside of the categories of equipment normally included in the EMC discipline. Like lightning and HIRF, the interaction of PED emissions with aircraft electrical/electronic systems is complex and could be global.

The electrical and electronic systems that perform functions considered to be flight critical must be identified by the applicant with the concurrence of the cognizant Federal Aviation Administration Aircraft Certification Office (FAA ACO) by conducting a functional hazard assessment and, if necessary, preliminary system safety assessments (see SAE ARP 4761). The term critical refers to those functions whose failure would contribute to, or cause, a catastrophic failure condition (loss of aircraft). Table 5.1 provides the relationship between function failure effects and development assurance levels associated with those systems that implement functions that can affect safe aircraft operation.

Terms such as "Level A" designate particular system development assurance levels, which refer to the rigor and discipline of processes used during system development (design, implementation, verification

TABLE 5.1 Nomenclature Cross Reference Between AC25.1309 and SAE-ARP 4754

Failure Condition Classification	Development Assurance Level
Catastrophic	Level A
Severe major/hazardous	Level B
Major	Level C
Minor	Level D
No effect	Level E

and certification, production, etc.). It was deemed necessary to focus on the development processes for systems based upon "highly integrated" or "complex" (those whose safety cannot be shown solely by test and whose logic is difficult to comprehend without the aid of analytical tools) elements; that is, primarily digital electronic elements.

Development assurance activities are ingredients of the system development processes. As has been noted, systems and appropriate associated components are assigned "development assurance levels" based on failure condition classifications associated with aircraft-level functions implemented by systems and components. The rigor and discipline needed in performing the supporting processes will vary depending on the assigned development assurance level. There is no development process for aircraft functions. Basically, they should be regarded as intrinsic to the aircraft and are categorized by the role they play for the aircraft (control, navigation, communication, etc.). Relative to safety, they are also categorized (from FAA advisory material) by the effect of their failures, such as catastrophic, severe major/hazardous, major and minor.

EMC has been included in FAA regulations since the introduction of radio and electrical/electronic systems into aircraft. Electrical equipment, controls, and wiring must be installed so that operation of any one unit or system of units will not adversely affect the simultaneous operation of any other electrical unit or system essential to aircraft safe operation. Cables must be grouped, routed, and located so that damage to essential circuits will be minimized if there are faults in heavy current-carrying cables. Critical environmental conditions must be considered in demonstrating compliance with aircraft electrical/electronic system safety requirements with respect to radio and electronic equipment and their installations. Radio and electronic equipment, controls, and wiring must be installed so that operation of any one component or system of components will not adversely affect the simultaneous operation of any other radio or electronic unit or system of units required by aircraft functions.

Relative to safety and electrical/electronic systems, the systems, installations, and equipment whose functioning is required for safe aircraft operation must be designed to ensure that they perform their intended functions under all foreseeable operating conditions. Aircraft systems and associated components that are part of any Level A-C function, considered separately and in relation to other systems, must be designed so that the following are true:

- The occurrence of any failure condition that would prevent the continued safe flight and landing of the airplane is extremely improbable.
- The occurrence of any other failure condition that would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions is improbable.

5.3.2 EME Energy Propagation

As has been noted in the introductory paragraph and illustrated in Figure 5.2, lightning and HIRF are threats to the overall aircraft. Since they are external EME elements, of the two, lightning produces the most intense environment, particularly by direct attachment. Both lightning and HIRF interactions produce internal fields. Lightning can also produce substantial voltage drops across the aircraft structure. Such structural voltages provide another mechanism (in addition to internal fields) for energy to propagate into electrical/electronic systems. Also, the poorer the conductivity of structural materials, the greater the possibility that the following conditions exist:

- Voltage differences across the structure
- Significant lightning diffusion magnetic fields
- Propagation of external environment energy

Figure 5.3 gives the HIRF spectrum and associated aircraft/installations features of interest. Not shown are GPS and Mode 5 frequencies (1-2 GHz).

In general, the propagation of external EME energy into the aircraft interior and electrical/electronic systems is a result of complex interactions of the EME with the aircraft exterior structures, interior structures, and system installations (see Figures 5.3 through 5.7). Figure 5.8 gives representative transfer

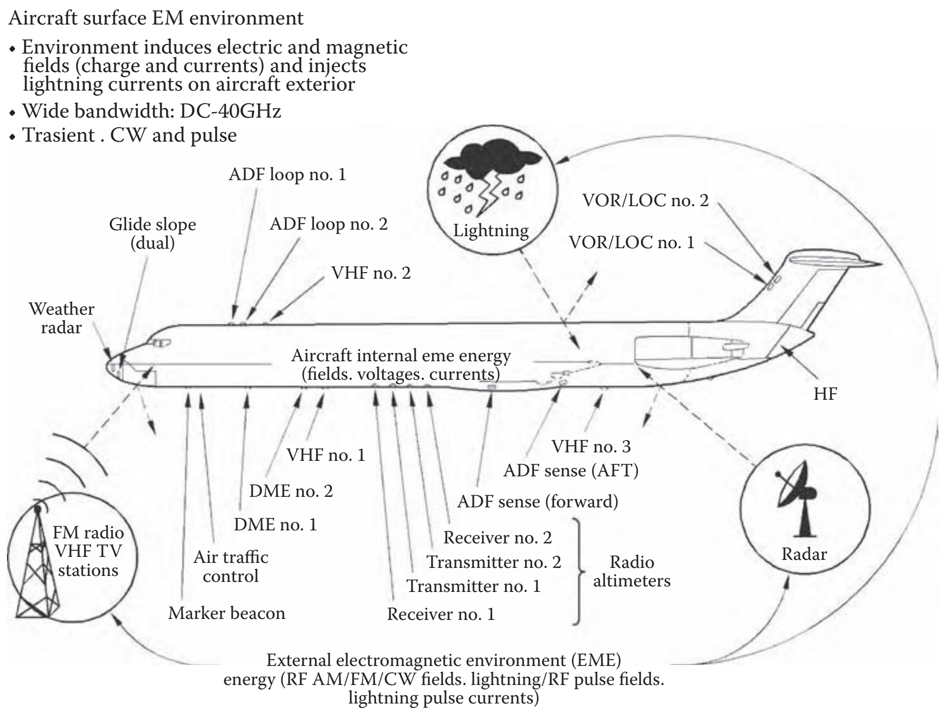

Aircraft surface EM environment
• Environment induces electric and magnetic fields (charge and currents) and injects lightning currents on aircraft exterior
• Wide bandwidth: DC-40GHz
• Trasient . CW and pulse
ADF loop no. 1
Glide slope (dual)
ADF loop no. 2
Lightning
VOR/LOC no. 2
VOR/LOC no. 1
VHF no. 2
Weather radar
Aircraft internal eme energy (fields. voltages. currents)
HF
VHF no. 3
ADF sense (AFT)
VHF no. 1
DME no. 2
ADF sense (forward)
DME no. 1
FM radio VHF TV stations
Receiver no. 2
Transmitter no. 2
Radio altimeters
Radar
Air traffic control
Transmitter no. 1
Marker beacon
Receiver no. 1
External electromagnetic environment (EME) energy (RF AM/FM/CW fields. lightning/RF pulse fields. lightning pulse currents)

FIGURE 5.2 External EME (HIRF, lightning) interaction.

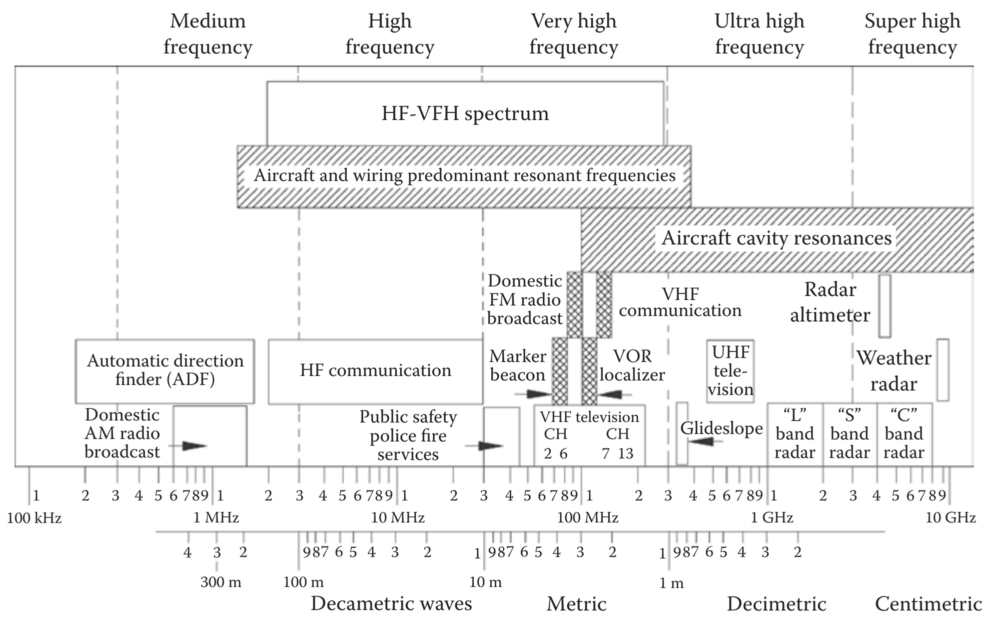


FIGURE 5.3 RF spectrum and associated installation dimensions of interest.

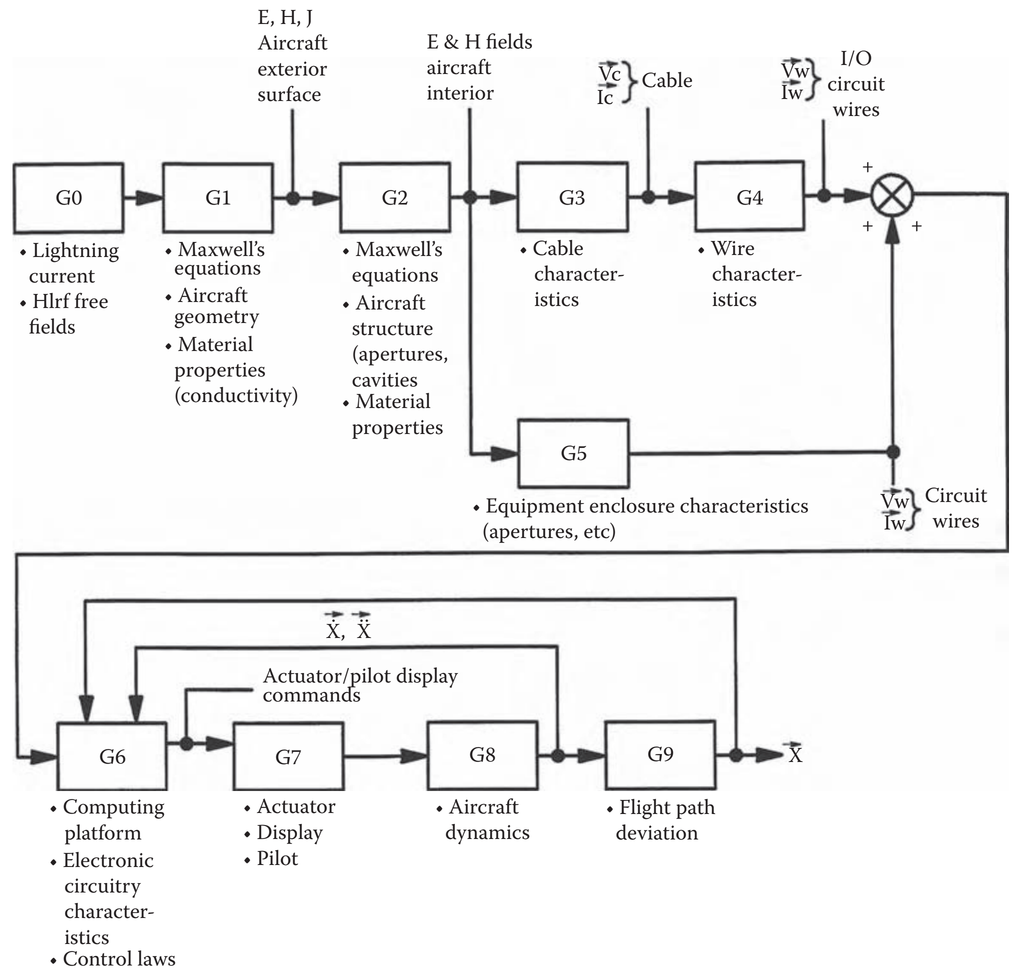
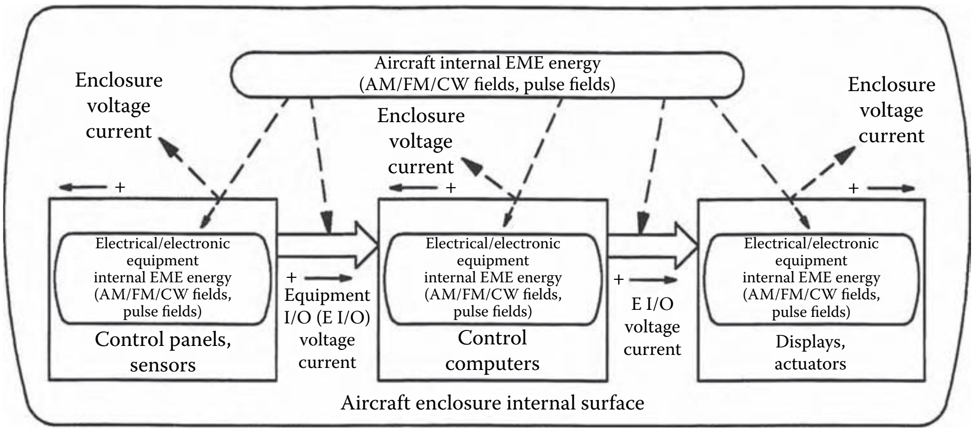


FIGURE 5.4 EME propagation process-transfer function perspective.

functions, in the frequency domain, of energy propagation into electrical/electronic systems, and Figure 5.9 provides time domain responses to a lightning pulse resulting from transfer functions having the low-frequency characteristic Vo(f) = kf[Hi(f)] and a high frequency "moding" (resonant) characteristic (e.g., open loop voltage of cabling excited by a magnetic field; see Figure 5.8).

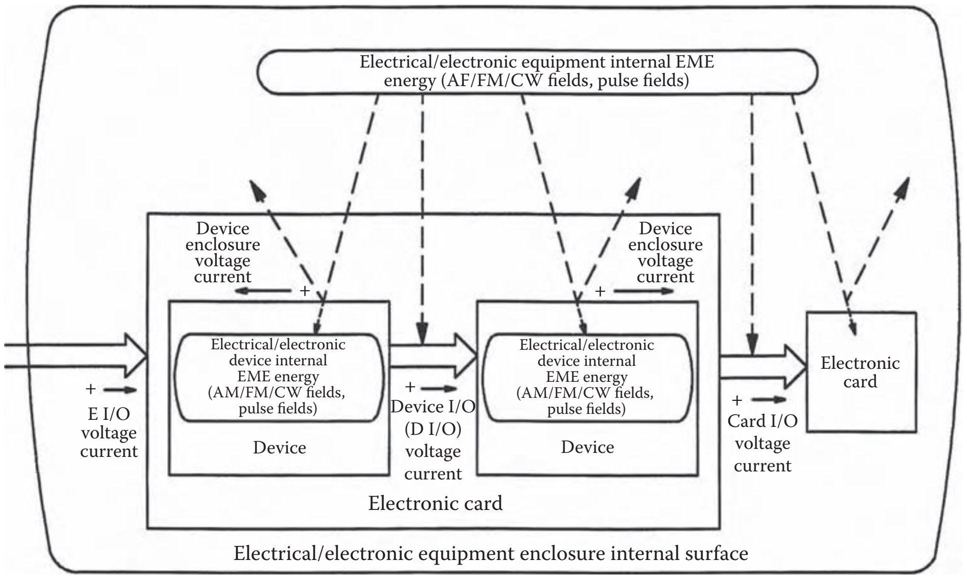
Paths of electromagnetic wave entry from the exterior to the interior equipment regions are sometimes referred to as points of entry. Examples of points of entry may be seams, cable entries, and windows. As noted, points of entry are driven by the local environment, not the incident environment. The internal field levels are dependent on both the details of the point of entry and the internal cavity. Resulting internal fields can vary over a wide range of intensity, wave shape, and wave impedance. Below 10 MHz within a metal aircraft, the magnetic fields due to lightning predominate because of the electric field shielding properties of metal skins. For HIRF "high-frequency" bands in some internal regions, internal field levels may exceed the incident field levels.

The EME local to the equipment or system within the installation (the EME energy coupled to installation wiring, which appears at equipment interface circuits) and the degree of attenuation or enhancement achieved for any region are the product of many factors, such as external EME characteristics, materials, bonding of structure, dimensions and geometric form of the region, and the location and size of any apertures allowing penetration into the aircraft (G0 through G5 of Figure 5.4, which could have any of the characteristics of Figure 5.8).


Aircraft internal EME energy (AM/FM/CW fields, pulse fields)
Enclosure voltage current
Enclosure voltage current
Enclosure voltage current
Electrical/electronic equipment internal EME energy (AM/FM/CW fields, pulse fields)
Control panels, sensors
Equipment I/O (E I/O) voltage current
Control computers
E I/O voltage current
Displays, actuators
Aircraft enclosure internal surface

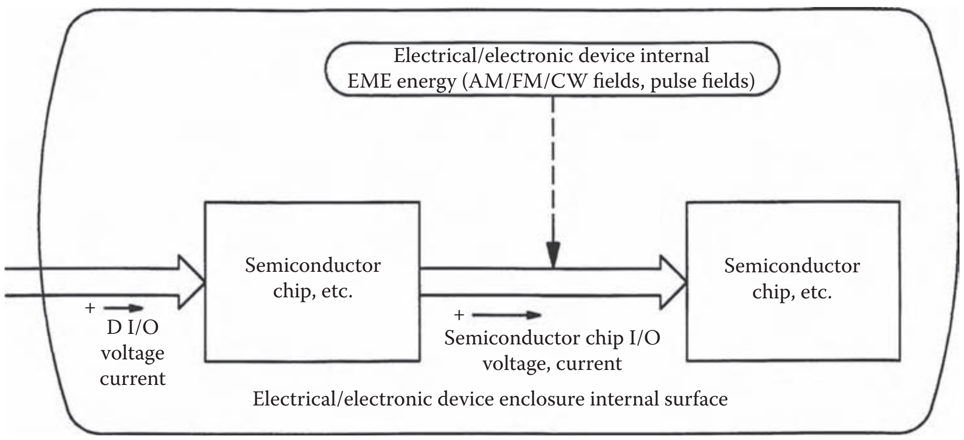
- External energy penetrates to interior via apertures, composites, seams, joints, and antennas
- Voltages and currents induced on flight control system components and cables
  - RF energy below 1 megahertz - induced coupling at these frequencies is inefficient and thus will probably be of lesser concern
  - RF energy between 1 and 300 megahertz is of major concern as aircarft wiring, when their lengths are on the order of a wavelength divided by two (λ/2) or longer at these frequencies, acts as a highly efficient antenna
  - RF energy coupling to aircraft wiring drops off at frequencies above 300 megahertz (at these higher frequencies, the EM energy tends to couple through box apertures rather than through aircraft wiring)

FIGURE 5.5 Aircraft internal EME energy electrical/electronic system.


Electrical/electronic equipment internal EME energy (AF/FM/CW fields, pulse fields)
Device enclosure voltage current
Device enclosure voltage current
Electrical/electronic device internal EME energy (AM/FM/CW fields, pulse fields)
Device
Electrical/electronic device internal EME energy (AM/FM/CW fields, pulse fields)
Device
E I/O voltage current
Device I/O (D I/O) voltage current
Card I/O voltage current
Electronic card
Electronic card
Electrical/electronic equipment enclosure internal surface

- Voltage, fields, currents and charge on system components penetrate into equipment enclosure interiors via holes, seams, and airplane wiring (cables)
- Energy (voltage and current) picked up by wires and printed conductors on cards and carried to electronic devices

FIGURE 5.6 Electrical/electronic equipment internal EME interaction electrical/electronic circuitry.


Electrical/electronic device internal
EME energy (AM/FM/CW fields, pulse fields)
Semiconductor chip, etc.
Semiconductor chip, etc.
+
D I/O voltage current
+
Semiconductor chip I/O voltage, current
Electrical/electronic device enclosure internal surface

- Card and device conductors carry energy to the semiconductor chips, etc.
- Possible effects
  - Damage
  - Upset

FIGURE 5.7 Electrical/electronic device internal EME interaction electrical/electronic circuitry.

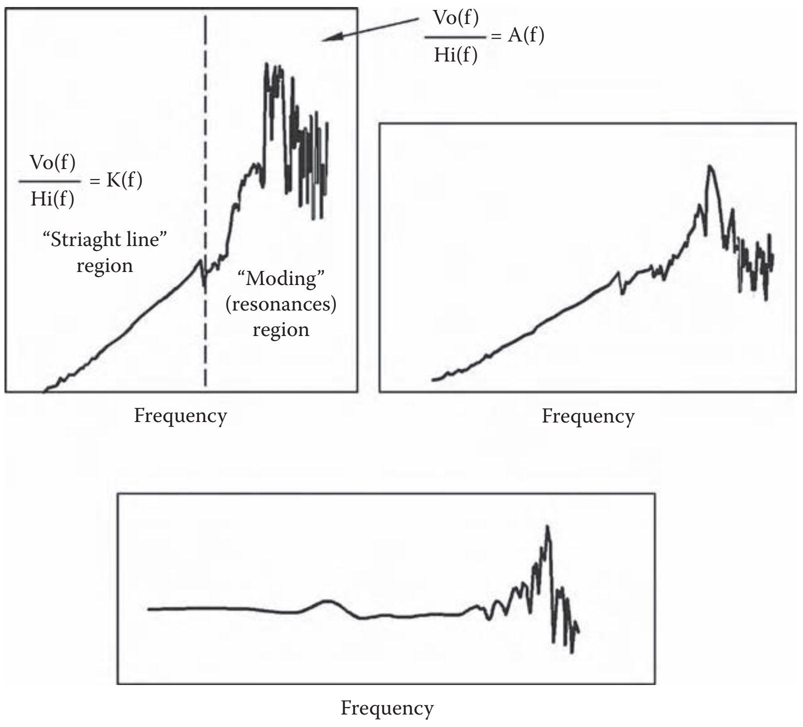


FIGURE 5.8 Frequency domain representation of EME energy attenuation/coupling transfer functions.

In HIRF high-frequency bands (frequencies 100 MHz and higher) the internal field resulting from such influences, as noted, will in most cases produce a nonuniform field within the region or location of the system or equipment. The field cannot be considered as uniform and homogeneous. The field will not necessarily allow the adoption of single-point measurement techniques for the accurate determination of the equivalent internal field to be used as the test level for systems. Several hot spots typically exist

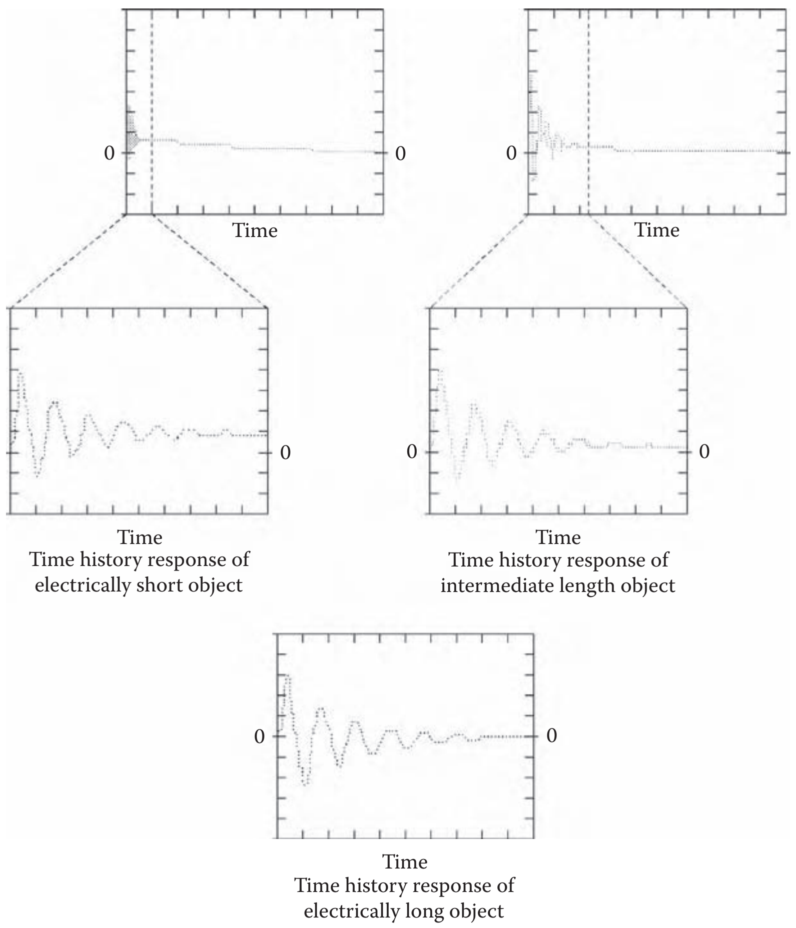


FIGURE 5.9 Responses for lightning EM pulse field interaction with objects of different "electrical lengths."

within any subsection of the aircraft. This is particularly true at cavity-resonant conditions. Intense local effects are experienced at all frequencies in the immediate vicinity of any apertures for a few wavelengths away from the aperture itself. For apertures that are small with respect to wavelength, measurements of the fields within the aperture would yield fields much larger than those further inside the aircraft because the fields fall off inversely proportional to radius cubed. For apertures a wavelength in size or larger, the fields may penetrate unattenuated.

The HIRF spectrum of RF energy that couples into aircraft wiring and electrical/electronic systems can be summarized into three basic ranges:

- *HIRF energy below 1 MHz* — Induced coupling at these frequencies is inefficient and thus will be of lesser concern.
- *HIRF energy between 1 and 400 MHz* — Induced coupling is of major concern since aircraft wiring acts as a highly efficient antenna at these frequencies.
- *HIRF energy above 400 MHz* — Coupling to aircraft wiring drops off at frequencies above 400 MHz. At these higher frequencies, the EM energy tends to couple through equipment apertures and seams and to the quarter wavelength of wire attached to the Line Replaceable Unit (LRU). In this frequency range, aspects of equipment enclosure construction become important.

The extension of electrical/electronic systems throughout the aircraft ranges from highly distributed (e.g., flight controls) to relatively compact. Wiring associated with distributed systems penetrates several aircraft regions. Some of these regions may be more open to the electromagnetic environment than others, and wiring passing through the more open regions is exposed to a higher environment. Thus, at frequencies below 400 MHz, the wiring of a highly distributed system could have a relatively wide range of induced voltages and currents that would appear at equipment interface circuits.

The flight deck of the aircraft is an example of an open region. The windscreen "glass" presents approximately zero attenuation to an incoming field at and above the frequency for which its perimeter is one wavelength. Some enhancement above the incident field level generally exists in and around the aperture at this resonance condition.

Lightning is a transient electromagnetic event, as is the resulting internal environment. Relative to a spectral representation, lightning energy would be concentrated in the zero to 50 MHz range (most energy is below 3 MHz). However, since lightning is such an intense transient, significant energy can be present up to and sometimes above 10 MHz. Relative to the higher frequency range (above 100 MHz), strong resonances of aircraft interior volumes (cavities) such as the flight deck and equipment bay could occur. At very high frequencies, the EME can be both very intense and very short in duration. From a cavity resonance issue, since the time constant of a relatively good cavity resonator is on the order of 1 ms, the pulse can be gone before significant field energy is developed within the cavity.

5.4 Architecture Options for Fault Mitigation

New system architecture measures have been evolving that could complement and augment traditional schemes to provide protection against EME energy effects. Architecture options can be applied at the overall system level or within the digital computing platform for the system. These options include the following:

- Distributed bus architecture
- Error Detection and Corrective (EDC) schemes
- Fiber optic data transfer
- Computation recovery

5.4.1 Electrical/Electronic System

In the past, soft faults in digital avionics were physically corrected by manual intervention, recycle power, and so on. More recently, system-level measures for the automatic correction of soft faults have begun to be developed. It is perceived that significant benefits can be gained through soft fault protection measures designed into the basic system mechanization. System-level soft fault protection methodologies provide the ability to tolerate disruption of either input/output data or internal computation. Accordingly, there are two distinct classes of disruption:

- Disruption at the system equipment interface boundary, causing corruption of data flowing to or from the affected subsystem.
- Disruption that reaches within system equipment to corrupt internal data and computation. As a worst-case scenario, it must be presumed that any logic state elements within the computational machine (registers, memory, etc.) may be affected at the time of disruption.

The short-term disruption of input/output data at an equipment boundary can be managed via a variety of existing methodologies. Data errors must be detected and the associated data suppressed until the error status is cleared. The data processing algorithm should tolerate data loss without signaling a hard fault. The length of time that can be tolerated between valid refreshes depends on the data item and the associated time constants (response) of the system and corresponding function being implemented.

The ability to tolerate disruption that reaches computation and memory elements internal to system equipment without propagation of the associated fault effect is a more difficult problem. For systems with redundant channels, this means tolerance of the disruption without loss of any of the redundant

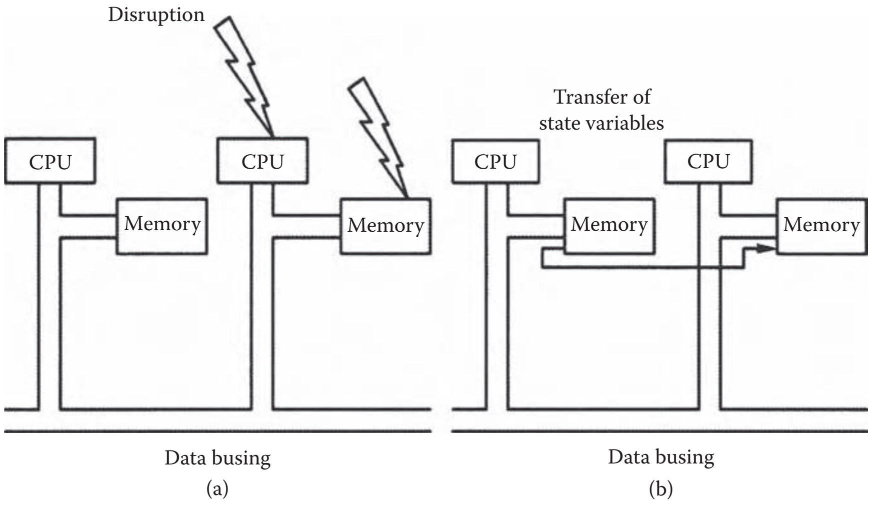


FIGURE 5.10 Redundant CPUs cross-lane recovery (can accomplish some degree of "rapid" recovery).

channels. Fault clearing and computation recovery must be rapid enough to be "transparent" relative to functional operation and flight deck effects. Such computational recovery requires that the disruption be detected and the state of the affected system be restored. Safety-critical systems are almost always mechanized with redundant channels. Outputs of channels are compared in real time, and an errant channel is blocked from propagating a fault effect. One means available for safety-critical systems to detect disruption is the same cross-channel monitor. If a miscompare between channels occurs, a recovery is attempted. For a hard fault, the miscompare condition will not have been remedied by the recovery attempt.

A basic approach to "rapid" computational recovery would be to transmit function state variable data from valid channels to the channel that has been determined faulted and for which a recovery is to be attempted (Figure 5.10). However, the cross-channel mechanization is ineffective against a disruption that has the potential to affect all channels.

5.4.2 Digital Computing Platform

The platform for the Airplane Information Management System (AIMS) used on Boeing 777 aircraft and Versatile Integrated Avionics (VIA) technology is an example of an architectural philosophy in the design of computing platforms. Essentially, VIA is a repackaged version of the AIMS technology. As mentioned, first-generation digital avionics have been plagued with high MTBUR (no-fault-found) rates. One primary goal of the Boeing 777 program was to greatly improve operational readiness and associated life cycle cost performance for the airlines. The AIMS functionally redundant, self-checking pairs architecture was specifically selected to attack these problems. The high integration supported by AIMS required a very comprehensive monitoring environment that is ideal for in-channel "graceful" recovery.

In AIMS, the more dramatic step of making hardware monitoring active on every CPU clock cycle was taken. All computing and input-output (I/O) management resources are lockstep compared on a processor cycle-by-cycle basis. All feasible hardware soft or hard faults are detected. In this approach, if a soft or hard fault event occurs, the processor module is immediately trapped to service handlers, and no data can be exported. In past systems, the latency between such an event and eventual detection (or washout) was the real culprit. The corrupted data would propagate through computations and eventually affect some output. To recover, drastic actions (reboots or rearms) were often necessary. In AIMS, critical functions such as displays (because the flight crew could "see" hiccups) have a "shadowing" standby computational resource. The shadow sees the same input set at the same time as the master self-checking pair. If the master detects an event, within nanoseconds the faulty unit is blocked from generating outputs.

The Honeywell SAFEbus® system detects the loss of output by the master and immediately passes the shadow's correct data for display.

In the faulted processor module, the core system has two copies of processor "state data" fundamental in the self-checking pair. Unlike past systems, in which the single thread processor may be so defective it cannot record any data, at least one half of the AIMS self-checking pair should be successful. Thus, the process of diagnosing hardware errors involves comparing what each half of the pair thought was going on. Errors, down to processor address, control, or data bits, can be easily isolated. If the event was a soft fault, the core system allows a graceful recovery before the processor module is again allowed to export data. On the surface, it appears to be a more sensitive system; however, even with the comprehensive monitoring (potentially a brittle operation), from the standpoint of a self-checking (dual-lock-step) pairs processor data comparison, in these platforms the automatic recovery capabilities should provide a compensating, more robust operation. In other words, from a macro time perspective, system functions will continue to be performed even though, on a micro time basis, a soft fault occurred.

In addition to the isolation of hardware faults (hard or soft), effective temporal and physical partitioning for execution of application software programs involving a variety of software levels has been achieved by the monitoring associated with the self-checking pairs processor and a SAFEbus® communication technology approach.

Defining Terms

DO-160: RTCA Document 160, Environmental Conditions and Test Procedures for Airborne Equipment, produced by RTCA Special Committee 135. Harmonized with ED-14.

ED-14: EUROCAE Document 14, Counterpart to DO-160, produced by EUROCAE Working Groups 14, 31, and 33. Harmonized with DO-160.

EMC: Electromagnetic compatibility is a broad discipline dealing with EM emissions from and susceptibility to electrical/electronic systems and equipment.

EME: Electromagnetic environment, which for commercial aircraft, consists of lightning, HIRF, and the electrical/electronic system and equipment emissions (intra and inter) portion (susceptibility not included) of EMC.

MTBF: Mean Time Between Failures (World Airlines Technical Operations Glossary).

MTBUR: Mean Time Between Unscheduled Removals (World Airlines Technical Operations Glossary).

EUROCAE: European Organization for Civil Aviation Equipment; for the European aerospace community, serving a role comparable to that of the RTCA and SAE.

PED: Portable Electronic Device, an emerging source of EM emissions not included in the EMC discipline.

References

Clarke, C.A. and Larsen, W.E., FAA Report DOT/FAA/CT 86/40, "Aircraft Electromagnetic Compatibility," June 1987.

Report AC25.1309-1A, "System Design and Analysis, Advisory Circular," U.S. Department of Transportation, Washington, D.C., 1988.

Hess, R.F., Computing Platform Architectures for Robust Operation in the Presence of Lightning and Other Electromagnetic Threats, *Proc. Digital Avionics Syst. Conf.*, Irvine, CA, October 1997.

Hess, R.F. Implications Associated with the Operation of Digital Data Processing in the Relatively Harsh EMP Environments Produced by Lightning, Int. Aerosp. Ground Conf. Lightning Static Elect., Paris, France, June 1985.

Hess, R.F., Options for Aircraft Function Preservation in the Presence of Lightning, Int. Conf. Lightning Static Electr., Toulouse, France, June 1999.

Aviation Regulations

AC/AMJ 20-136, “Protection of Aircraft Electrical/Electronic Systems Against the Indirect Effects of Lightning.”

EUROCAE ED-14D/RTCA DO-160D, “Environmental Conditions and Test Procedures for Airborne Equipment.”

MIL-STD-464, “Electromagnetic Environmental Effects Requirements for Systems.”

N8110.67 (FAA Notice), “Guidance for the Certification of Aircraft Operating in High Intensity Radiated Field (HIRF) Environments.”

SAE AE4L Report: AE4L-87-3 (“Orange Book”), “Certification of Aircraft Electrical/Electronic Systems for the Indirect Effects of Lightning,” September 1996 (original publication February 1987).

SAE ARP4754, *Certification Consideration for Highly Integrated or Complex Aircraft Systems*, SAE, Warrendale, PA, issued November 1996.

SAE ARP4761, *Guidelines and Tools for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,* SAE, Warrendale, PA, issued December 1996.

SAE ARP5412, Aircraft Lightning Environment and Related Test Waveforms, SAE, Warrendale, PA, 1999.

SAE ARP5413, Certification of Aircraft Electrical/Electronic Systems for the Indirect Effects of Lightning, SAE, Warrendale, PA, issued August 1999.

SAE Report: AE4L-97-4, “Aircraft Lightning Environment and Related Test Waveforms Standard,” July 1997.

XX.581 Lightning Protection

XX.1316 System Lightning Protection

6 RTCA DO-297/ EUROCAE ED-124 Integrated Modular Avionics (IMA) Design Guidance and Certification Considerations

Cary R. Spitzer
AvioniCon

6.1 Introduction

RTCA document DO-297, "Integrated Modular Avionics (IMA) Design Guidance and Certification Considerations," is one of several documents that are key to the approval of avionics and ultimately the certification of the host aircraft. These documents include DO-160, DO-178, DO-254, SAE ARP 4761, and ARP 4754, as well as many others, such as Technical Standard Orders (TSO). DO-297, like DO-178 and DO-254, provides guidance, not regulations, to be followed in the development of avionics. Consultation by the avionics developers and aircraft manufacturers with the certification authorities will determine which portions of the document will be used as part of the basis for certification of the aircraft. All of the listed RTCA documents (DO designation) become sanctioned by the Federal Aviation Administration (FAA) for certification use through the issuance of Advisory Circulars (AC). Presently, no AC has been issued for DO-297, so its use by the FAA is optional. (ACs have been issued for the other mentioned RTCA documents.) The European Organization for Civil Aviation Equipment (EUROCAE) has designated the document as ED-124. Any reference in this chapter to DO-297 also infers ED-124. As of press time ED-124 has not been approved by the EUROCAE Council and has *not* been released by it. DO-297, however, is available.

The need for DO-297 is derived from the emergence of pioneering IMA architectures on the Boeing B-777, Airbus A-380, and the Boeing B-787. Although DO-297 was too late to be part of the certification baseline for these aircraft, the lessons learned from them did guide, in part, the content of DO-297.

According to DO-297, IMA is a shared set of flexible, reusable, and interoperable hardware and software resources that, when integrated, form a platform that provides services, designed and verified to a defined set of safety and performance requirements, to host applications performing aircraft functions.

The document provides guidance for IMA developers, integrators, applicants, and those involved in the approval and continued airworthiness of IMA systems. It provides specific guidance for the assurance of IMA systems as differentiated from traditional federated avionics.

The key contribution of this document is guidance on the objectives, processes, and activities related to the development and integration of IMA modules, applications, and systems to *incrementally accumulate* design assurance toward the installation and approval of an IMA system on an approved aviation product. Incremental acceptance of individual items of the IMA platform (including the core software) and hosted applications enables the reduction of follow-on certification efforts without compromising system safety.

6.2 Outline

DO-297 contains six chapters:

- Chapter 1 is an overview of the document.
- Chapter 2 introduces the concept of IMA and highlights the system, hardware, and software characteristics.
- Chapter 3 provides a description of IMA-specific development and integration guidelines.
- Chapter 4 is the core of the document and provides guidelines for the acceptance of IMA and describes the relationship to the approval of the installed IMA system.
- Chapter 5 describes integral (umbrella) processes for IMA development.
- Chapter 6 gives guidelines for continued airworthiness of IMA systems.

Figure 6.1 illustrates the relationships between the chapters. The following four annexes are also included:

- Annex A has six tables that summarize the objectives to be satisfied. These objectives are based on the six tasks described in Chapter 4.
- Annex B is the glossary.
- Annex C is the list of acronyms and abbreviations used in the document.
- Annex D contains IMA examples.

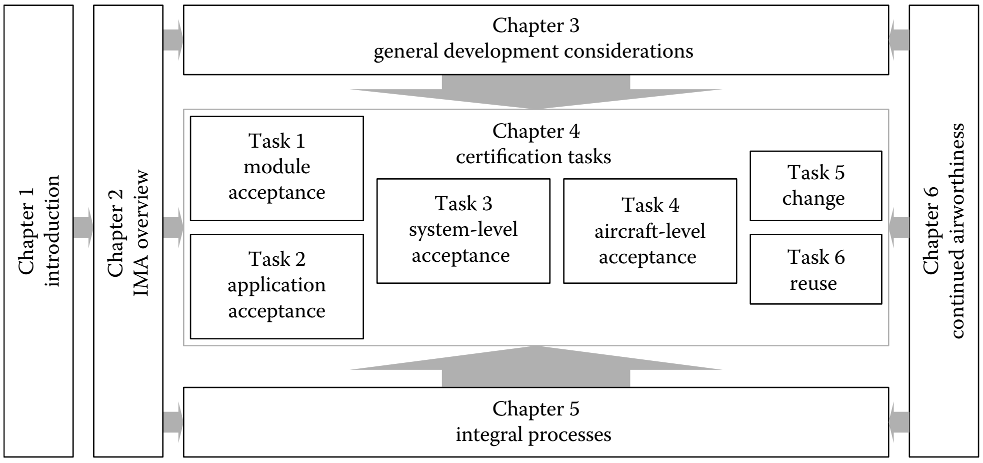


FIGURE 6.1 Chapters and their relationships. © RTCA. Used with permission.

TABLE 6.1 Task Index to Description and Objectives

Task	Reference/Objectives
Task 1: Module acceptance	Section 4.2/Table A-1
Task 2: Application acceptance	Section 4.3/Table A-2
Task 3: IMA system acceptance	Section 4.4/Table A-3
Task 4: Aircraft integration of IMA system – including validation and verification	Section 4.5/Table A-4
Task 5: Change of modules or applications	Section 4.6/Table A-5
Task 6: Reuse of modules or applications	Section 4.7/Table A-6

6.3 Step-by-Step IMA Development and Approval

Six tasks define the incremental acceptance of IMA systems in the certification process:

- Task 1: Module acceptance
- Task 2: Application software or hardware acceptance
- Task 3: IMA system acceptance
- Task 4: Aircraft integration of IMA system, including validation and verification
- Task 5: Change of modules or applications
- Task 6: Reuse of modules or applications

DO-297 provides detailed, comprehensive guidance on each of these tasks. Table 6.1 lists each task along with the section of DO-297 that presents the details on how to accomplish the task. Each task has multiple objectives spelled out in tables in Appendix A.

There are several traditional aspects of avionics development that take on added significance for IMA. Foremost among these are definition of the requirements, liaison among the stakeholders, and robust partitioning. Closely related to these items is the need for very close liaison with the certification authority.

Defining the IMA requirements is a significant challenge in that a very long-term perspective is required. In recognition of the burgeoning role of avionics in modern aircraft operations, coupled with the cost of new or upgraded avionics, aircraft customers demand a large amount of growth in the IMA platform capabilities. The customer may expect the newly delivered IMA to use perhaps only about half of its installed processing power, input/output, and memory. Typically, the IMA cabinet also will have growth slots for additional modules. It is recommended to be very liberal in establishing the spare capabilities, especially the growth slots in the IMA cabinet

In contrast to traditional avionics, the IMA platform and its functions may come from many stakeholders. It is the role of the integrator (typically the aircraft manufacturer) to ensure these stakeholders communicate with the integrator and with each other to ensure that all requirements are identified, validated, accommodated, and verified. Open, effective communication between the integrator and the application providers is especially important.

Partitioning is perhaps the toughest design challenge in developing an IMA system. Because of the multiple functions hosted on a typical IMA platform, the extra demands placed on the partitioning techniques cause them to be referred to as "robust partitioning." In the most basic, lowest-level properties of robust partitioning, no application function in one partition can (1) access memory of any another partition in an adverse manner, (2) affect the timing of any another partition in an adverse manner, or (3) adversely affect the resources used by any other partition. Proof of achieving robust partitioning is especially difficult (see *Avionics: Elements, Software and Functions*, Chapter 16, RTCA DO-178/EUROCAE ED-12, Software Considerations in Airborne Systems and Equipment Certification).

Table 6.2 is from DO-297 and shows the ten objectives to be accomplished in the IMA module or platform development. Other tables in Appendix A cover the other five tasks. Table 6.2 includes the reference to the text that spells out general guidance in Chapter 3 and specific task guidance in Chapter 4. Specific remarks on the documentation are also found in Chapter 4 of DO-297 as listed in the "Life

TABLE 6.2 IMA Module/Platform Development Process (Task 1) Objectives

ID	Objective	Doc Ref	Life Cycle Data Description	Life Cycle Data Reference	Control Category
1	Module/platform development and acceptance life cycle, and associated processes are planned and implemented consistently with the guidance of DO-160, DO-178, DO-254 and this document.	4.2.1a 3.1.1a	Module/platform acceptance plan	4.2.3	CC1
2	Module/platform requirements specifications are defined, traceable, and verifiable.	4.2.1b 3.1.1b	Module/platform requirements specifications	4.2.4	CC1
			Traceability Data	4.2.5	CC2
3	Module/platform design is documented and addresses the IMA unique failure modes, safety analysis, and functionality.	3.1.1c,d 4.2.1b,c 5.1	Module/platform design data	4.2.4	CC1
			Module/platform failure analyses and safety analyses	4.2.12b	CC1
4	Verification and development tools are assessed and qualified, as needed.	4.2.1i 3.4 5.2.3	Module/platform tool qualification data	4.2.12c	CC2 or CC1[1]
5	Partitioning ensures that the behavior of any hosted application is prevented from adversely affecting the behavior of any other application or function.	3.5 3.1.1c,d 4.2.1c,d 5.1, 5.3, 5.4	Partitioning analysis data	4.2.4j	CC1
6	Compliance with module requirements, resource requirements, etc. is demonstrated.	3.1.1d,e 4.2.1c 4.2.1d 4.2.1e	Module/platform V&V data	4.2.5	CC2
7	Ensure module users have the information needed to integrate and interface the module.	4.2.1g,k,l 3.4	Module/platform acceptance data sheet	4.2.10	CC1
8	Platform integration is complete.	4.2.1h 3.1.1d 5.3, 5.4	Platform integration, verification, and validation data	4.2.5	CC2
9	Health monitoring and fault management functions of the IMA platform are provided and documented for use by the hosted applications and the IMA system.	4.2.1c 3.6 5.1.5.5 5.1.5.6	Platform requirements specification	4.2.4f	CC1
10	Quality assurance, configuration management, integration, validation, verification, and certification liaison for the module/platform are implemented and completed.	4.2.1f,j,k 5.3 to 5.7	Module/platform QA records	4.2.6	CC2
			Module/platform CM records	4.2.8	CC2
			Module/platform V&V data	4.2.5	CC2
			Module/platform acceptance accomplishment summary	4.2.9	CC1
			Module/platform configuration index	4.2.7	CC1
			Module/platform problem reports	4.2.11	CC2

© RTCA. Used with permission

[1] Control category for tool qualification data is defined in DO-178/ED-12 (Ref. [2]) or DO-254/ED-80 (Ref. [6]).

