

Integrated Diagnostics: Operational Missions, Diagnostic Types, Characteristics, and Capability Gaps

Theodore F. Marz

September 2005

Acquisition Support Program

Unlimited distribution subject to the copyright.

Technical Note
CMU/SEI-2005-TN-035

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2005 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

Contents

Abstract	vii
Author's Note	ix
1 Introduction	1
1.1 Background.....	1
1.2 Examples and Observations.....	1
1.3 Document Scope and Audience	2
1.4 Document Overview	2
2 Types of Diagnostic Missions	3
2.1 Verification of Operational Readiness.....	3
2.2 Fault Detection and Characterization	3
2.3 Fault Isolation	4
2.4 Diagnosis and Repair of Repairables	4
2.5 Other Maintenance Actions.....	5
2.5.1 Installation and Configuration	5
2.5.2 Calibration and Alignment.....	5
2.6 The Diagnostic Mission Life Cycle.....	5
3 Types of Diagnostics	9
3.1 Closed and Open Loop Tests	9
3.1.1 Closed Loop Tests	9
3.1.2 Open Loop Tests	9
3.1.3 Why Aren't All Tests Closed Loop?	10
3.2 Test Environments	10
3.2.1 Testing in In Situ Environments	10
3.2.2 Testing in Depot Environments.....	12
4 Characteristics of Diagnostic Environments	13
4.1 Initiating Tests	13
4.2 Grouping Tests.....	13
4.3 Iterating Tests	14
4.4 Terminating Tests.....	14

4.5	Reporting Results	14
4.6	Degree of Platform Integration	14
4.7	Impact of a System Operational Profile on Diagnostics	15
5	Capability Gaps in Integrated Diagnostics.....	17
5.1	Test Coverage	17
5.2	Correlating Faults to Failures	17
5.2.1	Ability to Isolate Failures.....	17
5.2.2	Determining the Consequence of a Failure	18
5.3	Fraction of False Alarms – False Positives.....	18
5.4	Software Health Management.....	18
6	Advanced Testing Environments	21
	Bibliography	23

List of Figures

Figure 1: Diagnostic Mission Life Cycle.....	6
--	---

List of Tables

Table 1: Test Environments.....	10
---------------------------------	----

Abstract

The Acquisition Support Program at the Carnegie Mellon Software Engineering Institute assists organizations that develop Department of Defense hardware/software hybrid systems by participating in technical reviews and other validation activities. During recent customer engagements, it has been noted that some development teams do not demonstrate a consistent understanding of integrated diagnostic system needs. In particular, software engineers seem to lack the experience needed in this area. Since software engineers frequently derive the low-level requirements for developing diagnostic systems, a lack of knowledge about deployed testing environments can have significant impact. Failure to adequately address the integrated testing needs of a system profoundly impacts its supportability and, consequently, the cost of that system throughout its life cycle.

This report attempts to fill in these gaps in knowledge and experience by presenting an overview of the operational diagnostic life cycle of a system. In addition, it outlines how a system's operational profile impacts diagnostic tradeoffs.

Author's Note

Coincidentally, while I was developing this report, the following news story was posted on the U.S. Air Force news Web site:

Report: Flight control system problem caused F/A-22 crash

LANGLEY AIR FORCE BASE, Va. (AFPN) -- A flight control system problem caused an F/A-22 Raptor to crash on the runway at Nellis Air Force Base, Nev., on Dec. 20 [2004], according to an Air Force report released June 8.

The pilot ejected and sustained minor injuries. The \$133.3-million aircraft, assigned to the 422nd Test and Evaluation Squadron at Nellis, was destroyed when it crashed. Additional damage was limited to an arresting cable, runway guide sign, runway light and the runway itself.

The flight control system malfunction was caused by a brief power interruption to the aircraft's three rate sensor assemblies, which caused them to fail. The assemblies measure angular acceleration in all three axes: pitch, roll and yaw. With three failed assemblies, the F/A-22 is not able to fly, investigators said.

When the pilot shut down engines for maintenance servicing, he left the auxiliary power unit running. Based on technical order guidance, he believed the power unit would supply continuous power to the flight control system. However, there was a less-than-one second power interruption to the assemblies during engine shutdown.

There is no automatic warning of this condition. To discover it, the pilot would have had performed a diagnostic test. The pilot accomplished a successful test before engine shutdown, and because the power unit was on, he believed a second test was unnecessary. (Courtesy of Air Combat Command News Service)

1 Introduction

1.1 Background

During several technical reviews of software-intensive Department of Defense (DoD) systems, an insufficient focus on logistical support software has been noted. In this context, the phrase *logistical support software* denotes those items in hardware/software hybrid systems that provide system support, rather than satisfy “functional” requirements. One of the primary logistical support software environments in these software-intensive systems is an integrated diagnostics and health management (DHM) environment.

1.2 Examples and Observations

During an Independent Technical Assessment (ITA) of a large system, it was determined that a large and contentious problem existed in the area of operational supportability. The acquisition activity failed to properly interpret and implement the fault detection/fault isolation (FD/FI) requirements. This resulted in the need for highly skilled technicians to support the system. However, technicians with appropriate skills and detailed system knowledge were not available to support the system within the command (organically) and the force structure and policy demanded that this support be organic. This left the developers and users of the system at an impasse, which was only resolved with great difficulty.

