AN EMBEDDED MAINTENANCE SUBSYSTEM

Eugene A. Esker
William R. Simpson
John W. Sheppard
ARINC Research Corporation
2551 Riva Road
Annapolis, Maryland 21401

ABSTRACT

The analysis of complex electronic systems in terms of maintainability and testability has evolved to where information models of the system can be used dynamically to provide maintenance personnel with powerful field maintenance tools. At the same time, the evolution of the hardware for these systems reflects on-board processing, bulk memory, and data buses for information movement. The parallel evolution of anlaysis tools and hardware now makes it possible to extend dynamic models and provide maintenance and testability information directly to the system itself. The result is an embedded system that examines itself from a maintainability and testability perspective and diagnoses and reconfigures itself. This paper describes the architecture leading to the implementation of an embedded maintenance subsystem and the steps necessary to integrate information models with system architecture.

INTRODUCTION

The art of system design is moving toward ever larger and more complex systems. The ability to maintain such systems no longer relies on individual expertise, but on tools and standards such as modular automatic test equipment (MATE) and integrated diagnostic support system (IDSS). We are now reaching a new level of complexity in system design as a result of the need for dynamic system reconfiguration. Advanced systems are employing architectures that permit the addition or modification of functional capability with minimum impact on the host platform. Many supporting activities included in the architecture are developed to increase flexibility in functional configuration and increase fault tolerance. Two areas of complex systems are evolving in parallel:

- Tools and techniques, including standards and modeling methods, are becoming more sophisticated.
- Hardware architectures of complex systems now include communication buses for passing information, bulk memory, and embedded processing capability.

This parallel evolution provides the elements necessary for what we call an embedded maintenance subsystem—a facility that is a supporting subsystem of the system into which it is embedded.

An embedded maintenance subsystem is a self-contained maintenance processing facility that will:

- Perform frequent and periodic system "health" monitoring
- Order specialized built-in test (BIT) execution to improve fault isolation on initial detection of an anomaly (If desired, an operator may actively participate in this process.)
- Report all failures detected and localized to a data recorder and store information in mass memory for later retrieval
- Perform in situ diagnosis and report that diagnosis to a resource management system, the operator, and elsewhere as required
- Dynamically reconfigure the system to minimize the impact of detected failures
- Dynamically reconfigure its own model by incorporating system reconfiguration and failure data

Many of the elements required for embedded maintenance subsystems are currently required for other purposes, such as providing fault-tolerant systems.

BACKGROUND TO EMBEDDED MAINTENANCE CONCEPT

TOWARD FULLY INTEGRATED SYSTEMS

Because system complexity has increased over the years, formerly independent subsystems now communicate and share data. In some cases, they act together to achieve objectives and cross-correlate information obtained during task prosecution. Such systems often contain some basic elements that allow increased flexibility and on-board intelligence (e.g., communications buses for sharing information, mass memory for storing information, and on-board processing for combining the former with the latter). These capabilities have, however, made

CH 2793-8/90/0000-0331 $1.00

maintenance extremely difficult under current architectures. This is true not only for military applications where multimission aircraft are evolving, but also for the commercial market where competitive pressures make maintenance turnaround times an important factor.

Systems are now being developed with interrelated goals that include fault tolerance, shared assets, and reconfigurable equipment suites. These goals bring with them a new set of problems for on-board failure diagnostics, which in turn relate to maintenance. Accomplishment of each goal requires precise on-board diagnostic capability. Fault tolerance requires redundant or shared equipment, but in order to make full, efficient use of this redundant or shared equipment, failures must be identified. Reconfigurable equipment requires fault isolation that is flexible and adaptable to each new or potential configuration of the equipment.

MAINTENANCE AND DIAGNOSIS— A COLLECTION OF PROBLEMS

The maintenance and diagnosis of complex systems have been treated as separate problems. For simple systems, manual troubleshooting techniques resolved any maintenance problems. As systems became more complex, greater levels of technician expertise were required until the collection of test equipment and test sophistication exceeded abilities associated with well-trained technicians. With complex equipment development, a number of techniques to assist the technician were developed, including BIT, automatic test equipment (ATE), electronic manuals, and other diagnostic and maintenance aids.