Cycle Data Reference" column. The control category sets the document availability: Control Category 1 documents must be delivered to the certification authorities, while Control Category 2 documents may be retained by the organization responsible for their preparation but must be made available upon request to the certification authorities.

TABLE 6.3 Life Cycle Data To Be Submitted To Certification Authority

Life Cycle Data Item	Ref.	Task 1	Task 2	Task 3	Task 4
Module Acceptance Plan (MAP)	4.2.3	X			
Module Configuration Index(es) (MCI)	4.2.7	X			
Module Acceptance Accomplishment Summary (MAAS)	4.2.9	X			
Module Acceptance Data Sheet	4.2.10	X			
Plan(s) for Hardware Aspects of Certification (PHAC)	4.2.12.a 4.3.2	X	X		
Plan(s) for Software Aspects of Certification (PSAC)	4.2.12.a 4.3.2	X	X		
Software Configuration Indices (SCIs)	4.2.12.a 4.3.2	X	X		
Hardware Configuration Indices (HCIs) (i.e., Top-Level Drawings)	4.2.12.a 4.3.2	X	X		
Software Accomplishment Summary(ies) (SAS)	4.2.12.a 4.3.2	X	X		
Hardware Accomplishment Summary(ies) (HAS)	4.2.12.a 4.3.2	X	X		
Safety Assessment Analysis/Report(s)	4.2.12.b			X	X
Hosted Application Acceptance Data Sheet	4.3.2		X		
IMA Certification Plan (system- and aircraft-level)	4.4.3 4.5.3			X	X
IMA Verification and Validation Plan (system- and aircraft-level)	4.4.4 4.5.4			X	X
IMA Configuration Index (system- and aircraft-level)	4.4.5 4.5.5			X	X
IMA Accomplishment Summary (system- and aircraft-level)	4.4.6 4.5.6			X	X
Environmental Qualification Test (EQT) Plan	DO-160	X	X	X	X
Environmental Qualification Test (EQT) Reports	DO-160	X	X	X	X

© RTCA. Used with permission.

Defining Terms

Application software: The part of an application implemented through software. It may be allocated to one or more partitions.

Approval: The act or instance of giving formal or official acknowledgement of compliance with regulations.

Component: A self-contained hardware or software part, database, or combination thereof that may be configuration controlled.

Integral process: A process that assists the system, software or hardware development processes and other integral processes and, therefore, remains active throughout the life cycle. The integral processes are the verification process, the quality assurance process, the configuration management process, and the certification liaison process.

Module: A component or collection of components that may be accepted by themselves or in the context of an IMA system. A module may also comprise other modules. A module may be software, hardware, or a combination of hardware and software, which provides resources to the IMA system hosted applications.

Partition: An allocation of resources whose properties are guaranteed and protected by the platform from adverse interaction or influences from outside the partition.

Partitioning: An architectural technique to provide the necessary separation and independence of functions or applications to ensure that only intended coupling occurs.

References

RTCA DO-178/EUROCAE ED-12, Software Considerations in Airborne Systems and Equipment Certification.

RTCA DO-297/EUROCAE ED-124, Integrated Modular Avionics (IMA) Design Guidance and Certification Considerations.

FAA AC 20-148, Reusable Software Components.

FAA TSO-C153, Integrated Modular Avionics Hardware Elements.

ARINC 653, Avionics Application Software Standard Interface.

7 Certification of Civil Avionics

G. Frank McCormick
Certification Services, Inc.

7.1 Introduction

Almost all aspects of the design, production, and operation of civil aircraft are subject to extensive regulation by governments. This chapter describes the most significant regulatory involvement a developer is likely to encounter in the certification of avionics.

Certification is a critical element in the safety-conscious culture of civil aviation. The legal purpose of avionics certification is to document a regulatory judgment that a device meets all applicable regulatory requirements and can be manufactured properly. At another level, beneath the legal and administrative machinery of regulatory approval, certification can be regarded differently. It can be thought of as an attempt to predict the future. New equipment proposed for certification has no service history, so certification tries, in effect, to provide credible predictions of future service experience for new devices — their influences on flight crews, their safety consequences, their failure rates, and their maintenance needs. Certification is not a perfect predictor, but historically it has been a good one.

This chapter discusses certification activities, for the most part, appropriate to the United States Federal Aviation Administration (FAA). However, be aware that the practices of civil air authorities elsewhere, while generally similar to those of the FAA, often differ in detail or scope. In the original edition of *The Avionics Handbook*, this chapter discussed some representative differences between FAA practices and those of Europe's Joint Aviation Authorities (JAA). At this writing, the European Union is shifting certification responsibilities from the JAA to a new organization, the European Aviation Safety Agency (EASA). That shift is not complete.

Toward the end of this chapter, we will examine some implications of EASA's creation. Gross surprises are unlikely. The institutions involved are world leaders in their respective disciplines. In a sense, European regulators are simply bringing mature capabilities to a new organizational structure. Nevertheless, EASA's regulations and guidance could change in significant and unforeseeable ways, and expensive misunderstandings can result from differences among regulators. The rules and expectations of every authority, the FAA included, change over time. For current guidance, authoritative sources should always be consulted.

This chapter discusses the following topics:

- The FAA regulatory basis
- The Technical Standard Order (TSO) system for equipment approval
- The Supplemental Type Certificate (STC) system for aircraft modification
- Use of FAA Designees in lieu of FAA personnel
- Definition of system requirements
- Safety assessments
- Environmental qualification
- Design assurance of software
- Design assurance of complex electronic hardware
- Production approvals
- EASA

Conceptually, the certification of avionics is straightforward, indeed almost trivial: an applicant simply defines a product, establishes the product's regulatory requirements, and demonstrates that those requirements have been met. The reality is, of course, more problematic. It is a truism that, for any proposed avionics system, a suitable market must exist. As with any commercial pursuit, adequate numbers of avionics units must be sold at margins sufficient to recover investments made in the product. Development costs must be controlled if the project is to survive. Warranty and support costs must be predicted and managed. The choices made in each of these areas will affect and be affected by certification.

This chapter is an introduction to certification of avionics. It is not a complete treatment of the subject. Some important topics are discussed only briefly, and many situations that come up in real-life certification projects are not addressed. Good engineering should not be confused with good certification. A new avionics device might be brilliantly conceived and flawlessly designed, yet ineligible for certification. Good engineering is a prerequisite to good certification, but the two are not synonymous. Certification has a strong legalistic element and is more craft than science. It is not unusual for projects to raise odd regulatory approval quirks during development. Certification surprises are rarely pleasant, but surprises can be minimized or eliminated by maintaining open and honest communication with the appropriate regulators.

7.2 Regulatory Basis of the Federal Aviation Administration

The FAA, created in 1958, acts primarily through publication and enforcement of the Federal Aviation Regulations (FARs). FARs are organized by sections known as Parts. The following are FAR Parts covering most avionics-related activity:

- Part 1 — Definitions and Abbreviations
- Part 21 — Certification Procedures for Products and Parts
- Part 23 — Airworthiness Standards: Normal, Utility, Acrobatic, and Commuter Category Airplanes
- Part 25 — Airworthiness Standards: Transport Category Airplanes
- Part 27 — Airworthiness Standards: Normal Category Rotorcraft

- Part 29 — Airworthiness Standards: Transport Category Rotorcraft
- Part 33 — Airworthiness Standards: Aircraft Engines
- Part 34 — Fuel Venting and Exhaust Emission Requirements for Turbine Engine Powered Airplanes
- Part 39 — Airworthiness Directives
- Part 91 — General Operating and Flight Rules
- Part 121 — Operating Requirements: Domestic, Flag, and Supplemental Operations
- Part 183 — Representatives of the Administrator

Only a subset of these regulations will apply to any given project. A well-managed certification program identifies and complies with the complete but minimum set of applicable regulations.

7.3 FAA Approvals of Avionics Equipment

The FARs provide several different forms of approval for electronic devices installed aboard civil aircraft. Of these, most readers will be concerned primarily with approvals under the TSO system, approvals under an STC, or approvals as part of a Type Certificate, Amended Type Certificate, or Service Bulletin.*

7.4 Technical Standard Order

Approvals under the TSO system are common. TSOs are regulatory instruments that recognize the broad use of certain classes of products, parts, and devices. TSOs apply to more than avionics; they can apply to any civil-aircraft article with the potential for wide use, from seat belts and fire extinguishers to tires and oxygen masks. Indeed, that is the guiding principle behind TSOs — they must be widely useful. Considerable FAA effort goes into the sponsorship and adoption of a TSO. The agency would have little interest in publishing a TSO for a device with limited application.

TSOs contain product specifications, required data submittals, marking requirements, and various instructions and limitations. Many TSOs are associated with avionics: flight-deck instruments, communications radios, Instrument Landing System (ILS) receivers, navigation equipment, collision avoidance systems, and flight data recorders, to name just a few.

TSO-C113, "Airborne Multipurpose Electronic Displays," is representative of avionics TSOs. Electronic display systems are used for different purposes: display of attitude, airspeed, altitude, engine data or aircraft status, or *en route* navigation or guidance during precision approach, maintenance alerts, passenger entertainment, and so on. The same physical display device could potentially be used for any or all of these functions and on many different aircraft types. Recognizing this broad applicability, the FAA published TSO-C113 so that developers could more easily adapt a generic display device to a variety of applications. TSO-C113 is typical, calling out requirements for the following data:

- Explanation of applicability
- Exceptions and updated wording
- References to related regulations, data, and publications
- Requirements for environmental testing
- Requirements for software design assurance
- Requirements for the marking of parts
- Operating instructions
- Equipment limitations
- Installation procedures and limitations
- Schematics and wiring diagrams
- Equipment specifications

*Newly developed equipment has been installed as part of a field approval under an FAA Form 337, though this has become rare and is disallowed in most cases.

- Parts lists
- Drawing lists
- Functional test specifications
- Equipment calibration procedures
- Corrective maintenance procedures

If an avionics manufacturer applies for a TSO approval and that manufacturer's facilities, capabilities, and data comply with the terms of the TSO, the manufacturer receives a TSO Authorization (TSOA) from the FAA. A TSOA represents approval of both design data and manufacturing rights; that is, the proposed device is deemed to be acceptable in its design, and the applicant has demonstrated the ability to produce identical units.

In TSO-based projects, the amount of data actually submitted to the FAA varies by system type, by the FAA's experience with particular applicants, and by FAA region. In one case, an applicant might be required to submit a great deal of certification data; in another, a one-page letter from an applicant might be adequate for issuance of a TSOA. On any new project, it is unwise to presume that all regulatory requirements are known. Consistency is a goal of the FAA, but regional differences among agency offices do exist. Early discussion with the appropriate regulators will ensure that the expectations of agency and applicant are mutually understood and agreed on.

For more information on TSOs, see FAA Advisory Circular 20-110 (Revision L or later), "Index of Aviation Technical Standard Orders" and FAA Order 8150.1 (Revision B or later), "Technical Standard Order Program." The latter Order is helpful to TSO planning in which the proposed device (1) meets only a subset of its TSO specification or (2) performs functions covered by multiple TSOs.

Note that a TSO does *not* grant approval for installation in an aircraft. Although data approved under a TSO can be used to support an installation approval, the TSOA itself applies only to the equipment in question. Installation approvals must be pursued through other means (Section 7.5) and are not necessarily handled by an avionics equipment manufacturer.

7.5 Supplemental Type Certificate

An STC is granted to a person or organization, usually other than the aircraft manufacturer, that wishes to modify the design of an existing aircraft. Retrofits and upgrades of avionics equipment are common motivations for seeking STC approvals from the FAA. In an STC, the applicant is responsible for all aspects of an aircraft modification. Those aspects typically include the following:

- Formal application for an STC
- Negotiation of the certification basis of the relevant aircraft with the FAA
- Identification of any items requiring unusual regulatory treatment
- Preparation of a certification plan
- Performance of all analyses specified in the certification plan
- Coordination with the FAA throughout the project
- Physical modification of aircraft configuration
- Performance of all conformity inspections
- Performance of all compliance inspections
- Performance of all required lab, ground, and flight testing
- Preparation of flight manual supplements
- Preparation of instructions needed for continued airworthiness
- Preparation of a certification summary
- Support of all production approvals

An applicant for an STC must be "a U.S. entity," although the exact meaning of that phrase is not always clear. One common case is that of a nominally foreign firm with an office in the United States. It is acceptable to the FAA for that U.S.-based office to apply for and hold an STC.

An applicant for an STC begins the process officially by completing and submitting FAA Form 8110-12, "Application for Type Certificate, Production Certificate, or Supplemental Type Certificate," to the appropriate FAA Aircraft Certification Office (FAA ACO). Accompanying that application should be a description of the project and the aircraft type or types involved, the project schedule, a list of locations where design and installation will be performed, a list of proposed Designees (discussed later in this chapter), and, if desired, a request for an initial meeting with the FAA. The FAA will assign a project number, appoint a manager for the project, schedule a meeting if one was requested, and send to the applicant an acknowledgement letter with these details. In some cases, a lengthy or involved project might require the FAA regional office to coordinate with FAA national headquarters before committing its support.

The applicant must determine the certification basis of the aircraft to be modified. The "certification basis" is the sum of all applicable FAA regulations (at specified amendment levels) and any binding guidance that applies to the aircraft and project in question. Regulations tend to become more stringent over time. Complying with later rules may be more time-consuming and expensive than with earlier rules.

The starting point for a certification basis may be established by reference to the Type Certificate Data Sheet (TCDS) for each affected aircraft type. In most cases, the certification basis will simply be those rules currently in effect at the time of application and applicable to the STC in question. Until relatively recently, aircraft modifiers were routinely allowed to comply with an aircraft's original certification basis. Long-obsolete requirements were thus "grandfathered" into current acceptability. Such grandfathering is now essentially extinct. New modifications must comply with current rules.

Complex avionics systems, extensive aircraft modifications, and novel system architectures all raise the odds that something in a project will be unusual, that something will not fit neatly into the normal regulatory framework. For such activities, an applicant might wish to propose compliance based on other regulatory mechanisms, such as alternative means of compliance, findings of equivalent safety, exemptions, or special conditions. If so, generic advice is largely useless. By their nature, these activities are unusual and require close coordination with the FAA.

An STC applicant must prepare a certification plan. The plan should include the following:

- A brief description of the modification and how compliance is to be substantiated
- A summary of the Functional Hazard Assessment (see Section 7.9)
- A list of proposed compliance documentation, including document numbers, titles, authors, and approving or recommending Designees, if applicable (the role of Designees is described in more detail later in this chapter)
- A compliance checklist, listing the applicable regulations from the certification basis, their amendment number, subject, means of compliance, substantiating documents, and relevant Designees
- A definition of Minimum Dispatch Configuration
- If used, a list of the proposed FAA Designees, including name, Designee number, appointing FAA office, classification, authorized areas, and authorized functions
- A project schedule, including dates for data submittals, test plan submittals, tests (with their locations), conformity inspections, installation completion, ground and flight testing, and project completion

Some FAA ACOs require all Designated Engineering Representatives (see Section 7.7) participating in a project to sign an FAA Form 8110-3, "Statement of Compliance with the Federal Aviation Regulations," recommending approval of the project's certification plan.

Extensive analysis and testing are generally required to demonstrate compliance. Results of these analyses and tests must be preserved. Later in this chapter, four of the most important of these activities — safety assessments, environmental qualification, software assurance, and design assurance of complex electronic hardware — will be discussed, along with another engineering topic, development and handling of system requirements.

The FAA's involvement in an STC is a process, not an act. FAA specialists support multiple projects concurrently, and matching the schedules of applicant and agency requires planning. That planning is the applicant's responsibility. Missed deadlines and last-minute surprises on the part of an applicant can result in substantial delays to a project as key FAA personnel are forced to reschedule their time, possibly weeks or months later than originally planned.

The STC process assumes modification of at least one prototype aircraft. It is in the aircraft modification that all the engineering analysis — aircraft performance, structural and electrical loading, weight and balance, human factors, and so on — comes together. Each component used in an aircraft modification must either be individually examined for conformance to its complete specifications or, more commonly, manufactured under an approved production system. Individual examination is known as "parts conformity inspection." A completed aircraft modification is then subject to an "installation conformity inspection." In complex installations or even complex parts, progressive conformity inspections may be required. Conformity inspections are conducted by an FAA Inspector or a Designee authorized by the FAA — a Designated Manufacturing Inspection Representative (DMIR) or Designated Airworthiness Representative (DAR) (see Section 7.7).

Once again, a conformity inspection ensures that a part matches its specifications. By contrast, a compliance inspection verifies through physical examination that a modification complies with applicable FARs. Typical of compliance inspections are examinations of modified wiring on an aircraft or of visibility of required placards. A compliance inspection is conducted by an FAA engineer or authorized DER.

For significant projects involving ground and flight testing, the FAA will issue a Type Inspection Authorization (TIA). The TIA details all the inspections, ground tests, and flight tests necessary to complete the certification program. Prior to issuing a TIA, the FAA should have received and reviewed all of the descriptive and compliance data for the project. The FAA has recently added an item to its TIA procedures: the flight test risk assessment. The risk assessment seeks to identify and mitigate any perceived risks in flight tests that include FAA personnel, based on data supplied by the applicant.

New avionics equipment installed as part of an STC will usually impose new and different procedures on flight crews. An applicant will, in most cases, document new procedures in a supplement to an approved flight manual. In complex cases, it may also be necessary to provide a supplement to an operations manual.

An applicant must provide instructions for the continued airworthiness of a modified airplane. Penetrations of the pressure vessel by, say, wiring or tubing, may require periodic inspection. Actuators associated with a new subsystem may need scheduled maintenance. Instructions for continued airworthiness are usually supplements to maintenance manuals but may also include supplements to illustrated parts catalogs, structural repair manuals, structural inspection procedures, or component maintenance manuals.

Much of this discussion has been more applicable to transport aircraft than to smaller aircraft. Regulatory requirements for the smaller (FAR Part 23) aircraft are, in some respects, less stringent than for transport aircraft. Yet even for transports, not everything described above is required in every circumstance. Early discussion between applicant and regulator is the quickest way to determine what actually needs to be done.

Since the 1960s, some avionics developers have found it desirable to seek STCs through a type of firm called a Designated Alteration Station (DAS). A DAS can, if properly authorized by the FAA, perform all the work associated with a given aircraft modification and issue an STC. In this approach, the avionics developer might not deal with FAA personnel at all. Two key issues are ownership of the STC rights and handling of production approvals.

Since publication of the first edition of the *Avionics Handbook*, the FAA has begun an overhaul of its organizational delegations (primarily Designated Alteration Stations, Delegation Option Authorizations, and SFAR [Special Federal Aviation Regulation] 36 Authorizations). On October 13, 2005, the FAA published final rules (Federal Register, Volume 70, Number 197) that replace previous organizational delegations with a new framework known as Organization Designation Authorization (ODA). ODA consolidates previous organizational authorizations in a new Subpart D under FAR Part 183 and, in the process, addresses new flight standards functions. ODAs will be constituted to handle type certification, production certification, supplemental type certification, Technical Standard Order Authorization, major repair, major alteration, airworthiness, Parts Manufacturer Approval, and flight standards operational authorization. As this second edition of the *Handbook* is being prepared, an FAA Order for ODA guidance is in progress but is unavailable to the public. That order and related guidance should be available by the time you read this.

It will take some time to phase out previous organizations and shift them to ODA status. Under the 2005 rules, the shift must be complete by November 14, 2009. If, during the transition to an all-ODA regime, you contemplate use of an outside organization to handle your STCs or related approvals, you might need to assess both DASs and ODAs.

7.6 Type Certificate, Amended Type Certificate, and Service Bulletin

Approvals as part of a Type Certificate (TC), Amended Type Certificate (ATC), or Service Bulletin are tied to the certification activities of airframers or engine manufacturers, referred to collectively as original equipment manufacturers (OEMs). On programs involving TCs, ATCs, or Service Bulletins, an avionics supplier's obligations are roughly similar to those imposed by an STC project, though detailed requirements can vary greatly. Avionics suppliers participating in an aircraft or engine development program can and should expect to receive certification guidance from their OEM customers.

In such programs, suppliers should be cautious in their evaluations of proposals that shift extraordinary or ill-defined responsibilities for certification to suppliers. Outsourcing is a useful tool and can be implemented and managed effectively. Doing so requires a comprehensive understanding of obligations on the part of every party concerned. But comprehensive understanding is rarely available during, say, contract negotiations, when that understanding is crucial. Without it, projects can plunge ahead, fed more by optimism than by a clear-eyed view of potentially expensive obligations. Suppliers who accept certification responsibilities on behalf of an OEM — particularly those suppliers with little or no prior certification experience that is directly applicable to the systems proposed — should be prepared to do nontrivial homework in the subject and to address hard questions candidly with their customers and regulators before agreements are signed.

7.7 FAA Designees

In the United States, any applicant may deal directly with the FAA. The FAA does not collect fees for its services from applicants. This provision of services at no charge to the user is in contrast to the practices of other civil air authorities, some of which do charge set fees for routine regulatory matters. At its discretion, the FAA can, however, appoint individuals who meet certain qualifications to act on its behalf. These appointees, called Designees, receive authorizations under FAR Part 183 and serve in a variety of roles. Some are physicians who issue medical certificates to pilots. Others are examiners who issue licenses to new pilots. Still others are inspectors authorized to approve maintenance work.

Avionics developers are most likely to encounter FAA DERs, DMIRs, or DARs.

All Designees must possess authorizations from the FAA appropriate to their activities. DERs can approve engineering data. Flight Test Pilot DERs can conduct and approve the results of flight tests in new or modified aircraft. DMIRs and DARs can perform conformity inspections of products and installations. DARs can issue Airworthiness Certificates. When acting in an authorized capacity, a Designee is legally a representative of the FAA; in most respects, he or she *is* the FAA for an applicant's purposes. Nevertheless, there are practical differences in conduct between the FAA and its Designees.

The most obvious difference is that an applicant actually hires and pays a Designee, and thus has more flexibility in managing the Designee's time on a project. The resulting benefits in project scheduling can more than offset the costs of the Designee. Experienced Designees can be sources of valuable guidance and recommendations, whereas the FAA generally restricts itself to findings of compliance — that is, the agency will simply tell an applicant whether or not submitted data complies with the regulations. If data is judged noncompliant, the FAA will not, in most cases, tell an applicant how to bring it into compliance. A Designee, however, can assist an applicant with recovery strategies or, better yet, steer an applicant toward approaches that are predictably compliant in the first place.

The FAA often encourages the use of Designees by applicants. An applicant must define and propose the use of Designees, by name, to the FAA ACO for each project. If the proposed Designees are acceptable to the ACO, the ACO will coordinate with its manufacturing counterpart and delegate certain functions to the specified Designees. Those Designees then act as surrogates for the relevant FAA personnel on the project, providing oversight and ultimately approving or recommending approval of compliant data.

Although an applicant's use of Designees is discretionary, the realities of FAA workload and scheduling may make the use of Designees a pragmatic necessity. Whenever Designees are considered for inclusion in a project, their costs and benefits should be evaluated with the same care devoted to any other engineering resource. For more information, see FAA Order 8100.8 (Revision B or later), "Designee Management Handbook" and FAA Order 8110.37 (Revision C or later), "Designated Engineering Representatives (DER) Guidance Handbook."

This chapter has so far dealt mainly with the definitions and practices of FAA regulation. There is, of course, a great deal of engineering work to be done in any avionics development. Five engineering topics of great interest to the FAA are the handling of system requirements, performance of a safety assessment, environmental qualification, software assurance, and design assurance of so-called "complex electronic hardware."

7.8 System Requirements

Avionics developers must document the requirements of their proposed systems, ideally in ways that are easily controlled and manipulated. Many experienced practitioners regard the skillful capture of requirements as the single most important technical activity on any project. A system specification is the basis for descriptions of normal and abnormal operation, for testing, for training and maintenance procedures, and much else. Our brief treatment of the topic here does not imply that it can be approached superficially. On the contrary, system specification is so important that a large body of literature exists for it elsewhere (see Chapter 2 for a starting point). Requirements definition is supported by many acceptable methods. Each company evolves its own practices in this area.

Over the years, many types of avionics systems have come to be described by *de facto* standardized requirements, easing the burden of both engineering and certification. New systems, though, are free to differ from tradition in arbitrary ways. Applicants should expect such differences to be scrutinized closely by regulators and customers, who may demand additional justification and substantiation of the changes.

Proper requirements are the foundation for well-designed avionics. Whatever the sources of requirements and whatever the methods used for their capture and refinement, an applicant must be able to demonstrate that a new system's requirements — performance, safety, maintenance, continued airworthiness, and so on — have been addressed comprehensively. Some projects simply tabulate requirements manually, along with one or more means of compliance for each requirement. Others implement large, sophisticated databases to control requirements and compliance information. Compliance is generally shown through analysis, test, inspection, demonstration, or some combination thereof.

7.9 Safety Assessment

Early in a project — the earlier the better — developers should consider the aircraft-level hazards associated with their proposed equipment. This is the first of possibly several steps in the safety assessment of a new system. There is an explicit correlation between the severity of a system's hazards and the scrutiny to which that system is subjected. With a few notable exceptions,* systems that are inconsequential from

*For example, failures of flight data recorders, cockpit voice recorders, and emergency locator transmitters have no effect on continued safe flight and landing. Conventional safety-assessment reasoning would dismiss these devices from failure-effect considerations. Such systems obviously perform important functions, however, and the FAA defines them as worthy of more attention than suggested by a safety assessment. For more discussion of this topic, refer to Software Assurance in this chapter for a description of software levels assigned to flight data recorders a safety standpoint receive little attention. Systems whose improper operation can result in aircraft damage or loss of life receive a great deal of attention and require correspondingly greater engineering care and substantiation.

Unsurprisingly, there is an inverse relationship between the severity of a system's hazards and the frequency at which those hazards are tolerated. Minor annoyances might be tolerable every thousand or so flight hours. Catastrophic hazards must occur less frequently than once every billion flight hours. Most hazards fall somewhere between those two extremes. For transport aircraft, the regulations also require that no single random failure, regardless of probability, result in a catastrophic hazard, implying that any such hazard must arise from two or more independent failures.

Initial considerations of hazards should be formalized in a Functional Hazard Assessment (FHA) for the proposed system. An FHA should address hazards only at levels associated directly with operation of the system in question. For example, an autopilot FHA would consider the hazards of an uncommanded hardover or oscillation of a control surface. A display-system FHA would consider the hazards of blank, frozen, and active-but-misleading displays during various phases of flight.

In general, if an FHA concludes that misbehavior of a system has little or no effect on continued safe flight and landing, no further work is needed for its safety assessment. Conversely, if the FHA confirms that a system can pose nontrivial risk to aircraft or occupants, then investigation and analysis must continue. Additional work, if needed, will likely involve preparation of a Preliminary System Safety Assessment, a Fault Tree Analysis, a Failure Modes and Effects Analysis, a Common Cause Analysis, and a final System Safety Assessment.

In the absence of a specific aircraft installation, assumptions must be made regarding avionics usage to make progress on a safety assessment. This is true in TSO approvals, for example, if design assurance levels are not specified in the TSO or if developers contemplate hazards or usage different from those assumed in the TSO. There are pitfalls* in unmindful acceptance and use of generic hazard classifications and software levels (see Section 7.11), even for standard products. Technologies change quickly; regulations do not. The gap between what is technically possible and what can be approved sometimes leads to conflicting requirements, bewildering difficulties, and delays in bringing to market devices that offer improvements to safety, operating economics, or both. The solution is early agreement with the appropriate regulators concerning the requirements applicable to a new device.

The details of safety assessments are outside the scope of this chapter. For an introduction to safety-related analysis, refer to the following:

- ARP4754 — Certification Considerations for Highly-Integrated Or Complex Aircraft Systems; Society of Automotive Engineers Inc., November 1996
- ARP4761** — Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment; Society of Automotive Engineers Inc., December 1996
- NUREG-0492 — Fault Tree Handbook; U.S. Nuclear Regulatory Commission, 1981
- FAA Advisory Circular 25.1309-1A — System Design Analysis, 1988

*A given TSO might specify a software level (see Section 7.11), and a TSOA could certainly be granted on that basis. Note, though, that actual installation of such a device on an aircraft might require a higher software level. For example, an airspeed sensor containing Level C software could be approved under TSO-C2d. That approved sensor could not, however, be used to supply a transport aircraft with primary air data. The primary air data function on Part 25 aircraft invariably requires at least some Level A software.

**ARP4754 and ARP4761 (or their successors) are expected to be recognized by a new FAA advisory circular, AC 25.1309-1B. At this writing, the AC has not been adopted but exists as a relatively mature draft referred to as the "Arsenal version." The FAA and EASA have accepted proposals by applicants to use Arsenal on recent development programs. No date has been set by press deadlines for formal release of AC 25.1309-1B.

- FAA Advisory Circular 23.1309-1C* — Equipment, Systems, and Installations in Part 23 Airplanes, 1999
- *Safeware: System Safety and Computers* — Nancy G. Leveson, Addison-Wesley Publishing Company, 1995
- *Systematic Safety: Safety Assessment of Aircraft Systems* — Civil Aviation Authority (UK), 1982

Customers may demand that some failures, even those associated with minor hazards, occur less frequently than tolerated by regulation. In other words, the customer's requirement may be more stringent than the FAA's. Economic issues, such as dispatch reliability and maintenance costs, are the usual motivations. In such cases, meeting the customer's specification automatically satisfies the regulatory requirement.

Some TSOs refer to third-party guidance material, usually in the form of equipment-performance specifications from organizations such as RTCA** and the Society of Automotive Engineers. TSOs, Advisory Circulars, and third-party specifications can explicitly call out hazard levels and software assurance levels. If those specifications prescribe hazard levels and assurance levels appropriately for a given project, developers may simply adopt the prescriptions given for use in their own safety assessments. Developers must still, of course, substantiate their claims to the levels called for.

In addition to a safety assessment, analysis of equipment reliability may be required in order to predict average times between failures of the equipment. Although this analysis is often performed by safety analysts, the focus is different. Whereas a safety assessment is concerned with the operational consequences and probabilities of system failures, a reliability analysis is concerned with the frequency of failures of particular components in a system.

7.10 Environmental Qualification

Environmental qualification is required of avionics. The standard in this area is RTCA/DO-160 (Revision E or later), "Environmental Conditions and Test Procedures for Airborne Equipment" (RTCA, 2004). DO-160E specifies testing for temperature range, humidity, crashworthiness, vibration, susceptibility to radiated and conducted radio frequencies, lightning tolerance, and other environmental factors. It is the responsibility of applicants to identify environmental tests appropriate to their systems. Whenever choices for environmental testing are unclear, guidance from FAA personnel or DERs is in order.

To receive certification credit, environmental testing must be performed on test units whose configurations are controlled and acceptable for the tests in question. Conformity inspection may be necessary for test articles not manufactured in accordance with a production approval. An approved test plan, test-setup conformity inspection, and formal witnessing of tests by FAA specialists or Designees are often required. In all cases, an applicant must document and retain evidence of equipment configurations, test setups, test procedures, and test results.

7.11 Software Assurance

Software has become indispensable to avionics development and has a correspondingly high profile in certification. It is often the dominant consideration in certification planning. For software, regulatory compliance can be shown by conforming to the guidelines described in RTCA/DO-178 (Revision B or

*An applicant developing avionics exclusively for general aviation should pay special attention to Advisory Circular 23.1309-1C. That AC offers regulatory relief from many requirements that would otherwise apply. In particular, for some functions on several classes of small airplanes, the AC allows software assurance at lower levels than would be the case for transport aircraft.

**RTCA, Inc., formerly known as the Radio Technical Corporation of America, is a nonprofit association of U.S.-based aeronautical organizations from both government and industry. RTCA seeks sound technical solutions to problems involving the application of electronics and telecommunications to aeronautical operations. RTCA tries to resolve such problems by mutual agreement of its member and participating organizations (cf. EUROCAE).

later), “Software Considerations in Airborne Systems and Equipment Certification.” DO-178B was developed jointly by RTCA and the European Organization for Civil Aviation Equipment (EUROCAE)*.

DO-178B is not a development standard for software; it is an assurance standard. DO-178B is neutral with respect to development methods. Developers are free to choose their own methods, provided the results satisfy the assurance criteria of DO-178B in the areas of planning, requirements definition, design and coding, integration, verification, configuration management, and quality assurance.

DO-178B defines five software levels, A through E, corresponding to hazard classifications derived from the safety assessment discussed earlier. At one extreme, Level A software is associated with functions whose anomalous behavior could cause or contribute to a catastrophic failure condition for the aircraft. Obvious examples of Level A software include fly-by-wire primary control systems and full-authority digital engine controllers. At the other extreme, passenger entertainment software is almost all Level E, because its failure has no safety-related effects. A sliding scale of effort exists within DO-178B: the more critical the software, the more scrutiny that must be applied to it. Level A software generates more certification data than does Level B software, Level B generates more than does Level C, and so on.

Avionics customers sometimes insist on software assurance levels higher than those indicated by a safety assessment. This is purely a contractual matter. Confusion can be avoided by separating a customer’s contractual wishes from regulatory compliance data submitted to the FAA or to DERs. Certification submittals should be based on the safety assessment rather than on the contract. If a safety assessment concludes that a given collection of software should be Level C, but that software’s customer wants it to be Level B, then in general the applicant should submit to the FAA plans and substantiating data for Level C software. Any additional evidence needed to demonstrate contractual compliance to Level B should be an issue between supplier and customer. That evidence is not required for certification and should become a regulatory matter only in unusual circumstances.**

FAA guidance itself sometimes requires that software be assured to a level higher than indicated by conventional safety assessment. This is not uncommon in equipment required for dispatch but whose failures do not threaten continued safe flight and landing. For example, a flight data recorder must be installed and operating in most scheduled-flight aircraft. Note, however, that failure of a recorder during flight has no effect on the ability of a crew to carry on normally. Thus, from a safety-assessment viewpoint, a flight data recorder has no safety-related failure conditions. Based on that, the recorder’s software would be classified as Level E, implying that the software needs no FAA scrutiny. This, of course, violates common sense. The FAA plainly has a regulatory interest in the proper operation of flight data recorders. To resolve such mismatches, the FAA requires at least Level D for any software associated with a dispatch-required function.

Digital technology predates DO-178B. Many software-based products were developed and approved before DO-178B became available. If an applicant is making minor modifications to equipment approved under an older standard, it may be possible to preserve that older standard as the governing criteria for the update. More typically, the FAA will require new or changed software to meet the guidelines of DO-178B, with unchanged software “grandfathered” into the new approval. When transport airplanes are involved in such cases, an Issue Paper dealing with use of “legacy” software may be included in the certification basis of the airplane. In a few cases, the FAA may require wholesale rework of a product to meet current standards.

How much software data should be submitted to the FAA for certification? It is impractical to consider submitting all software data to regulators. An applicant can realistically submit — and regulators can

*RTCA/DO-178B is equivalent to EUROCAE/ED-12B, “*Considerations sur le Logiciel en Vue de la Certification des Systemes et Equipements de Bord*” (EUROCAE, 1992).

**It is usually prudent to avoid setting precedents of additional work beyond that required by regulation. Applicants are always free to do additional work, of course — developers often do, for their own reasons — and if the regulations seem inappropriate or inadequate, applicants should seek to improve the regulations. However, precedents are powerful things, for good and ill, in any regulatory regime. New precedents can have unintended and surprising consequences.

realistically review — only a fraction of the data produced during software development. Applicants should propose and negotiate that data subset with the FAA. Whether submitted formally or not, an applicant should retain and preserve all relevant data (see DO-178B, Section 9.4, as a starting point). The FAA can examine an applicant's facilities and data at any time. It is each applicant's responsibility to ensure that all relevant data is controlled, archived, and retrievable.

For more information on software-assurance guidelines, see Chapter ??? of this book and supplemental information to DO-178B, such as RTCA/DO-248B, "Final Annual Report for Clarification of DO-178B"; RTCA/DO-278, "Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance"; and FAA Order 8110.49, "Software Approval Guidelines."*

7.12 Complex Electronic Hardware

For more than a decade, regulators have watched with growing concern as ever more complex digital devices have moved into increasingly critical aircraft functions. Today, most aircraft systems rely in whole or in part on digital logic. Design errors in that logic imply hazards. Because of their discrete logic, digital devices resemble software more than they do traditional, physical, tangible things. Whereas a given aircraft's structural element might be characterized usefully and effectively in terms of its linearized bending, shear, and torsional loads and stresses, software and digital devices enjoy no such simplifications.

As with software, the behavior of a digital device is essentially arbitrary. The designer of digital logic has free rein to specify any relationship desired among inputs, outputs, and timing. The difficulty lies in getting those relationships right in all circumstances. Complex digital hardware can rarely be "proven" correct. Instead, designers and regulators accept certain forms of design assurance as evidence that the device in question is suitable for use on civil aircraft. The standard governing hardware design assurance is RTCA/DO-254, "Design Assurance Guidance for Airborne Electronic Hardware."

DO-254 is oriented toward process. It aims to remove errors from hardware products through scrutiny of planning, requirements, design, validation, verification, configuration management, test criteria, documentation, and so on. Five design assurance levels, A through E, are recognized. These levels correspond to hazards of Catastrophic, Hazardous, Major, Minor, and No Safety Effect, respectively. The assurance level assigned to a device determines its depth of documentation, its requirements for independence, and the applicability of particular objectives within DO-254.

In recent years, the FAA has paid ever greater attention to programmable logic devices: application-specific integrated circuits, field-programmable gate arrays, and so on. These devices are referred to collectively as complex electronic hardware. The agency's increased scrutiny of complex electronic hardware is intended to ensure that acceptable processes are being followed during development of such devices. FAA Advisory Circular 20-152 authorizes use of RTCA/DO-254 as a means of showing compliance to the airworthiness requirements of complex electronic hardware and describes application of DO-254 to those technologies. For more information, see Chapter 16 in *Avionics: Elements, Software and Functions.*

7.13 Manufacturing Approvals

It is not enough to obtain design approval for avionics equipment; approval to manufacture and mark production units must be obtained as well. Parts manufactured in accordance with an approved production system do not require parts conformity inspection. With a TSO, as explained earlier, the approvals of design and manufacturing go together and are granted simultaneously. In order to receive a TSOA,

*FAA Order 8110.49 covers the following topics: (1) the software review process, (2) required levels of FAA involvement in software projects, (3) software conformity inspections, (4) field-loadable software, (5) user-modifiable software, (6) Level D previously developed software, (7) software tool qualification, (8) software changes in legacy systems, (9) software change impact analysis, and (10) reuse of software life cycle data.

the applicant must demonstrate not just an acceptable prototype but also an ability to manufacture the article. An STC holder must demonstrate production capabilities separately. After obtaining an STC approval, the holder may apply for Parts Manufacturer Approval (PMA) authority to produce the parts necessary to support the STC. PMA approvals are issued by the FAA Manufacturing Inspection District Office responsible for the applicant. An STC applicant who will need subsequent PMA authority should plan and prepare for PMA from the beginning of a project. Alternatively, an STC holder may assign production rights to others, who would then hold PMA authority for the parts in question. For more information, see FAA Order 8110.42, "Parts Manufacturer Approval Procedures."

7.14 European Aviation Safety Agency

In the first edition of this handbook, a grouping of Europe's national civil air authorities — the JAA — was discussed. The purpose of that discussion was to highlight a few representative similarities and differences between the FAA and other influential regulatory authorities. Since that time, many of the JAA's responsibilities have been shifted to the EASA. Through new legislation by the European Parliament and the Council of the European Union, EASA began work officially in late September 2003. EASA is headquartered in Cologne, Germany. At this writing, some questions regarding European civil air regulation are answerable on, at best, a case-by-case basis. Nevertheless, the outlines of EASA's future are clear.

EASA already has responsibility for design approvals, continued airworthiness, design organization approvals, and environmental certification within the European Union. In addition, EASA will approve certain activities — production facilities, repair stations, and maintenance training organizations — outside the European Union. The agency is charged with standardization and oversight of all aviation safety certification activities among its member states, and, when the relevant requirements are in final form, EASA will assume responsibility for operational approvals and personnel licensing. A transition to full operations is expected to be complete by March 2007.

The national air authorities of European Union countries will retain responsibility for many production activities and maintenance functions within their own countries, and for acceptance of products into their own national registries. These national authorities will follow EASA procedures. The JAA will address operational and licensing issues pending full transition of those responsibilities to EASA. The JAA will also represent states that are members of the JAA but not of the European Union, and will accept EASA approvals for recommendation to those states.

It will take time for the new European regulatory structure to stabilize and mature. Until then, applicants should not assume that prior knowledge of specifics is necessarily accurate or useful.

Reinterpretations of existing bilateral agreements may affect a project's administrative requirements. The details of import or export will almost certainly change. Acceptance of an FAA validation by EASA, or vice versa, may be subject to new procedures. Certification bases, TCs, STCs, TSOs, repair authorizations, airworthiness approvals, continued airworthiness, and Airworthiness Directives may all be affected. Most of the unsettled or confusing issues between applicant and regulator will be resolved straightforwardly, sensibly, and amicably. There will surely be pitfalls, however, for the presumptuous or unwary applicant. For now, extra homework, extra coordination, and extra caution are in order.

For more information, see www.easa.eu.int, as well as FAA Order 8100.14 (Revision A or later), "Interim Procedures for Working with the European Community on Airworthiness Certification and Continued Airworthiness"; FAA Order 8110.52, "Type Validation and Post-Type Validation Procedures" (April 29, 2005); and the EASA Certification Policy Memorandum of June 1, 2005.

7.15 Summary

Certification can be straightforward, but like any other developmental activity, it must be managed. At the beginning of a project, applicants should work with their regulators to define expectations on both sides. During development, open communication should be maintained among suppliers, customers, and regulators. In a well-run project, evidence of compliance with regulatory requirements will be produced with little incremental effort, almost as a side-effect of good engineering, during the normal course of work. The cumulative result will, in the end, be a complete demonstration of compliance, soon followed by certification.

Regulatory officials, whether FAA employees or Designees, work best and are most effective when they are regarded as part of an applicant's development team. An applicant is obliged to demonstrate compliance with the applicable regulations, nothing more. However, partial information from an applicant can lead to misunderstandings and delays, and attempts to resolve technical disagreements with regulators through nontechnical means rarely have the desired effect. In the past, regrettably, large investments have been made in systems that could not be approved by the FAA. In order to avoid such outcomes, applicants are well advised to hold early discussions with appropriate FAA personnel or Designees.

Defining Terms

Certification: Legal recognition, through issuance of a certificate, by a civil aviation authority that a product, service, organization, or person complies with that authority's requirements

Certification basis: The sum of all current regulations applicable to a given project at the time application is made to a civil aviation authority to begin a certification process

Designee: An individual authorized by the FAA under FAR Part 183 to act on behalf of the agency in one or more specified areas

Issue Paper: Instrument administered by an FAA Directorate to define and control a substantial understanding between an applicant and the FAA, such as formal definition of a certification basis or a finding of equivalent safety, or to provide guidance on a specific topic, such as approval methods for programmable logic devices

PMA: Parts Manufacturer Approval, by which the FAA authorizes the production of parts for replacement and modification, based on approved designs

Special Condition: A modification to a certification basis, necessary if an applicant's proposed design features or circumstances are not addressed adequately by existing FAA rules; in effect, a new regulation, administered by an FAA Directorate, following public notice and a public comment period of the proposed new rule

STC: Supplemental Type Certificate, by which the FAA approves the design of parts and procedures developed to perform major modifications to the design of existing aircraft

TSOA: Technical Standard Order Authorization, the mechanism by which the FAA approves design data and manufacturing authority for products defined by a Technical Standard Order

References

Certification Services, Inc.: www.certification.com
European Aviation Safety Agency (EASA): www.easa.eu.int
European Organization for Civil Aviation Equipment (EUROCAE): www.eurocae.org
Federal Aviation Administration (FAA): www.faa.gov
Joint Aviation Authorities (JAA): www.jaa.nl
Radio Technical Corporation of America (RTCA): www.rtca.org
Society of Automotive Engineers (SAE): www.sae.org

Section II

Implementation

8 Fault-Tolerant Avionics

Ellis F. Hitt
Strategic Systems Solutions

Dennis Mulcare
Retired

8.1 Introduction

Fault-tolerant designs are required to ensure safe operation of digital avionics systems performing flight-critical functions. This chapter discusses the motivation for fault-tolerant designs and the many different design practices evolving to implement a fault-tolerant system. The designer needs to make sure the fault tolerance requirements are fully defined to select the design concept to be implemented from the alternatives available. The requirements for a fault-tolerant system include performance, dependability, and a methodology to assure that the design, when implemented, meets all requirements. The requirements must be documented in a specification of the intended behavior of a system, specifying the tolerances imposed on the various outputs from the system (Anderson and Lee, 1981). Development of the design proceeds in parallel with the development of the methods of assurance to validate that the design meets all requirements, including the fault tolerance. The chapter concludes with references to further reading in this developing field.

A fault-tolerant system provides continuous, safe operation in the presence of faults. A fault-tolerant avionics system is a critical element of flight-critical architectures, which include the fault-tolerant computing system (hardware, software, and timing), sensors and their interfaces, actuators, elements, and data communication among the distributed elements. The fault-tolerant avionics system ensures integrity of output data used to control the flight of the aircraft, whether operated by the pilot or by autopilot. A fault-tolerant system must detect errors caused by faults, assess the damage caused by the fault, recover from the error, and isolate the fault. It is generally not economical to design and build a system that is capable of tolerating all possible faults in the universe. The faults the system is to be designed to tolerate must be defined based on analysis of requirements including the probability of each fault occurring, and the impact of not tolerating the fault.

A user of a system may observe an error in its operation that is the result of a fault being triggered by an event. Stated another way, a fault is the cause of an error, and an error is the cause of a failure. A mistake made in designing or constructing a system can introduce a fault into the design of the system, either because of an inappropriate selection of components or because of inappropriate (or missing) interactions between components. On the other hand, if the design of a system is considered to be correct, then an erroneous transition can occur only because of a failure of one of the components of the system. Design faults require more powerful fault-tolerance techniques than those needed to cope with component faults. Design faults are unpredictable; their manifestation is unexpected, and they generate unanticipated errors. In contrast, component faults can often be predicted, their manifestation is expected, and they produce errors that can be anticipated (Anderson and Lee, 1981).

In a non-fault-tolerant system, diagnosis is required to determine the cause of the fault that was observed as an error. Faults in avionics systems are of many types; they generally can be classified as hardware, software, or timing related. Faults can be introduced into a system during any phase of its life cycle, including requirements definition, design, production, or operation.

In the 1960s, designers strived to achieve highly reliable safe systems by avoiding faults or masking faults. The Apollo guidance and control system employed proven, highly reliable components and triple modular redundancy (TMR) with voting to select the correct output. Improvements in hardware reliability, and our greater knowledge of faults and events that trigger them, has led to improved design methods for affordable fault-tolerant systems.

In any fault-tolerant system, the range of potential fault conditions that must be accommodated is quite large; enumerating all such possibilities is a vital, yet formidable, task in validating the system's airworthiness or its readiness for deployment. The resultant need to handle each such fault condition prompts attention to the various assurance-oriented activities that contribute to certification system airworthiness.

8.1.1 Motivation

Safety is of primary importance to the economic success of the aviation system. The designer of avionics systems must assure that the system provides the required levels of safety to passengers, aircrew, and maintenance personnel. Fault-tolerant systems are essential given the trend toward increasingly complex digital systems.