During a program office engineering augmentation engagement, it was noted that the distributed, multi-organizational development team did not have a detailed understanding of the diagnostic missions that the system would need to support. This meant that it was not possible for them to consistently derive appropriate requirements for the design and implementation of their DHM environment.

Programs consistently focus efforts on the satisfaction of “functional” requirements at the expense of the “non-functional” logistical support requirements of reliability, availability, supportability, and modifiability. These “non-functional” requirements are strong system and software architecture drivers. In particular, the successful definition and implementation of a DHM environment requires close interaction between hardware, software, systems, and logistics engineering teams.

Too often, DHM support activities are delegated to a single, isolated team for implementation. This leads to the implementation of environments that are underspecified and insufficiently usable or capable. This drives life-cycle costs upwards significantly and can yield unsatisfactory evaluations of operational suitability during Operational Test & Evaluation (OT&E) events.

1.3 Document Scope and Audience

This report defines the types of diagnostic missions that any operational system needs to support. It discusses the types of diagnostics required to support these missions and some of the characteristics of these diagnostics. It provides a context for developers to determine the proper level of support and a rationale for their decisions. This report also addresses some of the areas in which capability gaps currently exist in the state of practice.

The intended audiences of this report are the project software engineers and systems engineers responsible for refining and implementing requirements for integrated diagnostic capabilities in DoD hardware / software hybrid systems. Program Office personnel who define, capture, and document diagnostics-related requirements, or who monitor the development of systems with diagnostics requirements may also find this paper of interest.

1.4 Document Overview

This report is organized as follows

- Section 1 presents background information: the motivation for the paper and the intended scope and audience.
- Section 2 defines types of operational diagnostic missions and presents a diagnostic mission life-cycle model.
- Section 3 defines testing terms and environments.
- Section 4 defines characteristics that integrated diagnostic environments should have to support diagnostic missions. It also discusses the effect that the system operational profile has on the diagnostic environment characteristics.
- Section 5 outlines capability gaps that exist in current diagnostic environments.
- Section 6 addresses advanced topics in integrated diagnostics, such as the embedment of Interactive Electronic Technical Manuals and the concept of Prognostic Health Management.

2 Types of Diagnostic Missions

To operate and support a system, different diagnostics activities must be supported. In this report, we refer to these activities as *diagnostic missions*, because we are relating them to aspects of the system's operational mission. The diagnostic missions we will discuss include

- Verification of Operational Readiness
- Fault Detection and Characterization
- Fault Isolation
- Diagnosis and Repair of Repairables
- Other Maintenance Actions

Each of these diagnostic missions is described in more detail in the following sections.

2.1 Verification of Operational Readiness

The Verification of Operational Readiness diagnostic mission is the process that verifies to the operators of a system that the system is capable of meeting the requirements of their operational mission. This process typically begins with system power-on and ends with operator assurance that the system is operable. The intermediate steps consist of a number of offline diagnostics that run as expeditiously as possible with minimal operator involvement. These diagnostics may not be comprehensive in that they may not isolate failures, but they are complete in that all operational data and functional paths are verified.

For example, a flight simulator training suite performs a collection of power-on self tests, followed by a collection of open and closed loop tests to verify the proper operation of the simulator hardware prior to the first of the day's training missions. (See Section 3.1 for definitions of open and closed loop testing.)

2.2 Fault Detection and Characterization

After operational readiness has been achieved, the system commences operation. Fault Detection and Characterization is an ongoing diagnostic mission to detect possible faults in the functioning system. The goal is to detect the presence of faults and to characterize their effects so the operator can determine if the mission is affected. If the mission is affected the system may continue to operate in a degraded mode or it may be declared inoperable. If the system has failed, then the operator executes a fault isolation and repair process to restore the system to operation.

For example, an aircraft flight control computer performs continuous, periodic self-evaluations of its capability and performance. In the event of out-of-tolerance operation, the aircrew is alerted to the fault and a backup resource is granted control of the system.

2.3 Fault Isolation

Fault isolation is a refinement of the fault detection and characterization process. The goal of fault isolation is to locate the failure within a single (or small group of) replaceable items which if replaced, restore the system to operation. Diagnostic hardware and software may be aided by historical data and procedural activities to narrow the field of possible failure sources and focus the diagnosis and repair activities. The correlation of faults to the item or items that have failed can be very complex.

Example: A data cable fails in a satellite communications system and irregular faults are reported on a single network link from the modem unit. Technical manuals note the fault condition and attribute it to either the satellite modem, the boundary router of the network link, or the connecting cable. Replacement of the modem from spares fails to correct the fault. Since diagnostics from the router indicate that only this connection is having problems, the technicians replace the cable and the intermittent fault does not recur. Inspection of the cable reveals a crimp where the cable was pinched by a floor tile, causing crosstalk on the cable and an increase in the bit-error-rate of the resulting connection.

2.4 Diagnosis and Repair of Repairables

After a failed item has been isolated, removed from the system, and replaced, it must be evaluated for repairability, repaired or discarded if it is not repairable, and the repair verified for correct operation. The first and last of these steps are performed by the comprehensive tests available in the integrated diagnostics environment. In the event that these integrated diagnostics are insufficient to isolate the failure to a repairable unit (as opposed to a replaceable unit), then additional manual or automated test equipment (ATE) and custom diagnostics are employed. Manual, ATE, or custom diagnostic activities typically take place in either intermediate or depot maintenance support facilities, as depicted in Figure 1. In many cases, the depot support environment is the equipment vendor, rather than a military service depot.