BIT, historically, was designed to detect faults. If BIT provided any fault localization at all, it was to the subsystem or replaceable unit level only; BIT was not intended for fault isolation.[1] In the mid-1980s, however, an approach to using BIT for actual fault isolation was developed by Grumman for Rome Air Development Center (RADC).[2] The process was termed "smart BIT," a name applied to any BIT that provides more than simply fault detection. (More recently, the term "intelligent BIT" has been applied in the same context.[3])

The basic approach to the smart BIT is to use the diagnostic fault tree, an approach which can become extremely complex for even a stationary system (i.e., a system that does not reconfigure itself). An example of diagnostic fault tree complexity for a simple system may be found in Tong.[4] To assume that a complex fault-tolerant or reconfigurable system can be diagnosed by embedded fault trees is unrealistic.

Alternative approaches to fault isolation that have been evolving throughout the industry are the information model-based approaches; typically, the System Testability and Maintenance Program (STAMP®),[5] IDSS,[6] and a number of other approaches[7,8] Not all have reached the full maturity levels required. During the evolution of information model-based systems, hardware architecture has evolved to incorporate greater levels of cross-communication and inherent complexity. Combining information modeling with improved hardware architectures now assures the integrity of the entire maintenance process.The integrity is accomplished through a disciplined approach to the process, including fault detection and isolation, repair through changes in system configurations, and logging the maintenance actions taken for logistics purposes.

COMMON AVIONICS BASELINE FOR ADVANCED ARCHITECTURES

Advanced military architectures include requirements for in-flight reconfigurability, the use of common resources, and a high degree of fault tolerance. In order to achieve these objectives, a number of factors come into play. Of the many hardware system architectures currently evolving, the Joint Integrated Avionics Working Group (JIAWG) advanced avionics architecture provides a common avionics baseline. Figure 1 shows the architecture of a complex electronic system under the JIAWG common avionics baseline (CAB),[9] including many of the elements common to such systems:

- Embedded microprocessors for providing information processing and display services as well as data logging and interpretation
- A number of buses to move information around the system (Some buses are tailored, such as video buses or bulk transfer buses, to load information and instructions directly to processors; other buses serve as general purpose, high-speed information transfer devices [e.g., MIL-STD-1773].)
- Mass memory to store integral knowledge of missions and systems for use by on-board processors
- A reconfigurable network of some type to tailor performance to the mission and to keep mission-critical elements "healthy" (In the CAB, common elements may exist in many systems, and reconfiguration might include routing around failed elements of one system to a similar element in another.)

The evolution of the hardware system and maintenance strategy is shown in Figure 2. Previously (1970s), systems were designed and performance analyzed without an analysis structure. When a system was ready for fielding, an "expert's" recommendations were used to develop a maintenance strategy, which was then put into a field program or maintenance architecture that already existed (such as the Air Force three-level maintenance [AFTO-649] system), and little planning was required. The strategy primarily consisted of static fault trees and a repair hierarchy from field organizational-level maintenance, to intermediate-level maintenance, to depot-level maintenance. The current process (mid-1980s) is shown by the solid lines of Figure 2.

Testability analysis is used to establish the domains of isolatable and nonisolatable faults, and the results of the analysis of these domains govern an intelligent, economic decision on how much ambiguity is allowable. Ideally, testability and performance analyses are performed in parallel, and the