Many factors necessitate fault tolerance in systems that perform functions that must be sustained without significant interruption. In avionics systems, such functions are often critical to continued safe flight or to the satisfactory conduct of a mission; hence the terms flight-critical and mission-critical. The first compelling reality is that physical components are nonideal; that is, they are inescapably disposed to physical deterioration or failure. Clearly, then, components inherently possess a finite useful life, which varies with individual instances of the same component type. At some stage, then, any physical component will exhibit an abrupt failure or excessive deterioration such that a fault may be detected at some level of system operation. Due to low demand for high reliability, military-qualified integrated circuits (ICs), the avionics industry and their customers have elected to use commercial off-the-shelf (COTS) ICs in their designs for new avionics used in new aircraft and retrofit into existing aircraft to upgrade their functional capability. These COTS ICs have increasing clock frequencies, decreasing process geometries, and decreasing power supply voltages. Studies (Constantinescu, 2002) of these trends conclude that the dependability of COTS ICs is decreasing, and the expected life of ICs is also decreasing (Driscoll, et al, 2004) with the anticipated working life in the range of 5 to 10 years.

The second contributing factor to physical faults is the nonideal environment in which an avionics system operates. Local vibrations, humidity, temperature cycling, electrical power transients, electromagnetic interference, and so on tend to induce stress on the physical component, which may cause abrupt failure or gradual deterioration. The result may be a transient or a permanent variation in output, depending on the nature and severity of the stress. The degree of induced deterioration encountered may profoundly influence the useful life of a component. Fortunately, design measures can be taken to reduce susceptibility to the various environmental effects. Accordingly, a rather comprehensive development approach is needed for system dependability, albeit fault tolerance is the most visible aspect because it drives the organization and logic of the system architecture.

The major factor necessitating fault tolerance is design faults. Tolerance of design faults in hardware and software and the overall data flow is required to achieve the integrity needed for flight-critical systems. Relying on the hardware chip to produce correct output when there is no physical failure is risky, as demonstrated by the design error discovered in the floating point unit of a high-performance microprocessor in wide use. Because of the difficulty in eliminating all design faults, dissimilar redundancy is used to produce outputs that should be identical even though computed by dissimilar computers. Use of dissimilar redundancy is one approach to tolerating common-mode failures (CMFs). A CMF occurs when copies of a redundant system suffer faults nearly simultaneously, generally due to a single cause (Lala, 1994).

8.1.2 Definitional Framework

A digital avionics system is a "hard real-time" system producing time-critical outputs that are used to control the flight of an aircraft. These critical outputs must be dependable — both reliable and safe. Reliability has many definitions and is often expressed as the probability of not failing; another definition is the probability of producing a "correct" output (Vaidya and Pradhan, 1993). Safety has been defined as the probability that the system output is correct or the error in the output is detectable (Vaidya and Pradhan, 1993). Correctness is the requirement that the output of all channels agree bit-for-bit under no-fault conditions (Lala, 1994). Another design approach, approximate consensus, considers a system to be correct if the outputs agree within some threshold. Both approaches are in use.

Hardware component faults are often classified by extent, value, and duration (Avizienis, 1976). Extent applies to whether the errors generated by the fault are localized or distributed; value indicates whether the fault generates fixed or varying erroneous values; duration refers to whether the fault is transient or permanent. Several studies have shown that permanent faults cause only a small fraction of all detected errors, as compared with transient faults (Sosnowski, 1994). A recurring transient fault is often referred to as intermittent (Anderson and Lee, 1981). Figure 8.1 depicts these classifications in the tree of faults.

Origin faults may result from a physical failure within a hardware component of a system or may result from human-made faults. System boundary internal faults are those parts of the system's state

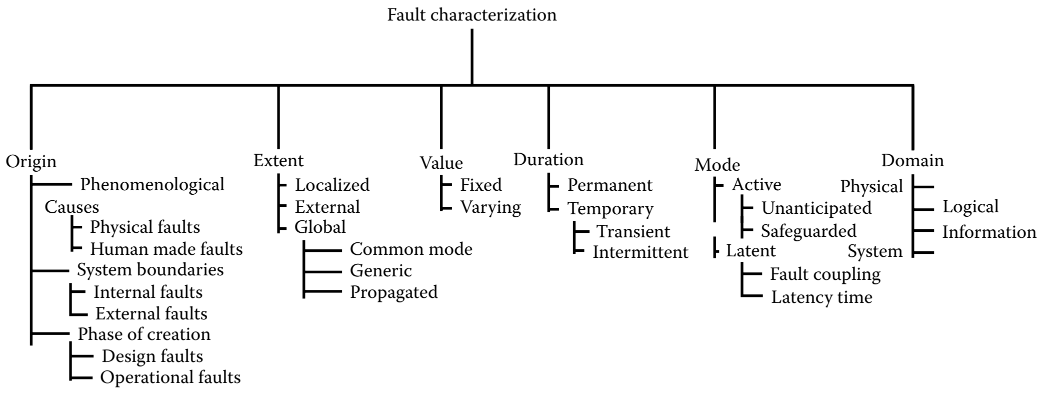


FIGURE 8.1 Fault classification.

which, when invoked by the computation activity, will produce an error, while external faults result from system interference caused by its physical environment, or from system interaction with its human environment. Origin faults classified by the time phase of creation include design faults resulting from imperfections that arise during the development of the system (from requirements specification to implementation), subsequent modifications, the establishment of procedures for operating or maintaining the system, or operational faults that appear during the system operation (Lala and Harper, 1994).

A fault is in the active mode if it yields an erroneous state, either in hardware or software; that is, a state that differs from normal expectations under extant circumstances. Alternatively, a fault is described as latent when it is not yielding an erroneous state. Measures for error detection that can be used in a fault-tolerant system fall into the following broad classification (discussed in Section 8.4) (Anderson and Lee, 1981):

1. Replications checks
2. Timing checks
3. Reversal checks
4. Coding checks
5. Reasonableness checks
6. Structural checks
7. Diagnostic checks

During the development process, it is constructive to maintain a perspective regarding the fault attributes of Domain and Value in Figure 8.1. Basically, Domain refers to the universe of layering of fault abstractions that permit design issues to be addressed with a minimum of distraction. Value simply refers to whether the erroneous state remains fixed, or whether it indeterminately fluctuates. While proficient designers tend to select the proper abstractions to facilitate particular development activities, the following associated fault domains should be explicitly noted:

- Physical — elemental *physical failures* of hardware components (underlying short, open, ground faults)
- Logical — manifested *logical faults* per device behavior (stuck-at-one, stuck-at-zero, inverted)
- Informational — exhibited *error states* in interpreted results (incorrect value, sign change, parity error)
- System — resultant *system failure* provides unacceptable service (system crash, deadlock, hardover)

These fault domains constitute levels of design responsibilities and commitments as well as loci of fault tolerance actions per se. Thus, fault treatment and, in part, fault containment, are most appropriately addressed in the physical fault domain. Similarly, hardware fault detection and assessment are most readily managed in the logical fault domain, where the fixed or fluctuating nature of the erroneous value(s) refines the associated fault identification mechanism(s). Lastly, error recovery and perhaps some fault containment are necessarily addressed in the informational fault domain and service continuation in the system fault domain.

For safety-critical applications, physical hardware faults no longer pose the major threat to dependability. The dominant threat is now CMFs, which result from faults that affect more than one fault containment region at the same time, generally due to a common cause. Approaches used in tolerating CMFs include fault avoidance, fault removal through test and evaluation or via fault insertion, and fault tolerance implemented using exception handlers, program checkpointing, and restart (Lala, 1994). Table 8.1 presents a classification of common mode faults; the X indicates the possible combinations of faults that must be considered that are not intentional faults.

Physical, internal, and operational faults can be tolerated by using hardware redundancy. All other faults can affect multiple fault-containment regions simultaneously. Four sources of common-mode failures need to be considered:

TABLE 8.1 Classifications of Common Mode Faults

Phenomenological Cause		System Boundary		Phase of Creation		Duration		
Physical	Human Made	Internal	External	Design	Operational	Permanent	Temporary	Common Mode Fault Label
X			X		X		X	Transient (External) CMF
X			X		X	X		Permanent (External) CMF
	X	X		X			X	Intermittent (Design) CMF
	X	X		X		X		Permanent (Design) CMF
	X		X		X		X	Interaction CMF

1. Transient (External) Faults, which are the result of temporary interference to the system from its physical environment such as lightning, High-Intensity Radio Frequencies (HIRF), heat, etc.
2. Permanent (External) Faults, which are the result of permanent system interference caused by its operational environment such as heat, sand, salt water, dust, vibration, shock, etc.
3. Intermittent (Design) Faults, which are introduced due to imperfections in the requirements specifications, detailed design, implementation of design, and other phases leading up to the operation of the system
4. Permanent (Design) Faults are introduced during the same phases as intermittent faults but manifest themselves permanently (Lala, 1994)

An **elemental physical failure**, which is an event resulting in component malfunction, produces a physical fault. These definitions are reflected in Figure 8.2, a state transition diagram that portrays four fault status conditions and associated events in the absence of fault tolerance. Here, for example, a Latent Fault Condition transitions to an Active Fault Condition due to a Potentiating Event. Such an event might be a functional mode change that caused the fault region to be exercised in a revealing way. Following the incidence of a sufficiently severe active fault from which spontaneous recovery is not forthcoming, a **system failure** event occurs wherein expected functionality can no longer be sustained. If the effects of a particular active fault are not too debilitating, a system may continue to function with some degradation in services. Fault tolerance can, of course, forestall both the onset of system failure and the expectation of degraded services.

The Spontaneous Recovery Event in Figure 8.2 indicates that faults can sometimes be transient in nature when a fault vanishes without purposeful intervention. This phenomenon can occur after an external disturbance subsides or an intermittent physical aberration ceases. A somewhat similar occurrence is provided through the Incidental Fault Remission Event in Figure 8.2. Here, the fault does not vanish, but rather spontaneously reverts from an active to a latent mode due to the cessation or removal of fault excitation circumstances. Table 8.2 complements Figure 8.2 by clarifying these fault categories.

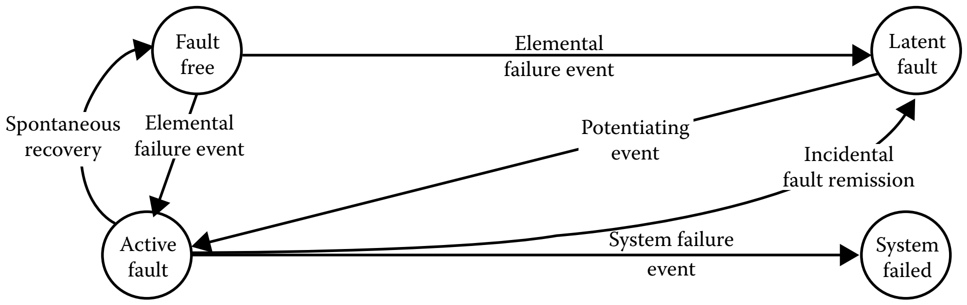


FIGURE 8.2 Hardware states (no corrective action).

TABLE 8.2 Delineation of Fault Conditions

Recovered Mode	Latent Mode	Active Mode
Spontaneous recovery following disruption	—	Erroneous state induced by transient disturbance
or		or
Marginal physical fault recovery	—	Passing manifestation of marginal fault
—	Hard physical fault remission	Passing manifestation of hard fault
	or	or
	Hard physical fault latency	Persistent manifestation of hard fault

Although these transient fault modes are thought to account for a large proportion of faults occurring in deployed systems, such faults may nonetheless persist long enough to appear as permanent faults to the system. In many cases then, explicit features must be incorporated into a system to ensure the timely recovery from faults that may induce improper or unsafe system operation.

Three classes of faults are of particular concern because their effects tend to be global regarding extent, where "global" implies impact on redundant components present in fault-tolerant systems. A common mode fault is one in which the occurrence of a single physical fault at a one particular point in a system can cause coincident debilitation of all similar redundant components. This phenomenon is possible where there is a lack of protected redundancy, and the consequence would be a massive failure event. A generic fault is a development fault that is replicated across similar redundant components such that its activation yields a massive failure event like that due to a common mode fault. A propagated fault is one wherein the effects of a single fault spread out to yield a compound erroneous state. Such fault symptoms are possible when there is a lack of fault containment features. During system development, particular care must be exercised to safeguard against these classes of global faults, for they can defeat fault-tolerance provisions in a single event. Considerable design assessment is therefore needed, along with redundancy provisions, to ensure system dependability.

Byzantine faults are of great concern with the move away from ICs built to avionics standards to the use of COTS ICs in avionics systems. A Byzantine fault presents different symptoms to different observers. A Byzantine failure is the loss of a system service due to a Byzantine fault (Driscoll et al., 2004). If a system uses a voting mechanism to reach consensus, Byzantine faults can cause Byzantine failures. Safety critical functions for avionics are required to have a failure probability of $<10^{-9}$ per flight hour. Research has identified Byzantine faults due to a digital signal stuck at $^1/_2$ a CMOS open, and slightly-off-specification (SOS) transmission timing as just a few faults that designers need to consider to achieve the required system dependability.

8.1.3 Dependability

Dependability is an encompassing property that enables and justifies reliance upon the services of a system. Hence, dependability is a broad, qualitative term that embodies the aggregate nonfunctional attributes, or "-ilities," sought in an ideal system, especially one whose continued safe performance is critical. Thus, attributes like safety, reliability, availability, and maintainability, which are quantified using conditional probability formulas, can be conveniently grouped as elements of dependability. As a practical matter, however, dependability usually demands the incorporation of fault tolerance into a system to realize quantitative reliability or availability levels. Fault tolerance, moreover, may be needed to achieve maintainability requirements, as in the case of on-line maintenance provisions.

For completeness sake, it should be noted, as depicted in Figure 8.3, that the attainment of dependability relies on fault avoidance and fault reduction, as well as on fault tolerance. This figure is complemented by Table 8.3, which emphasizes the development activities, such as analyzing fault possibilities during design to minimize the number and extent of potential fault cases. This activity is related to the criteria of containment in Figure 8.3 in that the analysis should ensure that both the number and propagation of fault cases are contained. The overall notion here is to minimize the number of fault

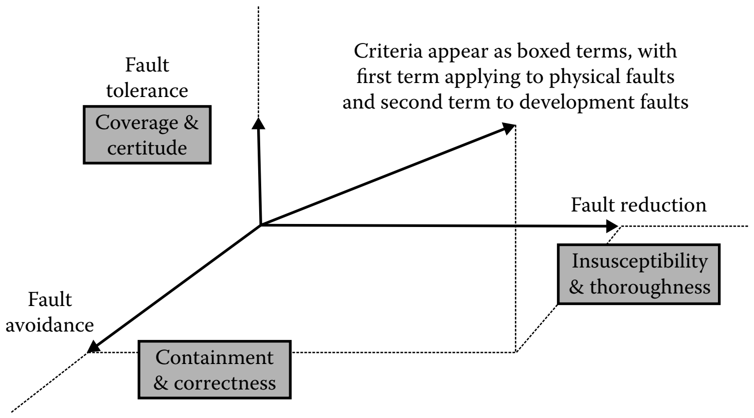

Fault tolerance
Coverage & certitude
Criteria appear as boxed terms, with first term applying to physical faults and second term to development faults
Fault reduction
Insusceptibility & thoroughness
Fault avoidance
Containment & correctness

FIGURE 8.3 Dependability.

TABLE 8.3 Ensuring Dependability

	Physical Faults	Development Faults
Fault avoidance	Minimize by analysis	Prevent by rigor development errors
Fault reduction	Selectively reduce the incidence of faults	Remove by verification
Fault tolerance	Ensure by redundancy	Testing

Note: Both physical and developmental fault handling may be present, but any deficiency revealed is a developmental defect.

possibilities, to reduce the prospects of their occurrences, and to ensure the safe handling of those that do happen in deployed systems.

8.1.4 Fault Tolerance Options

System reliability requirements derive from the function criticality level and maximum exposure time. A flight-critical function is one whose loss might result in the loss of the aircraft itself and possibly the persons on board as well. In the latter case, the system is termed safety-critical. Here, a distinction can be made between a civil transport, where flight-critical implies safety-critical, and a combat aircraft. The latter admits to the possibility of the crew ejecting from an unflyable aircraft, so its system reliability requirements may be lower. A mission-critical function is one whose loss would result in the compromising or aborting of an associated mission. For avionics systems, a higher cost usually is associated with the loss of an aircraft than with the abort of a mission (an antimissile mission to repel a nuclear weapon could be an exception). Thus, a full-time flight-critical system would normally pose much more demanding reliability requirements than a flight-phase mission-critical system. Next, the system reliability requirements coupled with the marginal reliabilities of system components influence the design of the fault tolerant system architecture and determine the level of redundancy to ensure system survivability. To ensure meeting system reliability requirements then, the evolution of a fault-tolerant design must be based on interplay between design configuration commitments and substantiating analyses.

For civil transport aircraft, the level of redundancy for a flight-phase critical function like automatic all-weather landing is typically single fail-operational, meaning that the system should remain operable after any potential single elemental failure. Alternatively, a full-time critical function like fly-by-wire primary flight controls is typically double fail-operational. It should be noted that a function that is not flight-critical itself can have failure modes that threaten safety of flight. In the case of active controls to alleviate structural loads due to gusts or maneuvering, the function would not be critical where the purpose is merely to reduce structural fatigue effects. If the associated flight control surfaces have the authority during a hardover or runaway fault case to cause structural damage, then such a failure mode is safety-critical. In such cases, the failure modes must be designed to be fail-passive, which precludes any active mode failure effect like a hardover control surface. A failure mode that exhibits active behavior can still be failsafe, however, if the rate and severity of the active effects are well within the flight crews' capability to manage safely.

Complexity of avionics communication, navigation, and surveillance functions increases as the avionics system on a single aircraft functions as part of a net-centric operational system involving a distribution of functions across other aircraft and ground and space-based systems. Cooperative exchange of correct time- and position-referenced information necessitates that the system have the capability to tolerate faults that arise internal to the avionics system on an aircraft as well as faults that occur external to the aircraft. These faults may originate in any node (aircraft, ground, or space-based) in the network and include network operation faults as well as faulty information. Net-centric operations options for ensuring a dependable fault tolerant system include fault avoidance by intentionally disabling vulnerable network elements when a threat is detected or predicted and fault elimination by reconfiguring the system to replace faulty components with known good components (Knight et al., 2002). Fault tolerant network survivability involves a control loop structure in which the network state is sensed and analyzed and required changes effected through reconfiguration of the network (Hill and Knight, 2003). The level of integrity required for net-centric operations information flow between multiple aircraft operating in highly congested airspace creates the need for designers to include not only fault-tolerance, but also to ensure that network survivability and resilience is embedded in the avionics system design.

8.1.5 Flight Systems Evolution

Beginning in the 1970s, NASA's F-8 digital fly-by-wire (DFBW) flight research program investigated the replacement of the mechanical primary flight control systems with electronic computers and electrical signal paths. The goal was to explore the implementation technology and establish the practicality of replacing mechanical linkages to the control surfaces with electrical links, thereby yielding significant weight and maintenance benefits. The F-8 DFBW architecture relied on bit-wise exact consensus of the outputs of redundant computers for fault detection and isolation (Lala and Harper, 1994).

The Boeing 747, Lockheed L-1011, and Douglas DC-10 used various implementations to provide the autoland functions, which required a probability of failure of $<10^{-9}$ during the landing. The 747 used triply redundant analog computers, the L-1011 used digital computers in a dual-dual architecture, and the DC-10 used two identical channels, each consisting of dual-redundant fail-disconnect analog computers for each axis. Since that time, the Airbus A-320 uses a full-time DFBW flight control system. It uses software design diversity to protect against common-mode failures. The Boeing 777 flight control computer architecture uses a three-by-three matrix of nine processors of three different types. Multiversion software is also used.

The Airbus A-380 and Boeing 787 avionics architecture use the Avionics Full Duplex Switched Ethernet (A664-P7)/ARINC 664 for data communication between the Integrated Modular Avionics cabinets. The A-380 and Boeing 787 avionics vendors use the ARINC Specification 653, "Avionics Application Software Standard Interface" (see *Avionics: Elements, Software and Functions*, Chapter 14). The 787 uses a Common Core System (CCS) that is similar to the Airplane Information Management System (AIMS) of the 777 but hosts more functions and provides greater communication bandwidth. The CCS includes the Common Computing Resource (CCR) cabinet, the A664-P7, and remote data concentrators.

The move away from federated avionics architectures to the integrated architectures of the A-380 and the 787 may also move dedicated functions such as the flight data acquisition unit and flight data recorder to a distributed implementation. Integrated Vehicle Health Management (IVHM) functions detect, identify, log, and isolate and contain faults. Some IVHM designs include recovery and repair to a nominal system state (Scandura and Garcia-Galan, 2004). The implementation of a dependable fault tolerant design, IVHM, and digital flight data recorder functions requires an integrated design approach to ensure that the errors observed do not result in the IVHM and fault-tolerant system detecting different faults for the same observed error. Various errors and faults are displayed to the aircrew, and the aircrew is trained to implement various actions based on the displayed information.

8.1.6 Design Approach

The design of a dependable fault-tolerant avionics system must be based on proven systems engineering processes and tools. The designers must identify all of the functions and the information and data flow between processes that implement these functions. Functions involving fault detection, identification, isolation, and recovery must use tools that accurately document the allocation of the processes to hardware, software, and the human (aircrew and maintenance personnel). The design, development, integration, and testing must trace to these allocations.

It is virtually impossible to design a complex avionics system that will tolerate all possible faults. Faults can include both permanent and transient faults and hardware and software faults, and they can occur singularly or concurrently. Timing faults directly trace to the requirement of real-time response within a few milliseconds and may be attributable to both hardware and software contributions to data latency as well as incorrect data. Incorrect data faults will be of increasing importance as the demands for more efficient use of air space reduces spacing between aircraft *en route* and in the terminal areas. Required navigation performance and four-dimensional (4-D) (aircraft position as a function of time) control of flight increases the demand for accurate transmission of aircraft state information when interrogated by air traffic control or another aircraft. Other data faults of concern are use of databases containing invalid data, such as terrain and man-made object locations, and flight plan information used by the flight management system.

The implementation of fault tolerance entails increased system overhead, complexity, and validation challenges. The overhead, which is essentially increased system resources and associated management activities, lies in added hardware, communications, and computational demands. The expanded complexity derives from more intricate connectivity and dependencies among both hardware and software elements; the greater number of system states that may be assumed; and appreciably more involved logic necessary to manage the system. Validation challenges are posed by the need to identify, assess, and confirm the capability to sustain system functionality under a broad range of potential fault cases. Hence, the design approach taken for fault-tolerant avionics must attain a balance between the costs incurred in implementing fault tolerance and the degree of dependability realized.

Design approach encompasses system concepts, development methodology, and fault tolerance elements. The system concepts derive largely from the basic fault-tolerance options introduced in Section 8.1.4, with emphasis on judicious combinations of features that are adapted to given application attributes. The development methodology reduces to mutually supportive assurance-driven methods that propagate consistency, enforce accountability, and exact high levels of confidence in system dependability. Fault tolerance design elements tend to unify the design concepts and methods by providing an orderly pattern of system organization and evolution. The following are fault tolerance elements, which, in general, should appear in some form in any fault-tolerant system:

- Error Detection — recognition of the incidence of a fault
- Damage Assessment — diagnosis of the locus of a fault
- Fault Containment — restriction of the scope of effects of a fault
- Error Recovery — restoration to a restartable error-free state
- Service Continuation — sustained delivery of system services
- Fault Treatment — repair of fault

A fundamental design parameter that spans these elements and constrains their mechanization is the granularity of fault handling. Basically, the detection, isolation, and recovery from a fault should occur at the same level of modularity to achieve a balanced and coherent design. In general, it is not beneficial or justified to discriminate or contain a fault at a level lower than that of the associated fault-handling boundary. There may be exceptions, however — especially in the case of fault detection, where a finer degree of granularity may be employed to take advantage of built-in test features or to reduce fault latency.

Depending on the basis for its instigation, fault containment may involve the inhibition of damage propagation of a physical fault and the suppression of an erroneous computation. Physical fault containment has to be designed into the hardware, and software error-state containment has to be designed into the applications software. In most cases, an erroneous software state must be corrected because of applications program discrepancies introduced during the delay in detecting a fault. This error recovery may entail resetting certain data object values and backtracking a control flow path in an operable processor. At this point, the readiness of the underlying architecture, including the coordination of operable components, must be ensured by the infrastructure. Typically, this activity relies heavily on system management software for fault tolerance. Service continuation, then, begins with the establishment of a suitable applications state for program restart. In an avionics system, this sequence of fault-tolerance activities must take place rather quickly because of real-time functional demands. Accordingly, an absolute time budget must be defined with tolerances for worst-case performance for responsive service continuation.

8.2 System-Level Fault Tolerance

8.2.1 General Mechanization

As discussed in Section 8.1.2, system failure is the loss of system services or expected functionality. In the absence of fault tolerance, a system may fail after just a single crucial fault. This kind of system, which in effect is zero fail-operational, would be permissible for noncritical functions. Figure 8.2, moreover, characterizes this kind of system in that continued service depends on spontaneous remission of an active fault or a fault whose consequences are not serious enough to yield a system failure.

Where the likelihood of continued service must be high, redundancy can be incorporated to ensure system operability in the presence of any permanent faults. Such fault-tolerant systems incorporate an additional fault status state, namely that of recovery, as shown in Figure 8.4. Here, system failure occurs

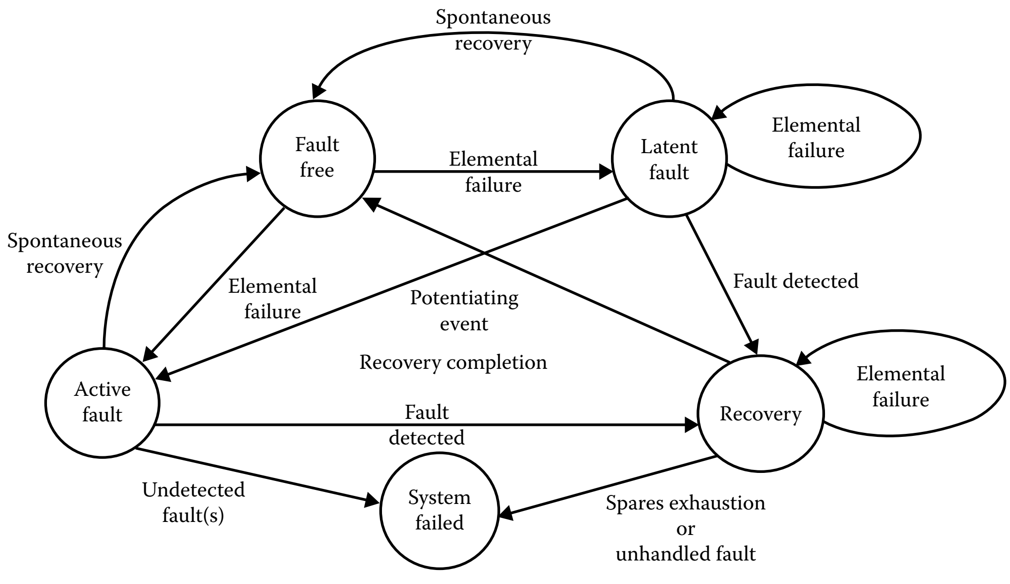


FIGURE 8.4 Hardware states (with corrective action).

only after the exhaustion of spares or an unhandled severe fault. The aforementioned level of redundancy can render it extremely unlikely that the spares will be exhausted as a result of hardware faults alone. An unhandled fault could occur only as a consequence of a design error, like the commission of a generic error wherein the presence of a fault would not even be detected.

This section assumes a system-level perspective and examines fault-tolerant system architectures and examples from this perspective. Still, these examples embody and illuminate general system-level principles of fault tolerance. Particular prominence is directed toward flight control systems, for they have motivated and pioneered much of the fault-tolerance technology. In the past, such systems have been functionally dedicated, thereby providing a considerable safeguard against malfunction due to extraneous causes. With the increasing prevalence of integrated avionics, however, the degree of function separation is irretrievably reduced. This is actually not altogether detrimental; more avionics functions than ever are critical, and a number of benefits accrue from integrated processing. Furthermore, the system developer can exercise prerogatives that afford safeguards against extraneous faults.

8.2.2 Redundancy Options

Fault tolerance is usually based on some form of redundancy to extend system reliability through the invocation of alternative resources. The redundancy may be in hardware, software, time, or combinations thereof. There are three basic types of redundancy in hardware and software: static, dynamic, and hybrid. Static redundancy masks faults by taking a majority of the results from replicated tasks. Dynamic redundancy takes a two-step procedure for detection of and recovery from faults. Hybrid redundancy is a combination of static and dynamic redundancy (Shin and Hagbae, 1994).

In general, much of this redundancy resides in additional hardware components. The addition of components reduces the mean-time-between-maintenance actions, because there are more electronics that can, and at some point will, fail. Since multiple, distinct faults can occur in fault-tolerant systems, there are many additional failure modes that have to be evaluated in establishing the airworthiness of the total system. Weight, power consumption, and cooling, are examples of other penalties for component redundancy. Other forms of redundancy also present system management overhead demands, like computational capacity to perform software-implemented fault tolerance tasks. Like all design undertakings, the realization of fault-tolerance presents trade-offs and the necessity for design optimization. Ultimately, a balanced, minimal, and valid table design must be sought that demonstrably provides the safeguards and fault survival margins appropriate to the subject application.

A broad range of redundancy implementation options exist to mechanize desired levels and types of fault tolerance. Figure 8.5 presents a taxonomy of redundancy options that may be invoked in appropriate combinations to yield an encompassing fault-tolerance architecture. This taxonomy indicates the broad

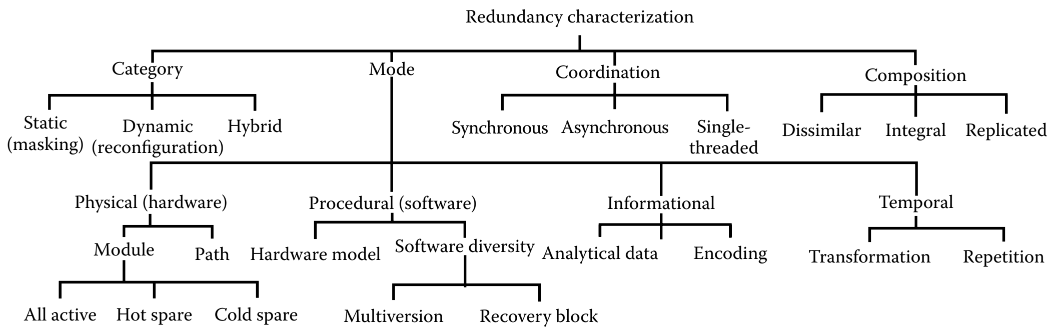


FIGURE 8.5 Redundancy classifications.

TABLE 8.4 Fault Avoidance Techniques and Tools

Technique
Use of mature and formally verified components
Conformance to standards
Formal methods
Design automation
Integrated formal methods and VHDL design methodology
Simplifying abstractions
Performance common-mode failure avoidance
Software and hardware engineering practice
Design diversity

range of redundancy possibilities that may be invoked in system design. Although most of these options are exemplified or described in later paragraphs, it may be noted briefly that redundancy is characterized by its category, mode, coordination, and composition aspects. Architectural commitments have to be made in each aspect. Thus, a classical system might, for example, employ fault-masking using replicated hardware modules that operate synchronously. At a lower level, the associated data buses might use redundancy encoding for error detection and correction. To safeguard against a generic design error, a backup capability might be added using a dissimilar redundancy scheme.

Before considering system architectures *per se*, however, key elements of Figure 8.5 need to be described and exemplified. Certain of these redundancy implementation options are basic to formulating a fault-tolerant architecture, so their essential trade-offs need to be defined and explored. In particular, the elements of masking versus reconfiguration, active versus standby spares, and replicated versus dissimilar redundancy are reviewed here.

No unifying theory has been developed that can treat CMFs the same way the Byzantine resilience (BR) treats random hardware or physical operational faults. Three techniques — fault-avoidance, fault-removal, and fault-tolerance — are the tools available to design a system tolerant of CMFs. The most cost-effective phase of the total design and development process for reducing the likelihood of CMFs is the earliest part of the program. Table 8.4 presents fault avoidance techniques and tools that are being used (Lala and Harper, 1994).

Common mode fault removal techniques and tools include design reviews, simulation, testing, fault injection, and a rigorous quality control program. Common mode fault tolerance requires error detection and recovery. It is necessary to corroborate the error information across redundant channels to ascertain which recovery mechanism (i.e., physical fault recovery or common mode failure recovery) to use. Recovery from CMF in real time requires that the state of the system be restored to a previously known correct point from which the computational activity can resume (Lala and Harper, 1994).

8.2.3 Architectural Categories

As indicated in Figure 8.5, the three categories of fault-tolerant architectures are masking, reconfiguration, and hybrid.

8.2.3.1 Fault Masking

The masking approach is classical per von Neumann's TMR concept, which has been generalized for arbitrary levels of redundancy. The TMR concept centers on a voter that, within a spares exhaustion constraint, precludes a faulted signal from proceeding along a signal path. The approach is passive in that no reconfiguration is required to prevent the propagation of an erroneous state or to isolate a fault.

Modular avionics systems consisting of multiple identical modules and a voter require a trade-off of reliability and safety. A "module" is not constrained to be a hardware module; a module represents an entity capable of producing an output. When safety is considered along with reliability, the module design affects both safety and reliability. It is usually expected that reliability and safety should improve with added redundancy. If a module has built-in error detection capability, it is possible to increase both reliability and safety with the addition of one module providing input to a voter. If no error detection capability exists at the module level, at least two additional modules are required to improve both reliability and safety. An error control arbitration strategy is the function implemented by the voter to decide what the correct output is, and when the errors in the module outputs are excessive so that the correct output cannot be determined, the voter may put out an unsafe signal. Reliability and safety of an n-module safe modular redundant (nSMR) architecture depend on the individual module reliability and on the particular arbitration strategy used. No single arbitration strategy is optimal for improving both reliability and safety. Reliability is defined as the probability the voter's data output is correct and the voter does not assert the unsafe signal. As system reliability and safety are interrelated, increasing system reliability may result in a decrease in system safety and vice versa (Vaidya and Pradhan, 1993).

Voters that use bit-for-bit comparison have been employed when faults consist of arbitrary behavior on the part of failed components, even to the extreme of displaying seemingly intelligent malicious behavior (Lala and Harper, 1994). Such faults have been called Byzantine faults. Requirements levied on an architecture tolerant of Byzantine faults (referred to as Byzantine-resilient [BR]) comprise a lower bound on the number of fault containment regions, their connectivity, their synchrony, and the utilization of certain simple information exchange protocols. No *a priori* assumptions about component behavior are required when using bit-for-bit comparison. The common mode failure is the dominant contributor to failure of correctly designed BR system architecture.

Fault effects must be masked until recovery measures can be taken. A redundant system must be managed to continue correct operation in the presence of a fault. One approach is to partition the redundant elements into individual fault containment regions (FCRs). An FCR is a collection of components that operates correctly regardless of any arbitrary logical or electrical fault outside the region. A fault containment boundary requires the hardware components be provided with independent power and clock sources. Interfaces between FCRs must be electrically isolated. Tolerance to such physical damage as a weapons hit necessitates a physical separation of FCRs such as different avionics bays. In flight control systems, a channel may be a natural FCR. Fault effects manifested as erroneous data can propagate across FCR boundaries; therefore, the system also must provide error containment by using voters at various points in the processing, including voting on redundant inputs, voting the result of control law computations, and voting at the input to the actuator. Masking faults and errors provides correct operation of the system with the need for immediate damage assessment, fault isolation, and system reconfiguration (Lala and Harper, 1994).

8.2.3.2 Reconfiguration

Hardware interlocks provide the first level of defense prior to reconfiguration or the use of the remaining nonfaulty channels. In a triplex or higher redundancy system, the majority of channels can disable the output of a failed channel. Prior to taking this action, the system will determine whether the failure is permanent or transient. Once that determination is made, the next step is to ascertain what functions are required for the remainder of the mission and whether the system needs to invoke damage assessment, fault isolation, or reconfiguration of the remaining system assets. The designer of a system required for long-duration missions may implement a design with reconfiguration capability.

8.2.3.3 Hybrid Fault Tolerance

Hybrid fault tolerance uses hybrid redundancy, which is a combination of static and dynamic redundancy — masking, detection, and recovery that may involve reconfiguration. A system using hybrid redundancy will have N-active redundant modules, as well as spare (S) modules. A disagreement detector detects if the output of any of the active modules is different from the voter output. If a module output disagrees with the voter, the switching circuit replaces the failed module with a spare. A hybrid (N,S) system cannot have more than (N−1)/2 failed modules at a time in the core, or the system will incorrectly switch out the good module when two out of three have failed.

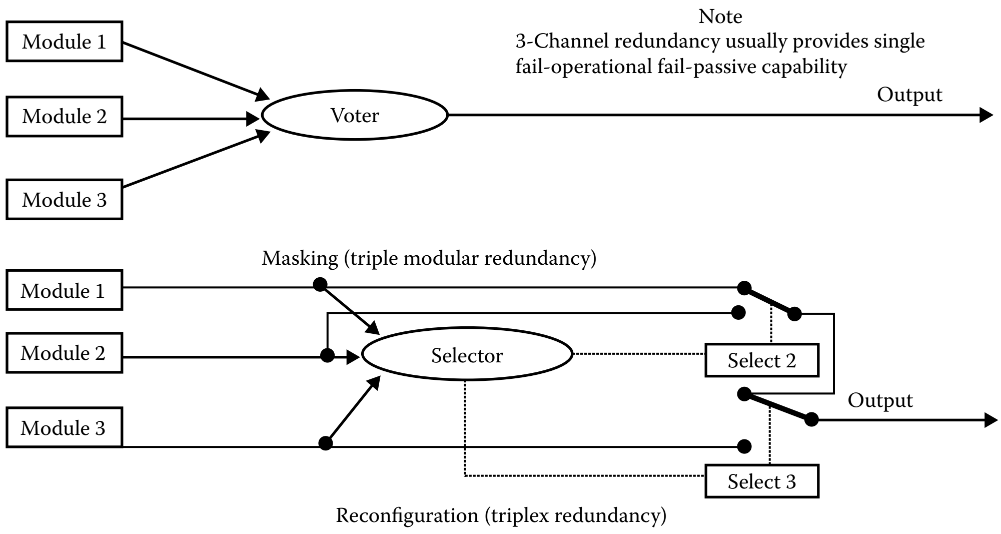

Note
3-Channel redundancy usually provides single fail-operational fail-passive capability
Module 1
Module 2
Module 3
Voter
Output
Masking (triple modular redundancy)
Module 1
Module 2
Module 3
Selector
Select 2
Select 3
Output
Reconfiguration (triplex redundancy)

FIGURE 8.6 Masking vs. reconfiguration.

Hybrid fault tolerance employs a combination of masking and reconfiguration, as noted in Section 8.2.3. The intent is to draw on strengths of both approaches to achieve superior fault tolerance. Masking precludes an erroneous state from affecting system operation and thus obviates the need for error recovery. Reconfiguration removes faulted inputs to the voter so that multiple faults cannot defeat the voter. Masking and reconfiguration actions are typically implemented in a voter-comparator mechanism, which is discussed in Section 8.3.1.

Figure 8.6 depicts a hybrid TMR arrangement with a standby spare channel to yield double fail-operational capability. Upon the first active channel failure, it is switched out of the voter-input configuration, and the standby channel is switched in. Upon a second channel failure, the discrepant input to the voter is switched out. Only two inputs remain then, so a succeeding (third) channel failure can be detected but not properly identified by the voter *per se*. When a voter selects the lower of two remaining signals and precludes a hardover output, a persistent miscomparison results in a fail-passive loss of system function. An alternative double-fail operational configuration would forego the standby channel switching and simply employ a quadruplex voter. This architecture is actually rather prevalent in dedicated flight-critical systems like fly-by-wire (FBW) flight control systems. This architecture still employs reconfiguration to remove faulty inputs to the voter.

The fault tolerance design elements described in Section 8.1.6 are reflected in the fault-tolerant architecture in Figure 8.6 by way of annotations. For example, error detection is provided by the comparators; damage assessment is then accomplished by the reconfiguration logic using the various comparator states. Fault containment and service continuation are both realized through the voter, which also obviates the need for error recovery. Lastly, fault treatment is accomplished by the faulty path switching prompted by the reconfiguration logic. Thus, this simple example illustrates at a high level how the various aspects of fault tolerance can be incorporated into an integrated design.

8.2.4 Integrated Mission Avionics

In military applications, redundant installations in some form will be made on opposite sides of the aircraft to avoid loss of functionality from battle damage to a single installation. Vulnerability to physical damage also exists in the integrated rack installations being used on commercial aircraft. Designers must take these vulnerabilities into account in the design of a fault-tolerant system.

8.2.5 System Self Tests

Avionics system reliability analyses are conditional on assumptions of system readiness at dispatch. For lower-criticality systems, certain reductions in redundancy may sometimes be tolerable at dispatch. For full-time flight-critical systems, however, a fully operable system with all redundancy intact is generally assumed in a system reliability prediction. This assumption places appreciable demands on the coverage and confidence values of system preflight self-testing. Such a test is typically an end-to-end test that exercises all elements in a phased manner that would not be possible during flight. The fault tolerance provisions demand particular emphasis. For example, such testing deliberately seeks to force seldom-used comparator trips to ensure the absence of latent faults like passive hardware failures. Analysis of associated testing schemes and their scope of coverage is necessarily an ongoing design analysis task during development. These schemes must also include appropriate logic interlocks to ensure safe execution of the preflight test, for example, a weight-on-wheels interlock to preclude testing, except on the ground. Fortunately, the programming of system self-tests can be accomplished in a relatively complete and high-fidelity manner.

Because of the discrete-time nature of digital systems, all capacity is not used for application functions. Hence, periodic self-tests are possible for digital components like processors during flight. Also, the processors can periodically examine the health status of other system components. Such tests provide a self-monitoring that can reveal the presence of a discrepancy before error states are introduced or exceed detection thresholds. The lead time afforded by self-tests can be substantial because steady flight may not simulate comparator trips due to low-amplitude signals. The longer a fault remains latent, the greater the possibility that a second fault can occur; therefore, periodic self-tests can significantly enhance system reliability and safety by reducing exposure to coincident multiple fault manifestations.

Self-monitoring may be employed at still lower levels, but there is a trade-off as to the granularity of fault detection. This trade-off keys on fault detection and recovery response and on the level of fault containment selected. In general, fault containment delineates the granularity of fault detection unless recovery response times dictate faster fault detection that is best achieved at lower levels.

8.3 Hardware-Implemented Fault Tolerance: Fault-Tolerant Hardware Design Principles

8.3.1 Voter Comparators

Voter comparators are very widely used in fault-tolerant avionics systems, and they are generally vital to the integrity and safety of the associated systems. Because of the crucial role of voter comparators special care must be exercised in their development. These dynamic system elements, which can be implemented in software as well as hardware, are not as simple as they might seem. In particular, device integrity and threshold parameter settings can be problematic.

Certain basic elements and concerns apply over the range of voter-comparator variants. A conceptual view of a triplex voter-comparator is depicted in Figure 8.7. The voter here is taken to be a middle signal selector, which means that the intermediate level of three inputs is selected as the output. The voter section precedes the comparators because the output of the voter is an input to each comparator. Basically, the voter output is considered the standard of correctness, and any input signal that persists in varying too much from the standard is judged to be erroneous.

In Figure 8.7, the respective inputs to each of the signal paths is an amplitude-modulated pulse train, as is normal in digital processing. Each iteration of the voter is a separate selection, so each voter output is apt to derive from any input path. This is seen in Figure 8.8, where the output pulse train components are numbered per the input path selected at each point in time. At each increment of time, the voter output is applied to each of the comparators, and the difference with each input signal is fed to a corresponding amplitude threshold detector. The amplitude threshold is set so that accumulated tolerances are not apt to trip the detector. As shown here, the amplitude detector issues a set output when

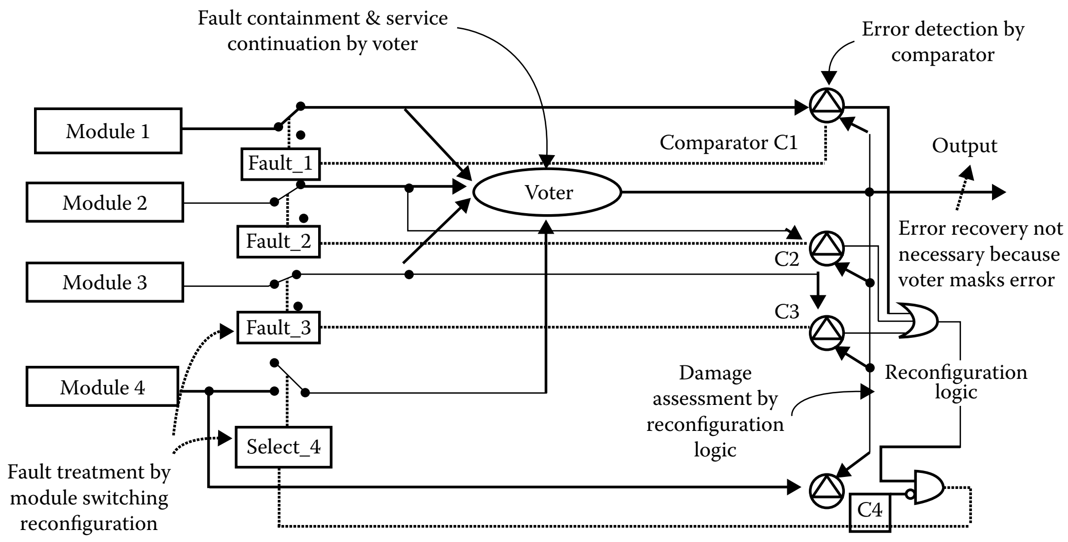

Fault containment & service continuation by voter
Error detection by comparator
Module 1
Module 2
Module 3
Module 4
Fault_1
Fault_2
Fault_3
Select_4
Voter
Comparator C1
C2
C3
C4
Output
Error recovery not necessary because voter masks error
Reconfiguration logic
Damage assessment by reconfiguration logic
Fault treatment by module switching reconfiguration

FIGURE 8.7 Hybrid TMR arrangement.

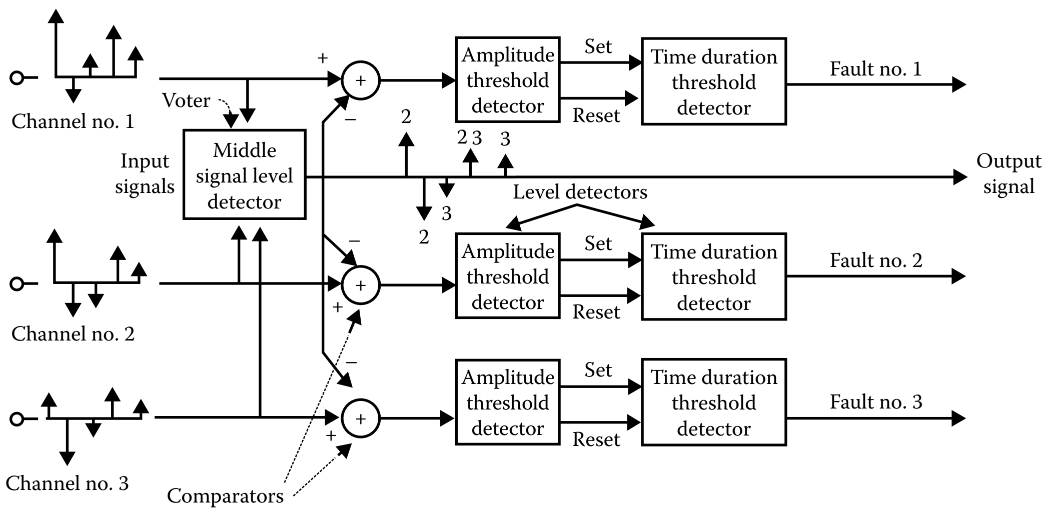


FIGURE 8.8 Triplex voter-comparator.

an excessive difference is first observed. When the difference falls back within the threshold, a reset output is issued.