Example: An aircraft RADAR data processor (RDP) failed built-in testing (BIT). The processor was removed from the aircraft and sent to the base avionics maintenance shop, which isolated the failure to a single card. This card was replaced with one from spares and the RDP was returned to inventory. The failed card was sent to the central avionics maintenance depot for further diagnosis and repair. The technicians discovered that tin whiskers were growing from an integrated circuit (IC) package down to the circuit itself and causing it to short out. *Tin whiskers* are electrically conductive, crystalline structures of tin that sometimes grow from surfaces where tin (especially electroplated tin) is used as a final

finish [NASA 05]. Since this failure mode was not previously known, the IC package was replaced to repair the board and a more permanent replacement solution was sought to increase overall reliability [Nordwall 86].

2.5 Other Maintenance Actions

Systems will typically need other maintenance actions that must take place either periodically or on an event-driven basis. These activities are most likely associated with installation, configuration, or re-configuration tasks, or are associated with on-going calibration and alignment activities. Another class of “other maintenance actions” is pre-programmed preventive maintenance actions, which are not specifically addressed in this report but can be logically grouped as calibration and alignment activities or as Verification of Operational Readiness diagnostic actions. Ideally, preventive maintenance actions evolve into Prognostic Health Management, an advanced topic covered briefly in Section 6.

2.5.1 Installation and Configuration

Installation and configuration maintenance activities are special activities that transition a system from its stored state into a ready-for-operation state. The extent of this activity varies depending on the type of system and installation. However, it is expected that an initial calibration and alignment activity must be performed as part of installation and configuration.

Example: An aircraft-mounted directional antenna is being installed on an aircraft body. Accurate antenna pointing is required, so the antenna mounting ring must be correctly aligned on the aircraft body and the antenna mounted on the ring. Following installation, local body effects on the transmitting and receiving antenna elements are tuned to the correct configuration.

2.5.2 Calibration and Alignment

Calibration and alignment actions are activities that are required to align system components to their proper orientation and to calibrate sensor components to their correct operational limits.

Example: A flight simulator visual system contains a steerable target projector. This projector is driven by synchro servos and position feedback is provided by a set of resolvers. In addition, each projector has a “home” position determined by a set of micro switches. The synchros and resolvers drift over time, both relative to each other and relative to absolute home, requiring periodic recalibration for correct operation.

2.6 The Diagnostic Mission Life Cycle

The five diagnostic missions described in this section exist within a diagnostic mission life cycle, which is depicted in Figure 1.

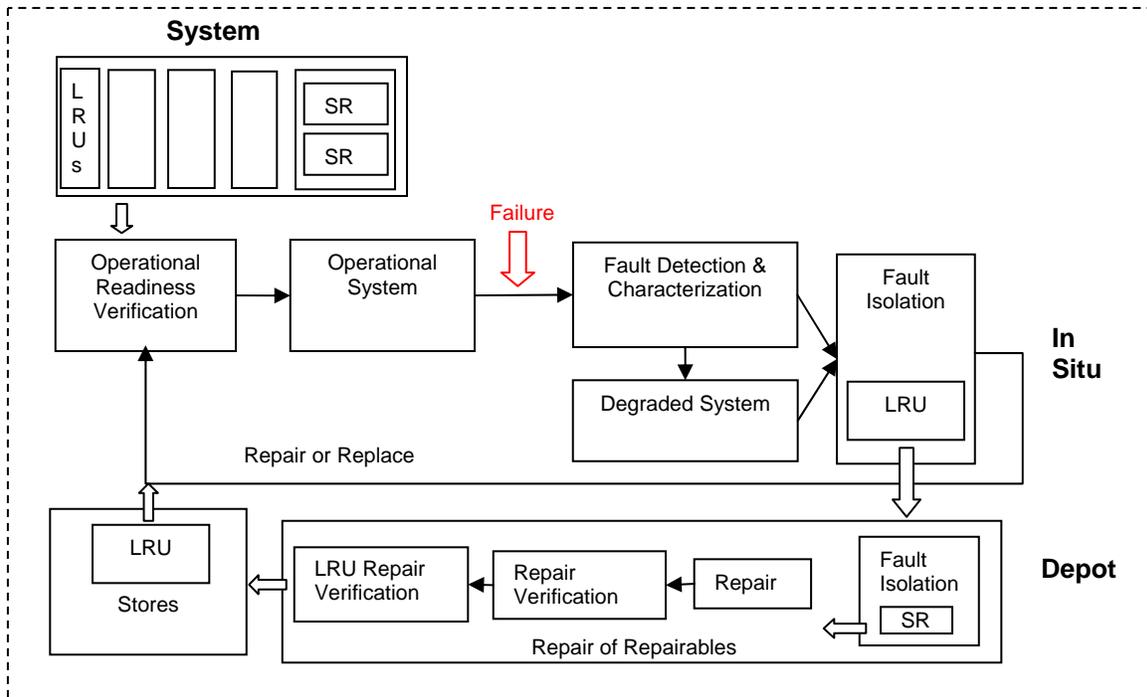


Figure 1: Diagnostic Mission Life Cycle

Systems consist of one or more configuration items, packaged as line-replaceable units (LRUs). LRUs are intended to function as the replacement items for organizational-level (O-level) maintenance activities. The LRUs themselves consist of one or more Shop Replaceable Units (SRUs). SRUs are intended to function as the replacement items for intermediate-level (I-level), and depot-level (D-level) or vendor repair. SRUs can themselves consist of repairable (or replaceable) item components for which repairs or replacements are typically performed at the vendor or supplier level.