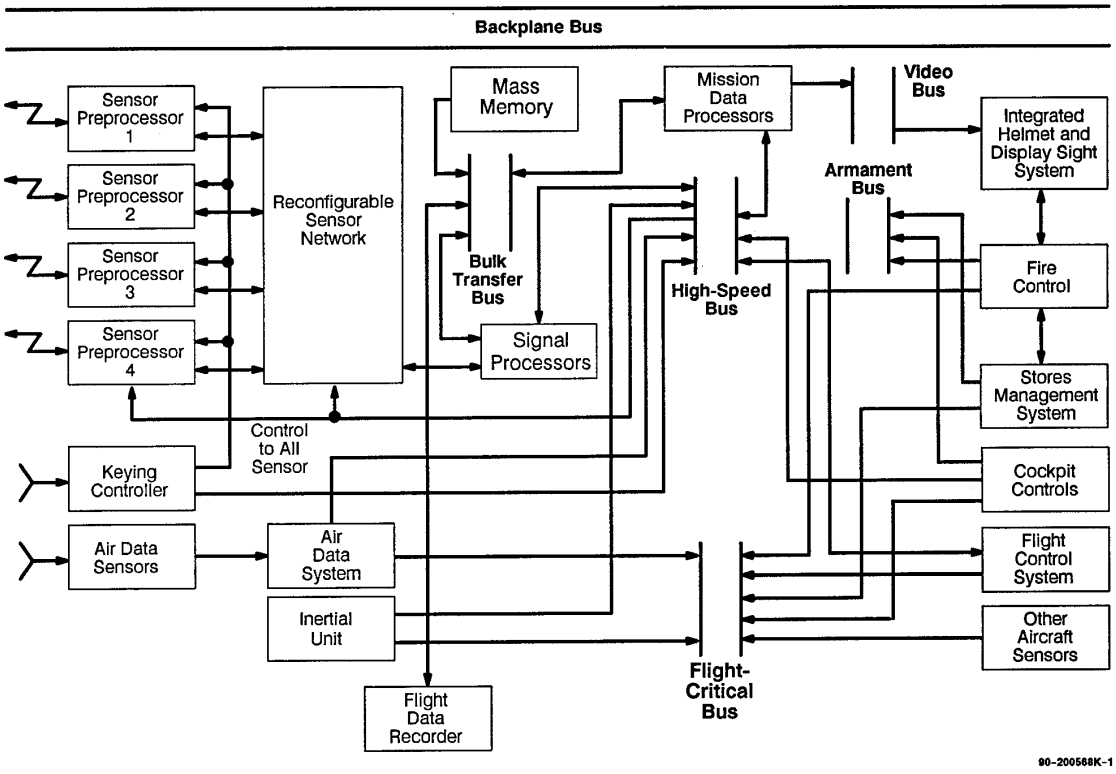

FIGURE 1. COMMON AVIONICS BASELINE

results then affect the system design. Maintenance architectures are usually determined during the system concept phase. In some cases (for example, the Combined Altitude Radar Altimeter), the customary three-level maintenance concept has been abandoned or modified. In other cases, attempts have been made to use smart BIT in maintenance architecture (such as for the F–16 shop replaceable unit on-board isolation).

In the past, the major approach to fault isolation, the static fault tree, either was embedded in BIT or was in maintenance manuals (or sometimes electronic manuals). Attempts at reconfigurable systems often used multivolume static trees (one for each configuration). By the late 1980s, we were capable of designing "dynamic" fault-isolation techniques using a model of the system to be maintained.

The process of developing an embedded maintenance model is shown in Figure 3. The first steps are similar to those currently used for system testability analysis (Figure 2), except that after each step of system maintenance model development, the information is embedded in the final product and is included in the design allocations. When the testability evaluation is complete, the model is then transferred to the embedded system and combined with an inferencing capability (inference engine) so that the model can be used for on-board diagnostics and system reconfiguration.

Figure 4 shows a simplified extension of the CAB architecture shown in Figure 1, including the reconfigurable maintenance model with accesses to buses for receiving BIT information, triggering extended BIT and BIT equipment, and providing information to reconfiguration systems. The maintenance model consists of six major elements:

- An internal representation of the system (including the maintenance subsystem)
- An optimization subsystem for choosing the next test
- An inference engine for reasoning about fault data
- A reconfiguration subsystem (and an optional learning subsystem for adjusting the model) for triggering fault repairs or workarounds
- A logging system for later retrieval and application of data to logistics data bases
- A user interface subsystem for communication with flight personnel and ground personnel

The reconfigurable maintenance model, as shown in Figure 4, has access to the backplane bus, where most health BIT indications can be accessed. This provides for fault detection, which is necessary as the first step in the maintenance process.