Because transient effects may produce short-term amplitude detector trips a timing threshold is applied to the output of each amplitude detector. Typically, a given number of consecutive out-of-tolerance amplitude threshold trips are necessary to declare a faulty signal. Hence, a time duration threshold detector begins a count whenever a set signal is received and, in the absence of further inputs, increments the count for each sample interval thereafter. If a given cycle count is exceeded, an erroneous state is declared and a fault logic signal is set for the affected channels. Otherwise, the count is returned to zero when a reset signal is received.

The setting of both the timing and amplitude thresholds is of crucial importance because of the trade-off between nuisance fault logic trips and slow response to actual faults. Nuisance trips erode user confidence in a system; their unwarranted trips can potentially cause resource depletion. On the other hand, a belated fault response may permit an unsafe condition or catastrophic event to occur. The allowable time to recover from a given type of fault, which is application-dependent, is the key to setting the thresholds properly. The degree of channel synchronization and data skewing also affect the threshold settings, because they must accommodate any looseness. The trade-off can become quite problematic when fast fault recovery is required.

Because the integrity and functionality of the system is at stake, the detailed design of a voter comparator must be subject to careful assessment at all stages of development. In the case of a hardware-implemented device, its fault detection aspects must be thoroughly examined. The main concern is passive failures in circuitry that is not normally used. Built-in test, self-monitoring, or fail-hard symptoms are customary approaches to device integrity. In the case of software-implemented voter comparators, their dependability can be reinforced through formal proof methods and in-service verified code.

8.3.2 Watchdog Timers

Watchdog timers can be used to catch both hardware and software wandering into undesirable states (Lala and Harper, 1994). Timing checks are a form of assertion checking. This kind of check is useful because many software and hardware errors are manifested in excessive time taken for some operation. In synchronous data flow architectures, data are to arrive at a specific time. Data transmission errors of this type can be detected using a timer.

8.4 Software-Implemented Fault Tolerance: State Consistency

Software performs a critical role in digital systems. The term "software implemented fault tolerance" is used in this chapter in the broader sense, indicating the role software plays in the implementation of fault tolerance, and not as a reference to the SRI International project performed for NASA in the late 1970s and referred to as SIFT.

8.4.1 Error Detection

Software plays a major role in error detection. Error detection at the system level should be based on the specification of system behavior. The outputs of the system should be checked independent of the system to assure that the outputs conform to the specification. Since they are implemented in software, the checks require access to the information to be checked, and therefore may have the potential of corrupting that information. Hence, the independence between a system and its check cannot be absolute. The provision of ideal checks for error detection is rarely practical, and most systems employ checks for acceptability (Anderson and Lee, 1981).

Deciding where to employ system error detection is not a straightforward matter. Early checks should not be regarded as a substitute for last-moment checks. An early check will of necessity be based on a knowledge of the internal workings of the system and hence will lack independence from the system. An early check could detect an error at the earliest possible stages and minimize the spread of damage; a last-moment check ensures that none of the output of the system remains unchecked. Therefore, both last-moment and early checks should be provided in a system (Anderson and Lee, 1981).

In order to detect software faults, it is necessary that the redundant versions of the software be independent of each other, that is, of diverse design (Avizenis and Kelly, 1982) (see Section 8.5).

8.4.1.1 Replication Checks

If design faults are expected, replication must be provided using versions of the system with different designs. Replication checks compare the two sets of results produced as outputs of the replicated modules. The replication check raises an error flag and initiates the start of other processes to determine which component or channel is in error (Anderson and Lee, 1981).

8.4.1.2 Timing Checks

Timing checks are used to reveal the presence of faults in a system, but not their absence (Anderson and Lee, 1981). In synchronous hard real-time systems, messages containing data are transmitted over data buses at a specific schedule. Failure to receive a message at the scheduled time is an error, which could be caused by faults in a sensor or data bus, for example. In this case, if the data were critical, a method of tolerating the fault may be to use a forward state extrapolation.

8.4.1.3 Reversal Check: Analytical Redundancy

A reversal check takes the outputs from a system and calculates what the inputs should have been to produce that output. The calculated inputs can then be compared with the actual inputs to check whether there is an error. Systems providing mathematical functions often lend themselves to reversal checks (Anderson and Lee, 1981).

Analytic redundancy using either of two general error detection methods — multiple model (MM) or generalized likelihood ratio (GLR) — is a form of reversal check. Both methods make use of a model of the system represented by Kalman filters. The MM attempts to calculate a measure of how well each of the Kalman filters is tracking by looking at the prediction errors. Real systems possess nonlinearity and the model assumes a linear system. The issue is whether the tracking error from the extended Kalman filter corresponds to the linearized model "closest to" the true, nonlinear system and is markedly smaller than the errors from the filters based on "more distant" models. Actuator and sensor failures can be modeled in different ways using this methodology (Willsky, 1980).

The GLR uses a formulation similar to that for MM, but different enough that the structure of the solution is quite different. The starting point for GLR is a model describing normal operation of the observed signals or of the system from which they come. Since GLR is directly aimed at detecting abrupt changes, its applications are restricted to problems involving such changes, such as failure detection. GLR, in contrast to MM, requires a single Kalman filter. Any detectable failure will exhibit a systematic deviation between what is observed and what is predicted to be observed. If the effect of the parametric failure is "close enough" to that of the additive one, the system will work.

Underlying both the GLR and MM methods is the issue of using system redundancy to generate comparison signals that can be used for the detection of failures. The fundamental idea involved in finding comparison signals is to use system redundancy — known relationships among measured variables to generate signals that are small under normal operation and display predictable patterns when particular anomalies develop. All failure detection is based on analytical relationships between sensed variables, including voting methods, which assume that sensors measure precisely the same variable. Using analytical relationships, we can reduce hardware redundancy and maintain the same level of fail-operability. In addition, analytical redundancy allows extracting more information from the data, permitting detection of subtle changes in system component characteristics. On the other hand, the use of this information can cause problems if there are large uncertainties in the parameters specifying the analytical relationships (Willsky, 1980).

The second part of a failure detection algorithm is the decision rule, which uses the available comparison signals to make decisions on the interruption of normal operation by the declaration of failures. One advantage of these methods is that the decision rule — maximize and compare to a threshold — are simple, while the main disadvantage is that the rule does not explicitly reflect the desired trade-offs. The Bayesian Sequential Decision approach, in which an algorithm is used for the calculation of the approximate Bayes decision, has exactly the opposite properties — that is, it allows for a direct incorporation of performance trade-offs — but it is extremely complex. The Bayes Sequential Decision Problem is to choose a stopping rule and terminal decision rule to minimize the total expected cost and the expected cost that is accrued before stopping (Willsky, 1980).

8.4.1.4 Coding Checks

Coding checks are based on redundancy in the representation of an object in use in a system. Within an object, redundant data are maintained in some fixed relationship with the (nonredundant) data representing the value of the object. Parity checks are a well-known example of a coding check, as are error detection and correction codes such as the Hamming, cyclic redundancy check, and arithmetic codes (Anderson and Lee, 1981).

8.4.1.5 Reasonableness Checks

These checks are based on knowledge of the minimum and maximum values of input data, as well as the limits on rate of change of input data. These checks are based on knowledge of the physical operation of sensors, and employ models of this operation.

8.4.1.6 Structural Checks

Two forms of checks can be applied to the data structures in a computing system. Checks on the semantic integrity of the data will be concerned with the consistency of the information contained in a data structure. Checks on the structural integrity will be concerned with whether the structure itself is consistent. For example, external data from subsystems is transmitted from digital data buses such as MIL-STD-1553, ARINC 429, or ARINC 629. The contents of a message (number of words in the message, contents of each word) from a subsystem are stored and the incoming data checked for consistency.

8.4.1.7 Diagnostic Checks

Diagnostic checks create inputs to the hardware elements of a system, which should produce a known output. These checks are rarely used as the primary error detection measure. They are normally run at startup, and may be initiated by an operator as part of a built-in test. They may also run continuously in a background mode when the processor might be idle. Diagnostic checks are also run to isolate certain faults.

8.4.2 Damage Confinement and Assessment

When an error has been discovered, it is necessary to determine the extent of the damage done by the fault before error recovery can be accomplished. Assessing the extent of the damage is usually related to the structure of the system. Assuming timely detection of errors, the assessment of damage is usually determined to be limited to the current computation or process. The state is assumed consistent on entry. An error detection test is performed before exiting the current computation. Any errors detected are assumed to be caused by faults in the current computation.

8.4.3 Error Recovery

After the extent of the damage has been determined, it is important to restore the system to a consistent state. There are two primary approaches — backward and forward error recovery. In backward error recovery, the system is returned to a previous consistent state. The current computation can then be retried with existing components (retry), with alternate components (reconfigure), or it can be ignored (skip frame). The use of backward recovery implies the ability to save and restore the state. Backward error recovery is independent of damage assessment. Forward error recovery attempts to continue the current computation by restoring the system to a consistent state, compensating for the inconsistencies found in the current state. Forward error recovery implies detailed knowledge of the extent of the damage done, and a strategy for repairing the inconsistencies. Forward error recovery is more difficult to implement than backward error recovery (Hitt et al., 1984).

8.4.4 Fault Treatment

Once the system has recovered from an error, it may be desirable to isolate and correct the component that caused the error. Fault treatment is not always necessary because of the transient nature of some faults or because the detection and recovery procedures are sufficient to cope with other recurring errors. For permanent faults, fault treatment becomes important because the masking of permanent faults reduces the ability of the system to deal with subsequent faults. Some fault-tolerant software techniques attempt to isolate faults to the current computation by timely error detection. Having isolated the fault, fault treatment can be done by reconfiguring the computation to use alternate forms of the computation to allow for continued service. (This can be done serially, as in recovery blocks, or in parallel, as in N-Version programming.) The assumption is that the damage due to faults is properly encapsulated to the current computation and that error detection itself is faultless (i.e., it detects all errors and causes none of its own) (Hitt et al., 1984).

8.4.5 Distributed Fault Tolerance

Multiprocessing architectures consisting of computing resources interconnected by external data buses should be designed as a distributed fault-tolerant system. The computing resources may be installed in an enclosure using a parallel backplane bus to implement multiprocessing within the enclosure. Each enclosure can be considered a virtual node in the overall network. A network operating system, coupled with the data buses and their protocol, completes the fault-tolerant distributed system. The architecture can be asynchronous, loosely synchronous, or tightly synchronous. Maintaining consistency of data across redundant channels of asynchronous systems is difficult (Papadopoulos, 1985).

8.5 Software Fault Tolerance

Software faults, considered design faults, may be created during the requirements development, specification creation, software architecture design, code creation, and code integration. While many faults may be found and removed during system integration and testing, it is virtually impossible to eliminate all possible software design faults. Consequently, software fault tolerance is used. Table 8.5 lists the major fault-tolerant software techniques in use today. The two main methods that have been used to provide software fault tolerance are N-version software and recovery blocks.

8.5.1 Multiversion Software

Multiversion software is any fault-tolerant software technique in which two or more alternate versions are implemented, executed, and the results compared using some form of a decision algorithm. The goal is to develop these alternate versions such that software faults that may exist in one version are not

TABLE 8.5 Categorization of Fault-Tolerant Software Techniques

Multiversion software
N-version program
Cranfield algorithm for fault-tolerance (CRAFT) food taster
Distinct and dissimilar software
Recovery blocks
Deadline mechanism
Dissimilar backup software
Exception handlers
Hardened kernel
Robust data structures and audit routines
Run-time assertions[a]
Hybrid multiversion software and recovery block techniques
Tandem
Consensus-recovery blocks

[a] Not a complete fault-tolerant software technique as it only detects errors.

Source: From Hitt, E. et al., Study of Fault-Tolerant Software Technology, NASA CR 172385.

contained in the other version or versions, and the decision algorithm determines the correct value from among the alternate versions. Whatever means are used to produce the alternate versions, the common goal is to have distinct versions of software such that the probability of faults occurring simultaneously is small and that faults are distinguishable when the results of executing the multiversions are compared with each other.

The comparison function executes as a decision algorithm once it has received results from each version. The decision algorithm selects an answer or signals that it cannot determine an answer. This decision algorithm and the development of the alternate versions constitute the primary error detection method. Damage assessment assumes the damage is limited to the encapsulation of the individual software versions. Faulted software components are masked so that faults are confined within the module in which they occur. Fault recovery of the faulted component may or may not be attempted.

N-versions of a program are independently created by N-software engineering teams working from a (usually) common specification. Each version executes independently of the other versions. Each version must have access to an identical set of input values and the outputs are compared by an executive which selects the result used. The choice of an exact or inexact voting check algorithm is influenced by the criticality of the function and the timing associated with the voting.

8.5.2 Recovery Blocks

The second major technique shown in Table 8.5 is the recovery block and its subcategories — deadline mechanism and dissimilar backup software. The recovery block technique recognizes the probability that residual faults may exist in software. Rather than develop independent redundant modules, this technique relies on the use of a software module, which executes an acceptance test on the output of a primary module. The acceptance test raises an exception if the state of the system is not acceptable. The next step is to assess the damage and recover from the fault. Given that a design fault in the primary module could have caused arbitrary damage to the system state and that the exact time at which errors were generated cannot be identified, the most suitable prior state for restoration is the state that existed just before the primary module was entered (Anderson and Lee, 1981).

8.5.3 Trade-Offs

Coverage of a large number of faults has an associated overhead in terms of redundancy, and the processing associated with the error detection. The designer may use modeling and simulation to determine the amount of redundancy required to implement the fault tolerance versus the probability and impact of the different types of faults. If a fault has minimal or no impact on safety, or mission completion, investing in redundancy to handle that fault may not be effective, even if the probability of the fault occurring is significant.

8.6 Summary

Fault-tolerant systems must be used whenever a failure can result in loss of life or loss of a high-value asset. Physical failures of hardware decreased when analog avionics were replaced with the first and second generation digital avionics. The third-generation digital avionics use of highly integrated multifunction COTS ICs and circuit cards implementing multiple functions have more complex failure mechanisms and may, at the card level, have higher failure rates than the second-generation digital avionics. Physical and design faults are virtually impossible to completely eliminate, so increased reliance on fault tolerance must occur to meet safety mandates.

8.6.1 Design Analyses

In applying fault-tolerance to a complex system, there is a danger that the new mechanisms may introduce additional sources of failure due to design and implementation errors. It is important, therefore, that the new mechanisms be introduced in a way that preserves the integrity of a design with minimum added complexity. The designer must use modeling and simulation tools to assure that the design accomplishes the needed fault tolerance.

Certain design principles have been developed to simplify the process of making design decisions. Encapsulation and hierarchy offer ways to achieve simplicity and generality in the realization of particular fault-tolerance functions. Encapsulation provides the following:

- Organization of data and programs as uniform objects, with rigorous control of object interaction
- Organization of sets of alternate program versions into fault-tolerant program modules (e.g., recovery blocks and N-version program sets)
- Organization of consistent sets of recovery points for multiple processes
- Organization of communications among distributed processes as atomic (indivisible) actions
- Organization of operating system functions into recoverable modules

The following are examples of the hierarchy principle used to enhance reliability of fault-tolerance functions:

- Organization of all software, both application and system type, into layers with unidirectional dependencies among layers
- Integration of service functions and fault-tolerance functions at each level
- Use of nested recovery blocks to provide hierarchical recovery capability
- Organization of operating system functions so that only a minimal set at the lowest level (a "kernel") needs be exempted from fault tolerance
- Integration of global and local data and control in distributed processors

That portion of the operating system kernel that provides the basic mechanisms the rest of the system uses to achieve fault-tolerance should be "trusted." This kernel should be of limited complexity so that all possible paths can be tested to assure correct operation under all logic and data conditions. This kernel need not be fault tolerant if the foregoing can be assured.

8.6.2 Safety

Safety is defined in terms of hazards and risks. A hazard is a condition, or set of conditions that can produce an accident under the right circumstances. The level of risk associated with the hazard depends on the probability that the hazard will occur, the probability of an accident taking place if the hazard does occur, and the potential consequence of the accident (Williams, 1992).

8.6.3 Validation

Validation is the process by which systems are shown through operation to satisfy the specifications. The validation process begins when the specification is complete. The difficulty of developing a precise specification that will never change has been recognized. This reality has resulted in an iterative development and validation process. Validation requires developing test cases and executing these test cases on the hardware and software comprising the system to be delivered. The tests must cover 100% of the faults the system is designed to tolerate, and a very high percentage of possible design faults, whether hardware, software, or the interaction of the hardware and software during execution of all possible data and logical paths. Once the system has been validated, it can be put in operation. In order to minimize the need to validate a complete Operational Flight Program (OFP) every time it is modified, development methods attempt to decompose a large system into modules that are independently specified, implemented, and validated. Only those modules and their interfaces that are modified must be revalidated using this approach (see Chapter 9).

Rapid prototyping, simulation, and animation are all techniques that help validate the system. Formal methods are being used to develop and validate digital avionics systems. There are arguments both in favor of and against the use of formal methods (Rushby, 1993; Williams, 1992).

8.6.4 Conclusion

For safety-critical systems, fault tolerance must be used to tolerate design faults that are predominately software- and timing-related. It is not enough to eliminate almost all faults introduced in the later stages of a life cycle; assurance is needed that they have been eliminated or are extremely improbable. Safety requirements for commercial aviation dictate that a failure causing loss of life must be extremely improbable — on the order of 10^{-9} per flight-hour. The designer of safety-critical, fault-tolerant systems should keep current with new development in this field since both design and validation methods continue to advance in capability.

References

Anderson, T. and Lee, P.A., *Fault Tolerance, Principles and Practices*, Prentice-Hall, London, 1981.

Anderson, T., Ed., *Safe and Secure Computing Systems*, Blackwell Scientific, Oxford, U.K., 1989.

Avizienis, A., Fault-tolerant systems, *IEEE Trans. Comput.*, C-25(12):1304–1312, 1976.

Avizienis, A. and Kelly, J., *Fault-Tolerant Multi-Version Software: Experimental Results of a Design Diversity Approach,* UCLA Computer Science Department, Los Angeles, 1982.

Avizienis, A., Kopetz, H., and Laprie, J.C., Eds., *The Evolution of Fault-Tolerant Computing*, Springer-Verlag, New York, 1987.

Best, D.W., McGahee, K.L., and Shultz, R.K.A., *Fault Tolerant Avionics Multiprocessing System Architecture Supporting Concurrent Execution of Ada Tasks*, Collins Government Avionics Division, AIAA 88-3908–CP, 1988.

Constantinescu, C., Impact of Deep Submicron Technology on Dependability of VLSI Circuits, in *Proc. Dependable Systems and Networks*, 2002.

Driscoll, K., et al., Byzantine fault tolerance, from theory to reality, Int. Conf. Computer Safety, Reliability and Security, 2004.

Gargaro, A.B., et al., Adapting Ada for Distribution and Fault Tolerance, in *Proc. 4th Int. Workshop Real-Time Ada Issues*, ACM, 1990.

Gu, D., Rosenkrantz, D.J., and Ravi, S.S., Construction of check sets for algorithm-based fault tolerance, *IEEE Trans. Comput.*, 43(6): 641–650, 1994.

Hill, J. and Knight, J. C., Selective Notification: Combining Forms of Decoupled Addressing for Internet-Scale Command and Alert Dissemination, Computer Science Department, University of Virginia, 2003.

Hitt, E., Webb, J., Goldberg, J., Levitt, K., Slivinski, T., Broglio, C., and Wild, C., Study of Fault-Tolerant Software Technology, NASA CR 172385, Langley Research Center, VA, 1984.

Hudak, J., Suh, B.H., Siewiorek, D., and Segall, Z., Evaluation and comparison of fault tolerant software techniques, *IEEE Trans. Reliability*, 1993.

Knight, J., et al., The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications, Intrusion Tolerance Workshop, DSN-2002, International Conference on Dependable Systems and Networks, Washington, D.C., June 2002.

Lala, J.H. and Harper, R.E., Architectural principles for safety-critical real-time applications, *Proc. IEEE*, 82(1): 25–40, 1994.

Lala, P.K., *Fault Tolerant and Fault Testable Hardware Design*, Prentice-Hall, London, 1985.

Papadopoulos, G.M., Redundancy Management of Synchronous and Asynchronous Systems, Fault Tolerant Hardware/Software Architecture for Flight Critical Functions, AGARD-LS-143, 1985.

Rushby, J., Formal Methods and Digital Systems Validation for Airborne Systems, NASA CR 4551, Langley Research Center, VA, 1993.

Scandura, P. and Garcia-Galan, C., A Unified System to Provide Crew Alerting Electronic Checklists and Maintenance Using IVHM, *Proc. 23rd Digital Avionics Syst. Conf.*, 2004.

Shin, K.G. and Parmeswaran R., Real-time computing: A new discipline of computer science and engineering, *Proc. IEEE*, 82(1): 6–24, 1994.

Shin, K.G. and Hagbae, K., A time redundancy approach to TMR failures using fault-state likelihoods, *IEEE Trans. Comput.*, 43(10): 1151–1162, 1994.

Sosnowski, J., Transient fault tolerance in digital systems, *IEEE Mi*cro, 14(1): 24–35, 1994.

Strunk, E., Knight, J. C., and Aiello, M. A., Distributed reconfigurable avionics architectures, *Proc. 23rd Digital Avionics Syst. Conf.*, 2004.

Tomek, L., Mainkar, V., Geist, R.M., and Trivedi, K.S., Reliability modeling of life-critical, real-time systems, *Proc. IEEE,* 82(1): 108–121, 1994.

Vaidya, N.H. and Pradhan, D.K., Fault-tolerant design strategies for high reliability and safety, *IEEE Trans. Comput.*, 42(10): 1195–1206, 1993.

Williams, L.G., Formal Methods in the Development of Safety Critical Software Systems, UCRL-ID-109416, Lawrence Livermore National Laboratory, Livermore, CA, 1992.

Willsky, A.S., Failure Detection in Dynamic Systems, Fault Tolerance Design and Redundancy Management Techniques, AGARD-LS-109, 1980.

Further Information

A good introduction to fault-tolerant design is presented in *Fault Tolerance, Principles and Practices*, by Tom Anderson and P.A. Lee. Hardware-specific design techniques are described by P.K. Lala in *Fault Tolerant and Fault Testable Hardware Design*. Other excellent journals are the IEEE publications *Computer, IEEE Micro, IEEE Software, IEEE Transactions on Computers,* and *IEEE Transactions on Software Engineering*.

9 Boeing B-777

Michael J. Morgan
Honeywell

9.1 Introduction

The avionics industry has long recognized the substantial cost benefits that could be realized using a large-scale integrated computing architecture for airborne avionics. Technology achievements by airframe, avionics, and semiconductor manufacturers allow implementation of these integrated avionics architectures resulting in substantial life cycle cost benefits. The Boeing 777 Airplane Information Management System (AIMS) represents the first application of an integrated computing architecture in a commercial air transport.

9.2 Background

Since 1988, the avionics industry has made a significant effort to develop the requirements and goals for a next-generation integrated avionics architecture. This work is documented in ARINC Project Paper 651 [1]. Top-level goals of the Integrated Modular Avionics (IMA) architecture are to reduce overall cost of ownership through reduced spares requirements (includes reduction in cost of spare Line Replaceable Modules [LRM] and reduction in number of LRMs required), reduce equipment removal rate, and reduce weight and volume in both avionics and wiring. In addition, IMA addresses the airlines' demand for better MTBUR/MTBF (Mean Time Between Unscheduled Removals as a fraction of Mean Time Between Failures), improved system performance (response time), increased airborne functionality, better fault isolation and test, and maintenance-free dispatch for extended intervals.

Technology trends in microprocessor and memory technology demand that airborne computing architectures evolve if the avionics industry is to meet the goals of IMA. By exploiting these developments in the microprocessor and memory industries very highly integrated architectures previously not technologically feasible or cost-effective may now be realized. These functionally integrated architectures minimize life cycle cost by minimizing the duplication of hardware and software elements (see Figure 9.1).

High levels of functional integration dictate availability and integrity requirements far exceeding the requirements for federated architectures. Resource availability requirements must be sufficient to probabilistically preclude the simultaneous loss of multiple functions utilizing shared resources. These

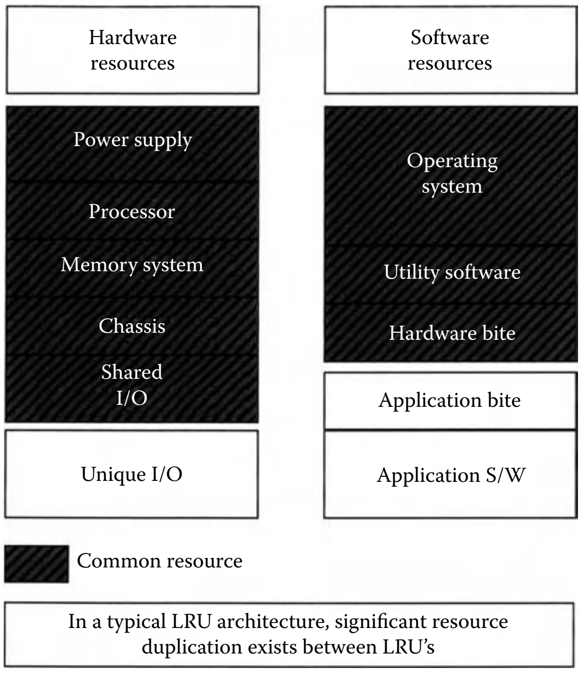


FIGURE 9.1 Components of a typical LRU.

availability requirements imply application of fault-tolerant technology. Although fault tolerance is required to meet the integrity and availability goals of IMA, it is also directly compatible with the airline goal for deferred maintenance. Furthermore, since fault-tolerant technology requires high-integrity monitoring, it also is compatible with airlines' desires for improved fault isolation, better maintenance diagnostics, and reduced unconfirmed removal rate (MTBUR). Current IMA implementations are realizing a more than six times improvement in unconfirmed equipment removals over a typical federated Line Replaceable Unit-based (LRU-based) architecture.

High functional integration also implies the requirement to maintain functional independence for software using any shared resource. Strict CPU separation is not sufficient to ensure that functions will not adversely affect each other. Input-output (I/O) resource sharing demands a backplane bus architecture that has extremely high integrity and enforces rigid partitioning between all users. Processor resource sharing requires a robust software partitioning system in which all partition protection elements are monitored to ensure isolation integrity.

Robust partitioning protection must be performed as an integral part of the architecture, and isolation must not be dependent upon the integrity of the application software. In this environment, the robust partitioning architecture would be certified as a standalone element allowing functional software to be updated and certified independently of other functions sharing the same computational or I/O resources. Since it is anticipated that airborne functionality will continue to increase and that the majority of this increase will be accommodated via software changes alone, this partitioned environment will provide flexibility in responding to evolving system requirements (e.g., CNS/ATM).

9.3 Boeing 777 AIMS

Now in its second-generation implementation, the Boeing 777 AIMS implements the IMA concept in an architecture supporting a high degree of functional integration and reducing duplicated resources to a minimum. In this architecture, the conventional LRUs, which typically contain a single function, are

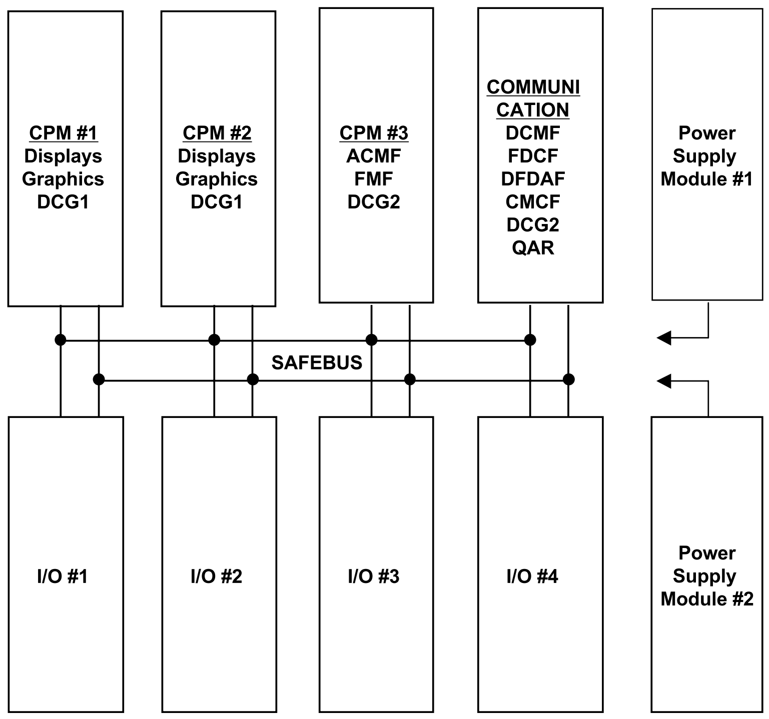


FIGURE 9.2 AIMS baseline functional distribution.

replaced with dual integrated cabinets, which provide the processing and the I/O hardware and software required to perform the following functions (see Figure 9.2):

- Flight Management
- Display
- Central Maintenance
- Airplane Condition Monitoring
- Communication Management (including flight deck communication)
- Data Conversion Gateway (ARINC 429/629 Conversion)

The integrated cabinets are connected to the airplane interfaces via a combination of ARINC 429, ARINC 629, and discrete I/O channels (see Figure 9.3; note that for clarity the 429 and discrete channels are not shown).

9.4 Cabinet Architecture Overview

The heart of the AIMS system consists of dual cabinets located in the electronics bay, each containing four core processor modules (CPMs), four input/output modules (IOMs), and two power supply modules. Additional space is reserved in the cabinet to add CPMs and IOMs as may be required for future growth (see Figure 9.4). The following are shared platform resources provided by AIMS:

- Common processor and mechanical housing
- Common input/output ports, aircraft-level power conditioning, and mechanical housing
- Common backplane bus (SAFEbus®) to move data between CPMs and between CPMs and IOMs
- Common operating system and built-in test (BIT) and utility software

Instead of individual applications residing in a separate LRU, applications are integrated on common CPMs. The IOMs transmit data from the CPMs to other systems on the airplane, and receive data from

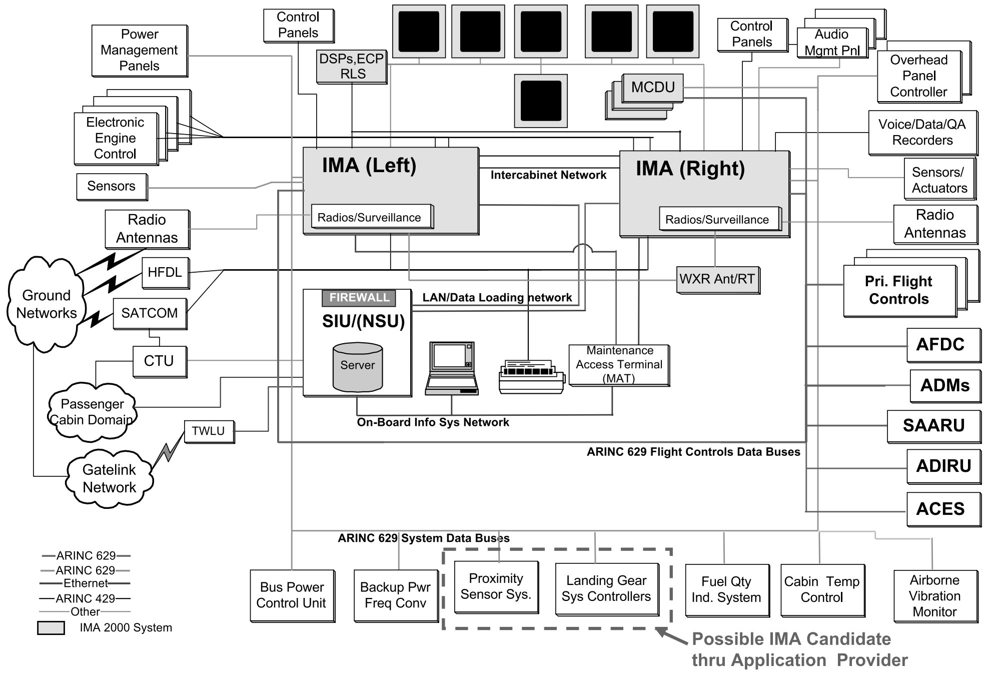


FIGURE 9.3 Airplane interface schematic.

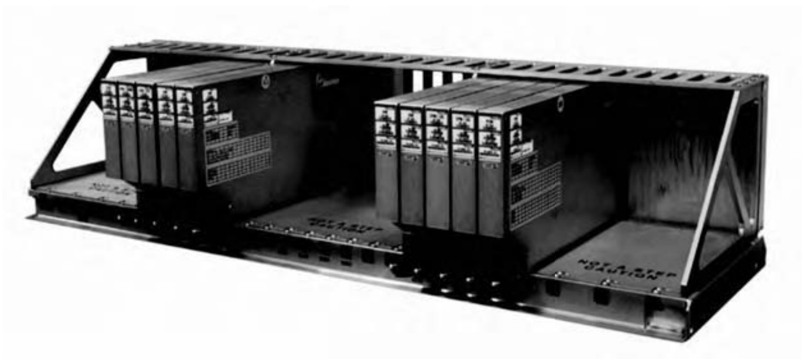


FIGURE 9.4 AIMS cabinet.

these other systems for use by the CPM applications. A high-speed backplane bus, called SAFEbus, provides a 60-Mbit/s data pipe between any of the CPMs and IOMs in a cabinet. Communication between AIMS cabinets is accomplished through four ARINC 629 serial buses.

The robust partitioning provided by the architecture allows applications to use common resources without any adverse interactions. This is achieved through a combination of memory management and deterministic scheduling of application software execution. Memory is allocated before run time, and only one application partition is given write-access to any given page of memory. Scheduling of processor resources for each application is also done before run time and is controlled by a set of tables loaded onto each CPM and IOM in the cabinet. This set of tables operates synchronously and controls application scheduling on the CPMs as well as data movement between modules across the SAFEbus.

Hardware fault detection and isolation is achieved via a lock-step design of the CPMs, IOMs, and the SAFEbus. Each machine cycle on the CPMs and IOMs is performed in lock-step by two separate processing channels, and comparison hardware ensures that each channel is performing identically. If a miscompare occurs, the system will attempt retries where possible before invoking the fault-handling and logging software in the operation system. The SAFEbus has four redundant data channels that are compared in real time to detect and isolate bus faults. The applications hosted on AIMS are listed below, along with the number of redundant copies of each application per shipset in parentheses:

- Displays (4)
- Flight Management/Thrust Management (2)
- Central Maintenance (2)
- Data Communication Management (2)
- Flight Deck Communication (2)
- Airplane Condition Monitoring (1)
- Digital Flight Data Acquisition (2)
- Data Conversion Gateway (4)

All of the IOMs in the two AIMS cabinets are identical. The CPMs have common hardware for processor, memory, power, and SAFEbus interface, but have the capability to include a custom I/O card to provide specific hardware for an application "client." The client hardware in AIMS includes the displays graphics generator, the digital flight data acquisition interface to the data recorder, Aircraft Communications Addressing and Reporting System (ACARS) modem interface, and the airplane condition monitoring memory. The following are other flight deck hardware elements that make up the AIMS system:

- Six flat panel display units
- Three control and display units
- Two Electronic Flight Instrument System (EFIS) display control panels
- Display select panel
- Cursor control devices
- Display remote light sensors

9.5 Backplane Bus

As stated, the cabinet LRMs are interconnected via dual high-speed serial buses called SAFEbus (see Figure 9.5) that provide the only communication mechanism between the processing and I/O elements of the integrated functions. As such, extremely high availability and integrity requirements are necessary to preclude the simultaneous loss of multiple functions and to preserve robust partitioning of I/O resources. In addition, SAFEbus itself is required to provide and enforce the integrity of this key shared resource. Absolute data integrity must be ensured independent of hardware and software failures within any of the CPMs. In this environment, SAFEbus behaves as a generic and virtual resource capable of supporting high levels of I/O integration.

The SAFEbus protocol is driven by a sequence of commands stored in the internal table memory of each Bus Interface Unit (BIU). Each command corresponds to a single message transmission. All BIUs are synchronized so that at any given point in time all BIUs "know" the state of the bus and are at equivalent points in their tables. Because buffer addresses are stored in tables, they do not need to be transmitted over the bus, and since all transactions are scheduled deterministically, there is no need to arbitrate the bus. This allows for extremely high bus efficiency (94%) with no bits required to be dedicated to address control and minimal bits required to control data. A more detailed description of SAFEbus operation can be found in ARINC Project Paper 659 and also in Reference 2.

9.6 Maintenance

The requirements for fault tolerance allow increased design flexibility and capability for deferred maintenance operation. By taking advantage of the high-integrity hardware monitoring, which fault-tolerant

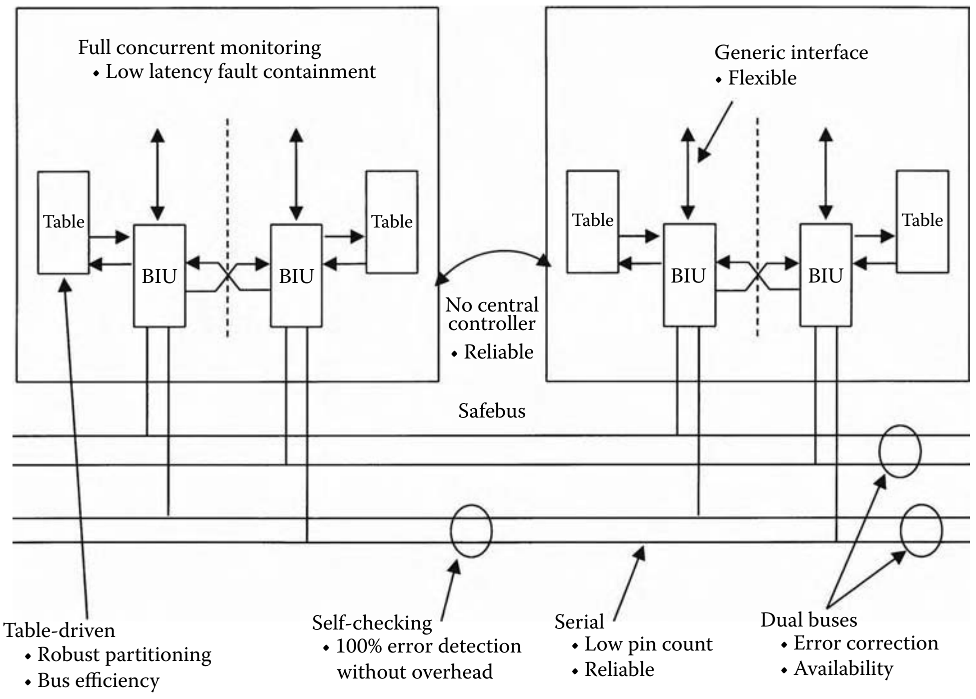


FIGURE 9.5 SAFEbus dual serial buses.

design mandates, the AIMS cabinets are capable of instantaneous fault detection and confinement. This increased fault visibility allows the cabinet to suppress most faults prior to producing a flight deck effect. This is an important step in reducing the Mean Time Between Removal (MTBR) of the equipment. In addition, fault tolerance provides the capability for deferring maintenance to regular (and thus schedulable) intervals. Depending upon the "fail-to-dispatch" probability that a particular airline may be willing to "endure," dispatch can continue for 10 to 30 days without maintenance following any first failure in the AIMS.

9.7 Growth

Functional growth is provided in the cabinets through two paths: (1) installed growth in the form of excess computing and backplane resources provided as part of the baseline AIMS and (2) spare LRM slots provided in each cabinet. Spare computing and backplane resources may be used by any function (new or existing) that requires additional throughput or I/O. Existing spare I/O hardware — for example 629 terminals, 429 terminals, and discrete I/O — are also available for use by any function integrated into the cabinet. Spare LRM slots may be used for additional processing, I/O, or additional unique hardware that may be required for a specific function. Additional processing modules may be added as required without changes to existing cabinet hardware. Addition of I/O may require wiring changes if new airplane interfaces are needed.

References

1. ARINC Project Paper 651, Draft 6, Design Guidance for Integrated Modular Avionics.
2. Hoyme, K., Driscoll, K., Herrlin J., and Radke, K., Honeywell, Inc., ARINC 629 and SAFEbus®: Data buses for commercial aircraft, *Scientific Honeyweller*, Fall 1991, pp. 57–70.

Further Information

This chapter is substantially a reprint of material originally presented in the following sources:

Morgan, M.J., Honeywell, Inc., Integrated modular avionics for next generation airplanes, *IEEE Aerospace and Electronic Systems Magazine*, 6:9–12, Aug. 1991.

Witwer, R., Honeywell, Inc., Developing the 777 airplane information management system (AIMS): A view from program start to one year of service," *IEEE Trans. Aerospace Elect. Syst.*, 33: 637–41, Aug. 1996.

10
New Avionics Systems — Airbus A330/A340

Peter Potocki de Montalk
Airbus Industrie

10.1 Overview

The A330/A340 project is a twin program — the first time that an aircraft has been designed from the outset both with four engines and also with two engines. Both aircraft types have essentially the same passenger and freight capacity. The four-engined A340 is optimized for long-range missions, but is also efficient at shorter ranges. With two engines, the A330 offers even better operating economics for the missions where an airline does not need the very long range of the A340.

The key to obtaining substantial commonality between the two products was the realization that on the two different aircraft very many features could, in fact, be engineered the same way without a penalty. This approach has provided very substantial advantages for the operators, the airframe manufacturer, and for the equipment vendors. In effect, by designing for both sister aircraft from the outset, the requirements were engineered in common, and any added features for either of the two aircrafts could be introduced at a point in the design where they cost very little extra in terms of price, weight, reliability/maintainability or fuel burn. As a result, the two aircrafts can use the same parts (except the engine-related parts), the same aircrews, and the same airport and maintenance environment, and cost almost the same to develop as a single aircraft. Also, both are supremely efficient.

The A340 is offered in two configurations, allowing operators to tailor capacity and capability to demand. The larger A340-300 has the same fuselage length as the A330 and, while seating 300 to 350 passengers, has seat-mile costs close to those of the latest 747, making it an economical alternative on long-range routes with lower traffic densities.

The A340-200, seating 250 to 300 passengers, has the longest range capability of any commercial airliner available. Its low trip costs, coupled with the operating flexibility of four engines, make it an ideal aircraft for taking over when long-range twins become uneconomic.

While the A340 serves very long routes, the A330 is designed to serve high-growth, high-density regional routes. At the same time, it has the capability to operate economically on extended-range international routes. With typically 335 seats in a two-class arrangement, the A330 has a range of 4500 nmi with a full complement of passengers and baggage and 3200 nmi with maximum payload, making it ideal as a direct replacement for the costlier trijets and as a growth replacement for earlier twinjets.

10.2 Highlights

The A330 and A340 are built on the technological background established by two previous, complementary, product lines. The A300/A310 series is the world's best-established twin-jet twin-aisle aircraft program, with a very large number of technologically advanced features that transfer to larger, longer-range aircraft. The A319/A320/A321 series is the world's most advanced single-aisle aircraft program, again offering a large number of features that are found on much larger aircraft.

During the entire development process, there has been an insistence on securing the maximum commonality that could be achieved with the other programs without loss of efficiency. Using selected features from each of these product lines and updating them as needed resulted in an all-new A330/A340 aircraft program remarkably free of teething troubles, while at the same time providing a new benchmark for aircraft in this size category. As an added benefit, the technological features of the A330/A340 can, in many cases, be used to improve the established older product lines.

10.3 Systems

Before the A330/A340 entered into service, the world's most technologically advanced airliner, in any category, was the A320. Its design formed the basis for the A330/A340 systems.

10.4 Cockpit

The A330/A340 cockpit is designed to be identical to that of the A320 from the point of view of the crew. The exceptions to this rule are associated with the size of the aircraft and the added needs of the long-range mission, such as improved dispatchability, polar navigation capability, and of course, engine-related features. The result is that the 130-seat-capacity short/medium haul A319 up to the 340-plus seat capacity very long-haul A340 have the most advanced flight deck of any airliner, enabling the same crews to fly any of these aircraft with minimum additional training.

10.5 User Involvement

The design of the A330/A340 cockpit has evolved from the same methods that were used successfully on the first Airbus A300. The initial design of the cockpit (and the systems) was based on three features:

- The existing cockpit from the previous aircraft (the A320, in this case)
- The geometry of the A330/A340 nose section (which is based on the geometry of the A300, A310, and 300-600)
- Applicable new research and development work carried out since the A320 had been designed

This initial design was reviewed by a task force consisting of pilots and engineers of each of the launch airlines in light of their experience with the A320 or with other aircraft operating on the intended routes for the A330 and A340.

The task force met a number of times over a period of over a year. At each of these steps, the design of the A330/A340 was refined, and certain features were mocked up for the next iteration in the review. The final design of the aircraft system and cockpit is essentially the one that the airline task force experienced and "flew" in the simulators during their final sessions.

10.6 Avionics

The avionics of the A330/A340 are highly integrated for optimal crew use and for optimal maintenance. As with all previous new and derivative aircraft since the A300FF of 1981, the primary data bus standard is ARINC 429 with ARINC 600 packaging. Other industry bus standards are used in specific applications in which ARINC 429 is not suitable.

10.7 Instruments

The six Cathode Ray Tubes (CRTs) on the main instrument panel display present flight and systems information to the pilots. This arrangement provides excellent visibility of all CRTs. Flight information is provided by the Electronic Flight Instrument System (EFIS) consisting of a Primary Flight Display (PFD) and a Navigation Display (ND) in front of each pilot. The Electronic Centralized Aircraft Monitor (ECAM), consisting of the engine/warning display on the upper screen and aircraft systems display on the lower screen, provides the systems information. Sensors throughout the aircraft continuously monitor the systems, and if a parameter moves out of the normal range, they automatically warn the pilot.

During normal flight, the ECAM presents systems displays according to the phase of flight, showing the systems in which the pilot is interested, such as some secondary engine data, pressurization, and cabin temperature. The pilot can, by manual selection, interrogate any system at any time. Should another system require attention, the ECAM will automatically present it to the flight crew for action. Should a system fault occur that results in a cascade of other system faults, ECAM identifies the originating fault and presents the operational checklists without any need for added crew actions. The information display formats currently in use enable the pilots to assimilate the operational situation of the aircraft much more easily than on the previous generation of aircraft.

There are substantial advantages on the maintenance side as well; the entire Electronic Instrument System (EIS) consists of only three Line Replaceable Unit (LRU) types, enabling significant dispatchability and spare stocks availability. In fact, all the flight information (including standby) is presented on only 11 instruments of 6 types. A new EIS using liquid crystal displays is installed on the A330/A340 and A320 family of aircraft, offering improved capabilities and cost of ownership.

10.8 Navigation

Dual Flight Management Systems (FMS; integrated with the Flight Guidance and Flight Envelope computing functions) combine the data from the aircraft navigation sensors, including the GPS installation. Backup navigation facilities are included in each pilot's multipurpose control/display unit (MCDU), allowing the aircraft to be dispatched with an inoperative FMS.