System operators perform regular operational readiness verification tests, usually at the start of a mission, to ensure that they have a capable system. If they do not have a capable system, or if the operating system fails, they (or their maintenance technicians) perform detailed fault detection and characterization activities. These activities are designed to determine if there is an actual system failure (false alarm rejection) and the operational impact of the detected failure. If the failure is not central to the mission, or if workarounds exist, they may choose to operate the system in a degraded mode for a period of time.

Eventually, however, the failure must be corrected. This task begins with fault isolation activities, which are designed to isolate the failure to a single or small group of LRUs. The failed unit is then replaced from stores and begins the repair of repairables process. The "repaired" system is verified by using the operational readiness evaluation and is then returned to normal operations.

Meanwhile, the failed LRU is routed into the logistics system for repair or replacement. A system of one or more depots exists to support enhanced diagnosis and repair of the failure. Depots can consist of localized I-level facilities, as well as facilities at a centralized facility

(D-level), or the system developer. Technicians at a depot attempt to isolate the failure progressively to the next smaller repairable or replaceable item (SRU or other component) that is compatible with the capabilities of the depot. After the failure is isolated, the component is repaired or replaced and the correction is verified up through the sub-assemblies until the completed item is returned to stores for reissue and reuse.

3 Types of Diagnostics

There are several types of diagnostics (tests or procedures) that have been developed to address the needs of the diagnostic mission areas. Diagnostics can be characterized by a few attributes

- interactive nature of the diagnostic (closed or open loop)
- location of the diagnostic (in situ or in depot)
- degree to which the diagnostic can be performed during normal system operation (online or offline)

3.1 Closed and Open Loop Tests

Diagnostic software can be categorized by the need for operator interaction to complete the test. These categories are generally known as *closed loop* and *open loop* tests.

3.1.1 Closed Loop Tests

Closed loop tests are tests that do not require operator interaction to complete successfully. A loopback path from the actuator (or data source) to a sensor (or data sink) which is used to verify correct operation. Because no human interaction is required in the test loop, the loop is said to be “closed.” Examples of a closed loop test include communications interface loopback tests, discrete and analog output loopback tests (where hardware support is available), and other special purpose electronic interfaces where diagnostic loopback paths have been designed into the hardware.

Closed loop tests can affect the operation of the system during their execution and may not be comprehensive. For example, a closed loop test may not verify end-stage hardware such as line drivers/receivers, lamps, and attached instrumentation.

3.1.2 Open Loop Tests

Open loop tests are tests that require some kind of operator interaction to complete and determine whether the test succeeded or failed. With an open loop test, there is no loopback path from the source to the sink that does not include an operator. A human is part of the test loop, so the loop is said to be “open.” Examples of open loop tests include lamp and gauge tests, switch tests, and other tests that usually involve operator controls and displays.

3.1.3 Why Aren't All Tests Closed Loop?

It is difficult and expensive to design special purpose hardware to “close the loop” on open loop tests that require a human to monitor operator controls and displays. For example, performing closed-loop testing of all keys on a keyboard requires additional hardware to actuate the keyboard switches. Building in this specialized hardware just for diagnostic purposes may not be cost-effective, and may actually reduce the overall reliability of the system.

3.2 Test Environments

Tests can also be categorized by the environment in which they exist or are exercised and the time or system state in which they are executed. Two testing environments are in situ and depot. In situ tests are characterized by tests that are executed offline (with respect to the basic function of the system) and those that are executed online. Depot tests are characterized by tests that are run where the equipment (or unit under test [UUT]) is installed outside of the normal operating environment.

Table 1: Test Environments

In Situ	Offline	Power-On Test	Basic
			Extended
		Initiated Test	Closed Loop
		Open Loop	
		Alignment & Calibration	
	Online	Continuous Test	
Depot	BIT Environment	As for In Situ testing above	
	ATE Environment		

3.2.1 Testing in In Situ Environments

In situ tests are executed at the O-level, with the UUT installed in its normal operating environment. In situ test are often referred to as BITs and can also be categorized by their execution time relative to system state. In situ test and BITs can be conducted offline or online. Offline tests are executed outside of normal system operation, often in a specialized test environment. Online tests are a collection of background diagnostics that can run in parallel with normal system operation without affecting it. Offline and online in situ test and BITs are described in more detail in the following sections.

3.2.1.1 Offline Tests

In situ offline tests are characterized by when and how they are executed. Typically, a collection of tests is run when the system is powered-on and initiated, with additional tests

being run on request. These tests are executed when the prime function of the system is non-operational, or “offline.” The power-on and initiation tests directly support the Verification of Operational Readiness diagnostic mission, while operator-initiated tests more directly support the Fault Isolation, Repair of Repairables, and Other Maintenance Action diagnostic missions.

Offline in situ tests can include power-on BITs, power-on self-tests, and initiated BITs, which are described in more detail in the next section.

3.2.1.1.1 Power-On BITs, Power-On Self-Tests, and Initiated BITs

Power-on BITs, often referred to as PBIT (power-on BIT) or IBIT (initial BIT) tests, are tests that run automatically when a system is powered-up and initialized. They are typically closed loop and support the Verification of Operational Readiness mission. PBIT tests can have an additional sub-category of tests called power-on self tests (POSTs). POSTs are a subset of PBITs. POSTs are a set of low-level tests, usually developed by a commercial off-the-shelf (COTS) hardware vendor or specialized hardware developer. They are hosted in non-volatile memory and execute prior to system software boot. One of the jobs of PBIT is to collect the diagnostic information from the various POST tests.