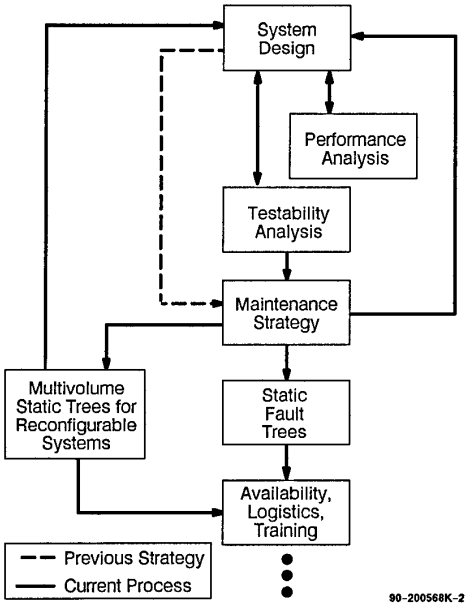

FIGURE 2. EVALUATION OF MAINTENANCE STRATEGIES

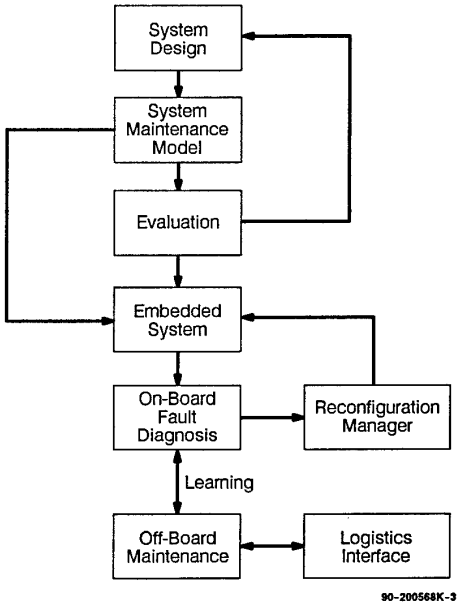

FIGURE 3. THE PROCESS OF EMBEDDED MAINTENANCE MODEL DEVELOPMENT

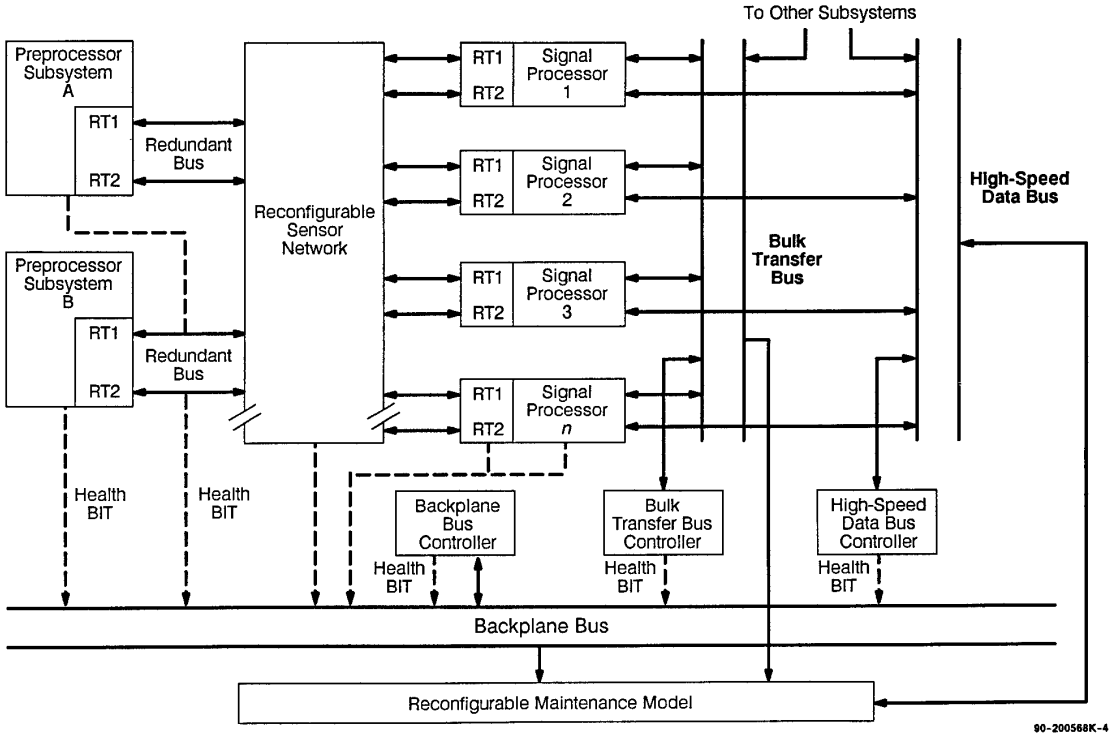

FIGURE 4. SIMPLIFIED EXTENSION OF EMBEDDED ARCHITECTURE

When the maintenance model detects an anomaly (i.e., background BIT), it can trigger specialized BIT (a fault-isolation test within the maintenance model) by sending instructions via the high-speed data bus. Responses to the specialized BIT can be received via this bus also. The maintenance model will dynamically select a set of tests to conduct using its current knowledge of the system and thus localize the fault. This localization can then be reported to the configuration manager for appropriate system reconfiguration. The resulting changes in configuration are reported to the maintenance model via the high-speed data bus, which the maintenance model loads from mass memory via the bulk transfer bus. The maintenance model can also send diagnostic information to bulk memory for later retrieval and to the flight data recorder if desired.

Figure 5 shows a fully integrated embedded system with detailed specific hardware requirements:

- Maintenance Data Processor (MDP)—The type of processor and amount of memory depend on the specific application.
- Mass Memory—This memory stores the modeling information to be used by the maintenance processor and the results of BIT.
- Bus for Access—Buses provide data access and some loading allocation.
- Maintenance Feedback Interface—This capability is not needed at the proof-of-concept stage, but with such a robust internal information source, provisions should be made to incorporate it at the next level. At a minimum, it could be provided in the flight data recorder.

THE NEXT STEP

Technology has evolved both in terms of analysis techniques for systems and in terms of hardware architectures. This evolution provides an opportunity to integrate the two and provide a more versatile maintenance approach in the 1990s. The requisite modeling techniques and the appropriate hardware architectures have been developed; now they must be integrated and tuned to work in the combined environment.