The FMS permits the crew to select an optimal flight plan for their route from a selection in the airline navigation data base, allowing the aircraft to fly automatically, through the autopilot or flight director, from just after take-off until the crew elects to carry out a precision approach and automatic landing. The "canned" flight plan captures the data needed for flight from the specifications entered by the crew prior to departure, as well as along the route as conditions change and more current information on weather and routing becomes available. New FMSs, with improved cost of ownership and capability, are installed on aircraft delivered from mid-2000. The same new FMSs are being installed on the A320 family.

10.9 Flight Controls

The flight control system of the A330/A340 is essentially the same as that of the A320, with five computers of two different types allowing the pilot to control the aircraft in pitch, roll, and yaw. The layout of the pilot controls is essentially the same as that of the A320 series, as are the handling qualities. The technology features are also essentially the same, with extensive use of dissimilarity in the hardware and in the software, and extensive segregation in the hydraulic and electrical power supplies and signaling lanes. As with the A320 series, mechanical signaling is used for the rudder and for the horizontal stabilizer trim backup.

Detail changes have been introduced reflecting the longer mission times, especially of the A340, to provide better access to the system, and the opportunity has been taken to reduce the variety of backup submodes that the crew must use, making the aircraft even easier to fly.

Like the A320 series, the A330/A340 is a conventional, naturally stable airliner. The electronic flight controls offer a number of advantages to the pilot. There is a large reduction in manually operated mechanical parts, easier troubleshooting, and no need for rigging. Optimal use of the control surfaces is facilitated, as is the use of maneuver load alleviation. The passengers and crew benefit as well, since the aircraft is more comfortable and easier to fly with precision in turbulence, while the flight envelope and structural protection features allow the crew to immediately use the whole capability of the aircraft should it be needed in an extreme situation.

10.10 Central Maintenance System

The A320, with its Central Fault Display System (CFDS), pioneered the industry standard for Central Maintenance Systems (CMS). This industry background of experience has been built into the A330/A340 CMS. It enables troubleshooting and return-to-service testing to be carried out rapidly and with a high degree of confidence from the cockpit. Much of the CMS information may also be accessed remotely, via Aircraft Communications Addressing and Reporting System (ACARS), allowing the aircraft to be greeted upon arrival by a maintenance technician who already has a good idea of the exact nature of any defect and has likely been able to procure from stores the proper spare LRU required to resolve the fault.

Compared with the previous generation of CMSs, such as the A320 CFDS, the A330/A340 CMS has been improved in a number of respects, allowing troubleshooting to take place on more than one system at a time and with even clearer data available for the job. There has been a significant improvement in dependability as well. The systems designers and the equipment vendors have paid great attention to maintainability standards, and a maintenance message filter facility has been incorporated that enables known false messages to be eliminated by the CMS, so that the mechanic does not apply the maxim "Falsus in Uno, Falsus in Omnibus."

10.11 Communications

There is a quiet revolution going on in the way that the crew communicates with the ground. This has been taking place in two ways. The A330/A340 uses the same full-capability standardized flight crew audio and frequency selections system as used on the A320 series, which is also largely used on other recent derivative aircraft. This is a break from the traditional highly customized lower-capability systems.

The other aspect of the revolution is more far-reaching: voice communication is giving way to data communications, with the advantages of lower error rates, more timely service, and lower costs. This started with highly customized ACARS systems for company communications, using very high frequencies (VHFs). The A330/A340 is equipped with a standardized ACARS system that can be used by any customer, with allowance for customers to easily introduce their own custom features to reflect their own needs.

These initial ACARS systems have been extended to offer worldwide coverage, even in mid-ocean and sparsely inhabited areas, using the Inmarsat facilities and high-frequency (HF) data link, and to cover not only company communications but also Air Traffic Control (ATC) services, starting with predeparture and oceanic clearances. On aircraft delivered since 1998, the ACARS unit has been replaced by the Air Traffic Services Unit (ATSU), which is designed to also accommodate safety-related ATC functions using the Aeronautical Telecommunications Network (ATN), offering the majority of ATC and other communication services now using voice and, more importantly, offering profitable migration to the ATN. The ATSU is the first unit to host software from a number of different vendors. The same ATSU is also used on the A320 family of aircraft. The ATN upgrade is being implemented to be available when the corresponding communication and ATC services are in service.

10.12 Flexibility and In-Service Updates

The first generation of aircraft with widespread digital systems, such as the A300FF, A310, and B767, suffered from some of the same disadvantages as their analog-system predecessors because their avionics were not designed to accommodate unplanned changes. Once a design change was made, equipment had to be removed from the aircraft, program memories had to be reloaded in the avionics workshops (sometimes by physically changing parts), and the equipment reinstalled. At some point, the airframe manufacturer usually got involved to certify the change. There was an advantage in the avionics shop, because reloading a program and retesting is a faster and cheaper activity than installing a kit of new electronic parts, but the major cost of carrying out the change on the aircraft stayed the same.

The A330/A340 systems have, to a large extent, overcome this disadvantage, at greatly reduced cost, by having *in situ* facilities on the aircraft for those digital LRUs that have been identified as requiring in-service change. Two techniques are used, depending on the criticality of the LRU concerned and other practicalities:

1. On-board replaceable memories (OBRMs) — memory modules located on the (accessible) front panel of an LRU — come in industry-standard sizes, cost much less than the LRU itself, and can be "recycled" many times. The visible part of the OBRM contains the LRU's software part number section. OBRMs comply with the toughest criticality criteria, enable classic configuration control of the LRU, and require no tools to change. They have been in use on the A320 since 1988.
2. On-board data loading, using 3.5-in diskettes and other media, is a little slower. Although it is even less expensive than OBRMs, it does require a data loader to be carried to or installed on board the aircraft and an adaptation to the aircraft's classic configuration control techniques. The same data loader is used for the FMS data base.

Both techniques enable software updates to be carried out overnight on the whole fleet.

Another aspect is flexibility in dealing with airlines' changing needs. The basic equipment for the aircraft is designed with a number of pin-programmable features that correspond to frequently requested airline changes and other systems like the FMS, in which the airline loads a database that specifies its own preferences. These features allow airlines to pool databases and standard spares at outstations and still obtain the kind of operation that they need. Another feature is partitioned software, in which heavily customized systems like ACARS can be certified just once for all users, with one set of "core" software. The airline may load its own additional operational software on top of this core to reflect its own needs.

Lastly, certain systems like the optional Aircraft Condition Monitoring System (ACMS), which used to be heavily customized, use a combination of these techniques to enable an airline to select the features that it needs out of a very extensive selection that forms a superset of the needs of all the customers.

10.13 Development Environment

The development of each Airbus aircraft has been supported by an Iron Bird whole-aircraft systems rig and by supporting systems rigs that enable work to proceed simultaneously without mutual interference. The A330/A340 model is no exception, and a number of facilities have been constructed specifically for this program. These methods are now being used by other airframe manufacturers.

Proper software development is an essential part of systems development throughout the aircraft, and a number of software tools have been developed, notably in the areas of formal methods, rapid prototyping, automatic coding, and rapid data recovery and analysis. These are supplemented by large, fast data recording and telemetry facilities on the test aircraft fleet, associated with real-time and rapid-playback test data displays for the benefit of the flight test observers on board the test aircraft and for the test and systems engineers on the ground.

The result of this environment, the proper use of features from previous programs, and the proper management of test data flow and the resulting decision process has created an aircraft that has had a remarkable trouble-free period of introduction into service. This is true both in terms of customer satisfaction and in terms of measurable parameters such as delay rate, which have been up to an order of magnitude better for A340 than for the previous derivative long-range aircraft that entered service.

10.14 Support Environment

The A330/A340 airplanes have a number of unique support features, apart from those previously described. As with other Airbus aircraft, an airframe-wide Automatic Test Equipment unit is available to customers, along with an airframe-wide test program suite. No other airframe manufacturer offers this facility for avionics. The Aircraft Maintenance Manual and Trouble Shooting Manual have been carefully designed to integrate with the CMS for easier, faster fault rectification. For those airlines that wish to use it, a software package for a PC-compatible laptop is available to further speed fault-finding. The documentation is also compatible with open industry computer text and graphics standards to facilitate the introduction of intelligent maintenance documentation systems.

11 McDonnell Douglas MD-11 Avionics System

Gordon R.A. Sandell
The Boeing Corporation

11.1 Introduction

While the MD-11 is a derivative of the DC-10 airplane, the avionics system is an all-new system that represented the state of the art at the time of its introduction into service in December 1990. The MD-11 flight deck, shown in Figure 11.1 is designed to be operated by a two-pilot crew.

The crew is provided with six identical 8-in. color CRT displays, which are used to display flight instrument and aircraft systems information. A navigation system based on triple Inertial Reference Systems (IRS) and dual Flight Management Systems (FMS) is provided to automate lateral and vertical navigation, and reduce pilot workload. An Automatic Flight System (AFS) based on dual Flight Control Computers (FCC) is also installed to provide full flight regime autopilot and autothrottles, including fail-operational Category IIIb autoland capability.

The hydraulic, electrical, environmental, and fuel systems, that on previous aircraft were the responsibility of a flight engineer, were automated with the system management now performed by Aircraft System Controllers.

The avionics equipment on a commercial airliner can be divided into two general categories, Seller-Furnished Equipment (SFE) and Buyer-Furnished Equipment (BFE). The BFE avionics comprises the type of equipment that is largely standard from airplane to airplane, such as radios, sensors, and entertainment systems. Airlines generally buy this direct from competing suppliers, and may well use a common supplier for a particular piece of equipment for many different airplane types. Most of this BFE avionics is defined by industry standards, published by ARINC for the Airlines Electronic Engineering Committee (AEEC). The SFE avionics consists of the type of equipment that is specific to the airplane type, and is provided by the airframe manufacturer. It includes such systems as the Auto-Flight System (AFS), the Electronic Instrument System (EIS), Flight Management System (FMS), and the various system controllers. On the MD-11 most

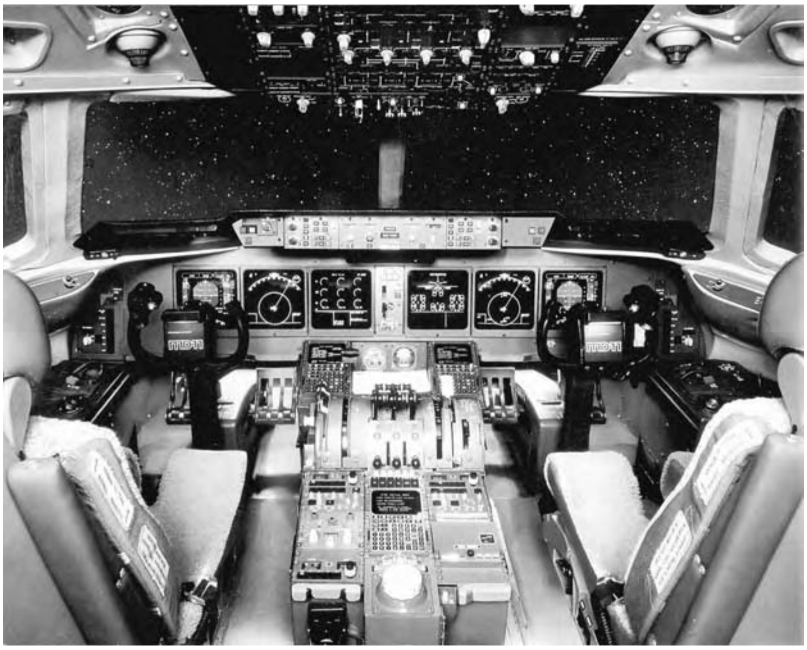


FIGURE 11.1

of this SFE avionics is supplied by Honeywell under a partnership agreement which shared the systems integration function between McDonnell Douglas and Honeywell.

In commercial aviation, the various systems on an airplane are identified under chapter numbers that are defined by the Air Transport Association (ATA). The architectures of each of the systems (communication, navigation, displays, etc.) are discussed below under their respective ATA chapters. Simplified schematic diagrams are provided where appropriate. Note, though, that since the interfaces between the systems are largely ARINC 429 databuses (and in some cases discrete or video), they have been simplified for the purposes of illustration. Some of the data flows, for example, are shown as a single bi-directional arrow. Clearly this is not possible with the point-to-point ARINC 429 databuses, and the single line must therefore represent more than one databus.

11.2 Flight Controls (ATA 22-00 and 27-00)

A dual-dual (four-channel) Auto Flight System (AFS) is installed on the MD-11 to provide autopilot/autothrottle capabilities. The functions of the AFS include:

- Flight Director (FD)
- Automatic Throttle System (ATS)
- Automatic Pilot (AP)
- Autoland (to Cat IIIb minima)
- Yaw damper
- Automatic stabilizer trim control
- Stall warning
- Wind shear protection (detection and guidance)
- Elevator load feel
- Flap limiter
- Automatic ground spoilers

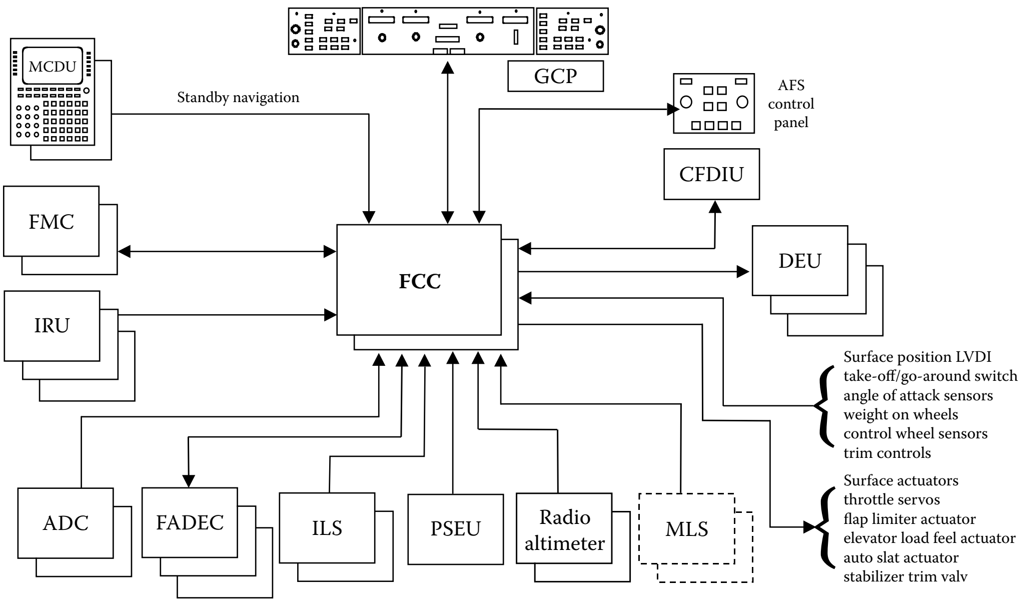


FIGURE 11.2 Auto Flight System (AFS) architecture.

- Altitude alerting
- Longitudinal Stability Augmentation System (LSAS)

The system architecture of the AFS is shown in Figure 11.2. It is built around the dual-dual Flight Control Computers (FCC) and the Glareshield Control Panel (GCP). The GCP is used by the pilots to select AFS modes for pitch, roll, and thrust control and to select targets (e.g., speed or altitude) for those modes. The AFS Control Panel is used by the crew to reconfigure the system in the event of a failure.

The dual-dual FCC architecture is designed around the fail-operational Cat IIIb autoland requirement. Each FCC has two independent computational lanes to provide added integrity and redundancy. Each of these lanes consists of a power supply, two dissimilar microprocessors with dissimilar software (with one of these microprocessor types common to both lanes), and servo-electronics to drive the actuators that move the aircraft's control surfaces. This triple dissimilarity is intended to limit the possibility of software errors in one lane resulting in unsafe operation. This architecture is used for the functions that require high integrity (e.g., autoland and LSAS). Functions having a lower criticality only use a single version and are spread across the different processors. The system is designed to allow the airplane to be dispatched with only one FCC operational, though in this case it would not be able to perform a Cat IIIb autoland.

A key element of flight control system design is the need to provide appropriate levels of redundancy in the interfaces to the actuators for the flight control surfaces. A number of related issues have to be considered:

- Dispatch with one Flight Control Computer (FCC) or one lane inoperative.
- Protection against both random and generic hardware and software failures/errors.
- Minimize the probability of a multi-axis hardover.

Figure 11.3 shows how the elevator, aileron and rudder actuators interface to the various channels of the FCC to satisfy these requirements. The control surfaces are also interconnected mechanically, so driving only one elevator, for example, will actually result in all elevator panels moving. Sufficient control authority is retained in the event of loss of a single channel or even of a complete FCC. The diagram does not show the stabilizer control (which is also driven by the FCC) or the spoilers, which are driven by the aileron/spoiler mechanical mixers and are thus driven indirectly by the aileron actuators.

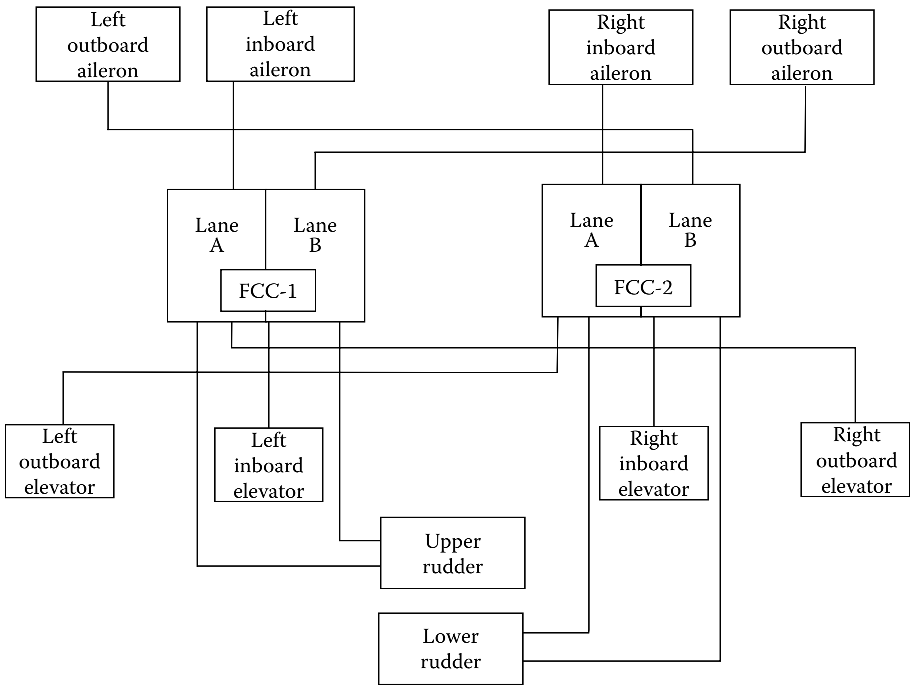


FIGURE 11.3 AFS actuator architecture.

11.3 Communications System (ATA 23-00)

The communication system installed on the MD-11 is a highly integrated system, designed to reduce the workload of the two-man crew while providing the required levels of redundancy. It includes voice communication with the ground via VHF, HF, and SATCOM, as well as data link communications using an optional Aircraft Communications Addressing and Reporting System (ACARS) over the VHF radio, SATCOM, or HF data link (HFDL). The HF and VHF radios are controlled by the Communication Radio Panels located in the pedestal on the flight deck. Selective calling capability is provided by a SELCAL unit. The architecture is shown in Figure 11.4. The basic features of this architecture, in terms of the communication facilities provided, are dictated by Federal Aviation Regulations (FAR) Part 25, which mandate dual independent communication facilities be provided throughout the flight.

The Audio Management Units (AMU) are the heart of the voice communication system for the pilots, and provide flight and service interphone capabilities, as well as supporting the aural alerts on the flight deck generated by the Central Aural Warning System (CAWS), Traffic Alert and Collision Avoidance System (TCAS), and Ground Proximity Warning System (GPWS). The Cockpit Voice Recorder (CVR) records all transmissions by the pilots. Audio Control Panels are provided for all crew on the flight deck (including the observer's station) to control volume, etc. Similarly, jack panels are provided for each crew member's headset.

One feature which is becoming more common on transport aircraft today is the SATCOM system, and the MD-11 has provisions to allow this to be installed. Either a single (6-channel) or a dual (12-channel) system may be installed to provide facilities for both passengers and crew transmissions. The data channel provided for crew use comes into its own for the new CNS/ATM environment (discussed below), while the other channels can be used to provide facsimile and telephone services for the passengers and the cabin crew. It is this passenger service use that typically pays for the system's installation on the airplane.

With all these communication systems, and the navigation systems described below, there is a need for a very large number of antennas on the airplane, and the total installation has to be designed to

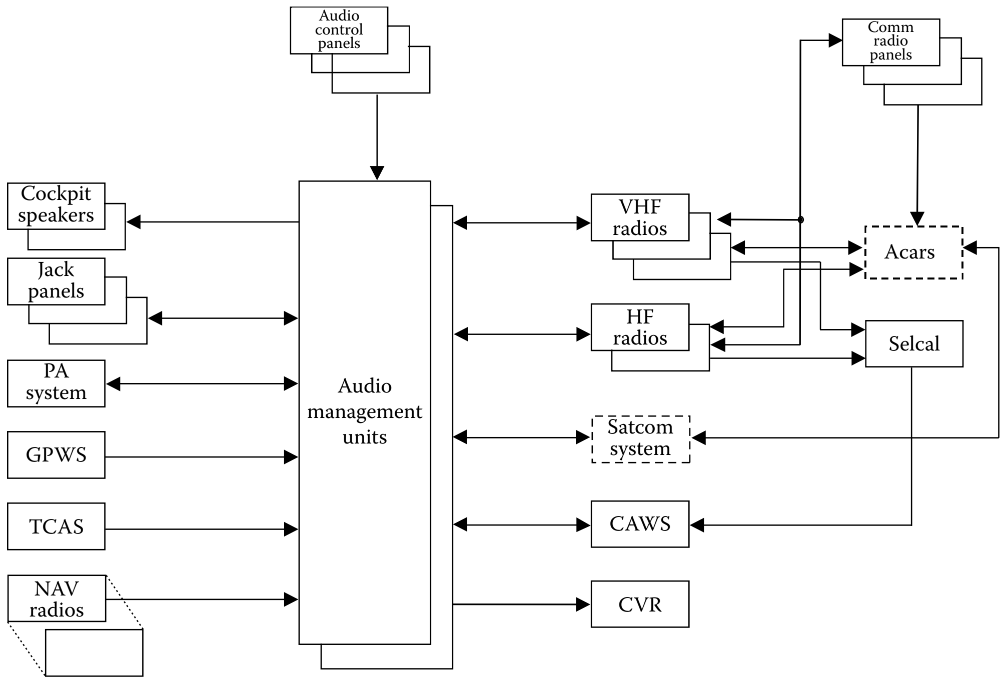


FIGURE 11.4 Communication system architecture.

preclude interference between the different systems. The antenna layout on the MD-11 is shown in Figure 11.5.

Much of the communications equipment is defined by standard ARINC characteristics and is procured by the operators as BFE. Thus multiple suppliers are certified, and the operators may choose which they prefer. The same applies to the navigation radios described below.

11.4 Entertainment System (23-00)

On an airplane such as the MD-11, which can have as many as 410 seats, the entertainment system comprises the vast majority of the avionics Line Replaceable Units (LRUs). With one passenger control unit per seat, one seat electronics box per seat group, and one in-seat video monitor per seat (for those operators that provide in-seat video), it can amount to well over 1000 LRUs.

It is, however, typically a system that is almost totally outside the control of the airplane's manufacturer. It is usually BFE, and an airline may well upgrade it significantly (or replace it totally) several times during the life of the airplane. An airline will usually select a supplier for their entire fleet and then have the supplier adapt it to the particular airplane installation. It is therefore not really practicable to talk about a standard MD-11 entertainment system.

The only standard feature is the audio entertainment system/service system, which interfaces with the Passenger Address (PA) system to allow any safety-related announcements to override the entertainment audio.

11.5 Display System (ATA 31-00)

The most prominent feature of the MD-11 flight deck is the Electronic Instrument System (EIS). This consists of six 8-in. by 8-in. Cathode Ray Tube (CRT) Display Units (DU) arranged in two horizontal groups of three. The outer two DUs are Primary Flight Displays (PFD). Inboard of these are two Navigation Displays (ND), which can display any of five different formats. The center two DUs provide

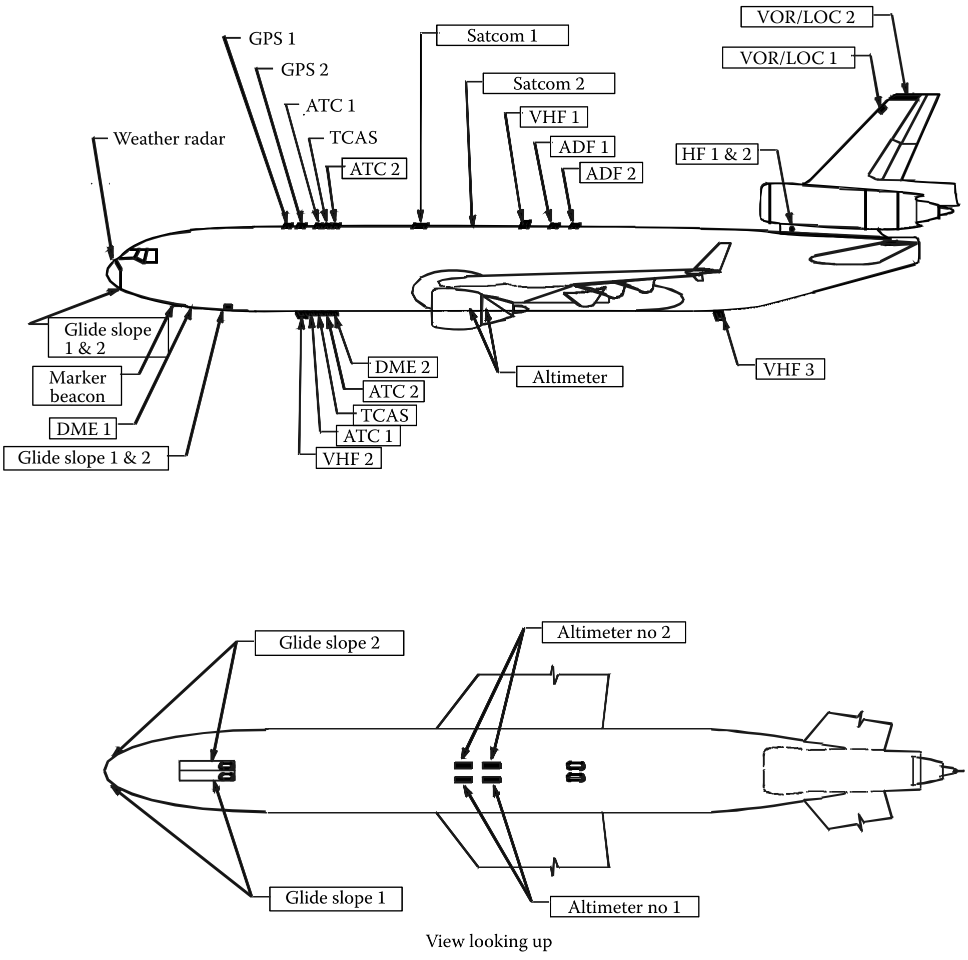


FIGURE 11.5 Antenna layout.

the Engine and Alert Display (EAD) and the System Display (SD). The SD has 10 selectable pages to allow synoptic displays for any of the airplane systems to be presented. The SD pages are selected from the System Display Control Panel (SCP).

The architecture of the EIS is shown in Figure 11.6. This is a very simplified presentation. Any Display Electronics Unit (DEU) can support all six DUs, thus allowing the flight to continue in the event of a failure, and dispatch of the airplane with one inoperative. It is also possible to dispatch with one DU inoperative. In the event of loss of one or more DUs, the system will automatically reconfigure to provide the appropriate displays according to a fixed priority scheme. The lowest priority is accorded to the First Officer's Navigation Display (ND), and the highest priority to the Captain's Primary Flight Display (PFD).

In addition to these displays, standby displays of air data (airspeed and altitude) and a standby attitude indicator are provided on the main instrument panel. These are completely independent of the EIS, thus providing an additional level of backup. These standby displays are mandated by Federal Aviation Regulations (FAR).

On the MD-11 the Engine and Alert Display is part of the EIS. The DEUs thus contain all the alerting logic for the airplane and drive the Master Caution and Warning indicators. They also provide outputs to the Central Aural Warning System (CAWS) to generate voice alerts.

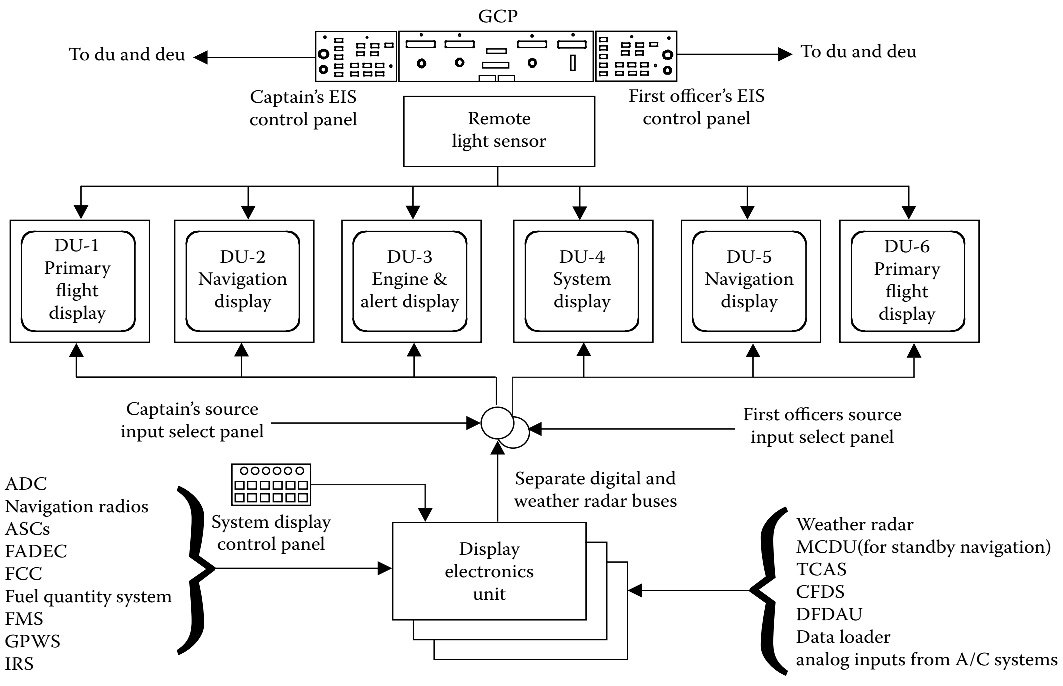

GCP
To du and deu
To du and deu
Captain's EIS control panel
First officer's EIS control panel
Remote light sensor
DU-1 Primary flight display
DU-2 Navigation display
DU-3 Engine & alert display
DU-4 System display
DU-5 Navigation display
DU-6 Primary flight display
Captain's source input select panel
First officers source input select panel
Separate digital and weather radar buses
System display control panel
Display electronics unit
ADC
Navigation radios
ASCs
FADEC
FCC
Fuel quantity system
FMS
GPWS
IRS
Weather radar
MCDU(for standby navigation)
TCAS
CFDS
DFDAU
Data loader
analog inputs from A/C systems

FIGURE 11.6 Display system architecture.

11.6 Recording Systems (ATA 31-00)

A number of in-flight recording capabilities are provided on the MD-11, for both voice and data storage. These include:

Cockpit Voice Recorder (CVR)
Flight Data Recorder (FDR)
Auxiliary Data Acquisition System (ADAS)
In-Service Data Acquisition System (ISDAS)

The CVR was discussed along with the communications systems, to which group it properly belongs. The FDR is the recorder mandated by Federal Aviation Regulations to allow investigations of incidents that have occurred. Figure 11.7 shows how this is driven by a Digital Flight Data Acquisition Unit (DFDAU), which receives data from the other avionics systems in both digital and analog form. There is an FAA Notice of Proposed Rule-Making (NPRM) that mandates expanding the amount of data to be recorded. The MD-11 already records most of these parameters, but will be adapted as necessary to record the remainder.

The Auxiliary Data Acquisition System (ADAS) is also shown in the figure. It uses the Data Management Unit (DMU), and allows both the FAA mandatory data and additional data from the aircraft systems to be recorded for future access via the Quick Access Recorder (QAR). Typically an operator would have the DMU programmed to record specific data for specific events (e.g., recording data for analysis of an aircraft exceedance, or recording aircraft and engine performance parameters to allow trend analysis). The ADAS is an optional feature. The form of the ADAS shown in the figure is one version. There is also a version that uses a combined DFDAU/DMU.

The final recording system is the In-Service Data Acquisition System (ISDAS). Several of the major avionics LRUs have a databus that can be programmed to output an operator-defined set of parameters to allow in-service troubleshooting of the system. The FMS, EIS, and AFS are among those systems having this capability.

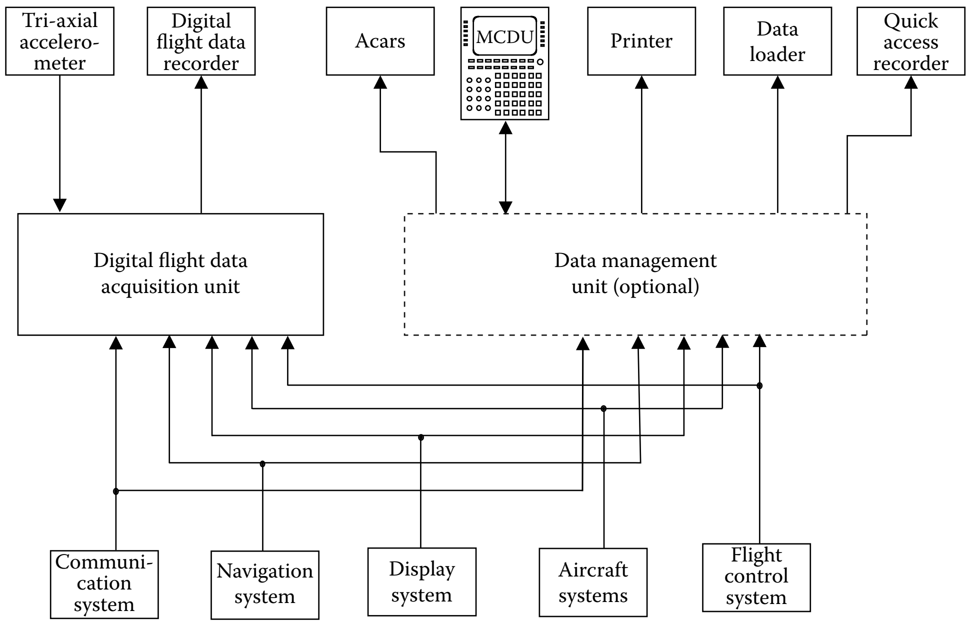


FIGURE 11.7 Recording system architecture.

11.7 Navigation Systems (ATA 34-00)

The navigation system for the MD-11 is built around a triple Inertial Reference System (IRS) and a dual Flight Management System (FMS). The navigation system is shown schematically in Figure 11.8. The FMS is, of course, much more than just a lateral navigation system. It provides a number of functions that are central to operation of the airplane:

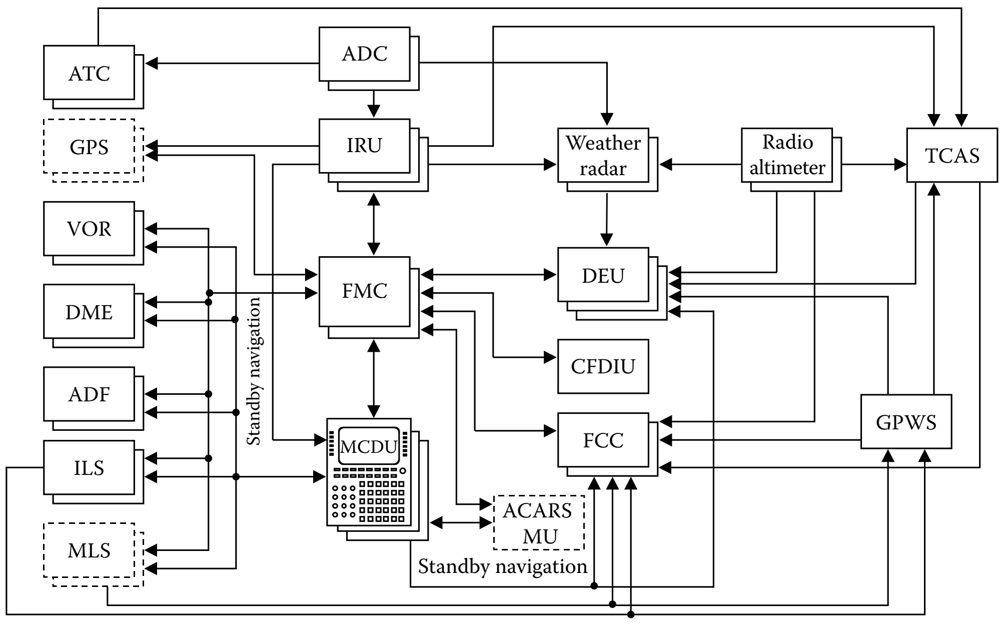


FIGURE 11.8 MD-11 navigation system architecture.

- Ability to create flight plans, including airways, Standard Instrument Departures (SIDs), and Standard Terminal Arrival Routings (STARs) by keyboard entry or data link.
- Multisensor navigation using inertial reference data, together with inputs from GPS, DME, VOR, and ILS.
- Performance predictions for the complete flight plan, including altitude, speed, time of arrival, and fuel state.
- Guidance to the flight plan in three dimensions and controlling arrival time.
- Take-off and approach speed generation.
- Providing the VOR beam guidance mode.

On a long-range airplane, such as the MD-11, being able to dispatch the airplane when it is several thousand miles from the airline's maintenance facility and one navigation system has failed is very important to securing the bottom line for the operator. Such airplanes therefore usually have triple navigation systems. This capability to dispatch with a single failure is provided on the MD-11 by having triple IRS (thus allowing for a failure in this system) and having a standby navigation function provided in the Multipurpose Control/Display Units (MCDU), thus allowing for an FMS failure.

The Inertial Reference System provides a good independent position solution for short-term operation, or even for long-term operation within its capability of a drift of up to 2 nmi/h. However to provide the accuracy necessary for the area navigation required in today's airspace system or for terminal area operations, radio updating is necessary. This is provided on the MD-11 by having dual VHF Omni-Range Receivers (VOR) and dual Distance Measuring Equipment (DME) transceivers. Automatic Direction Finding (ADF) for flying nonprecision approaches and Instrument Landing System (ILS) for precision approach and landing are also provided. At the time that the MD-11 was designed, Microwave Landing System (MLS) was the up and coming system, and provisions are included to install this, although no airline has yet installed MLS. Global Navigation Satellite Systems for en-route operation and even in the future as a precision approach sensor are now the means of navigation, and the option to install this on the MD-11 is now available. All of these navigation sensors are BFE equipment, and thus operators have a choice of suppliers to select from.

The antennas are not shown on the diagram, but one point that calls for a comment is that because of the geometry of the MD-11, the glideslope antennas for the ILS, which are installed in the radome, have to be replicated on the nose landing gear and the ILS must use the gear-mounted antennas on final approach. This is to meet the FAA requirement to have the antenna less than 19 ft above the wheels when crossing the runway threshold. The same rule, obviously, applies to the equivalent MLS antennas.

A dual air data system is also installed to provide airspeed, altitude, etc. for display to the crew and as inputs for the other systems (AFS, FMS, etc.) that need such data. Selection of baro reference is provided on the Glareshield Control Panel (GCP) which is part of the AFS (ATA 22-00) and is described there. Metric altitude displays and barosets are provided in addition to the English units normally used. There is an option to add a third air data system, in which case it is configured as a hot spare with a separate switching unit.

Additionally, dual weather radar systems (with a single flat plate antenna) are provided, together with radio altimeters, ATC transponders, and Traffic Alert and Collision Avoidance System (TCAS). The weather radar is now available with the capability to detect windshear ahead of the airplane. TCAS is a requirement for U.S. operators and foreign operators flying in U.S. airspace. Freighter aircraft (since the FAA regulation applies to aircraft with more than 30 seats) and government-operated aircraft do not require it, although most of these operators do, in fact, install it. These systems, again, are BFE, thus allowing the operators to select from competing suppliers.

All of this equipment is connected to the Centralized Fault Display System (CFDS) to provide fault reporting on each of the units, although for clarity only the FMCU is shown connected to the CFDIU in Figure 11.8. This system is discussed in more detail under ATA 45-00.

11.8 Maintenance Systems (ATA 45-00)

The maintenance system on the MD-11 consists of two main elements, the Centralized Fault Display System (CFDS) that is standard on the airplane, and the On-board Maintenance Terminal (OMT) which is available as a customer option.

The CFDS consists of a Centralized Fault Display Interface Unit (CFDIU) and any of the three MCDUs, with the capability to interface to all the major avionics subsystems on the aircraft, using ARINC 604 protocols, as shown in Figure 11.9. The functions provided by the CFDS are

- A summary of Line Replaceable Units (LRUs) that have reported faults on the last flight
- Capability to select individual "Reporting LRUs" for review of current faults and fault history
- Initiation of Return-to-Service testing of aircraft components (on-ground only)
- Capability to view sensor data
- Erasure of LRU maintenance memory (on-ground only)
- Ability to declare components inoperative for the Aircraft System Controllers (ASC)

The CFDS can also interface to an ACARS Management Unit to transmit fault data to the ground and to a printer on board the airplane.

The optional On-board Maintenance Terminal (OMT), shown in Figure 11.9, expands the capability of the CFDS and automates many of the required maintenance tasks. It contains a fault message database, which allows it to correlate each fault message to a specific flight deck effect. The displays on the OMT are customized by the individual airlines, but a typical display can show which LRU is involved, the fault message, the flight deck effect and alert, the Minimum Equipment List (MEL), documentation reference, and other useful information.

The OMT also incorporates a mass storage device, allowing it to store the aircraft maintenance documents, including the Aircraft Maintenance Manual (AMM), Fault Isolation Manual (FIM), MEL, wiring diagrams, etc. The built-in references to this documentation allow these data to be provided automatically as part of the fault-tracing process.

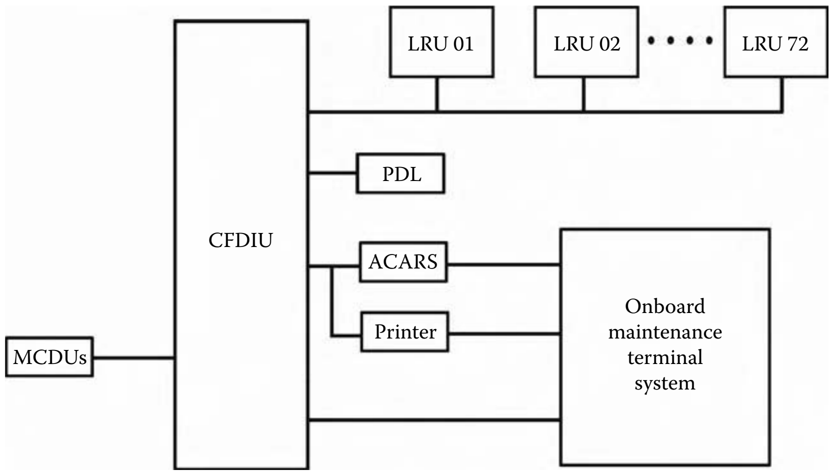


FIGURE 11.9 MD-11 maintenance system architecture.

11.9 Aircraft Systems

One of the major challenges in redesigning the DC-10 to become the MD-11 was to convert the three-crew flight deck of the DC-10 to a two-pilot flight deck, which is the current standard for the industry. To enable the MD-11 to be flown by a two-pilot crew, the normal, abnormal, and emergency system functions performed by the Flight Engineer on the DC-10 have been automated. This was made possible by advances in system control technology that enabled automatic execution of proven DC-10 procedures without extensive redesign or modification of the systems.

The status of the aircraft and its systems are provided to the crew on the main instrument panel Display Units (DUs) without any need to review the overhead panels, which provide backup annunciation and manual interface capability. Alert information is displayed on the Engine and Alert Display (EAD), while system status is displayed in schematic form as system synoptics on the System Display (SD), allowing a rapid assessment of system failures and their consequences.

The aircraft systems are monitored for proper operation by the Automatic Systems Controllers (ASC). In most cases, system reconfiguration as a result of a malfunction is automatic, with manual input being required for irreversible actions, such as engine shutdown, fuel dumping, fire agent discharge, or generator disconnect. A "Dark Cockpit" philosophy has been adopted. During normal operations, all annunciators on the overhead panel will be extinguished, thus confirming to the crew that all systems are operating normally and are properly configured.

The MD-11 aircraft systems can be controlled manually from the overhead panel area of the cockpit. The center portion of the overhead panel is composed of the primary aircraft system panels, including air, fuel, electric, and hydraulic systems, and are easily accessible by either pilot. Each panel is designed so that the left portion controls system #1, the center portion system #2, and the right hand portion system #3.

Each Aircraft System Controller (ASC) has two automatic channels and a manual mode. If the operating automatic channel fails or is shut off by its protection devices, the ASC will automatically select the alternate automatic channel and continue to operate normally. If both automatic channels fail, then the controller reverts to manual operation. The crew would then employ simplified manual procedures for the remainder of the flight. In order to allow operators to maximize the dispatch reliability of the airplane, the MD-11 is certified for dispatch with up to two systems operating in manual mode. The general architecture for Aircraft System Controllers is shown in Figure 11.10.

Automatic System Controllers (ASC) are provided for the primary systems as follows:

- Environmental System Controller (ESC)
- Hydraulic System Controller (HSC)
- Electrical Power Control Unit (EPCU)
- Fuel System Controller (FSC) and Ancillary Fuel System Controller (AFSC)

Pneumatic system controller, air conditioning controllers, and cabin pressure controllers are also provided to control their respective subsystems.

11.10 Interchangeability

The MD-11 has been designed to allow interchangeability with the systems installed on other aircraft. The aircraft radios (HF, VHF, SATCOM, GNSSU, VOR, ILS, DME, ADF, ATC transponder, and radio altimeter) and other systems, such as ACARS, recorders, TCAS, and weather radar) are buyer-furnished equipment (BFE). They are all defined by industry standards (ARINC characteristics). Thus an operator is able to select a standard unit that is used on several airplane types and purchase in bulk for his entire fleet. With this standardization, he is also able to obtain spares from the vendor or other airlines at locations where he does not have a spares depot.