Initiated BITs, (IBITs) also referred to as CBITs (commanded BITs) are tests that run when an operator initiates them. IBITs support the Fault Isolation and Diagnosis and Repair of Repairables missions. Initiated BITs usually consist of a subset of PBIT tests augmented with additional diagnostics. The additional diagnostics can include open loop tests to verify controls and displays and tests to verify proper communication and interaction with other connected systems. Initiated BIT tests can run as a single iteration or as a repeating set of one or more tests. Repetitive execution of the tests is necessary to help detect and isolate intermittent failures.

3.2.1.1.2 Other Maintenance Actions

Other maintenance actions are diagnostic actions that may not be tests per-se, but are necessary for proper system operation. Other maintenance actions can include running software to support initial system set-up and verification and performing periodic tasks to verify system alignment and calibration.

3.2.1.2 Online Tests

In situ online tests are run periodically or continuously in the background during normal system operation. *Continuous BITs* or *periodic BITs* are online tests that run in the background to support the Fault Detection and Characterization diagnostic mission. They are typically either closed loop tests or statistics collection activities that attempt to verify that data is flowing properly across the entire system. Examples of continuous BIT tests include the collection of network interface error statistics (such as checksum failures or parity errors) and other closed loop tests that can be run without affecting normal system operation.

The purpose of these tests is to support the Fault Detection and Characterization diagnostic mission and to provide sufficient Fault Isolation to support line maintenance operations. *Line maintenance operations* consist of component replacement and other adjustments made at the LRU level.

3.2.2 Testing in Depot Environments

Depot or “vendor” tests are run when the UUT is installed outside of the normal operating environment. The goals of testing in a depot environment are to verify that new components function properly and to support the Repair of Repairables diagnostic mission.

Testing in a depot environment typically includes, but may not be limited to, all of the in situ tests run for an LRU. These tests are run to verify that an LRU has failed or has been repaired and to focus additional diagnostic and repair activities that are available only to the depot, vendor, or hardware developer.

Depot tests that are executed below the LRU level typically require the support of additional ATE. The development, standardization, configuration, and use of ATE is its own complex domain and is beyond the scope of this report.

4 Characteristics of Diagnostic Environments

The diagnostic missions defined in Section 2 determine some overall requirements for testing environments. This section discusses the derived needs of the in situ, offline testing environment, which includes power-on BIT, POST, and Initiated BIT. Several of the derived needs discussed also apply to depot testing environments.

4.1 Initiating Tests

PBITs are initiated automatically when power is applied (Phase 1) and the system is initialized (Phase 2). Phase 1 tests typically consist of low-level POSTs run on the machine's processor, memory, and attached hardware. Phase 2 tests typically consist of additional closed loop testing to verify subsystem interconnects and other configuration-determined items. Initiation diagnostics should include sufficient closed loop testing to provide reasonable assurance of the correct operation of the system; additional open loop testing can be required to verify controls and displays.

Initiated BITs are initiated by an operator. Because initiated BITs interfere with normal system operation, the system should be placed in non-operational mode either prior to BIT initiation (recommended) or as a consequence of BIT initiation. Other aspects of the BIT initiation environment are addressed in Sections 4.2 and 4.3.

4.2 Grouping Tests

The set of available discrete Initiated BIT tests may be large to provide comprehensive test coverage. Some of the tests can take an extended period of time to complete (e.g., comprehensive memory diagnostics). In addition, a system may contain a large number of LRUs, each requiring separate testing. Also, to support effective fault isolation, BIT tests must be able to be grouped at the LRU level. The tests should be able to be grouped at the SRU level.

Consider the need for operator interaction when providing the capability to group tests. Closed loop tests should be able to be grouped separately from open loop tests. To support repair verification, tests can be grouped but should be able to be selected individually for execution.

4.3 Iterating Tests

To diagnose intermittent failures, BIT test groups should be capable of multiple iterations with a single invocation. This capability should support both a fixed iteration count and an indefinite number of repetitions. When indefinite repetition is an option, the test should be configurable so that an operator-defined failure statistic terminates the iteration. This “stop on failure statistic” capability should be available even for fixed iteration counts and at a minimum, should support “halt on first failure.”

4.4 Terminating Tests

BITs for which failing to complete is a possibility must be capable of being terminated. An operator command and a timeout value should both be available to terminate the test. In addition, any BIT that can be iterated indefinitely must be able to be terminated.

Since terminating a BIT prematurely can leave a system in an unknown and potentially hazardous state, test termination should be followed by activities that return the system to a known, operable state.

4.5 Reporting Results

Test completion reports should include summary status (“go” or “no-go”) for the test group. In the event of a failure, the reports should contain more detailed status information. A detailed status should characterize the failure to at least the LRU level. To support the Repair of Repairables mission, the detailed status report should characterize the failure to the lowest level of replaceable components.

In the event that a failure cannot be isolated to a single component, the completion report should report the set of possibly failed components, in probability order, high to low. Note, however, that implementing this capability requires detailed design analysis to be performed for both failure modes and effect and for the installed reliability of components.

Information about test configurations and failure reports should have the ability to be exported from the system in an open usable and format. They should then be forwarded with the failed item into the depot repair system.