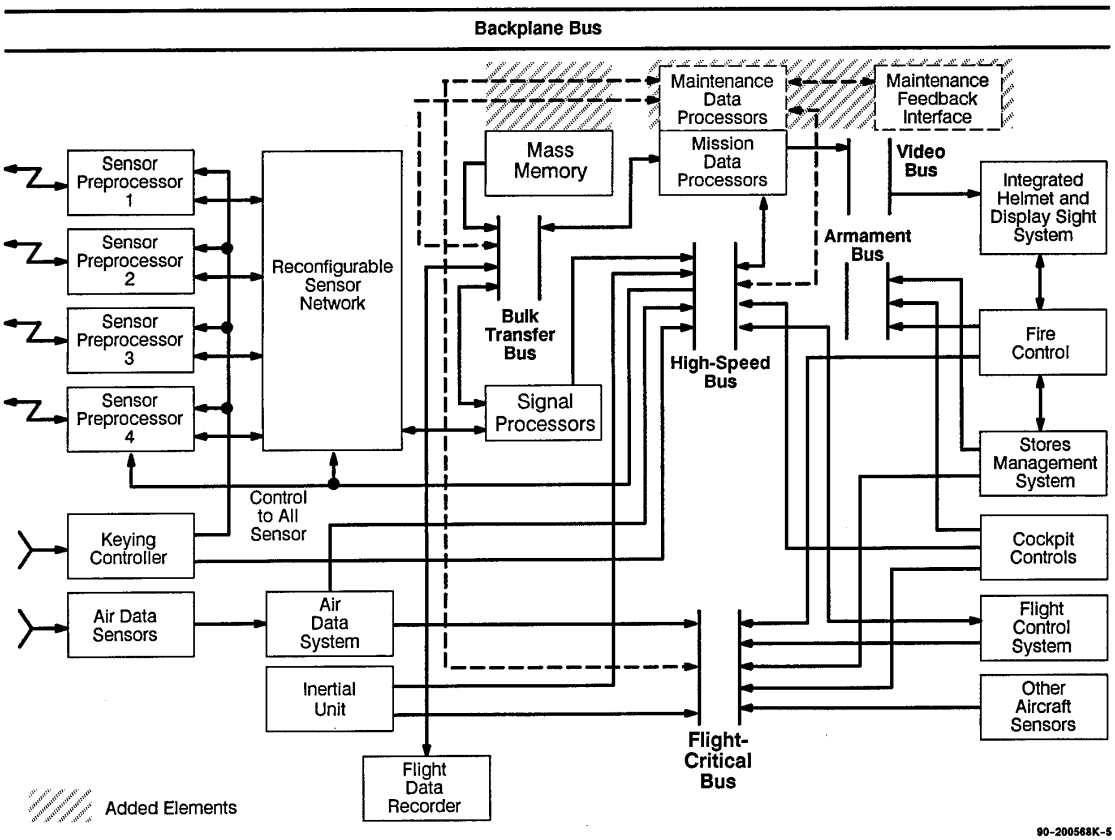

FIGURE 5. FULLY INTEGRATED EMBEDDED SYSTEM

One set of software tools has evolved to the point that this integration may begin. The ARINC Research-developed STAMP[5] modeling software has been applied to a number of hardware systems over the last nine years, often giving spectacular improvements over existing field maintainability procedures.[10] More recently, ARINC Research has developed a companion piece of software called the Portable Interactive Troubleshooter (POINTER™).[11] This software tool enables interactive use of the maintenance model and has been applied as an ATE driver program[12] and as a portable maintenance aid driver.[13] The STAMP and POINTER software systems are well suited for an overall integrated approach to maintainability. [14] The STAMP model, together with the POINTER inference engine, can be embedded directly into a complex electronic system to provide the necessary elements of an embedded maintenance subsystem. This subsystem will then allow the objectives of fault tolerance, shared assets, and reconfiguration.

REFERENCES

1. Hughes Aircraft Company, *BIT/External Test Figures of Merit and Demonstration Techniques*, RADC-TR-79-309, Contract Report for Rome Air Development Center, December 1979.

2. K. A. Haller, et al., *SMART BIT*, RADC-TR-85-148, Rome Air Development Center, March 1985.

3. D. W. Richards and J. A. Collins, "Intelligent Built-in Test and Stress Measurement," *AUTOTESTCON '89 Conference Record*, Philadelphia, Pennsylvania, September 1989.

4. D. W. Tong, et al., "Diagnostic Tree Design with Model-Based Reasoning," *AUTOTESTCON '89 Conference Record*, Philadelphia, Pennsylvania, September 1989.