Another key element in providing interchangeability is with the MD-11's "single part number" philosophy. Under this philosophy, all the options required by various airlines for a particular system are

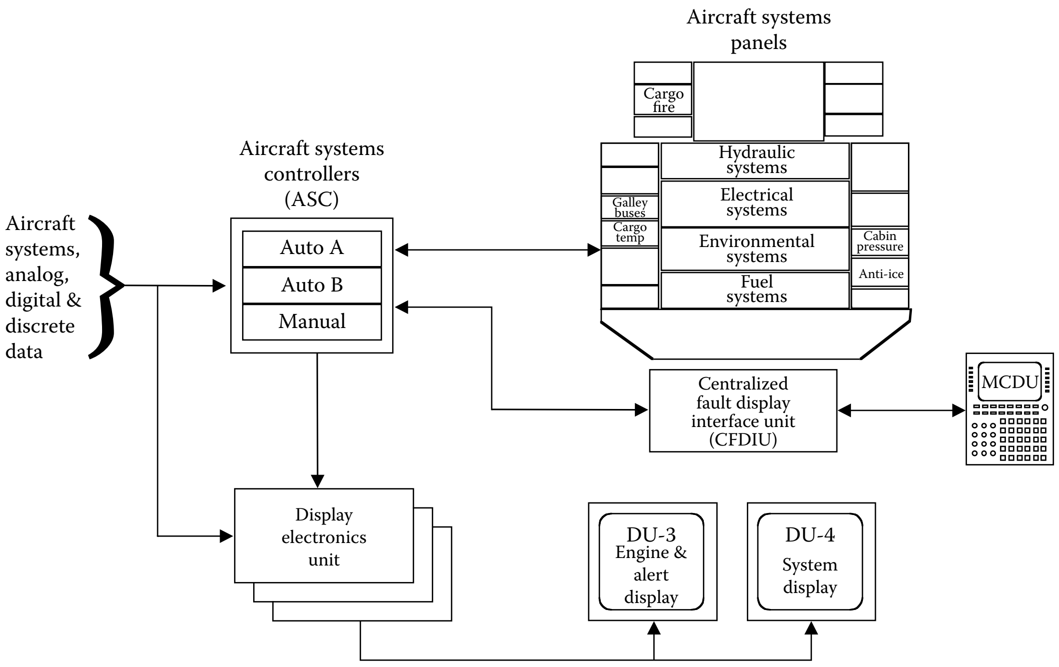


FIGURE 11.10 Generalized architecture for aircraft system controllers (ASC).

incorporated in a single standard version of that system, and the individual airline then selects the appropriate options by means of a program pin or software option code. Additionally, as improved versions of the systems are developed, each MD-11 operator receives the latest version of that system (with the new features, if there are any, being selected again by program pin or option code). The advantages of this are considerable. At any time there is only one version of each system in the field to be supported, the airlines stay abreast of system improvements and avoid obsolescence, and there is a single part number available as a spare for airlines, thus improving their availability at remote locations.

Most of the changes to the various avionics systems that have to be kept updated under this single part-number philosophy are software changes. Most of the major avionics systems are software loadable, that is they can have their software loaded via the front or rear connector *in situ*. This simplifies the airlines' task in keeping their aircraft current.

11.11 CNS/ATM Architecture

One of the major changes affecting aircraft manufacturers and operators today is the need to operate in the new Communication, Navigation, Surveillance/Air Traffic Management (CNS/ATM) environment. This began with the ICAO Committee on Future Air Navigation Systems (FANS), and its first in-service application was on the Boeing 747-400 FANS-1 package in the South Pacific. This introduces a number of new CNS features in the airplane avionics systems:

- Controller/Pilot Data Link Communications (CPDLC) to communicate with ATC
- Global Navigation Satellite System (GNSS) navigation
- Required Navigation Performance (RNP) certification
- Required Time of Arrival (RTA) navigation to control arrival times at waypoints
- Automatic Dependent Surveillance (ADS) to provide surveillance data to ATC and the airline

The MD-11's original avionics architecture lends itself well to adaptation for these new functions. In the MD-11 CNS/ATM architecture, the FMC provides the computing resources for the new functions,

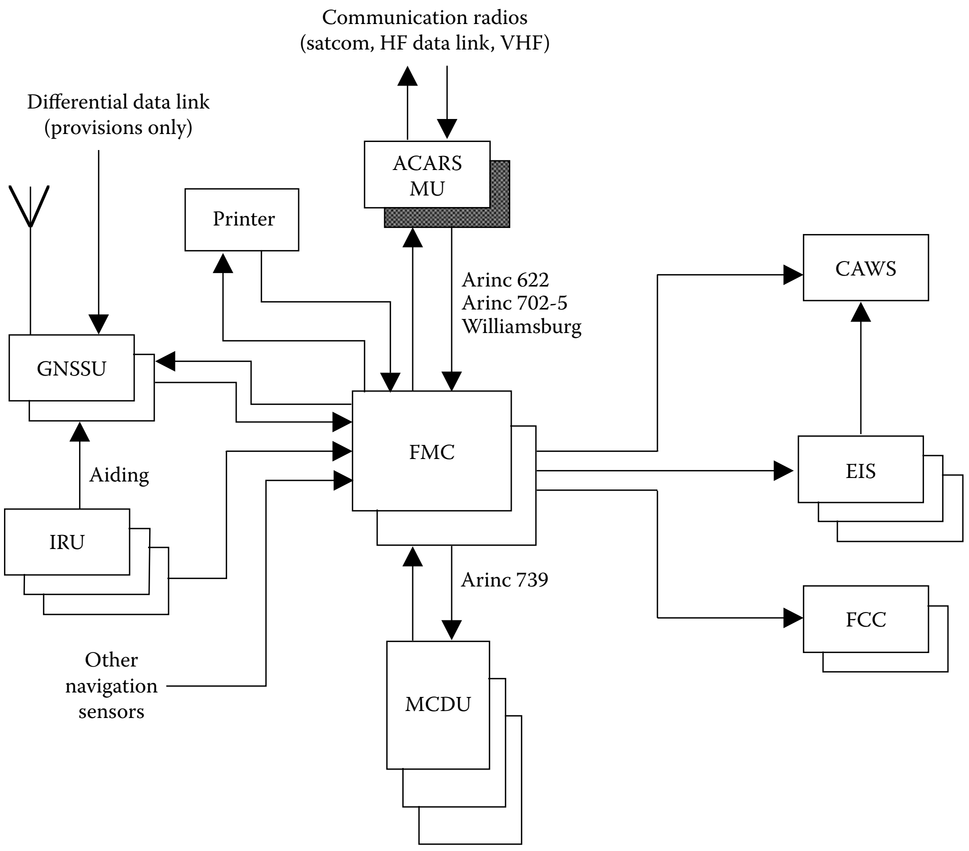


FIGURE 11.11 MD-11 CNS/ATM architecture.

with the ACARS MU (or CMU) used as a communications link to the ground via the SATCOM, VHF, and HF Data Link (HFDL) to the airline dispatch and ATC centers on the ground. The architecture is shown in Figure 11.11.

Having the FMC host the applications for the new CNS/ATM functions, and in particular the communication/surveillance functions of CPDLC, AOC, and ADS, allows changes to these functions to be largely limited to the FMC. Similarly changes to the communications protocols can be kept to a single LRU, in this case the ACARS MU. With the migration to the Aeronautical Telecommunications Network (ATN), the ACARS MU will be replaced by an ARINC 758 Communications Management Unit (CMU), which will provide the necessary application gateway between the ATN Open System Interconnect (OSI) protocols and the FMS.

The GNSSU provides position, velocity, and time (PVT) to the Flight Management System (FMS), which determines the multisensor navigation solution based on best available data, i.e., the IRS combined with GNSS data or data from whichever other radio is most suitable. GNSS data are not provided to either the IRU or the MCDU, since standby GNSS navigation is not provided.

The GNSSU is provided today, and will be sufficient for the en route, terminal area, and non-precision approach accuracies required. As GNSS comes into use as a precision approach aid, it will be replaced by a unit such as the ARINC 755 Multi-Mode Receiver (MMR) to provide both precision approach capabilities and the outputs for the FMS.

The printer is interfaced directly to the FMC to allow the crew to print the flight plan clearance messages transmitted by ATC.

Obviously the architecture selected for the MD-11 was not the only possibility. AEEC has documented in ARINC 660 a number of possible architectures, some of which adopt a federated approach where the Air Traffic Services (ATS) functions are split between the FMC and the CMU.

McDonnell Douglas has elected to use the integrated approach described here for a number of reasons:

- Keeping the FMS/CMU interfaces simple
- Ability to re-use existing FANS-1 designs in the FMS
- Avoiding having to certify multiple different ATS applications with multiple ACARS/CMU suppliers
- Ability to provide a common ATS capability for all operators and provide commonality with other airplane types

The existing Flight Management Computer was designed to provide sufficient capabilities for the features required at the time of the MD-11's entry into service in 1990. It could probably provide sufficient computational resources (throughput and memory) to allow inclusion of the FANS-1 functionality, albeit with some degradation of the existing response times. However, growth in this configuration would be very limited. It therefore makes sense for such a major change to introduce the new hardware that will provide the necessary growth for the future. This is the Pegasus processor. The Pegasus processor is an Advanced Micro Devices (AMD) 29xxx family processor chip set, as used on the Airplane Information Management System (AIMS) designed by Honeywell for the Boeing 777. (See Chapter 9.)

11.12 Derivatives

The MD-11 flight deck was very much the standard to beat at the time of its introduction to service in 1990. In most respects, it still is, although technology developments in some areas, such as flat panel displays, have gone beyond the capabilities introduced by the MD-11. The flight deck layout is, however, still widely regarded as one of the best in the industry.

It was only to be expected, therefore, that the MD-11 flight deck would form the basis for future flight decks on McDonnell Douglas* commercial transport aircraft. For these aircraft, the Advanced Common Flight Deck (ACF) was developed, with the capability to be adapted to the specific needs of the airplane type. The ACF is based on MD-11 flight deck layout, and in many cases MD-11 components, and takes advantage of a number of technology advances that have occurred in the last few years:

- High-speed Reduced Instruction Set Computer (RISC) processors
- Dual lock-step processing to monitor for hardware errors
- Time and space partitioning to allow software of mixed criticality to exist in the same box
- High levels of functional integration, as in the AIMS cabinet
- Flat panel display technology for the main display units and for the standby unit
- Integrated Air Data and Inertial Reference System (ADIRS)
- CNS/ATM capabilities

The ACF uses these technologies not just for the sake of using technology, but because each offers real cost benefits to the operators of the airplane and to the manufacturer. The result is the architecture shown in Figure 11.12.

The heart of the ACF is the Versatile Integrated Avionics (VIA). This unit provides a similar level of integration as is provided by the 48" AIMS cabinets, but contained in an 8MCU Line Replaceable Unit

*In August 1997, The Boeing Company and McDonnell Douglas Corporation merged. Following that merger, the McDonnell Douglas name disappeared, and products of the former McDonnell Douglas Corporation now appear under the Boeing name. The term *McDonnell Douglas* has been used here where it would be historically inaccurate to use the term Boeing.

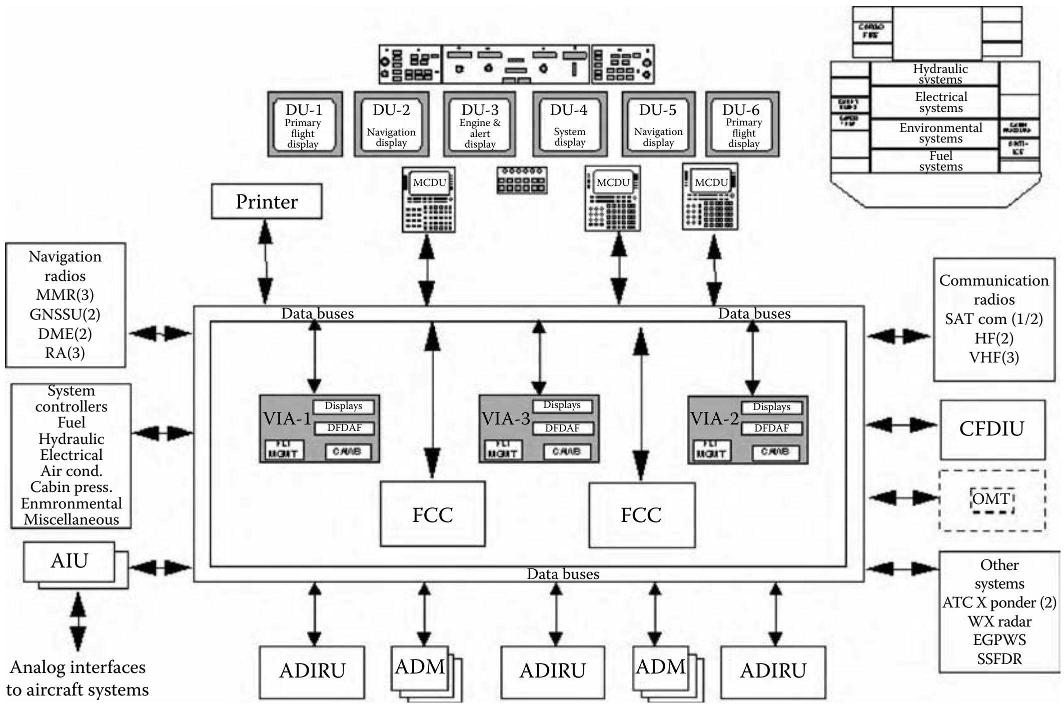

DU-1 Primary flight display
DU-2 Navigation display
DU-3 Engine & alert display
DU-4 System display
DU-5 Navigation display
DU-6 Primary flight display
MCDU
MCDU
MCDU
Printer
Hydraulic systems
Electrical systems
Environmental systems
Fuel systems
Navigation radios MMR(3) GNSSU(2) DME(2) RA(3)
System controllers Fuel Hydraulic Electrical Air cond. Cabin press. Enmronmental Miscellaneous
AIU
Analog interfaces to aircraft systems
Data buses
Data buses
Data buses
VIA-1
VIA-3
VIA-2
Displays
DFDAF
FLT MGMT
CAWS
FCC
FCC
Communication radios SAT com (1/2) HF(2) VHF(3)
CFDIU
OMT
Other systems ATC X ponder (2) WX radar EGPWS SSFDR
ADIRU
ADM
ADIRU
ADM
ADIRU

FIGURE 11.12 Advanced common flight deck (ACF) architecture.

(LRU). On a long-range aircraft, such as derivatives of the MD-11, three of these units will be installed to allow dispatch with one inoperative. On smaller, short-range aircraft, such as the MD-95, only two are installed. The VIAs will have the same hardware, and largely common software on all aircraft types. The key to this is the use of Aircraft Interface Units (AIU). These are data concentrator units that convert most of the analog data to digital form, and allow the VIA to process only digital data received via ARINC 429 databuses. MD-11-type control units (MCDU, Glareshield Control Panel, System Control Panel, etc.) are used to give the flight deck the look and feel of the MD-11.

This advanced flight deck is now in production for the Boeing 717-200 and a two-crew version of the DC-10 known as the MD-10. The diagram in Figure 11.12, in fact, shows the architecture for the MD-10 airplane. For application on a short-range airplane, such as the 717, the architecture is simplified by deleting one of the VIAs. Similarly, if Cat IIIb autoland capability is not required, some of the sensors (ADIRU, ILS, Radio Altimeter) can be reduced from triple to dual installations.

12 Genesis Platform

Randy Walter
Smiths Aerospace LLC
Chris Watkins
Smiths Aerospace LLC

12.1 Genesis Platform Concept

The Genesis Platform is a hardware/software platform that provides computing, communication, and input-output (I/O) services for implementing real-time embedded systems, known as hosted functions. Multiple systems can be architected and overlaid on the partitioned platform resources to form a highly integrated system with the unique characteristic of full isolation and independence of each individual system. As such, the Genesis Platform provides the starting point for the synthesis of a highly integrated real-time system, hence the name (Generic Networked Elements for the Synthesis of Integrated Systems). The platform elements are architected to maintain a high-integrity, fault-tolerant environment, necessary for hosting critical system functionality.

Genesis is a new class of integrated computing platforms that encompasses Integrated Modular Avionics (IMA) as well as other real-time computing platforms. It is targeted for supporting highly critical applications, but can be scaled to appropriately support lower levels of application integrity. Genesis is characterized by the following features and architectural advantages:

Significant Features:
- Integrated systems architecture
- Computing, communication, and I/O
- Network-centric communications
- Composable and extensible architecture
- Robust partitioning
- Real-time deterministic system
- High-integrity platform
- Fault containment
- Fail-passive design
- Asynchronous component clocking
- Change containment
- Open system environment
- Legacy Line Replaceable Units (LRUs) compatibility

Architectural Advantages:
- Reduced system size
- Reduced system weight
- Reduced system power
- Reduced system cost
- Reduced part numbers
- Minimized interconnect wiring
- Supported efficient sensor and effecter placement
- Minimized wire lengths for passive cooling, which in turn improves electrical signal quality
- Life-critical applications hosting capability

Hosted functions are allocated to the platform resources to form a "functional" architecture specific to each system to meet the availability, operational, safety, and topological requirements for each function. Hosted functions can "own" unique sensors, effectors, devices, and nonplatform LRUs, which become part of the functional system architecture. Multiple hosted functions share the platform resources within a virtual system environment enforced by partitioning mechanisms that are implemented as part of the platform design. The virtual system partitioning environment guarantees that hosted functions are isolated from each other; therefore, they cannot interfere with each other regardless of faults that may occur within the hosted functions or the platform resources. The platform design guarantees each hosted function its allocated share of the computing, network, and I/O resources. Those resource allocations are predetermined and communicated to the platform components via loadable configuration files, which become the source of information for the run time enforcement of the hosted function resource guarantees.

The Genesis Platform is a scalable platform that allows the integrated system architecture to be sized for each platform element. The number of computing elements, I/O elements, and communication links can be scaled to meet the needs of a specific implementation. This ability, along with diverse interface types offered with the I/O element, allows the Genesis Platform to be used at any level of system integration deemed appropriate, without the need to redesign the platform.

The Genesis Platform enables the implementation of highly integrated system architectures, which benefit aerospace and other industries with significantly reduced system weight, part numbers, and equipment cost while maintaining lower cost-of-change characteristics associated with federated systems.

12.1.1 Comparison to Traditional Federated Architectures

The Genesis architecture is provided in contrast to the traditional architecture characterized by federated systems. Federated systems are architected to provide the following services in each system:

- Separate processing
- Separate infrastructure
- Separate I/O
- Internal system bus

In addition, I/O is routed point-to-point between sensors/effectors and systems in a federated system architecture. A federated system architecture diagram is presented in Figure 12.1.

In contrast to federated systems, Genesis is architected to provide the following services for an integrated set of systems:

- Common processing with robustly partitioned application software
- Common infrastructure
- Specific I/O via shared, Remote Interface Units (RIUs)
- Distributed Systems Bus (Avionics Full Duplex [A664-P7] Network)

The Genesis architecture utilizes network-centric communications. I/O is routed using the A664-P7 network between the GPM (General Processing Modules), LRU, and RIUs to sensors/effectors and non-A664-P7 (ARINC 429, Controller Area Network (ISO 11898) (CAN)) busses. A diagram of the Genesis system architecture is presented in Figure 12.2.

As shown in Figure 12.2, the Genesis architecture presents a "virtual system" concept to replace the "physical" systems as packaged in a federated architecture. Figure 12.2 portrays four "virtual systems" that are intended to be equivalent to the four physical systems shown in Figure 12.1. The "virtual system" consists of the same logical groupings of components as contained by a physical system:

- Application software
- Infrastructure/Operating System (OS)
- Processor
- System bus
- I/O

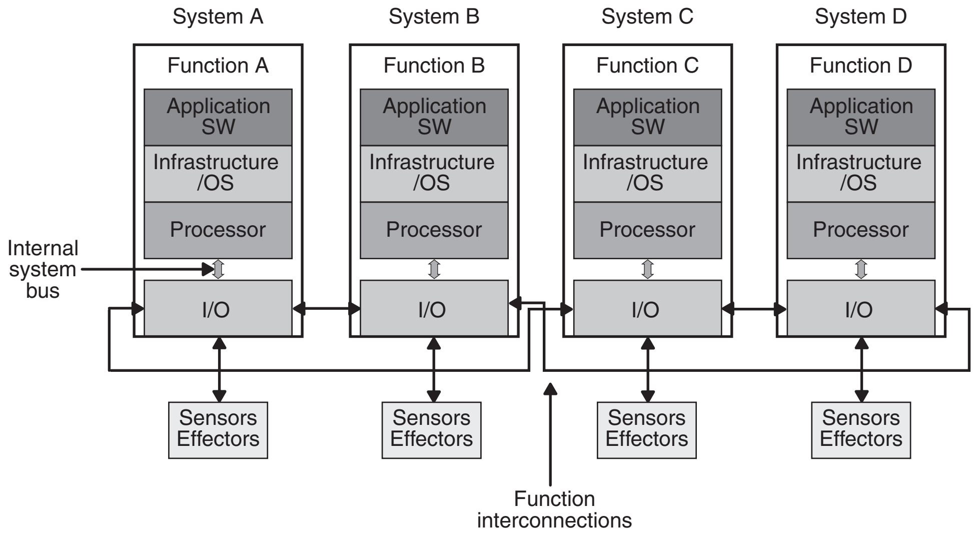


FIGURE 12.1 Federated system architecture.

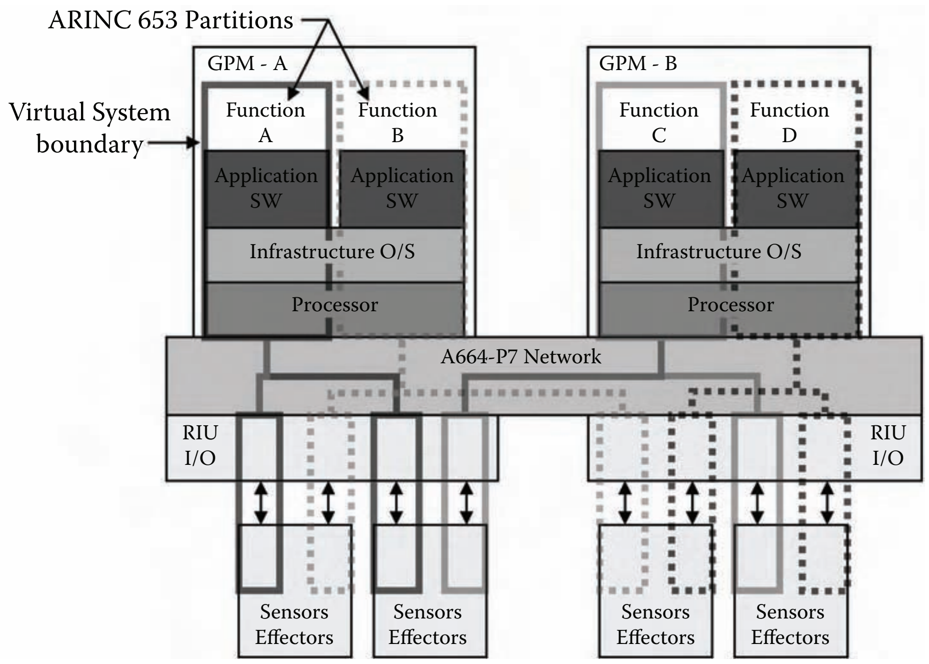


FIGURE 12.2 Genesis architecture identifying the "virtual systems."

Therefore, a key difference between Genesis and federated architectures is the definition of the logical system. In a federated architecture the logical system *is* the physical system. In the Genesis architecture, the logical system is different than the physical system and is thus referred to as a "virtual system."

12.1.1.1 Virtual Systems Compared to Physical Systems

In a federated architecture, the target computer and the software application are typically packaged as a "physical" system. The application is usually linked with the OS and Board Support Package (BSP), and the resultant software executable is verified as a single software configuration item. The application normally utilizes its own system bus to interface with its dedicated I/O. Multiple "physical" systems are then integrated in order to perform a specific set of aircraft functions.

In contrast, the major components of the Genesis architecture (GPM, A664-P7 network, RIU) provide a "virtual system" environment. The platform hosts the software application on a GPM, which is a computing resource shared between many software applications. The GPM hardware and platform software, along with configuration data developed by the system integrator, forms the equivalent of a target computer for RTCA DO-178B purposes. When a software application is integrated with the target computer, it forms a "virtual system." Multiple "virtual systems" can be provided by a single GPM (see Figure 12.2). The distinction between the application "virtual system" in the GPM and an application system in the federated environment is that the application "virtual system" in the GPM is a software configuration item without hardware. Each GPM is comparable to one physical system.

The "virtual system" concept extends to the A664-P7 network. The federated system's internal data bus is represented by a network-centric data communications environment. However with the Genesis architecture, the "virtual systems" share the A664-P7 network as a data transport mechanism. The Virtual Links (VLs) provide network transport partitioning for the application data messages. Each VL is allocated guaranteed network bandwidth (data size and rate) and maximum network delivery latency and jitter.

The "virtual system" concept also extends to the RIU, which is configured to provide I/O services for multiple "virtual systems." The RIU employs temporal partitioning mechanisms through scheduled read/write operations and allows for physical separation between I/O contained within multiple independent fault zones (IFZs) to segregate functional signals as determined by the system integrator. The IFZ boundaries ensure that RIU faults do not affect I/O interfaces outside of the faulted IFZ.

12.1.2 Platform Architecture

The platform consists of the following major components:

- General Processing Modules (GPM) to support functional processing needs
- RIUs to support system analog and serial digital interfaces (the working plant)
- Avionics Full Duplex (A664-P7) Switched Ethernet network for communication between platform elements

These elements, including the quantity of each element, can be packaged in a variety of ways to form a specific physical implementation of the platform. Elements can be packaged as LRUs (sealed units of replaceable equipment) or in module or card form. Modules and cards can then be grouped within cabinets or integrated LRUs that may share common resources such as power supplies and cooling.

The Common Core System (CCS) developed by Smiths Aerospace for the Boeing 787 Dreamliner is a specific implementation of the Genesis Platform that groups GPMs and some A664-P7 switches within a cabinet structure. The CCS implements the RIU as a Remote Data Concentrator (RDC) and the remaining A664-P7 switches as LRUs that are distributed throughout locations within the airplane to facilitate separation and minimize wiring to subsystems, sensors, and effectors.

The platform uses an "open system" environment that enables independent suppliers to architect and implement their systems on the platform by complying with industry standard interfaces at all levels within the platform. The key open standards utilized are ARINC 653, ARINC 664, ARINC 429, ARINC 665, and ISO 11898 (CAN bus).

The Genesis Platform is an asynchronous system, meaning that component schedules are independent of each other. Each unit internally controls when data is produced; there is no attempt to order operations between units at the platform level. This decouples the elements at the network interface, helping to prevent individual unit behavior from propagating through the system and perturbing the operation of other units. This unit-level independence emulates the federated system environment, producing the same system level characteristics. Applications must account for this environment in their design, avoiding implementations that depend on synchronous data behavior. Asynchronicity makes the total system more robust by eliminating any dependency on synchronization mechanisms.

Genesis is a configurable resource platform. Functions are allocated the resources they require to perform their task in the form of sufficient processing time and memory, network I/O communication, and interface resources for both analog signals and other digital bus types. These resource allocations are mechanized within the platform through specific configuration tables loaded into each platform unit. The configuration tables represent the resource allocations that are guaranteed to each function to perform its task. These resource guarantees, along with the platform's system partitioning characteristics, form the cornerstone of hosted system independence and, therefore, change containment within the system. These properties allow individual functions to change without collateral impact to other functions. This platform characteristic becomes the certification basis for incremental change, enabling unit acceptance for certification at the individual function level instead of the entire integrated system level (all functions hosted on the platform).

The Genesis Platform architecture represented in Figure 12.3 depicts the building-block approach utilized to scale the platform for a specific implementation. The major building blocks consist of computing elements, network elements, and I/O elements. The platform is scalable by adding (or subtracting) building blocks for each of the resources depending upon the specific needs of the hosted functions for a given set of systems. Utilization of this scalability attribute does not alter the fundamental architecture or operation of the platform and does not impact the existing hosted functions as long as their associated resource allocation guarantees remain intact.

Figure 12.4 shows a generic architecture for a fully integrated system, and Figure 12.5 shows the architectural implementation for the 787 CCS. The main difference between these two architectural representations is that the 787 CCS implementation minimizes the interconnect wiring between network switches. From a strategic standpoint, the system integrator must weigh the cost of the switch interconnects to the level of flexibility and redundancy afforded to the network communication paths.

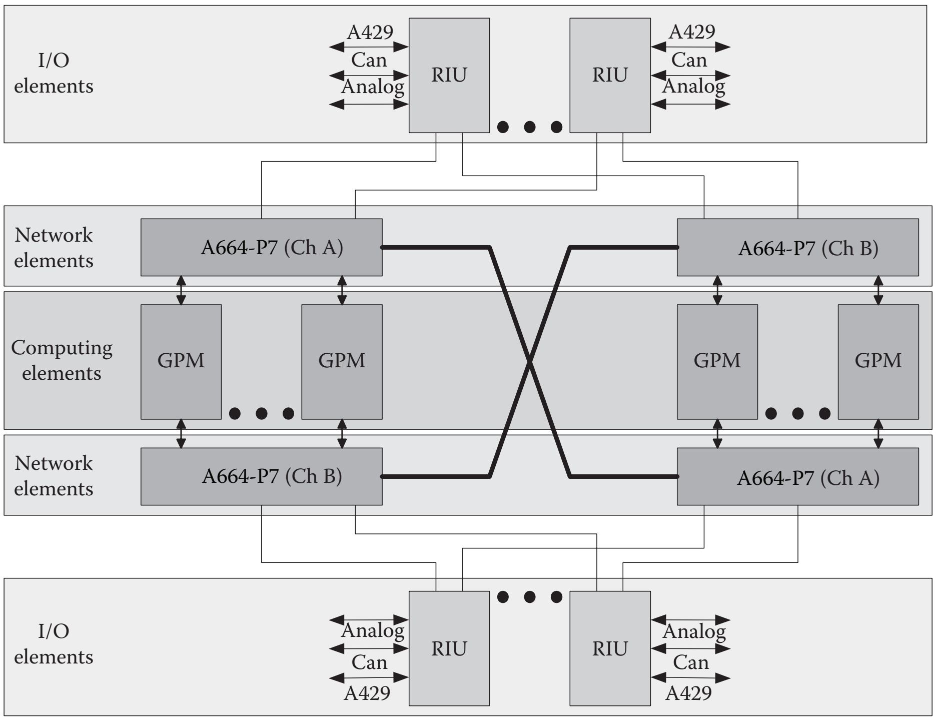


FIGURE 12.3 Platform building blocks.

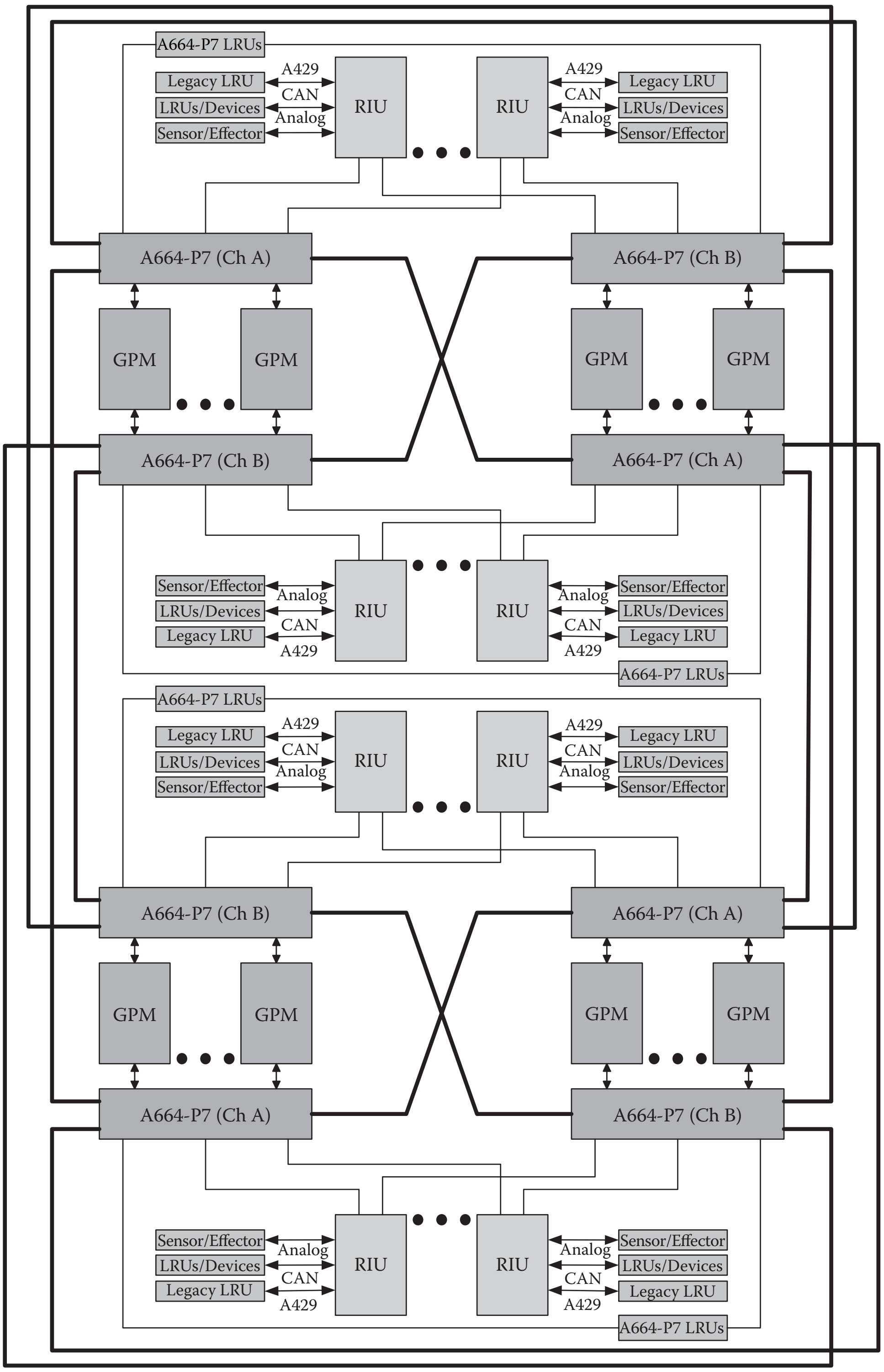


FIGURE 12.4 Integrated system architecture.

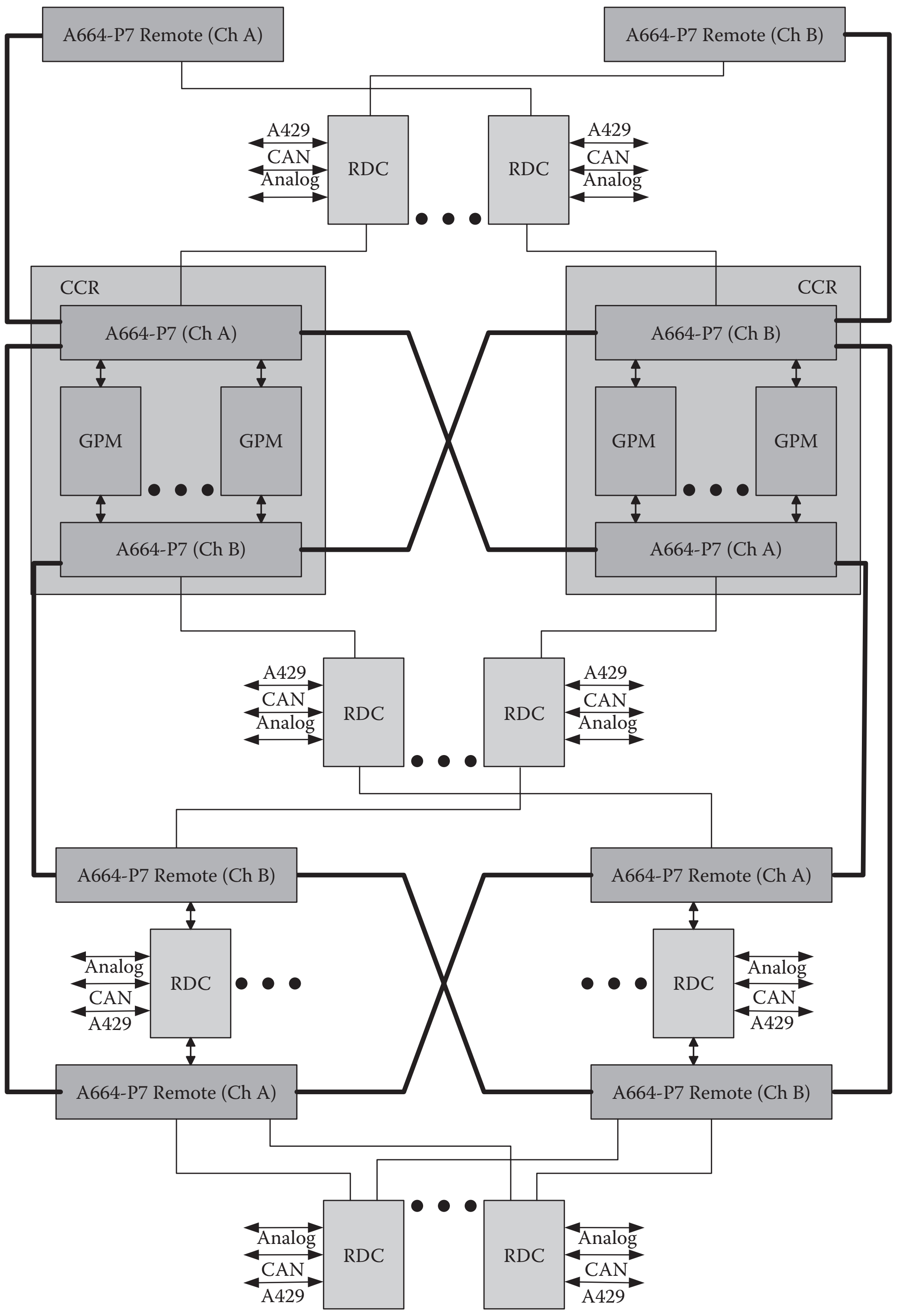


FIGURE 12.5 787 CCS platform implementation.

The Genesis Platform readily accepts integration of other "specialized" or legacy LRUs into the overall integrated system by providing interfaces for the network and other serial digital interfaces for ARINC 429 and CANbus. This allows the architectural flexibility to integrate "specialized" functions that may require unique hardware.

The platform components are located to provide physical segregation for redundant system elements as well as the network and I/O topology to minimize interconnect wiring. Sensors and effectors can be positioned where they perform best or are needed. The RIU is a remote, "bolt on" LRU designed for passive cooling. This allows signal conversion and concentration onto the platform communication media to be done in a manner local to the source, negating the need to run several long signal wires to a central location. This not only reduces installation weight but also improves signal quality for analog device interconnect.

12.1.3 Genesis Platform Major Component Characteristics

Certain attributes are required for each major component for the platform to provide system independence and an environment that simplifies implementations of all hosted functions. These attributes dictate integrity, availability, partitioning, and fault containment requirements for each component. The platform must stay "function neutral" to minimize the cost of change and to maintain the system flexibility. This concept drives certain system architectural decisions concerning how Genesis accomplishes its platform mission.

12.1.3.1 GPM Characteristics

The GPM is an independent computing platform that hosts core software and provides the hosted applications a robust partitioned environment and infrastructure services including I/O, health monitor, and nonvolatile file storage and retrieval based on the ARINC 653 standard. Execution time window, cyclic period, and memory needs, as well as remedial recovery action for process and partition level faults, are allocated to each application and conveyed to the core software through configuration files specific to each GPM. All these allocations are enforced by the partitioning mechanisms designed into the core software and hardware. Likewise, each application specifies its input and output data needs in the form of messages, which form a logical data group of one to many parameters. These groupings represent ARINC 653 data ports, which form the direct communication link with an application.

Application communication can be internal, A653 to A653, or external A653 to network. Internal A653 to A653 communication is memory-to-memory transfer. It is a facility used for GPM-specific Application Program Interfaces (APIs) and when necessary to meet demanding latency requirements for multiple partition applications residing in the same GPM. When an application requests access to an assigned 653 port mapped to the network, the core software executes network end system interface drivers to read or write the requested message from or to the network end system message buffers. These network message buffers are referred to as network communication ports (com port).

There is a direct correspondence between a 653 port and a specific com port (Figure 12.6). The core software has knowledge of the 653/653 and 653/com port mapping from the application-specific portion of the configuration file. Both A653 and com ports can be sampling or queuing. Sampling ports overwrite old data with the latest update. The contents of the sampling port remain after being read, allowing unlimited reads of the port. If a periodic data source ceases to provide updated data, then the last receipt of sampling port data remains and is left to grow "stale." Queuing ports aggregate messages in the queue buffer until read by the host. If the queue buffer is filled prior to the host reading it, any intervening messages are lost. Once the host reads the queue, the contents are emptied until the next message arrives. The time required to access the com port is part of the execution time allocated to that application. Most of these network com ports are actually User Datagram Protocol (UDP) ports, used in conjunction with the Internet Protocol (IP) from a protocol point of view. However, applications do not have to be aware of the underlying network protocol. The platform does provide a trivial file transfer protocol (TFTP) library function that can be included in any partition.

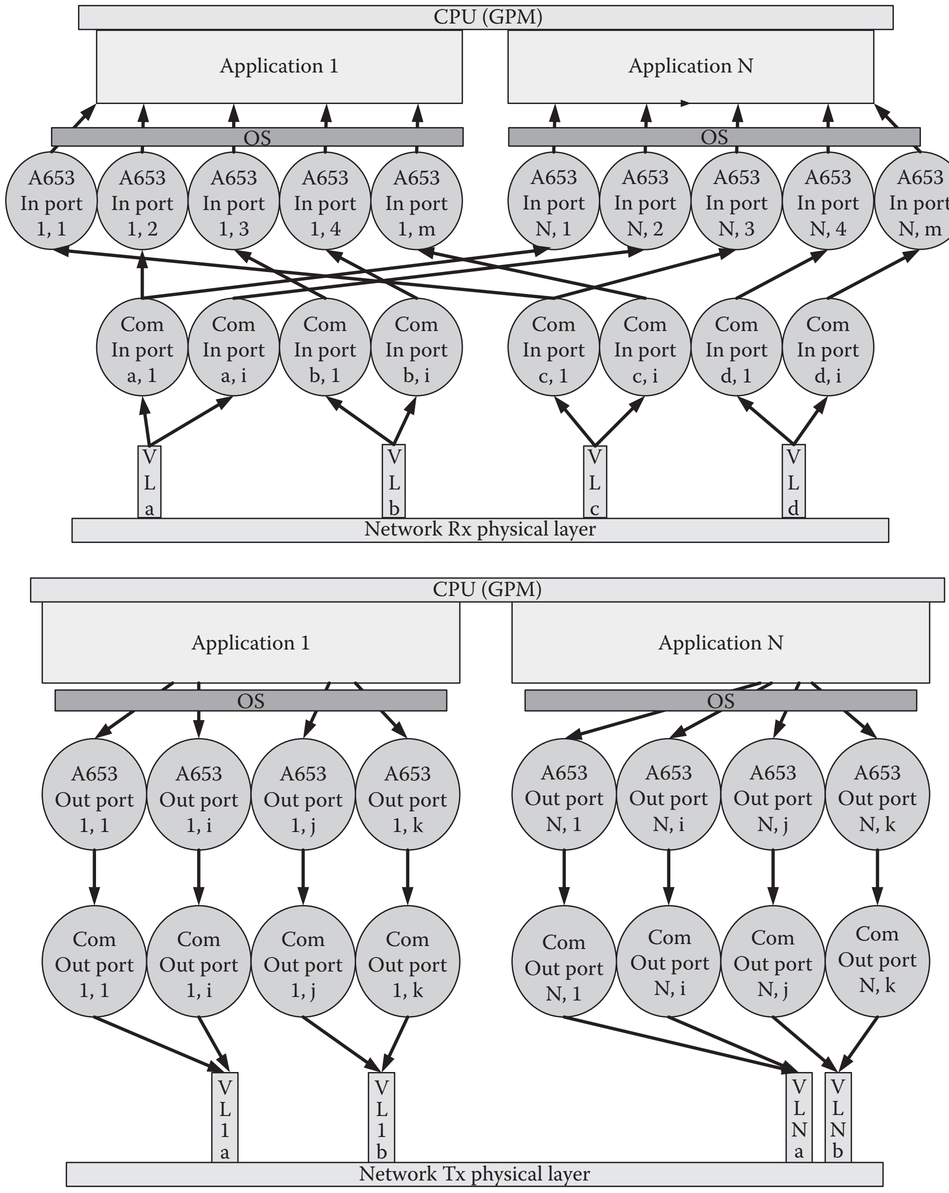


FIGURE 12.6 A664-P7 and A653 port mapping.

One or more com ports from a single application are assigned to a VL, which provides the network transport partitioning for the applications data messages. When data is written to a com port, it is transmitted on the network when the VL is eligible for transmission. Eligibility is managed through the communications mechanism for bandwidth regulation, referred to as the Bandwidth Allocation Gap (BAG). The data message is immediately transmitted if it is BAG-eligible and the transmission queue depth is zero. The transmission is delayed if it is not BAG-eligible or the transmission queue depth is not zero. Com port data messages are defined by their source application in logical data groupings by virtue of their associated ARINC 653 port definitions. Assignment of com ports to VLs are made by the integrator based on identifying both targeted and general consumers of particular data flows and upon transmission performance constraints.

The GPM hardware is designed to be fail-passive, which means there are no single faults that can cause erroneous behavior and that undetected fault sequences resulting in erroneous behavior are extremely improbable (consistent with critical function category requirements). This means that each GPM has autonomous integrity properties, allowing critical functions to be implemented without the need for cross-channel consolidation schemes (voting) between redundant computing elements to achieve the required integrity. Redundant applications may choose to synchronize their functional states using communication links provided by the A664-P7 network. Redundant applications for the computing domain are only necessary to achieve function availability requirements. If a GPM fault occurs, the GPM will fail "silent," ceasing network communication. Data that is sourced by applications contained within the GPM will quickly go "stale" on the network, conveying the application failure to any subscriber of that data. Likewise, a debilitating failure contained within application-specific system resources will result in no data transmission for that application. This is enforced via core software health monitor mechanisms. The fail silent behavior embodies the cornerstone for system redundancy management and fault tolerant system behavior. A faulted GPM will reset and attempt recovery, and data transmission will resume only after successful recovery to normal operation.

12.1.3.2 A664-P7 Network Characteristics

The communication backbone of the platform is an ARINC 664 part 7 network (A664-P7) comprised of the network end system hosted in each connecting end node and multiple network switches. The network is arranged in a dual-channel switched star topology with each end node having a redundant, full duplex point-to-point connection with two independent communication pathways (A and B channels). End systems may only employ a single connection to the network through a single switch if their availability requirement allows. The A/B dual-channel connection allows redundant transmission and reception of data through two independent network paths, ensuring that a loss of communication is extremely improbable. The channel redundancy is managed in the end system, which is contained in each end node host; therefore, the end system only presents a single data stream to the consuming host and only requires a single data stream from the source host. If there is a loss of a redundant data path, the end system continues to stream data from the alternate channel. The loss is essentially a transparent event for the data publishers and subscribers; however, a separate indication for the loss of channel redundancy is provided to the virtual systems that depend on network redundancy for their functional availability.

Figure 12.7 depicts the suggested A664-P7 network switch interconnection topology. Notice that there are two separate redundant channels of communication (labeled as A and B channels). Each switch on a channel is directly connected to all other switches on that channel. This topology of interconnected switches maximizes the capabilities for redundant network paths and minimizes the impact when a switch is lost. The topology allows the system integrator to best optimize the network routes during initial installation and provides the greatest flexibility for future growth of the virtual systems by minimizing bottlenecks in the network paths.

Each network channel is designed to contain all faults that may occur between Tx and Rx UDP ports, meaning that there are no single faults that can cause erroneous behavior and undetected fault sequences and that erroneous behavior is extremely improbable (consistent with critical function category requirements). This fault-containment behavior is mechanized using an end-system-to-end-system integrity algorithm implemented in the network end system, which allows the redundant channels to be used for high availability and ensures that function integrity is not compromised by the network, eliminating the need for application cross-comparisons of multiple communication paths to achieve communication integrity. A dual end system in a synchronized mode can be implemented for full fail-passive end system behavior.