4.6 Degree of Platform Integration

Designers should consider the degree of platform integration required by the integrated diagnostics environment when a system is to be installed on a different platform (e.g., lamp testing of a composite system console by using a single control).

4.7 Impact of a System Operational Profile on Diagnostics

The operational profile of a system affects the capabilities and relative importance of various in situ test environments and must be considered in the design of the integrated diagnostics environment. An *operational profile* describes the normal operating period for a system, such as 24 hours a day, 7 days a week, 365 days a year; or for the duration of a normal 2-hour mission.

For example, if a system has an operating profile to run constantly, without defined preventive maintenance periods being set aside, the importance and capabilities of the online test environment must increase dramatically. This is because comprehensive offline tests that must exist to support the Fault Isolation diagnostic mission, cannot be executed without affecting the system's operational profile adversely.

Similarly, if the system being analyzed is installed on a vehicle, the vehicle's mission cycle provides the opportunity for running comprehensive power-up tests relatively frequently. However, if the vehicle's operational profile contains a "quick launch" requirement, then the power-up tests must be run in an expedited manner.

5 Capability Gaps in Integrated Diagnostics

A number of diagnostic capability gaps currently exist in DoD hybrid systems. Some of these capability gaps appear to be the result of the increasing use of commercial items in defense systems. Others seem to be the result of considering only faults in the hardware domain, while the increasingly important area of software health management is ignored. Some of these capability gaps are addressed below.

5.1 Test Coverage

One continuing challenge is to ensure that an adequate portion of the system is covered by diagnostic testing environments. With the increasing use of COTS software products, this coverage may be available only during POSTs and even then, only with cursory coverage. Another challenge is that offline vendor diagnostics may exist, but may be difficult to integrate into the overall diagnostics system. If continuous health monitoring is required, then additional capabilities need to be created to cover the in situ, online diagnostics to satisfy the Fault Detection and Characterization diagnostic mission.

5.2 Correlating Faults to Failures

As the complexity of computerized systems increases, in-depth knowledge of the architectures of a system is required to effectively correlate a collection of error reports (faults) to what is wrong (failures). This shortfall has two aspects: the ability to isolate a collection of faults to a failed unit or units, and the ability to determine the impact of the failed unit on system operations.

5.2.1 Ability to Isolate Failures

Diagnostic activities can generate a collection of test failure reports or fault reports. Fault reports are not always easily correlated to a specific failure. To effectively isolate failures, operators need information from the system's Failure Modes and Effects Criticality Analysis (FMECA) as to the number and type of possible failures and an assessment of the likelihood that a particular fault or pattern of faults represents a failure in a particular component. This detailed system information is not generally available to anyone but the developing contractor but is critical to the cost-effective support of the system. Some of this data is captured in maintenance technical orders, but significant detail can be lost due to inadequate modeling.

Systems requirements increasingly specify that some or all FMECA information be incorporated into the diagnostics environment, but this capability is neither widespread nor easily achieved.

5.2.2 Determining the Consequence of a Failure

Additional complexity in interpreting the results of diagnostic tests lies in determining if a failure has operational impact. For systems that have many interfaces, some failures can clearly be categorized as critical and catastrophic. Other failures can represent components that are not currently in use or are not critical to the current mission. To effectively determine the consequence of a failure, operators must have detailed knowledge about the component that has failed, detailed knowledge about their current mission, and how this component supports or does not support the mission. It is unlikely that this level of assessment will be able to be automated in any significant way in the near future.

5.3 Fraction of False Alarms – False Positives

A continuing problem in diagnostic environments is the number of *false positive* fault reports or reports of anomalous behavior that do not represent failures. In some cases, resolution of false positive tests cause significant expense in the logistics support environment, causing the need for additional spare components to be held in inventory and requiring that repetitive and often futile detailed testing activities be performed. For example, in some systems the “could not duplicate” rate of “failed” units exceeds 40% [Neumann 89; Nordwall 86]. Reducing the fraction of false alarms and false positive, while maintaining a responsive online diagnostic environment, is a critical but elusive goal for existing and emerging systems.

5.4 Software Health Management

While hardware diagnostic environments have a long history in defense systems, software diagnostic environments—diagnostic environments that monitor software systems for faults—are in their infancy. Many of the tools in use for assessing hardware faults, such as FMECA, are not yet accepted in the software domain. Indeed, due to the large number of both direct and indirect connections that software systems possess, performing meaningful software FMECA may not be possible or cost-effective.

There are a number of things that properly architected software systems can do to detect and report failures. Software can be created to monitor the system at the operating system level, ensuring that the tasks that are intended to be present actually are. Software components can maintain activity counters that are monitored to ensure that individual components or threads of execution do not terminate or become trapped in an infinite loop. Certain classes of systems can also contain “oracle” applications or software code that models the expected outcome of dynamic control activities and traps inappropriate behaviors. These oracle

applications include system resource monitors that can detect, report, and manage impending overload conditions.

All of these techniques are possible, but are usually outside of the systems engineers' experience base. Further, several of these concepts are fairly new and may be outside of the experience of the lead software engineers and architects. Regardless of whether diagnostic capabilities are implemented by using hardware or software, acquirers should demand that systems have the capability to be monitored for correct operation and that operational faults be reported for failure analysis and repair.