5. F. Johnson and R. Unkle, "The System Testability and Maintenance Program (STAMP): A Testability Tool for Aerospace Systems," Proceedings of the AIAA/NASA Symposium on Maintainability of Aerospace Systems, Anaheim, California, July 1989.

6. J. R. Franco, "Experiences Gained Using the Navy's IDSS Weapon System Testability Analyzer," *AUTOTESTCON '88 Conference Record*, Minneapolis, Minnesota, September 1988.

7. R. A. DePaul, Jr., "Logic Modeling as a Tool for Testability," *AUTOTESTCON '85 Conference Record*, Uniondale, Long Island, New York, October 1985.

8. Naval Electronic Systems Command (ELEX-8111), *Testability Program for Electronic Systems and Equipment*, MIL-STD-2165, Washington, D.C.: Naval Electronic Systems Command, January 26, 1985.

9. Joint Integrated Avionics Working Group, *JIAWG Advanced Avionics Architecture Standard J87-01 (CAB II)*, Dayton, Ohio, January 1989.

10. W. R. Simpson, "STAMP Testability and Fault Isolation Applications, 1981-1984," *AUTOTESTCON '85 Conference Record*, Uniondale, Long Island, New York, October 1985.

11. W. R. Simpson, J. W. Sheppard, and C. R. Unkle, "POINTER—An Intelligent Portable Maintenance Aid," *AUTOTESTCON '89 Conference Record*, Philadelphia, Pennsylvania, September 1989.

12. H. H. Dill, "Test Program Sets—A New Approach," *AUTOTESTCON '90 Conference Record*, San Antonio, Texas, September 1990.

13. C. R. Unkle and M. D. Donovan, "A Smart Portable Troubleshooting Aid for Power Plant Controls," 1989 Conference on Power Plant Controls and Automation, Miami, Florida, February 1989.

14. J. W. Sheppard and W. R. Simpson, "Integrated Diagnosis—A Hierarchical Approach," *AUTOTESTCON '90 Conference Record*, San Antonio, Texas, September 1990.