Network data partitioning is enforced using VLs as depicted in Figure 12.8. The VL is the network transport "pipe" for data packets from one publisher to one or more subscribers, analogous to an ARINC 429 point-to-multipoint bus. VLs have guaranteed allocated network bandwidth (data size and rate) and guaranteed maximum network delivery latency and jitter. Further, the ordering of data packets within the VL is maintained from publisher to subscriber. The network switches route VLs from a configured input physical port to a configured output physical ports or ports. During this process, the switch checks

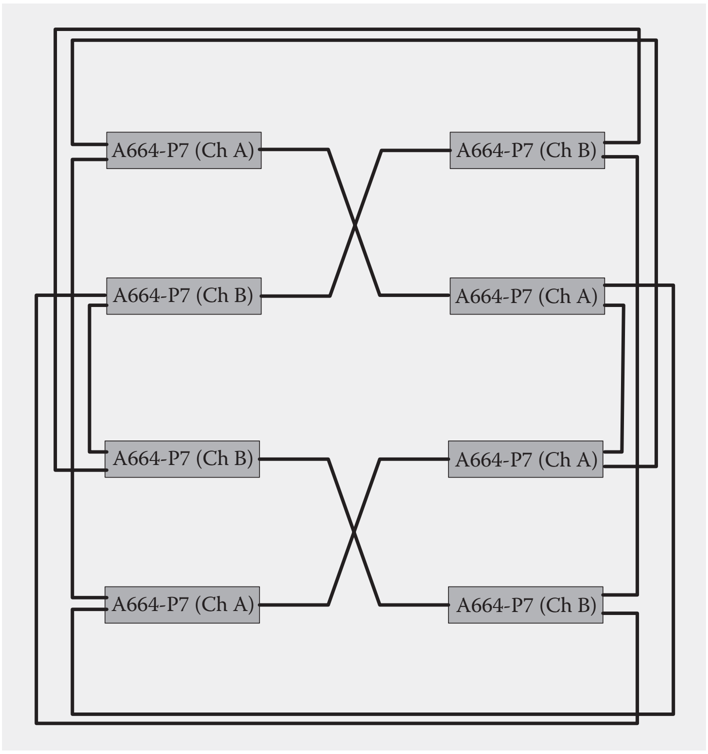


FIGURE 12.7 Topology for A664-P7 network switch interconnection.

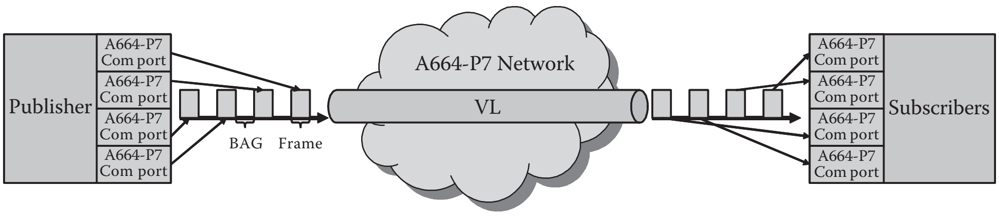


FIGURE 12.8 Virtual links.

to ensure that the size and transmission rate of each VL does not exceed its allocation. If these limits are exceeded, the messages are dropped, thereby containing any data flow misbehavior and enforcing the network partitioning. The switch has knowledge of these allocated VL attributes through network configuration files.

As shown in Figure 12.9 VLs can be comprised of up to four sub VLs, which carry messages from their associated com port buffers. The sub VLs are utilized in order to regulate data flow within the VL by "prioritizing" VL messages through sub VL groupings. The sub VL transmissions are serviced round-robin within each VL. The com port buffer represents messages of one or more parameters from a single source. Com ports are the basic data element and are statically configured to be updated at different rates or aperiodically. Messages will only be transmitted when they are written by the source publisher and the allocated VL is BAG eligible, which represents the predefined transmit interval for the group of ports assigned to a VL. If a message is larger than the maximum payload size for a network frame, then the message is segmented into multiple network frames for transmittal. A buffered network frame is limited in size according to the VL-defined maximum frame size, and only one network frame can be transmitted during each BAG interval. BAG is the primary means for regulating data flow through the network and protects the network from babbling sources.

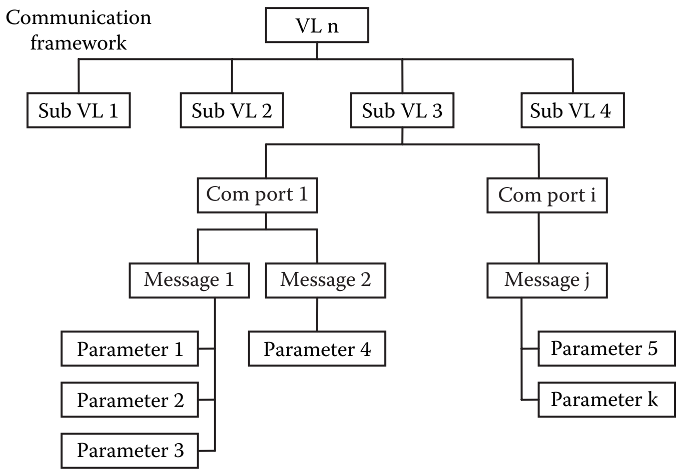


FIGURE 12.9 Communication framework.

VL priorities can be configured at the switch ports to increase performance for performance-critical messages. In such a configuration higher-priority messages are routed through the switch port before lower priority messages. The message delivery performance is exchanged between the VL priority levels, thereby decreasing the guaranteed transport time for higher priority VL messages and increasing the guaranteed transport time for lower priority VL messages. The system configuration tools include traffic analysis to ensure that high priority VLs won't be able to "starve" out low priority VLs by consuming too much of the switch port bandwidth.

The A664-P7 end system can be configured for various modes of operation depending on the host's particular needs. It supports dual channel communications (referred to as network channels A and B), which can be configured by VL for dual redundant operation (identical data on both channels), independent operation (different data on both channels), or single operation (only one channel active). For dual redundant operation, the end system can be configured by VL to perform the redundancy management between the dual channels, allowing the host to instruct the end system to select between redundant messages and provide a single message presentation to the host. In this configuration, an error in either network channel is transparent to the host.

12.1.3.3 RIU Characteristics

The RIU is the gateway between the A664-P7 network, analog devices, legacy A429 busses, and linear CAN subnets. As such, the RIU provides analog-to-digital and digital-to-analog conversion services along with network formatting, range checking, scaling, offset, linearization, threshold, and filter services that are specific for each signal. An analog device interface such as a pressure sensor, valve, motor, or synchro can be comprised of one or many analog signals. Devices such as sensors may require excitation by the RIU. The RIU provides conversions at the analog signal level (also known as electrical primitive) but treats the appropriate grouping of signals as an interface. It has knowledge of the electrical primitives (by pin) assigned to specific interfaces along with the other selected services through configuration files.

The RIU provides a digital-to-digital gateway for A429 and CAN linear subnets. Fundamentally, the RIU accumulates bytes received from these busses in buffers specific to each connected bus and maps this data to specific A664-P7 com ports within its A664-P7 end system. Similarly for transmissions, the RIU retrieves bytes from specific com ports in its A664-P7 end system and maps these to transmit buffers for each connected bus. The RIU has knowledge of the A664-P7 com port and connected bus mapping through configuration files.

The primary partitioning method for the RIU I/O complement is to provide multiple I/O IFZs using physical separation to segregate functional signals as determined by the system integrator. To that end, each RIU contains several I/O independent fault zones that share a common high-integrity A664-P7 ES

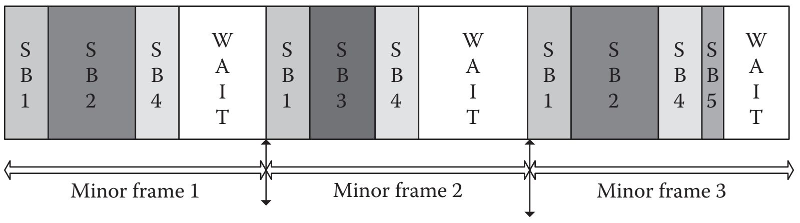


FIGURE 12.10 Major/minor frame scheduling policy.

interface. The A664-P7 ES interface is designed in a manner to prevent cross-corruption between IFZs that share the interface, allowing the system integrator to ensure segregation between functional I/O signals by grouping function specific signals within the same IFZ. Gateway operations (both analog and digital) are performed on a repetitive cyclic schedule specified in the configuration file. The allocated schedule (or update rate) is specified for each interface and each connected digital bus. There is a maximum rate for any gateway service defined for the architecture. The interface schedule consists of a major frame within which there are several minor frames. The duration of both the major frame and the minor frames is a configurable fixed period. Each minor frame consists of a number of scheduling blocks (SBs) that provide I/O processing, which means that all scheduling blocks and associated I/O will have a guaranteed period within which they will be run. Each minor frame will be implemented in such a way that a degree of free time (wait function will loop until a given time) will be allocated at the end (see Figure 12.10). This has a two-fold advantage. First, it ensures that SBs cannot overrun into the next minor frame. Second, depending on the amount of spare time that has been allocated, the free time allows for a certain amount of future expansion.

The RIU maps between interfaces and A664-P7 com ports. The mapping granularity can be allocated to parameters within the com ports or the entire com port depending on the specific interface. Analog signals always map at the parameter level. ARINC 429 interfaces always map at the com port level and CAN interfaces map at either level. The RIU has knowledge of this mapping through unique configuration files loaded into each RIU.

The RIU architecture is comprised of two distinct integrity regions, one that is shared across all resource users and a second that provides independent resources that can be used on an individual user basis. The shared region is designed such that either there are no single faults or fault sequences that can cause undetected erroneous behavior are extremely improbable. The independent resource region is designed such that undetected erroneous behavior is improbable. The importance of this architecture is that it prevents faults in the common resource from causing simultaneous malfunction of the independent resources and the functions that use them. This architecture eases the safety considerations associated with resource allocation of the RIU complement and cross-corruption of functions. Each individual RIU output is forced to a safe state in the presence of a fault or absence of a valid command for that interface. Likewise, a detected fault in any analog or gateway input interface is tagged as invalid data in the transmitted network packet. RIU service guarantees are enforced by an independent safety monitor that uses a separate time base from the processor. The safety monitor must be "kicked" by a keyword from the processor within a time window, otherwise the outputs will be put in a safe state via a hardware mechanism.

12.1.4 Platform Integrity and Fault Containment

Platform integrity is characterized by the following properties:

- Redundancy
- Fault isolation mechanisms
- "Virtual System" partitioning
- Fault containment zones

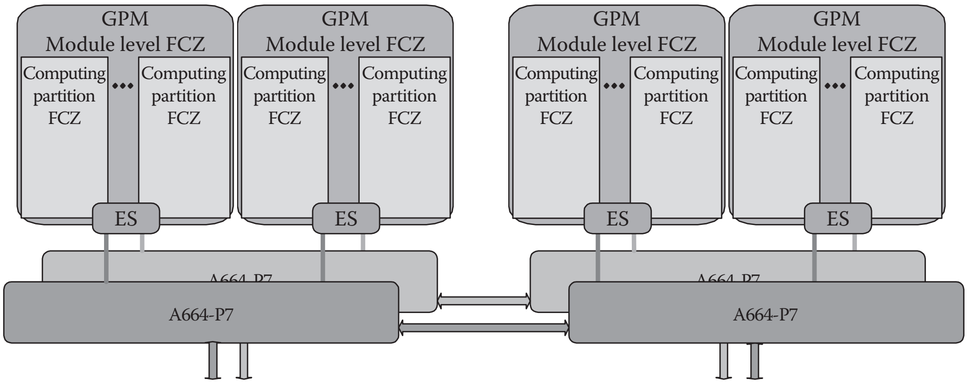


FIGURE 12.11 Computing FCZs.

As part of the Genesis architecture, redundancy is provided for all physical components of the system to meet the fault tolerance objectives for the functional systems. The fault isolation philosophy is to contain faults at the Genesis "virtual system" boundaries, allowing redundant elements to continue with a seamless flow of data. The concept of a "virtual system" describes the architectural feature in which a function is contained within a portion of a larger physical system. Multiple "virtual systems" can reside within one physical system. The "virtual system" is isolated through temporal and spatial partitioning mechanisms as opposed to physical partitioning mechanisms that define the boundaries of a physical system. An integral characteristic of the "virtual system" is that faults must be contained within the boundaries defined for the "virtual system."

GPMs are designed for fail-passive operation. There are no single fault exposures for erroneous behavior, and undetected fault sequences are extremely improbable. The fault containment zone for the computing resource is the "virtual system" defined at the partition level and bounded by the physical boundaries of the GPM. Module level failures that affect all partitions are contained within a GPM as shown in Figure 12.11.

Since each network channel (A and B) is designed to detect and contain all faults, the fault containment zone for the network forms the boundary of the A664-P7 UDP port in the transmitting and receiving end system for each channel. Most VL transmission faults are isolated at the switch boundary, and all faults are isolated at the receiving A664-P7 ES port level boundary. This prevents data corrupted during transport within the network from propagating to the host, as shown in Figure 12.12.

The I/O domain (RIU) leverages the fact that sensor/effectors and A429 buses are not (generally) high-integrity devices and require downstream consolidation to achieve high integrity, so any major essential or life-critical system requires multiple independent channels for I/O. The RIU architecture provides multiple independent fault zones within each RIU. Each IFZ provides fault detection for internal faults, which is generally an order of magnitude better than the sensors/effectors they interface to the system. However, the design of the shared resources within the RIU prevents fault propagation across multiple IFZs. Containment of all I/O faults, as in a federated system architecture, is still incumbent upon providing physically separated signals through independent fault zones. This allows consuming applications to use Downstream Consolidation (DC) methods to detect I/O faults, including the remaining small percentage attributable to the RIU. This is shown in Figure 12.13.

This fault-containment scheme has implications for those hosted functions (essential systems) that do not require multiple-channel I/O. These systems require sufficient system segregation at the I/O level to preclude misbehavior of multiple essential systems due to an undetected fault in a single RIU. To that end, the RIU complement provides independent fault zones to achieve segregation of essential system I/O.

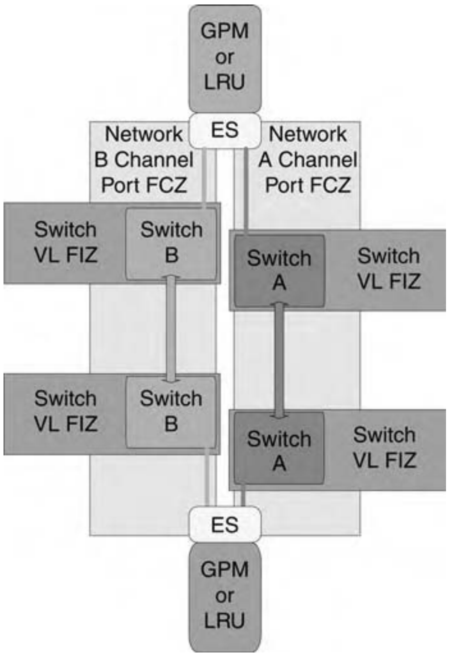


FIGURE 12.12 Network FCZ.

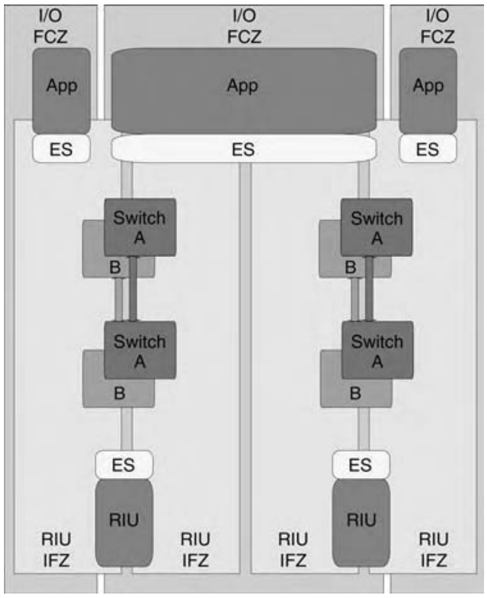


FIGURE 12.13 I/O FCZ.

12.1.4.1 How a Hosted Function Identifies a System Fault

The platform fault-containment zones form the foundation for communicating platform faults to hosted functions. Computing, network, and I/O faults are “seen” directly by hosted functions through platform component “fail silent” attributes. If a platform fault occurs, published data will stop updating within the A664-P7 com ports or dataset functional status will indicate a dataset specific fault. Each hosted

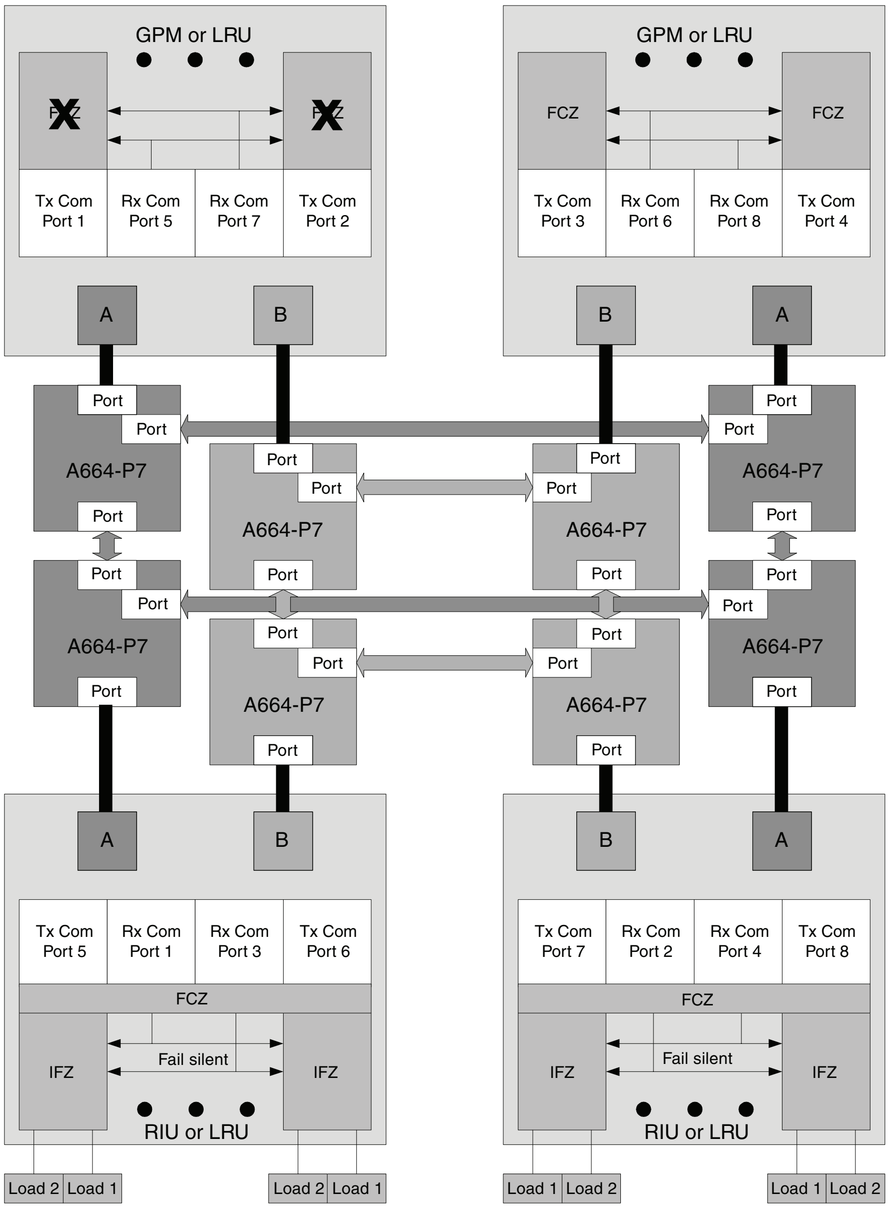


FIGURE 12.14 GPM module-level fault.

function can only recognize faults applicable to its subscribed data. This allows each function to determine its own functional response to the data failure. The A664-P7 end system reports network A and B channel status directly to hosted functions as header information associated with each com port. For hosted function effecter outputs, the RIU returns an output status message that includes output monitor values and network redundancy status for the output command. This allows hosted functions to monitor their analog output interfaces and a view of their communication redundancy.

The example platform fault scenario shown in Figure 12.14 depicts the platform fault indications used by hosted functions to determine their functional health if a GPM fails:

- All data published by partitions within the GPM fail silent (stops transmitting).
- This is seen by subscribers of data as A664-P7 Rx com ports going stale as determined by the difference in end system (ES) receive time compared to current ES time.

- Each subscriber determines how long to wait before declaring data too old.
- ES receive time is contained in each A664-P7 Rx com port header.

12.1.5 Platform Fault Tolerance

Platform fault tolerance is characterized by the following properties:

- Redundancy
- Fail-silent behaviors
- Data consolidation (at A664-P7 end system and hosted function)

Functional fault tolerance is architected into the "virtual system" by allocating sufficient redundant functional elements to support the required availability of that function. This can take the form of multiple copies for processing elements and multiple "virtual" channels (independent fault zones) for I/O elements. The consuming functions are provided redundant copies of input data from multiple independent sources. As mentioned in earlier sections, the fault detection and containment philosophy used by the platform is for the source to cease valid data transmissions in response to uncorrectable hard or soft faults. This characteristic becomes a built-in data validity indication to consumers of that specific data because the receiving port stops receiving "new" data. Receive-time tagging of the message ports allows the consumer to determine the freshness of data and decide when a source has gone "invalid." This allows the consuming function to select a source or sources based on its view of received data validity. The interconnecting platform network is a dual-channel arrangement that the network end systems manage. Network messages are typically configured for redundant transmission on each independent network. The receiving end system actually uses only one of the redundant messages to present to applications based on a redundancy algorithm, discarding the redundant copy. Any error in the transmission of a message due to either hard or soft faults is effectively screened from the functional system. From a systems standpoint, the data consumer sees a seamless flow of "good" data until the aggregate redundancy is exhausted. Applications may also implement data consolidation schemes for redundant sources to augment the integrity of source data that have insufficient standalone integrity. Figure 12.15 represents a simplified diagram depicting the hosted function redundancy supported by the architecture.

12.1.5.1 GPM Fault Tolerance

The A653 port read services hosted as part of the OS on the GPMs use the A664-P7 time tagging capability to set the A653 validity flag associated with the A653 sampling port. A similar mechanism is provided for A653 queuing ports, which will be reflective of "new data received since last read." This becomes a direct indication of data source failures for hosted applications. For the GPM, infrastructure software will only write application-requested data (through A653 port service calls) into the assigned A664-P7 communication ports if that partition and the associated platform hardware are healthy. This property supports the fail-silent behavior for applications hosted on the GPM acting as data sources.

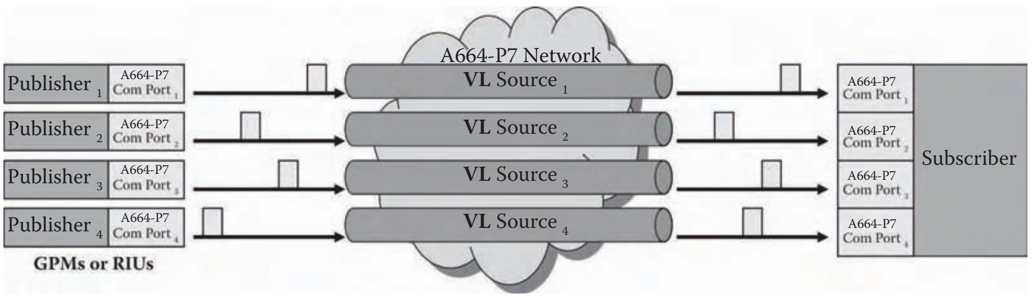


FIGURE 12.15 Generic redundancy architecture.

12.1.5.2 RIU Fault Tolerance

The same data source redundancy management scheme applies to signaling sources represented by the RIU. However, since the I/O hardware string (including sensor) generally has an undetected failure rate that is improbable, the consumption scheme for higher integrity functions typically is to receive data from all valid redundant I/O sources. It is incumbent upon the consuming application to provide integrity augmentation using downstream consolidation schemes (such as voting). The RIU ceases transmission of data on the network if it detects an internal RIU fault that affects all the data within a com port. Similarly, the RIU flags data as invalid if a fault associated with that specific data is detected.

The RIU can source-select between redundant data to drive a specific output based on a predetermined order of consumption for redundant sources. Redundant source data is represented as a set of com ports that the RIU will read and use in a specified priority order. The consumption scheme uses the first com port with valid data to drive the specific output.

12.1.6 Platform Health Management and Fault Isolation

The platform health manager provides the following services:

- Provides platform fault isolation
- Monitors connectivity of all platform and nonplatform LRUs
- Communicates configuration status to a higher-level health manager
- Controls return to service tests
- Determines when data-load of CCS LRMs (Line Replaceable Modules) and LRUs is possible

The platform has a built-in health manager function that aggregates and reports the health of the platform and its components (Figure 12.16). Based on its internal built-in test equipment (BITE), each platform LRU periodically reports its health to the platform health manager. The platform health manager also monitors the connectivity of all LRUs interfaced to the network using a simple request and response protocol. The health manager controls return to service tests (initiated BITE) and data-load of platform units to ensure the integrated system can remain operational during these maintenance actions. The platform health data is correlated by the Genesis health manager to provide definitive fault isolation for the platform, supporting both production build and operational maintenance. The health manager also indicates the dispatch readiness of the platform resources based on a predefined minimum equipment list for the platform unit complement, indicating any platform failures that would prevent safe dispatch.

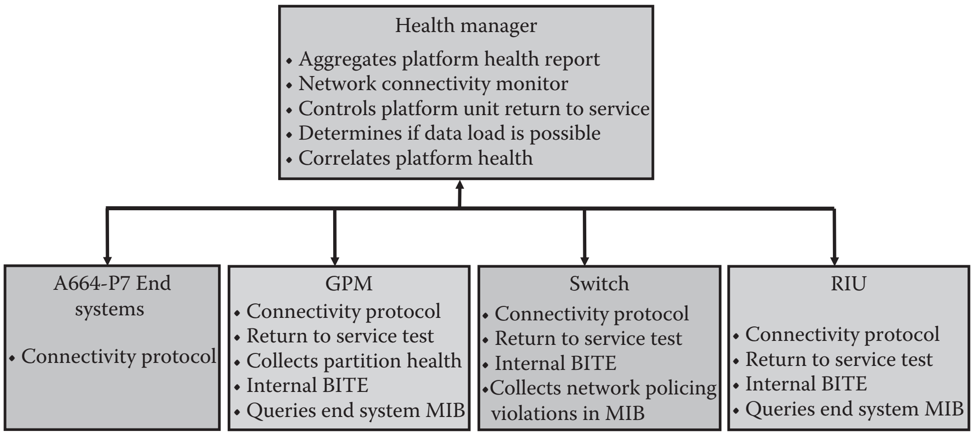


FIGURE 12.16 Genesis health management.

12.1.7 Platform Configuration Management

The platform configuration management function provides the following services:

- Validates hardware and software compatibility
- Provides consistency check for hosted function software
- Communicates configuration status to a higher-level configuration manager
- Generates a part-number report for maintenance personnel

The platform has a built in configuration manager that ensures that the installed platform unit level hardware and software are compatible with the installation and with each other. The compatibility levels that are validated by the configuration manager are unit-level and system-level compatibility as well as hosted function software consistency. The configuration manager also provides communication of configuration status to a higher-level configuration manager as a means to provide an "all go" indication or identify specific configuration problems for correction by maintenance action.

Unit-level compatibility is performed internal to a platform unit by comparing unit specific hardware and software version numbers. The version numbers do not reflect specific part numbers but rather groups of elements that are compatible. This allows interoperability of compatible part numbers within the platform so that benign unit modifications do not force massive retrofit of platform elements.

The primary means used to confirm the platform system configuration is a platform manifest — a list of software numbers intended for a specific installation. The platform manifest is loaded into the configuration manager and broadcast to each unit after it is validated against hardwired installation program pins. Using the Loadable Software Part (LSP) numbers contained in the manifest, units confirm they are compatible with the other units that comprise the platform and are proper for their installation. This ensures interface and communication compatibility for the specific installation.

The configuration manager also provides a consistency check for hosted function software loaded on each GPM, ensuring that each of the redundant copies are consistent across the platform. This is accomplished by instrumenting GPMs to report the LSP description and associated LSP number for each hosted function application they contain. The configuration manager then compares LSP numbers for the same LSP description.

All the platform units periodically report all hardware and LSP numbers. The platform configuration manager removes redundancies and consolidates an LSP numbers report for maintenance display. This report communicates any existing configuration faults in a manner that allows easy correction of the configuration problem by maintenance personnel.

12.2 Functional System Implementation on the Platform

12.2.1 Architecting Systems on the Platform

The process for architecting a system on the Genesis Platform is as follows:

- Define the functions that comprise the system.
- Define the system elements required to implement each function.
- Define the resource usage demands for each system element.
- Define the data exchange between system elements and functions.
- Optimize the logical architecture for the platform.
- Allocate the system elements to the platform to define the physical architecture.

The first step in synthesizing an integrated system to be hosted on the platform is to define the functions that comprise the system. From this function list, system elements emerge as processing applications, I/O needs, and special equipment (nonplatform elements). Attributes are then defined for each of these system elements such as availability, integrity, and performance needs. From these attributes, a "logical" system architecture is formulated that is used to scale the platform to determine the number of platform elements required to host the integrated system. The next step is to define the data exchange between system elements

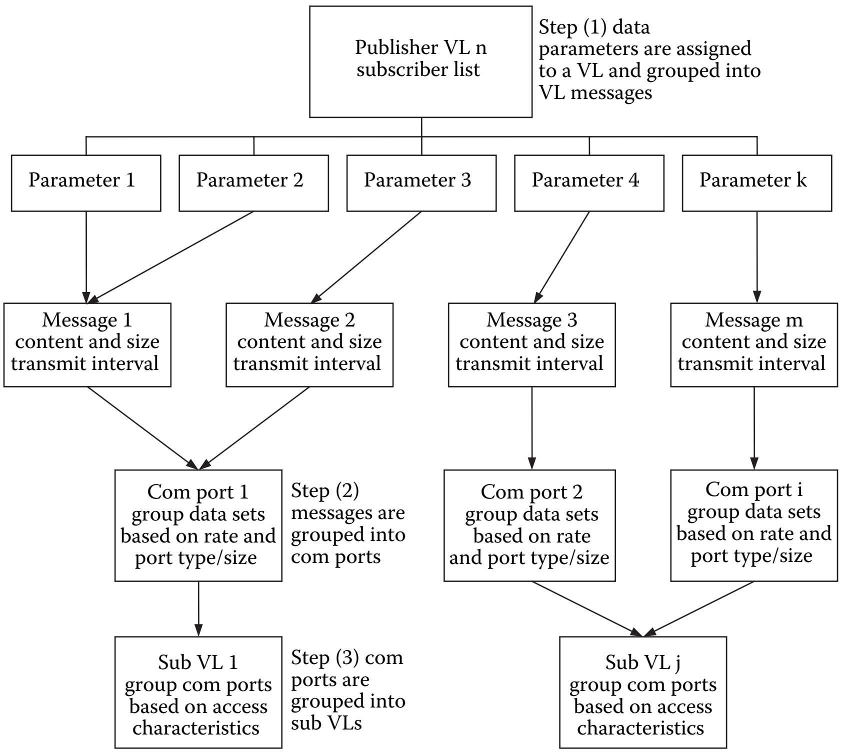

Publisher VL n subscriber list
Step (1) data parameters are assigned to a VL and grouped into VL messages
Parameter 1
Parameter 2
Parameter 3
Parameter 4
Parameter k
Message 1 content and size transmit interval
Message 2 content and size transmit interval
Message 3 content and size transmit interval
Message m content and size transmit interval
Com port 1 group data sets based on rate and port type/size
Step (2) messages are grouped into com ports
Com port 2 group data sets based on rate and port type/size
Com port i group data sets based on rate and port type/size
Sub VL 1 group com ports based on access characteristics
Step (3) com ports are grouped into sub VLs
Sub VL j group com ports based on access characteristics

FIGURE 12.17 Process for defining the communication framework.

and other systems. The definition of the functional communication forms the basic logical communication structure within which actual network messages and data sets are organized. The logical communication links define the VL framework for the platform network (Figure 12.17). This framework is associated with functional system elements, and when those elements are allocated to physical resources, the associated communication link structure follows the functional assignments to form the physical system architecture. By necessity, the communication link structure must be defined by the interfacing functions because they best know how their system exchanges data with other systems and the sensor and effector complement of the system. The optimization of the network resource is extremely dependent upon how this structure is defined, thereby keeping system-specific targeted communication isolated from more general purpose multiple-user data. The task of structuring the communication framework is a top-down process that identifies the primary data exchanges between publishers and subscribers, the data content required for the exchange and the port structure that best mechanizes the framework. (See Figure 12.18.)

The network configuration process begins with definition of the communication link structure, which is performed from a systems view, much like defining A429 connections between LRUs in a federated system approach. Each system must identify its associated publish communication link structure with a view of the subscribers and the data they need. The integrated system architect starts by identifying targeted users links, multiple user links, gateway in links, and gateway out links. Every system should by default start with these four categories, adding within a category or removing categories as the system need dictates. A maximum latency requirement is defined and associated with each link.

For each communication link defined in the communication structure, the message elements need to be identified at a high level. For example, a VL connecting an aircraft Flight Management System (FMS) to a display unit may contain some periodic parametric data, such as position, velocity, time, wind vector, and a periodic file, such as an A661 nav display file. As part of this high level definition, update rates and estimated data sizes must be defined.

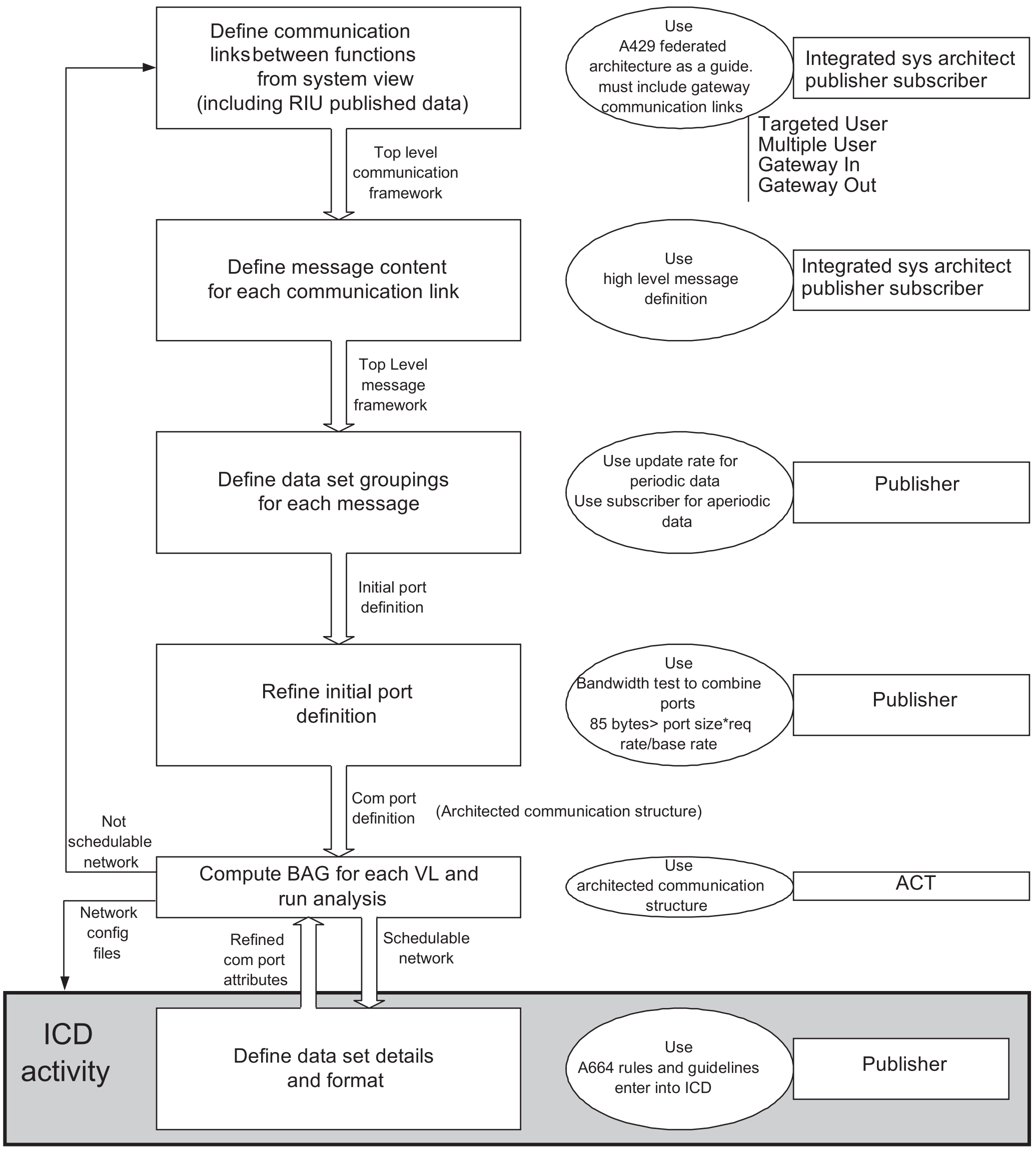

Define communication links between functions from system view (including RIU published data)
Top level communication framework
Define message content for each communication link
Top Level message framework
Define data set groupings for each message
Initial port definition
Refine initial port definition
Com port definition
(Architected communication structure)
Compute BAG for each VL and run analysis
Not schedulable network
Network config files
Refined com port attributes
Schedulable network
ICD activity
Define data set details and format
Use A429 federated architecture as a guide. must include gateway communication links
Integrated sys architect publisher subscriber
Targeted User
Multiple User
Gateway In
Gateway Out
Use high level message definition
Integrated sys architect publisher subscriber
Use update rate for periodic data Use subscriber for aperiodic data
Publisher
Use Bandwidth test to combine ports 85 bytes> port size*req rate/base rate
Publisher
Use architected communication structure
ACT
Use A664 rules and guidelines enter into ICD
Publisher

FIGURE 12.18 Network configuration process.

The actual com ports and characteristics can be defined from the message framework. In order to determine optimal bandwidth usage, the system architect separates periodic data into sampling and queuing update rate groupings, combining groups only if it is a lower bandwidth requirement to combine a group. This analysis compares the com port overhead penalty to the additional bandwidth usage that is created by adding lower rate parameters to a higher rate com port. In general, file transfers and A429 should be set up as queuing ports, with parametric data being set up as sampling ports. This process will result in a set of sampling and queuing ports defined for each VL structure.

12.2.2 Architecture Configuration Toolset

The Genesis Platform is a configurable set of resources. An unconfigured platform provides no functionality; each platform component would sit idle and no resources could be utilized. The configuration of the platform is the “glue” that binds the system together. The configuration is not only responsible for

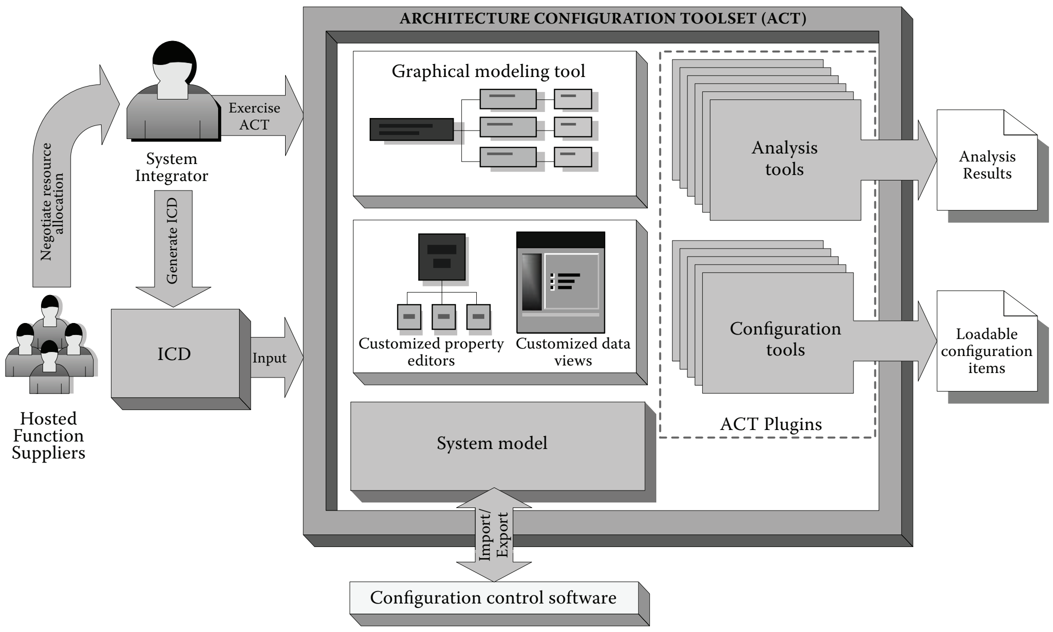

ARCHITECTURE CONFIGURATION TOOLSET (ACT)
Negotiate resource allocation
System Integrator
Exercise ACT
Generate ICD
ICD
Input
Hosted Function Suppliers
Graphical modeling tool
Customized property editors
Customized data views
System model
Analysis tools
Configuration tools
ACT Plugins
Analysis Results
Loadable configuration items
Import/ Export
Configuration control software

FIGURE 12.19 Architecture configuration toolset.

defining the system interfaces but also orchestrates the system performance. The system integrator's resource allocation guarantees are established by the platform configuration. The task of optimizing the system resources and the pledge of assurance for the validity of generated configuration files are integral responsibilities that should not be underestimated.

Once an integrated system logical architecture is defined, the logical architecture is allocated to the scaled platform architecture to form the physical architecture for the integrated system. To support this system allocation, an Architecture Configuration Toolset (ACT) is provided that allows graphical representations of the architecture resources to be created. (See Figure 12.19.) The integrator graphically structures the physical instantiation of the architecture using building block system elements and allocates these resources to specific system elements in the form of applications and their associated communication structure.

Definition of the hosted functions and allocation of the platform resources to those functions is necessary to achieve successful functional integration while ensuring efficient use of the available resources. The ACT is designed to aid in defining, configuring, modeling, and analyzing the integrated system. Hosted function resource allocation model elements are updated, through the use of the toolset, to reflect the current state and understanding of the functions, ensuring a complete and updated systems model is maintained for comprehensive allocation of platform resources. From this system model the appropriate platform element configuration files are produced to implement the integrated system.

The basic functions of the Architecture Configuration Toolset include:

- *Defining the physical architecture* – This allows the user to graphically construct the physical architecture including platform elements and system unique elements and allows overlay of the logical architecture onto specific resources.
- *Defining I/O messages* – The toolset allows the user to define and organize the I/O messages used in the communication of data between the system elements and other systems.
- *Analyzing Genesis performance* – For each hosted function, the toolset estimates the system performance relative to end-to-end response time requirements and latency requirements of the input and output signals as configured by the integrator.

- *Generating a model of integrated system* – Use analysis tools to evaluate the physical architecture definition and generate an integrated system model that satisfies the guaranteed resource definitions (resource quantity and performance) for all hosted systems.
- *Defining the Genesis configuration* – The toolset allows the user to generate configuration files for all platform components: GPMs, network switches, end systems, and RIUs. A GPM configuration file defines its software partitions and communications ports. Network configuration files define how the network data flow is controlled and bandwidth is allocated. A RIU configuration file defines the conversion operations the RIU will perform on the I/O signals and the gateway interconnect for serial digital busses (A429, CAN).
- *Managing configuration* – The toolset has a configuration management capability for storing, cataloging, and securing versions of the system configurations, I/O messages, and associated analysis and requirements data.
- *Assisting the user in making incremental Genesis changes* – Changes to the system configuration are expected not only after Entry Into Service but even during the integration and test phases of an initial program. Many changes will be limited in scope where only relatively small changes are needed in one or a few hosted functions. The toolset is designed to assist the user in making incremental Genesis changes and performs a "difference" of all the configuration files, providing a change-impact assessment to validate limited verification for the new configuration.

12.2.3 Contract-Based Approach to Integration of Modular Systems

The Genesis Platform is a modular architecture that utilizes an "open system" environment to allow independent suppliers to implement their systems on the platform. This developmental independence and partitioning of responsibilities provides the system integrator, platform provider, and hosted function supplier with an efficient project organization to separately complete their designs, integration, verification, and certification efforts. While a degree of developmental independence exists, the dependencies that remain must be properly recognized and managed accordingly.

The hosted function uses the platform architecture and, as a result, is ultimately *dependent* upon the resources provided by the platform. The benefits of developmental independence, therefore, cannot be fully realized unless the mechanisms of dependence are properly characterized and accounted for. This can be accomplished through a contract-based approach for integrating the hosted functions with the platform.

The "contract" between suppliers of the platform and the hosted functions forms a formal mode of communication that describes what resources or services can be guaranteed to the hosted functions. The contract defines and characterizes the interdependencies. The objective of the contract is two-fold. First, it communicates (and guarantees) the platform properties to the developers of the hosted functions. The contract provides a formal record of platform properties that the developers can rely upon when developing their own systems that depend upon the system or architecture. Second, the contract forms a piece of certification evidence. Since the hosted functions are allocated to the platform resources to form a "functional" architecture, each hosted function must identify their dependence upon the specific claims of guarantees in their safety and certification cases submitted to the certification authorities.

Multisystem integration methods are not a new concept, but due to the heightened degree of integration in the Genesis architecture, it is more important to formalize detailed communications between system providers than in the integration efforts for traditional federated systems.

12.2.3.1 Contract Roles and Responsibilities

The basic roles and responsibilities for the contract are split between the system integrator, the platform provider, and the hosted function supplier. The system integrator documents the platform resource allocation guarantees, including, for example, processor time allocation, memory allocation, communication bandwidth, and I/O allocation. These allocations form guarantees for the hosted function supplier. The platform provider documents the unallocated platform guarantees in their contract. The platform provider is then responsible for verifying their claims of guarantees and providing the evidence to the certification authorities (delivery to the hosted function supplier is not necessary). The hosted function supplier accepts the contracts listing the claims of guarantees and retains it as a formal record on which to base their platform assumptions and certification arguments. In lieu of providing platform-related evidence, the hosted function supplier is able to reference the platform contract in their certification and safety arguments.

12.2.3.2 Using Contractual Guarantees to Form Safety and Certification Arguments

In order to minimize the costs of change for the integrated system, it is important to form separate safety and certification arguments for each hosted function. This does not alleviate the need for a top-level argument for the integration of the individually argued systems, but the complexity of the top-level argument is reduced. If the safety and certification arguments made for the integrated system do not reason at the individual hosted function system level, then the system-wide arguments will be very complex and a change to any hosted function would require the entire argument to be reevaluated and revalidated.

An important aspect of the Genesis architecture is that the platform is provided independent of the hosted functions that will eventually utilize it. Therefore, the arguments for platform safety and certification are independent of any specific functionality that is formed when a hosted function is integrated with the platform. While the platform can be developed in isolation, the hosted function supplier must also consider its dependencies upon the platform. The system integrator is left to reason about the integration of the all the hosted functions together upon the platform.