6 Advanced Testing Environments

It's often difficult to diagnose a system failure by examining a collection of fault reports. Some small failures can generate a large collection of cascading fault reports. Other failures cannot be uniquely isolated to a single replaceable component, so additional isolation procedures must be performed. In each of these cases, diagnosing a system failure requires a deep and detailed understanding of the design, construction, and installation of the system.

Efforts to install this system knowledge in Interactive Electronic Technical Manuals (IETMs) are ongoing [IETM 05]. The use of IETMs helps solve the problem of out-of-date, incorrect, or unavailable paper technical manuals. In addition, these and other information bases are being encoded into expert systems knowledge bases. These diagnostic expert systems are intended to more quickly and accurately support the complicated process of fault isolation. Collectively, this area is known as Computer Aided Logistics Support (CALs).

Recently, to improve system operational availability, maintenance focus is changing from a diagnostic posture to a prognostic one. A *prognostic* is defined as a "forecast of future performance and/or condition" [Howard 04]. In order to support prognostic maintenance, detailed component and system reliability models must be created and maintained. These models may be augmented with direct measurement data so that they can then drive focused preventive maintenance activities.

Bibliography

This list consists both of references cited in the text and a bibliography of papers, journal articles, and conference presentations reviewed during the development of this report. Also listed are additional sources of information and participation.

Many of these references date from the late 80s and early 90s and relate to avionics systems. This is attributable to the Pave Pillar avionics research program, which led to the development of the integrated avionics environments used in the F-22 (Raptor), RAH-66 (Comanche), and F-35 (JSF) programs.

The NDIA Systems Engineering Division is currently leading efforts to re-invigorate the Systems Engineering process including Diagnostics and Supportability. The author encourages readers to further investigate and participate in this organization.

URLs are valid as of the publication date of this document.

- [Bohr 99]** Bohr, J.D. “Diagnostic Metrics—A Critical Element of the Diagnostic Architecture,” 215–221. *Proceedings of AUTOTESTCON ‘99. IEEE Systems Readiness Technology Conference*. San Antonio, TX, Aug. 30–Sept. 2, 1999. Piscataway, NY: IEEE, 1999.
- [Dean 96]** Dean, J.S. “Integrated Diagnostics: Confusion and Solutions,” 436–440. *Proceedings of the AUTOTESTCON ‘96. IEEE Systems Readiness Technology Conference*. Dayton, OH, Sept. 16, 1996. Piscataway, NY: IEEE, 1996.
- [DoD 97]** DoD. *MIL-HDBK-2097A: Acquisition of Support Equipment and Associated Integrated Logistics Support*, 1997.
- [Doskocil 88]** Doskocil, D.C. “A Comprehensive Strategy for Implementing BIT [Weapon Systems],” 211–215. *Proceedings of the AUTOTESTCON ‘88 Symposium*. Minneapolis, MN, Oct. 4, 1988. Piscataway, NY: IEEE, 1988.

- [Esler 90]** Esler, E.A.; Simpson, W.R.; & Sheppard, J.W. "An Embedded Maintenance Subsystem," 331–336. *Proceedings of the AUTOTESTCON '90. IEEE Systems Readiness Technology Conference*. San Antonio, TX, Sept. 17, 1990. Piscataway, NY: IEEE, 1990.
- [Freschi 98]** Freschi, S. & Shombert, L.A. "Optimizing the Use of Component-Level BIT in System-Level Test and Diagnosis: A Consistent Diagnostic Information Architecture Approach," 103–110. *Proceedings of AUTOTESTCON '98*. Salt Lake City, UT, Aug. 24, 1998. Piscataway, NY: IEEE, 1998.
- [Gaffney 94]** Gaffney, K.S. & Morones, P. "Advanced-Aircraft Integrated-Diagnostics System–Concept Evaluation," 20–25. *Proceedings of the 1994 Reliability and Maintainability Symposium*. Anaheim, CA, Jan. 24, 1994. Piscataway, NY: IEEE, 1994.
- [Giordano 90]** Giordano, P.J. & Nolan, M. "Integrated Diagnostics from a Concurrent Engineering Perspective," 589–594. *Proceedings of AUTOTESTCON '90. IEEE Systems Readiness Technology Conference*. San Antonio, TX, Sept. 17, 1990. Piscataway, NY: IEEE, 1990.
- [Howard 04]** Howard, P.L. "Prognostics and Health Management for Electronic Systems Workshop."
http://www.calce.umd.edu/whats_new/upcoming/2004/workshop/02.pdf (2004).
- [IETM 05]** IETM. Interactive Electronic Technical Manuals.
<http://www.ietm.net> (2005).
- [Keiner 90]** Keiner, W.L. "A Navy Approach to Integrated Diagnostics," 443–450. *Proceedings of AUTOTESTCON '90. IEEE Systems Readiness Technology Conference*. San Antonio, TX, Sept. 17, 1990. Piscataway, NY: IEEE, 1990.
- [Mosley 90]** Mosley, A. "Why an Air Force Centralized Integrated Diagnostics Office?" 473–476. *Proceedings of AUTOTESTCON '90. IEEE Systems Readiness Technology Conference*. San Antonio, TX, Sept. 17, 1990. Piscataway, NY: IEEE, 1990.
- [NASA 05]** NASA. NASA Goddard Space Flight Center Tin Whisker Homepage. <http://nepp.nasa.gov/whisker/> (2005).

- [Neumann 89]** Neumann, G. W. "Integrated Diagnostics—A Help or Hindrance?" 1316–1320. *Proceedings of the IEEE 1989 National Aerospace and Electronics Conference*. Dayton, OH, May 22, 1989. Piscataway, NY: IEEE, 1989.
- [Nichols 91]** Nichols, D.L.; Vicen, P.M.; & Marcum, R.B. "Software—Missing Piece in the Integrated Diagnostics Puzzle," 1220–1224. *Proceedings of the IEEE 1991 National Aerospace and Electronics Conference*. Dayton, OH, May 20, 1991. Piscataway, NY: IEEE, 1991.
- [Nolan 89]** Nolan, M.E. "Integrated Diagnostics: Needs, Technologies and Current Initiatives," 1252–1256. *Proceedings of the IEEE 1989 National Aerospace and Electronics Conference*. Dayton, OH, May 22, 1989. Piscataway, NY: IEEE, 1989.
- [Nordwall 86]** Nordwall, B. "Air Force Links Radar Problems to Growth of Tin Whiskers." *Aviation Week and Space Technology Vol. 124*, No. 26 (June 1986): 65, 67, 69.
- [Samson 90]** Samson, R. & Golas, M. "Overview of the GIMADS ID Process," 209–220. *Combined proceedings of the 1990 and 1991 Leesburg Workshops on Reliability and Maintainability Computer-Aided Engineering in Concurrent Engineering*. Leesburg, VA, Oct. 9, 1990. Piscataway, NY: IEEE, 1990.
- [Sheppard 98]** Sheppard, J.W. & Simpson, W.R. "Managing Conflict in System Diagnosis." *IEEE Computer* 31, 3 (March 1998): 69–76.
- [Sheppard 93]** Sheppard, J.W. & Simpson, W.R. "Performing Effective Fault Isolation in Integrated Diagnostics." *IEEE Design & Test of Computers* 10, 2 (June 1993): 78–90.
- [Sheppard 92]** Sheppard, J.W. & Simpson, W.R. "Applying Testability Analysis for Integrated Diagnostics." *IEEE Design & Test of Computers* 9, 3 (September 1992): 65–78.
- [Sheppard 91]** Sheppard, J.W. & Simpson, W.R. "A Mathematical Model for Integrated Diagnostics." *IEEE Design & Test of Computers* 8, 4 (December 1991): 25–38.

- [Sheppard 90a]** Sheppard, J.W. & Simpson, W.R. “Integrated Diagnosis—A Hierarchical Approach,” 477–483. *Proceedings of AUTOTESTCON '90. IEEE Systems Readiness Technology Conference*. San Antonio, TX, Sept. 17, 1990. Piscataway, NY: IEEE, 1990.
- [Sheppard 90b]** Sheppard, J.W. & Simpson, W.R. “Incorporating Model-Based Reasoning in Interactive Maintenance Aids,” 1238–1242. *Proceedings of the IEEE 1990 National Aerospace and Electronics Conference*. Dayton, OH, May 21, 1990. Piscataway, NY: IEEE, 1990.
- [Simpson 93]** Simpson, W.R. & Sheppard, J.W. “Fault Isolation in an Integrated Diagnostic Environment.” *IEEE Design & Test of Computers* 10, 1 (March 1993): 52–66.
- [Simpson 92]** Simpson, W.R. & Sheppard, J.W. “System Testability Assessment for Integrated Diagnostics.” *IEEE Design & Test of Computers* 9, 1 (March 1992): 40–54.
- [Simpson 91]** Simpson, W.R. & Sheppard, J.W. “System Complexity and Integrated Diagnostics.” *IEEE Design & Test of Computers* 8, 3 (September 1991): 16–30.
- [Stacey 88]** Stacey, T.R. & Whaley, M.D. “Diagnostics Integration—A TAC Maintainer’s View,” 1448–1450. *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*. Dayton, OH, May 23, 1988. Piscataway, NY: IEEE, 1988.
- [Weber 92]** Weber, J.P. “Integrated Diagnostics for Software,” 565–571. *Proceedings of the 1992 National Aerospace and Electronics Conference*. Dayton, OH, May 18, 1992. Piscataway, NY: IEEE, 1992.

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE September 2005	3. REPORT TYPE AND DATES COVERED Final		
4. TITLE AND SUBTITLE Integrated Diagnostics: Operational Missions, Diagnostic Types, Characteristics, and Capability Gaps		5. FUNDING NUMBERS FA8721-05-C-0003		
6. AUTHOR(S) Theodore F. Marz				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2005-TN-035		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES				
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE		
13. ABSTRACT (MAXIMUM 200 WORDS) The Acquisition Support Program at the Carnegie Mellon Software Engineering Institute assists organizations that develop Department of Defense hardware/software hybrid systems by participating in technical reviews and other validation activities. During recent customer engagements, it has been noted that some development teams do not demonstrate a consistent understanding of integrated diagnostic system needs. In particular, software engineers seem to lack the experience needed in this area. Since software engineers frequently derive the low-level requirements for developing diagnostic systems, a lack of knowledge about deployed testing environments can have significant impact. Failure to adequately address the integrated testing needs of a system profoundly impacts its supportability and, consequently, the cost of that system throughout its life cycle. This report attempts to fill in these gaps in knowledge and experience by presenting an overview of the operational diagnostic life cycle of a system. In addition, it outlines how a system's operational profile impacts diagnostic tradeoffs.				
14. SUBJECT TERMS operational mission, operational system, software-intensive systems, systems engineering, testing		15. NUMBER OF PAGES 38		
16. PRICE CODE				
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	