The logic used to form these safety and certification arguments is based upon a compositional reasoning approach - referred to as "Assume-Guarantee" or "Rely-Guarantee" reasoning - that is commonly utilized for modular systems in the domain of computer science. This type of reasoning can be visualized as shown in Figure 12.20, where X represents a hosted function system that is integrated with platform Y.*

System Y (platform) must guarantee its resources and services, and system X (hosted function) must assume that the guaranteed resources and services hold true. Since system X depends on system Y, system X can only hold true when the guaranteed properties of Y hold true. However, the converse is not true; system Y does not maintain any dependence upon system X.

This reasoning is formalized in Figure 12.21, which denotes the logic as $<p>X<q>$, asserting that if X is a system where p holds true, then the system must satisfy q. This notation is applied to our example, thus stating that if we assume the properties of P_1, then Y guarantees P_2. Likewise, by assuming the properties of P_2 and the additional properties P_3, X guarantees P_4. This form of reasoning allows the system providers to reason separately about their systems Y and X, and then the system integrator is able to deduce properties about the composition of Y and X, denoted Y||X, given the properties < P_1 and P_2 and P_3 >. The properties P_1, P_2, and P_3 represent guarantees that must be made between system providers so that the arguments for safety and certification at the integrated system level can be completed.

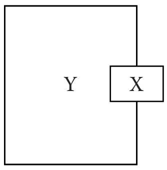


FIGURE 12.20 Visualization of the concept of compositional reasoning.

* J. Rushby. Modular Certification. NASA Contractor Report. CR-2002-212130, NASA Langley Research Center, Dec. 2002.

$$\frac{\left\langle P_1 \right\rangle Y \left\langle P_2 \right\rangle \qquad \left\langle P_2 \wedge P_3 \right\rangle X \left\langle P_4 \right\rangle}{\left\langle \text{true} \right\rangle Y \| X \left\langle P_1 \wedge P_2 \wedge P_3 \right\rangle}$$

FIGURE 12.21 A formal representation of compositional reasoning for two systems.

12.3 Certification Aspects

12.3.1 Certification Approach

The Genesis architecture presents some philosophical differences for the certification approach as compared to traditional aircraft certification approaches. The first difference is that the integrated system is organized into modules (including platform and hosted function modules) that are incrementally accepted for certification. Each module requires its own set of design assurance and certification data. The modules are submitted for acceptance by the certification authorities prior to system-level and aircraft-level submittals. Once all modules have been submitted for acceptance, the integrated system certification can be sought under a system-level certification argument that is based upon the individual arguments for module acceptance. The certification process of incremental acceptance can be characterized by six distinct phases:*

- Phase 1: Platform acceptance
- Phase 2: Hosted function acceptance
- Phase 3: Integrated system acceptance
- Phase 4: Aircraft integration and certification with Genesis system
- Phase 5: Incremental change of platform or hosted functions
- Phase 6: Reuse of platform or hosted functions

The second philosophical difference under the Genesis architecture is that hosted functions are certified according to a logical architecture rather than a physical architecture. The technical merit of this approach is justified through the contract-based integration mechanism provided by the platform guarantees. A hosted function's arguments supporting acceptance for certification are based upon a guaranteed set of resources provided by the platform elements. The guarantees describe the logical architecture provided to the hosted function. The physical architecture can be manipulated without disturbing the certification case as long as the modified physical architecture is shown to satisfy the platform guarantees made for the hosted function.

12.3.1.1 Minimal Cost of Change

Cost of change is the cost (time, effort, and money) incurred when there is a change in a hosted function or a change in a platform component. The cost of change concept is especially important in the Genesis architecture due to the high degree of integration between systems. Without the appropriate system design and certification approach, the changes in a hosted function can cause dramatic and unnecessary costs for the suppliers of hosted functions sharing the platform resources, the system integrator, end users, and even the changed hosted function. Similarly, updates to platform components (hardware or software) due to technology refresh or obsolescence could cause significant cost impact to hosted functions. The benefits realized by the Genesis architecture can be outweighed by high costs of change if change is not managed appropriately in the architecture implementation.

* Similar phases are referenced RTCA-DO-297, entitled, "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations."

The following are targets for minimum cost of change:

- Eliminating costs associated with unaffected hosted functions (HFs) when changes are made (the primary objective of reducing cost of change)
- Limiting costs related to scaling, extending, and upgrading the platform
- Limiting impacts on unaffected platform elements when technology refreshes occur
- Incrementally accepting platform or HF components for certification

In terms of design, the platform implementation must support computing robust partitioning, separation of software applications from computing hardware, network partitioning, I/O update rate segmentation, and qualified configuration mechanisms and tools.

In terms of certification, the platform certification arguments must support incremental acceptance and logical architectures. If the certification arguments are based upon a physical architecture, then any changes to the physical architecture (even those that do not disrupt the logical architecture) will require the certification arguments to be revalidated.

12.3.1.2 Regulatory Guidance

When the Genesis architecture was conceived, the common regulatory documents for the certification of complex aircraft systems did not specifically address the incremental approach to certification nor an approach to certifying systems according to a logical architecture (traditional guidance is focused on the physical architecture). Due to these deficiencies, a minimal cost of change was not fostered by these traditional sources of regulatory guidance:

- RTCA DO-178B "Software considerations in airborne systems and equipment certification"
- RTCA DO-254 "Design assurance guidance for airborne electronic hardware"
- SAE ARP-4754 "Certification considerations for highly-integrated or complex aircraft systems"

Fortunately, recent developments have generated new guidance that more appropriately addresses the unique properties of the platform. This new generation of guidance is likely to increase as architectures such as the Genesis Platform continue to grow in popularity. At the time of this publication, the following guidance was available:

- Federal Aviation Administration (FAA) TSO C-153 "Integrated Modular Avionics hardware elements"
  - Addresses reusability of certification credit for hardware elements (but not software)
  - Typical interpretations still focus on integrated system and do not allow for stand-alone architectural elements
- AC20-145 "Guidance for Integrated Modular Avionics (IMA) that implement TSO-C153 authorized hardware elements"
  - Supplement to TSO C-153 that offers no additional guidance for incremental acceptance or reuse
- AC20-148 "Reusable Software Components"
  - Supports incremental certification of software components
- RTCA-DO-297 "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations"
  - Supports most of the certification approach for the Genesis Platform
  - Currently is not officially recognized as an acceptable means of compliance but provides a valuable source of reference

12.3.2 Platform Acceptance for Certification

The platform is independently accepted for certification apart from the hosted functions. While the hosted functions rely upon the platform for their arguments for certification acceptance, the platform arguments are made in isolation. To adhere to the incremental approach to certification, the platform can be broken down into modules that can be incrementally accepted for certification. The platform developer can decide on the appropriate breakdown based upon the distributed roles and responsibilities for the program. Typical platform module breakdowns may individually seek certification acceptance for the processing module, operating system, infrastructure applications, network elements, and I/O elements.

A set of module acceptance data that establishes the module's physical characteristics, functional characteristics, performance characteristics, and interfaces (physical, electrical, and software as applicable) should be provided to the certification authorities with each module. In order to maximize reuse the usage domain can be described, including usage limitations and guidance for implementation. As in all certification approaches, the module's validation and verification evidence must be provided to the certification authorities. A portion of the platform evidence, if not the whole, will be required by the hosted function suppliers when developing their own evidence. As a platform-dependent system, the hosted function suppliers will need to reference the platform evidence when they apply for incremental acceptance.

12.3.3 Incremental Change and Module Reuse

A great benefit of a modular structure of incrementally accepted certification arguments is that this organization aids in providing a minimal cost of change during an incremental change in the platform or a hosted function. Traditional certification techniques require a single, complex argument that embraces all certification details. Under the Genesis approach, rather than revalidating the entire complex argument, only the smaller, less complex, modified argument must be revalidated. If the modular argument for certification is changed due to an incremental system change, then any arguments that rely on the modified argument would also need to be revalidated; otherwise, if the modified system continues to satisfy the certification argument such that it remains unchanged, then other arguments that depend upon it do not require revalidation.

The platform architecture, which is based upon a similar foundation as incremental change, provides incrementally accepted modules to be reused in alternate instances of the platform at minimal cost. The claim for "minimal cost" is predicated on the assumption that the original arguments for module certification acceptance can be shown to be satisfied by the new platform. The system-level arguments for the alternate platform would need to be formed, but most of the underlying logic that supported the module's acceptance in the original certification effort can likely be reused.

12.4 Conclusions and Summary

In summary, the Genesis Platform is a hardware/software platform that provides computing, communication, and I/O services for implementing real-time embedded systems. This architecture contrasts with the traditional federated architecture by utilizing shared resources to host multiple "virtual systems" that were traditionally hosted as separate "physical systems." The Genesis architecture employs a network-centric environment of computing, I/O, and communication elements and provides connections for other nonplatform systems (connected via A664-P7, A429, CAN, or digital I/O). The architecture uses robust partitioning mechanisms to establish fault containment capabilities and to ensure a high degree of "virtual system" integrity.

As compared to traditional certification strategies, the main differences for the suggested certification approach include the processes of incremental acceptance and certification by logical system (instead of by physical system). A contract-based approach to the integration of modular systems is proposed to support these certification methodologies.

Genesis is a composable and extensible architecture that is targeted for supporting highly critical applications but can be scaled to appropriately support lower levels of application integrity. As compared to the federated architectures, the Genesis architecture offers advantages of a reduced system size, weight, power, and cost. The platform provides system integrators with a foundation that allows them to quickly and efficiently build up an integrated system that takes advantage of a common development effort (open environment), common set of resources (less resource waste), and a common set of integration objectives for each system developed (increases coordination between system suppliers).

13 Advanced Distributed Architectures

Jim Moore
Smiths Aerospace LLC

In recent years a considerable momentum has built up on the assumption that future avionics will be based on centralized racks or integrated cabinets containing numbers of electronic modules of various types. Integrated Modular Avionics (IMA) in many sectors of the air transport industry has largely become the assumed way forward for the implementation of future avionics. Progress has already been demonstrated with first-generation civil IMA systems such as the Electrical Load Management System (ELMS) and Airplane Information Management System (AIMS) (see Chapter 9) on the B777. These are quite different implementations, having been optimized for their specific systems domains, and therefore appear to go only part of the way towards meeting the ultimate goals anticipated by the industry.

However, there are a number of reasons to suggest that relatively large groupings of tightly coupled and integrated computing resources may not provide the optimum architecture of future aircraft in the face of rapid technological change and ever-increasing business pressures.

This chapter will raise awareness of these trends and of alternative architectures in which the computing and interface resources are decentralized and operate autonomously within a networking communications environment.

13.1 Drivers and Trends

In addressing these issues it is important to identify and consider the changes that have taken place within the industry and the underlying trends to these changes and drivers

13.1.1 Technology Advance

Perhaps the greatest driver of change is the continuing rapid advance of electronics technology, affecting practically all areas of avionics systems — computers, sensors, displays, data buses, etc. Already there are examples of avionics computers introduced less than a few years ago that are now available at half the size, weight, and power consumption, but with considerably enhanced performance and functionality. For instance, a Flight Management Computer that 10 years ago required a size 8 MCU size box (10 in. wide) can now be implemented as a single-card module (<1 in. wide). Likewise there have been comparable advances in display technology, data buses, control panels, bulk memory media, sensor technology, etc.

These technology advances are being driven primarily by the demands of the large commercial markets, e.g., information technology, communications, PCs, consumer products, automotive, etc. The competitiveness and size of these markets are driving down costs and more and more are setting the performance standards. Aerospace components and communication standards are now being increasingly replaced by commercial ones, e.g.

- Microprocessors, microcontrollers
- Data buses (e.g., Ethernet, CANbus)
- Flat panel AMLCD displays and interfaces (e.g., OpenGL graphics language)

These trends are now firmly established in both the civil and military aviation markets.

The principal drawback is the relatively short lifetime (availability) of commercial components, which can become obsolete within just a few years, in much less time than the production cycle of the average civil aircraft programme. Therefore appropriate strategies are required to deal with part obsolescence.

The rapid and continuing advance of electronic technology is constantly driving down the cost of hardware, as the number and cost of the components used falls and much more functionality is achieved with less and less hardware. Figure 13.1 illustrates the growth in microprocessor performance over time and adherence to Moore's law and the expectation that performance will continue to double every 18 to 24 months, i.e., grow by a factor of over 1000 times in the next 20 years.

This illustrates the need for care in choosing hardware architectures with interface boundaries between the principal elements, which are the least likely to be affected by changes in the hardware technology, i.e., to prevent obsolescence/upgrade repercussions from propagating from the affected element (e.g., module) to other elements. This also applies to the interface to the functional software to protect it from the impact of changes in the processing platform on which it resides.

13.1.2 Increasing Functional Complexity

Associated with the enhanced capability afforded by the technology, and as driven by the competitive pressures of the civil transport aircraft market, the functionality of avionics systems has continued to escalate. It is evident that avionics have long since become central to the ability of a manufacturer to

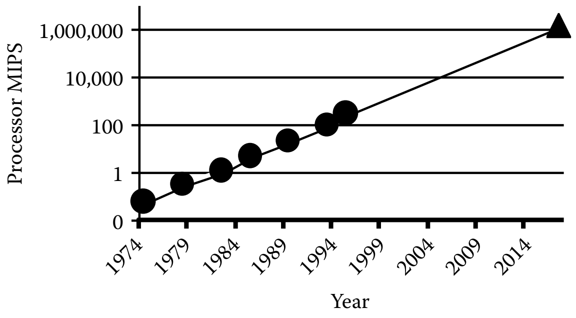


FIGURE 13.1 Increase in microprocessor performance.

manufacture and sell competitive aircraft and therefore are key to ongoing business viability. For example;

- Fly-by-wire flight controls
- FMS, full flight regime 4D flight management
- Full glass cockpits, large multifunction displays
- Future Air Navigation System (FANS) capability to operate in the new air traffic management environment
- Passenger entertainment systems and commercial/business services
- On-board central maintenance computers and electronic documentation

Most of this added functionality is, of course, effected in software, and so the quantity of code embedded in avionics continues to rise. For the suppliers, more software generally leads to increased development and certification cost. High-level languages such as Ada and C are widely used to simplify the programming effort, and where possible, automatic coding is used to further simplify the process and reduce cost.

Much effort is being applied these days to ensure that application software can be reused on different hardware, i.e., be independent of the hardware. The primary driver for this is to avoid the high cost of new software for the application being hosted on different processors, for example, in the event that during the life cycle of the product the processor becomes obsolete or has insufficient capability to support growth in required functionality.

Hardware-independent application software requires an embedded software Operating System (OS) or executive which provides a generic interface to the application code and which translates it for the particular processor and hardware architecture being used. By the use of such an executive and a standardized application interface definition it is possible to achieve compatibility of applications developed by different suppliers capable of being used on a variety of different hardware platforms. Hence the concept of APEX (Application-Executive interface) as documented ARINC 653. (See *Avionics: Elements, Software and Functions*, Chapter 14.)

13.1.3 Hardware/Software Cost Ratio Continually Falling

The combination of the above two trends suggests that, for the foreseeable future at least, the computing hardware cost is going to be less and less significant as a component of the overall cost of systems. The main cost drivers will be for software and integration. Already, today, software typically consumes 70 to 80% of the development budget and represents by far the highest schedule and cost risk to any new program.

It is also important to recognize that there are recurring elements of software cost. Historically suppliers have included amortized software development costs in the equipment recurring price. But the actual value or market worth of the embedded software has been invisible to the purchaser. However, as the material (hardware) cost falls and as suppliers use or adapt outsourced COTS software elements (e.g., embedded operating systems, data-bus protocol software, networking software) the recurring cost and value of such software will become more significant. In the limit where the product is only software, for example an FMS to be integrated into the customers host system, the price will be driven by the market value per copy of that function, assuming the development costs have already been recovered. After all, it is the software that embodies the supplier's know-how, systems expertise, and would carry copyright.

The lesson this suggests as we consider the next generation of avionics systems architectures is to beware of overcomplicating the software and level of integration just to minimize the quantity of hardware employed.

13.1.4 Integration

As technology has advanced, so there has been a continuing trend of avionics integration. In the past most integration has been concerned with processes or functions which were already interdependent, and by so doing savings could be made in interface hardware, aircraft wiring, power supplies, etc. with little risk to the certification of the system. This is usually referred to as "vertical" integration. The integration of inertial sensors with computers to produce Inertial Reference Systems (IRS) is an example of vertical integration.

In other cases separate processors have been co–located into a single box because they can share the same I/O hardware on which they both depend, thereby eliminating one set of I/O hardware and simplifying the aircraft wiring.

More latterly there is a trend to go one step further by the integration of largely unrelated avionics functions onto shared processors. This may be referred to as "horizontal" integration. There are significantly higher risks because, while additional hardware resource may be saved, there are added complexities to provide "equivalent independence" or partitioning within the processing platform. Partitioning is used to ensure that malfunctions within one application (function) cannot affect the others, or that modifications made to one may be certified without the need to revalidate the others. A further complication may be that additional levels of hardware redundancy and associated monitoring and configuration management facilities may be needed to offset the risk of failures within the shared processor, causing simultaneous loss of all the otherwise unrelated application functions.

The trade between savings of hardware (e.g., processor modules) on the one hand, and the escalation in cost to achieve acceptable levels of segregation and integrity on the other, needs to be carefully weighed. This is a trade that would appear to be more difficult to support as the hardware proportion of overall cost continues to fall with time. For example a "cluster" of separate low-cost processor chips, each dedicated to a system function and each including it's own program memory, communicating via shared data memory areas and external bus interfaces, may become a cheaper and better alternative.

13.1.5 Modularity

The concept of modular design for both hardware and software has been with us for a long time. Most avionics Line Replaceable Units (LRUs) employ a modular approach to some extent or other. The object is to build up the design by repetitively using building blocks of a minimum number of different types. A supplier can substantially reduce costs if new designs reuse existing modules. There are benefits of scale in purchased parts and manufacturing. The modules used in greater numbers more rapidly reach maturity and modifications and repairs can be more readily implemented and tested. The number and cost of spare modules needed to support maintenance shops is much reduced.

However modularity also generally adds some complexity, e.g., for connectors and interface circuits that otherwise would not be needed. Generally, there will also be elements added to allow a generic design to meet specific requirements. Thus there is an overhead in modularity which tends to reduce the benefits of integration.

A primary goal of IMA is to establish the application of a standard set of hardware modules, directly line-replaceable, encompassing as much of the total avionics suite as possible. However, the adherence to prescribed module boundaries, e.g., packaging processors separately from I/O modules and from power converter modules, etc. but all within a centralized cabinet, sets up real barriers to the benefits that can be gained from the rapid advances in electronics technology.

This concept stems from previous generations of equipment in which these distinct electronic functions each required relatively large amounts of real estate and could not be economically integrated into the same circuit board or module. Also, it was believed that the industry standards would lead to a market for competitive sources of such modules and therefore yield freedom of choice. However the reality is that this approach delivers only a "closed" architecture, no effective industry standards, and no market for choice of modules.

13.1.6 Business Pressures

The aircraft manufacturers are constantly driving to reduce manufacturing lead times and cost at the same time as the product complexity increases and technology continues to change.

Manufacturers place contracts for large packages of systems to single suppliers or consortia, i.e., the "one-stop shop." This not only reduces the overhead and diversity but also reduces the OEMs integration task and engineering burden. For the smaller manufacturers who may be unable to support large engineering resources this is perhaps the only way to cope with the problem of increasing functional complexity and technological change.

For the larger manufacturers there is the added question of the extent to which they want to also be able to manage the internal "package" integration and modification task independently of the supplier. Coupled with this may be the need to ensure long-term freedom of choice of suppliers and ability to exercise control. This is a key issue bearing upon the degree to which the package of systems or functions has open internal interfaces and is supported with reliable tools.

Because of the high value and high degree of dependence upon such suppliers the manufacturers are increasingly requiring risk-sharing partnerships. This is also evident in the trend towards partner design and manufacture of major airframe parts, or modular aircraft manufacture. Costs, build time, and risk can be reduced when the interfaces between the parts support rapid assembly and the parts are delivered fully dressed and pretested for their systems content.

From a systems viewpoint this is made considerably easier when the distributed elements located in the various major airframe parts are interfaced via serial data buses, perhaps through Remote Data Concentrators (RDCs) or local controllers.

13.2 Integrated Modular Avionics (IMA)

13.2.1 The Concept

The concept of integrated modular avionics has come about in response to the above trends and pressures.

Various implementation architectures are described in ARINC 651, the industry's overall design guidance document for integrated modular avionics. They share the common theme of a number of "cabinets" that are connected together and with other peripheral equipment by means of a number of multiple-access serial data buses (refer to Figure 13.2)

Each cabinet is seen as a high-power computing center which, in essence, replaces a number of today's application-specific avionics computers (line replaceable units, LRU). Each cabinet contains a selected mix of line replaceable modules (LRM) e.g., core processor, input-output (I/O), power supply, and other special LRMs, all interconnected via an internal backplane bus. The cabinet enclosure is designed to

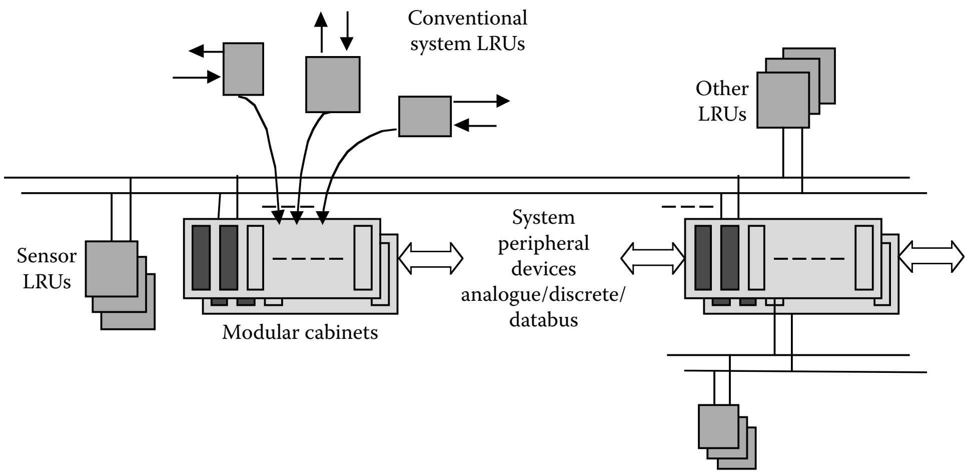


FIGURE 13.2 IMA overall architecture.

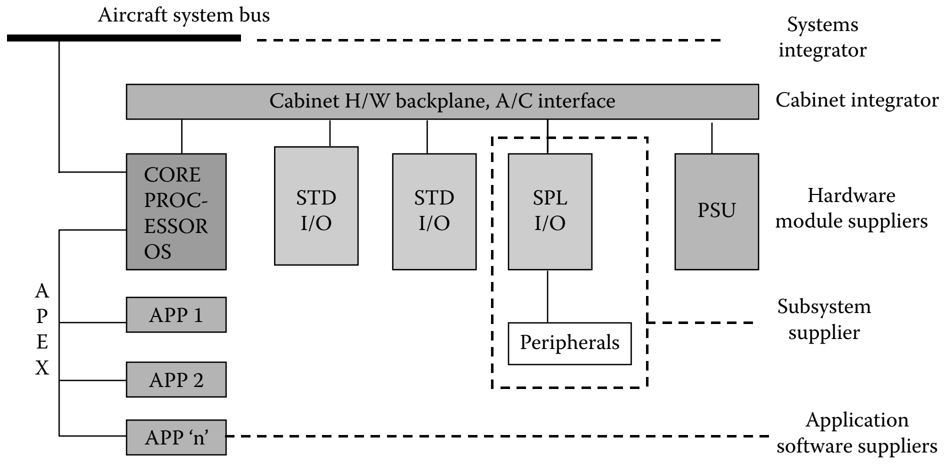


FIGURE 13.3 IMA concept.

provide a standard environment for the LRMs and effectively replaces conventional avionics equipment racks. Each cabinet is intended to have adequate processing and interface capacity to support the required integration of avionics functionality with spare capacity to meet life cycle growth requirements. The modules and backplane need to be fault tolerant to protect the group of integrated functions against single failures, or to provide high dispatch reliability, particularly for long-range or large aircraft. The system-specific application software is designed to meet the standard application-executive interface specification (APEX) to allow reusability on core processors of different hardware design.

13.2.2 Modular Architecture and Supplier Roles

The goal of standard, reusable, and interchangeable modules is central to the concept of IMA (refer to Figure 13.3). By rigorous definition and control of each module, both hardware and software, and of its interfaces, the aim is to permit a minimum number of different module types to support the broad range of current and future avionics system functions. The concept is expected to allow the systems integrator to procure each type of module from the most competitive source and to allow the user to support the aircraft with a greatly reduced number of spares. The concept, therefore, promises significant reductions in overall life-cycle cost.

It also will have a major impact on the way avionics business is conducted, since it opens up the possibility for LRMs to be supplied separately from the cabinet, software supplied separately from the hardware, and the whole cabinet to be integrated by an outside party.

To date, civil aircraft solutions like the B777 ELMS and AIMS have gone some way towards the envisaged IMA goals. Both approaches were optimized for the application domain and have brought benefits to the manufacturer and airline alike. It is important to recognize that neither AIMS nor ELMS are "open" architectures and all the parts and functions are manufactured, qualified, and integrated by the system supplier.

A key question now is whether and how a truly open common architecture and module suite can be developed and applied across the wider spectrum of avionics system domains. The following issues are considered relevant to this discussion.

13.2.3 Industry Standard Modules

Despite the original intentions, it is now fairly obvious that the industry will not be able to produce standard modules independently of the cabinet supplier/integrators. The specification process is too complex and the means are not available to independently validate and qualify modules for compatibility with the platform. The best that industry committees and agencies can do is to define form, fit, and interface requirements as a means of guidance and recommendations. It is also unlikely that the industry will be able to mandate a particular cabinet architecture. Apart from the complexities of these processes the time involved to achieve mature specifications is generally too long and will be overtaken by technology advance and real program-driven solutions.

13.2.4 Commercial Modules

The parallel is often drawn with the PC industry. The assumption being that because commercial interface standards have become widely established and the large market provides a wide choice of PC-compatible cards for the PC supplier and user, that the same “plug and play” approach will apply to the civil aircraft industry. In fact, for some avionics LRUs PC interfaces and modules are being used. However, the key difference is that the aircraft system integrator will generally have to prove and validate the use of any component used in the platform to a far higher level than used in the nonaviation world. This is likely to include a significant level of testing with the applications for a particular platform. The integrator carries the certification responsibility and will have to approve any alternative module sources.

13.2.5 Achieving the Wider Goals for IMA

The only realistic possibilities for achieving a wider application of common modules and integrated platforms appear to be the following:

- One platform supplier establishes his solution and module suite as the *de facto* standard, or
- A collaborative basis (probably aimed at a new aircraft program) in which the manufacturer and suppliers work together to define the standards and qualify a sufficient choice of interoperable parts.

Clearly the former is unlikely to generate freedom of choice and competitive component supply, and may simply establish a monopolistic market. However, some manufacturers may opt for this approach if they are not concerned about dependence on one supplier.

It is difficult to see how the larger vision of common modules across the wider systems domain along with freedom of choice and competition can be totally realized. Even the collaborative approach is likely to yield only a limited choice of supply. During the formative years the hardware and avionics functions will still be linked, if only to establish the certification principles. Then the technology will progress and the modules are likely to be obsolete for the next aircraft generation, requiring the cycle to be repeated. However, it is possible to see for a new aircraft programme that some module commonality could be achieved across platforms of different application domain, by collaboration between the platform suppliers.

13.2.6 Control of the Interfaces—Open Systems

A related issue is the question of the degree to which the manufacturer intends to be able to exercise control over the interfaces within the integrated system, e.g., within the platform between the hardware modules and between the application software and the operating system. For example he may wish to be in control of the integration of third-party software and variants arising from customer changes during the aircraft life cycle.

This is understood to be a clear objective for at least the manufacturers of large civil transport aircraft. In which case the same principles and features are needed as for integration management at the intersystems/aircraft level. It is this requirement, probably above all others, that will determine whether open architectures really do become established, and also the extent to which the platform internal architecture integration is maximized.

The key interfaces requiring control are the intermodule databuses (backplane) and the application software interface (APEX). It is vital for the aircraft buses to be truly open the communication over these interfaces (at the interoperability level) must be independent of the technology used in the system. Also, the interfaces must place minimum possible constraints on the functioning of the modules and equipment operation.

13.2.6.1 Software Interface

For the application software interface the manufacturer will require the same programming and integration tools as used by the platform supplier integrator. These tools would have been qualified by the platform supplier and would need to be maintained through the life cycle of the aircraft. Appropriate contractual arrangements will be necessary to define responsibility and liability associated with modifications or added functionality implemented by the manufacturer independently of the platform supplier.

13.2.6.2 Hardware Interface

For the module interfaces, it will be much more difficult for the manufacturer to add modules or introduce an upgraded performance module (for example) and then revalidate the platform if the platform architecture is tightly coupled, i.e., one in which the modules are controlled in lock-step fashion by the applications as an extension of the processor internal bus. In effect this is a wholly integrated platform, with the modules having only physical independence from each other. It may be open in the sense that the backplane bus meets a standard such as ARINC 659 or VME, but it does not readily permit the user to change or add hardware functionality independently of the integrator.

13.3 Aircraft and Systems Architecture Issues

13.3.1 "Smart" Peripherals

The technology advances that have enabled rack-mounted LRUs to be reduced in size or be integrated together over the years have also, in combination with serial data bus interfaces and other advances, enabled electronics to be used increasingly in various locations around the aircraft. Some examples of "smart peripherals" or "distributed systems" are

Full Authority Engine Control (FADEC) — Engine mounted
Air Data Modules (ADM) — Fuselage skin
Cabin entertainment systems — Cabin distributed
Smart autothrottle actuators — Under floor
Smart display heads — Flight deck
Smart systems panels — Flight deck/cabin

With today's advanced microcircuits it is already possible to provide considerable functionality within many system peripheral devices without significantly affecting the space they occupy or significantly reducing their inherent reliability. As costs fall we are likely to see data-bus-interfaced smart peripherals being used increasingly, e.g., primary and secondary displays, flight control actuators, landing gear sensors/actuators, tank-mounted fuel quantity processing, remote data concentrators, radio sensors/transceivers, etc.

The main drivers are

- Reduction in installation complexity — wires, connectors.
- Robustness of peripheral interfaces to noise, improved HIRF/EME immunity.
- Reduced uncertainties in identifying fault location, i.e., whether fault is in the peripheral, the computer, or the wiring.
- Simplification in data interfaces, by incorporating special peripheral processing at the "point of action" and reducing communication data flow to higher-order parameters.

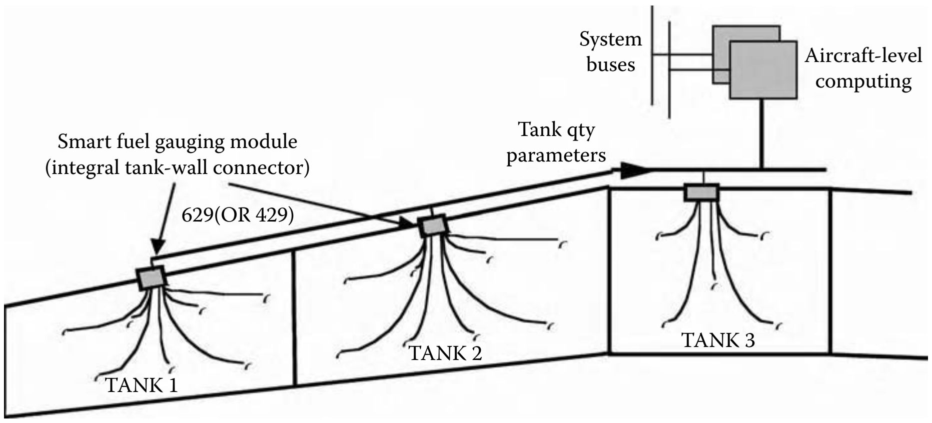


FIGURE 13.4 Distributed fuel gauging system.

- Architectural flexibility; the I/O interface is available on aircraft-level data buses for direct use elsewhere.
- Enabling the interface to be independent of the technology of the peripheral.
- Simplified procurement boundary promoting freedom of choice and control by the integrator.

Electronics is used at the peripherals and serial data buses provide the interface, so the systems become digital from end to end.

Figure 13.4 illustrates an example of a distributed fuel gauging system using the principal of smart peripherals. In a conventional high accuracy system the individual tank probes would be wired back to a gauging computer in the avionics bay. In a large aircraft there could be over a 100 probes and therefore at least 200 wires running over a considerable distance through the airframe, carrying sensitive analogue signals of very low strength. By locating highly reliable electronics close to the tanks this wiring is greatly simplified. All the probe signals for each tank are now combined onto a serial data bus. The signals are preprocessed into a form which is now independent of the probe technology and, being digital, are much less sensitive to wiring abnormalities and electrical interference. The processing task for the central computer is also simplified so that the gauging computation can now more easily be accomplished on a standard computing resource such as an Avionics Computer Resource (ACR).

The greater use of smart peripherals and remote data concentrators will also significantly diminish the need for analogue and discrete I/O modules within IMA cabinets and computer LRUs (see Figure 13.5).

13.3.2 High Speed Serial Data Buses

Serial digital data buses have been used on both civil and military aircraft of all types for many years. ARINC 429, which is a single-transmitter multiple-receiver topology bus operating at either 12 Kbps or 100 Kbps, is used on all modern civil transport aircraft. The disadvantage of this topology is that a single bus can carry data only from one source, and so each item of equipment needs as many inputs as the number of buses (sources) that it expects to receive data from. Accordingly, there are numerous bus spur connections and wires in the aircraft associated with most of the avionics boxes.

The Boeing 777 was the first civil transport aircraft to use a multiple access data bus in which all connected units can both transmit and receive via a single port connection (there are, of course, multiple redundant buses). This bus operates at 2 Mbps and uses active couplers to connect the unit port via a stub cable to the linear bi-directional bus itself. This is ARINC 629. This topology simplifies equipment and the aircraft wiring installation because one port and one bus stub connection only are needed for each unit to communicate to all the other units on the same bus. The bus is very deterministic in its operation and each terminal has high integrity control over the associated unit's access to the bus, the

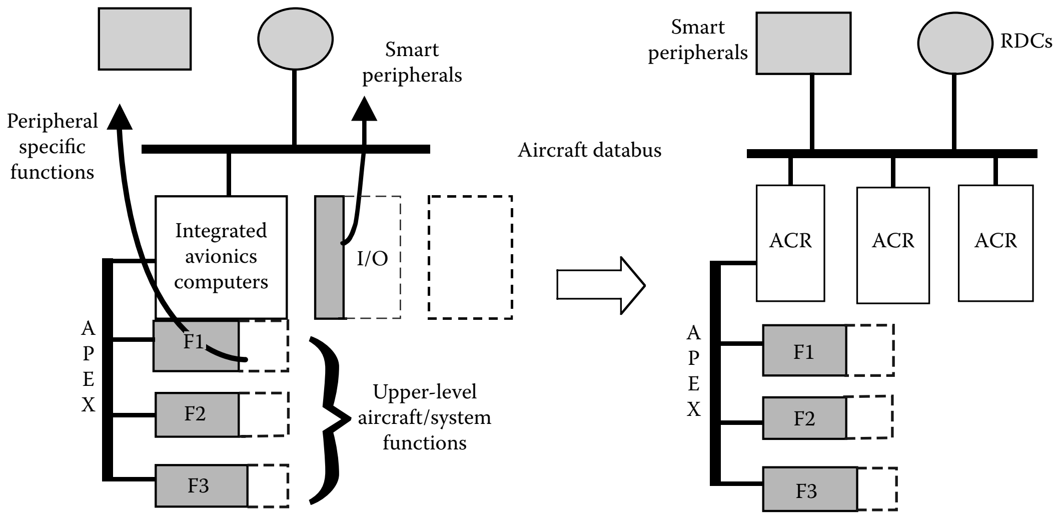

Peripheral specific functions
Smart peripherals
Integrated avionics computers
I/O
A P E X
F1
F2
F3
Upper-level aircraft/system functions
Aircraft databus
Smart peripherals
RDCs
ACR
ACR
ACR
A P E X
F1
F2
F3

FIGURE 13.5 RDCs reduce need for analogue and discrete I/O modules.

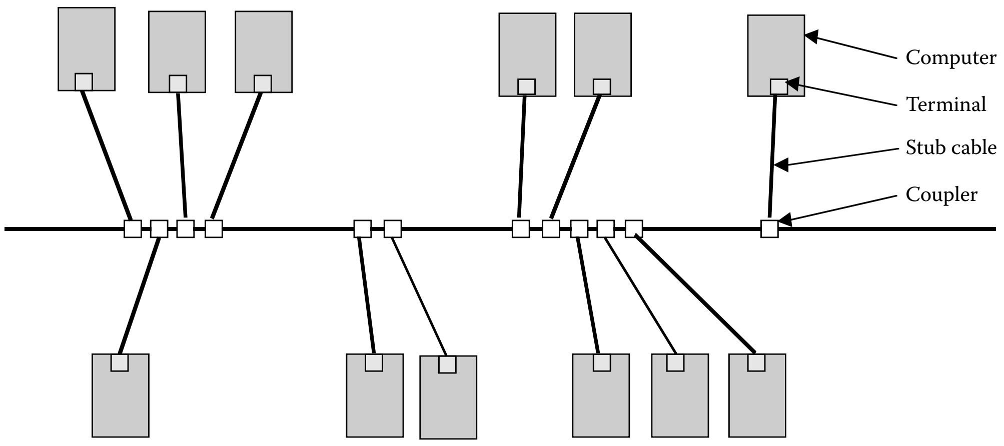


FIGURE 13.6 ARINC 629 topology.

correct use of allocated bandwidth, and the selection of transmitted and received data. Its principal limitation, however, is the 2 Mbps rate. Also, the components are relatively expensive and there can be electrical performance factors which limit the physical spacing of coupler connections along the bus. Figure 13.6 illustrates the ARINC 629 topology.

More recently, fast networking data buses based on commercial standards are starting to appear in civil avionics systems. The industry has started standardization work on Ethernet and the first example of FDX (full duplex Ethernet) applied on a civil transport aircraft is for the interfaces between the various display units and computers in the cockpit primary display system on the B767-400 aircraft.

Airbus is using a variant of FDX (A664-P7) as the main avionics and systems buses on the A380. These buses typically have a star topology in which the equipment ports are connected to each other via stub cables and the star point (switch unit). Again each item of equipment can communicate with all the others on the same bus through one port. In this case the data rate may be typically 100 Mbps, much

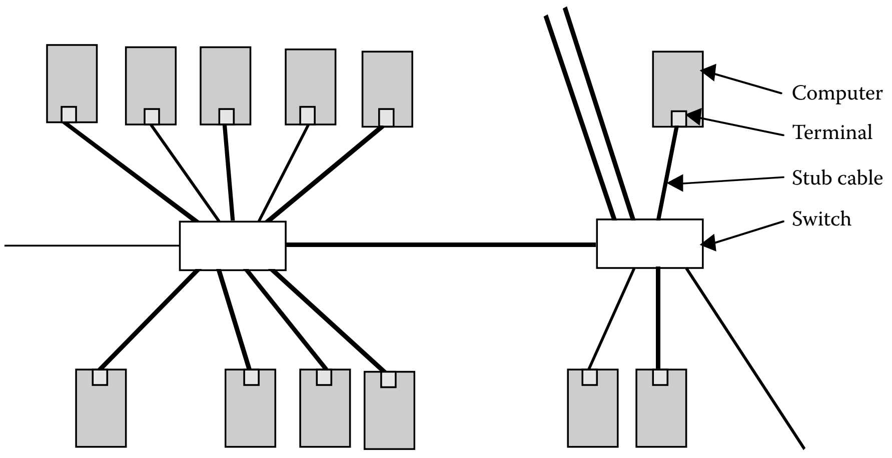


FIGURE 13.7 FDX (A664-P7) topology.

faster than ARINC 629, although some of this bandwidth is used by higher formatting and protocol overheads. The concept is potentially much cheaper, since it can use the commercial standard directly (at least for the physical layer components), and it is likely to become the next civil aircraft industry bus standard. For most aircraft applications, however, modifications to the services and functions of standard Ethernet are needed to ensure a statically configured network, additional integrity, and controls on message routing and channel bandwidth usage. These changes impact upon the design of both the switch and the user terminal. The topology is illustrated in Figure 13.7.

An important advantage of buses like A664-P7 is that they can form a continuous network covering both the aircraft level (e.g., communication between LRUs, IMA centers, and other smart peripheral equipment) and the systems domain level (e.g., communication between modules associated with a particular group of systems). In an IMA context the switch provides the backplane connectivity for a domain and provides a direct interface to the aircraft-level part of the network, without the need for conversion gateways. In addition, only one set of bus management tools is needed to integrate the communication at all levels and for all domains across the aircraft.

13.3.3 Procurement Boundaries

An important architectural driver for the aircraft manufacturer is the procurement boundary, i.e., the interfaces through which a system or equipment to be procured operates in relation to its neighbours. Simplifying these interfaces is one of the goals inherent in determining where the boundaries should be. This is a necessary process to minimze the specification and integration task, reduce aircraft wiring and data flow, and achieve correct interoperability with minimum risk. Figure 13.8 illustrates these principles with a hypothetical example.

System functions A, B, and C (FA, FB, and FC) are highly related to each other and have complex interfaces between them, but have relatively simple or standard interfaces with sensors S1, S2, and S3. In addition FB and FC communicate with FD via a small number of readily specified signals. It is assumed that suppliers exist who are competent in all these system functions. The grouping together of functions A, B, and C makes sense because it significantly simplifies the manufacturer's engineering and procurement workload compared, say, to specifying and procuring FA combined with its associated sensors S1 and S2, separately from FB and FC. The (procurement) boundaries around the FA, FB, and FC group are all relatively simple, easily controlled, and not dependent on the technology of the peripheral sensors. Therefore these functions are also likely to be good candidates for integration within a common computer, with least risk of requirements capture errors or overall integration problems within the aircraft.

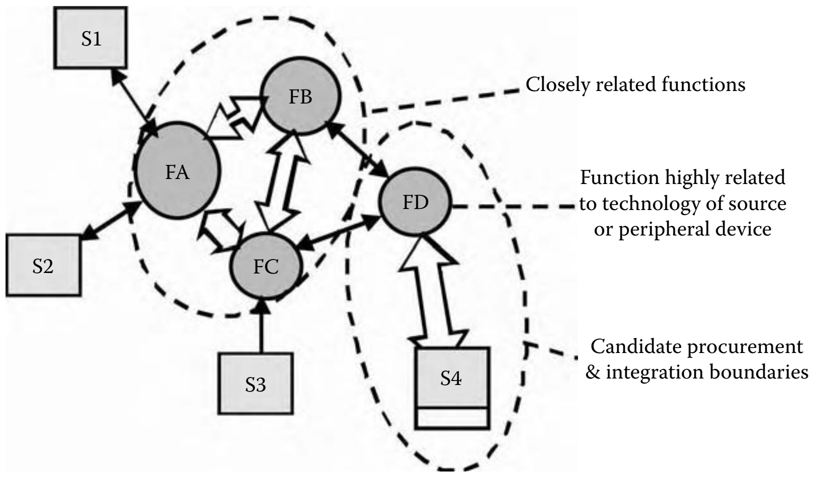

S1
S2
S3
S4
FA
FB
FC
FD
Closely related functions
Function highly related to technology of source or peripheral device
Candidate procurement & integration boundaries

FIGURE 13.8 Integration and procurement boundaries.

On the other hand, FD is very closely related to sensor 4. The signal interfaces are complex or the function is highly dependent on the technology or specific design of the sensor. The simplest procurement boundary, therefore, encloses, both, and once again there is the potential for further simplification through integration of the function with the sensor.

Analyses of functional relationships and interdependencies across all the aircraft systems are required to determine the best options for functional grouping and integration. A top-down hierarchical approach is needed to establish the optimum groupings at the global systems level and the systems domains level. This approach assists the manufacturer to ensure that the functional groupings simultaneously meet technical and commercial requirements.

13.4 Conclusions

The foregoing acknowledges that benefits can be achieved by the careful application of cabinet-based integrated modular avionics for any given aircraft programme. The degree of integration, as distinct from modularity within an IMA platform or cabinet, and the degree of openness of the internal platform interfaces for any particular aircraft programme, will depend largely upon the purchaser's (e.g., aircraft manufacturer's) desire for control over these interfaces and the integration process.

If the aircraft manufacturer is to readily fulfil the role of platform integrator, a less tightly integrated backplane and module communication concept is needed. Here the modules would have autonomy of operation and communicate with identified data sets or messages. Each would be specified for its functionality and communication requirements. Each could be validated to a greater extent independently of the application environment. This concept is not dissimilar to the way LRUs interoperate. Indeed, it is to some considerable advantage to the manufacturer and the end user if the same data bus standard can be used for the backplane as is used for aircraft-level communications. Then exactly the same integration tools and maintenance tools can be used at both the aircraft and system domain level.

In the longer term, as the aircraft systems become more and more digital end-to-end through the use of smart peripherals and remote data concentrators, the need for analogue and discrete IO interfaces within centralized computing platforms will diminish. In addition, the processing within these platforms need not be burdened with functions associated with peripheral devices and can be simplified to that needed to perform upper-level system functions only.

It is also evident that the conventional segregation of processors, IO, and power converter hardware into distinct and separate modules is too restrictive and uneconomic compared with single modules containing all of this capability using the latest advances in technology. Hence, low-cost compact LRMs or LRUs providing a computing platform capability with substantial capacity and without internal architectural constraints are now achievable and are likely to become increasingly more competitive as time passes. Though these may differ internally from different suppliers, they can be designed to meet performance and interface standards (e.g., as for the ACR) including the APEX interface for abstracted application software and hardware interfaces for standard and special IO, both digital and analogue.

It is possible, therefore, to envisage the advantages hoped for from IMA to be obtained by the use of multiple ACRs, each hosting a number of system applications, RDCs, and smart peripherals, networked together using high-speed data buses such as A664-P7.

Index

Electrical Engineering

AVIONICS
DEVELOPMENT AND IMPLEMENTATION

In the short time since Cary Spitzer's *The Avionics Handbook* was published, new technologies and standards have fueled advances in digital avionics technologies. Reflecting the increasingly digital nature of modern avionics, the second edition of this bestselling handbook features a new title: the *Digital Avionics Handbook*. But the title is not the only change to this edition. In addition to updated material and several completely new chapters, this essential reference is now presented as a set of two books focused on a specific area of avionics.

The second installment in this set, **Avionics: Development and Implementation** explores the practical side of avionics. It demonstrates several platforms deployed in a variety of civil and military aircraft and considers the process from system development through certification and operation. Each of the book's carefully updated chapters reflects the knowledge and experience of accomplished experts to supply the most accurate and reliable information. The contributors examine such topics as modeling and simulation, electronic hardware reliability, certification, fault tolerance, and several examples of real-world applications. New chapters discuss RTCA DO-297/EUROCAE ED-124 integrated modular avionics development and the Genesis platform.

Features

- Provides sharply focused coverage for anyone involved in building, integrating, and approving avionics systems
- Works systematically through the development and implementation process
- Includes the most up-to-date information on the latest technologies and certification standards
- Offers real examples of deployments in modern aircraft such as the Boeing B-777 and the Airbus A-330/340

Written by practitioners for practitioners, **Avionics: Development and Implementation** offers useful guidance and real-world insight for anyone developing, deploying, and maintaining modern avionics systems.

CRC Press
Taylor & Francis Group
an informa business
www.taylorandfrancisgroup.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
270 Madison Avenue
New York, NY 10016
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

This title is part of a set.