Integrated System Health Management (ISHM) Technology Demonstration Project Final Report

Principal Investigators

Ryan Mackey
Jet Propulsion Laboratory

David Iverson
NASA Ames Research Center

Team Leads

Greg Pisanich
NASA Ames Research Center

Mike Toberman
NASA Dryden Flight Research Center

Ken Hicks
Jet Propulsion Laboratory

December 15, 2005

PROJECT TEAM MEMBERS

ARC	DFRC	JPL
Scott Christa	Bob Antoniewicz	Edmund Baroth
Anthony Gross	David Dowdell	Len Day
David Iverson	Gayle Patterson	Martin Feather
David Lawrence	Marlin Pickett	Kenneth Hicks
Dougal Maclise	James Stewart	Mark James
Greg Pisanich	Mike Toberman	Ryan Mackey
	Ron Wilcox	Raffi Tikidjian

OTHERS	
Dan Itsara	U.S. Air Force
David Wang	MIT

TABLE OF CONTENTS

PROJECT TEAM MEMBERS 2
TABLE OF CONTENTS 3
INTRODUCTION 5
Project Background 5
Aircraft as Spacecraft Proxy 6
Test bed Advantages 7
OBJECTIVES AND SUCCESS CRITERIA 7
Goals and Objectives 7
Success Criteria 7
TEAM COMPETENCIES 8
Dryden Flight Research Center (DFRC) – Aircraft and Hardware Support 8
Jet Propulsion Laboratory (JPL) – Beacon-based Exception Analysis for Multi-missions (BEAM) Systems Engineering Support 8
Ames Research Center (ARC) – Inductive Monitoring System (IMS) and Systems Integration 9
X-Works Team Concept 9
Team Organization 9
Project Communications 10
Technical Approach 10
Project Budget and Schedule 12
TECHNICAL PROJECT COMPONENTS 13
Aircraft Subsystem Selection 13
Data Selection and Access 13
Data Processor Hardware 15
Software and Algorithm development 16
Dispatcher 16
Inductive Monitoring System (IMS) 18
State Space Recognition 19
Alert Logic 19
Scaling of Database 19
Control and Status Vectors 19
Statistical Weighting 20
Engine State Matrix and Logic Partitioning 20
Beacon-based Exception Analysis for Multi-missions (BEAM) 21
BEAM Data Coverage 23
Building the BEAM State Model 24
Building the BEAM Statistics Model 25
Software Configuration 27
Simulation Testing 27
F/A-18 Flight Simulator 28
Ames 1553 Bus Analyzer 29
F/A-18 Flight Test Aircraft 29
Processor Environmental Testing 29

Processor-Aircraft Integration 30
PRE-FLIGHT PROCEDURES AND FLIGHT TESTING 32
Hardware and Software Ground Tests 32
Flight Approval and Initial Test Flights 32
Data Download and Analysis 33
RESULTS AND CONCLUSIONS 34
Success Criteria Exceeded 34
Probable Anomalies Detected 34
Transient Effects 34
Repeated Intermittent 36
Unanticipated Operating Mode 41
Seeded Faults 41
F/A-18 Technology Accelerator was a Successful Test Bed 43
Team Synergy 44
X-Works 44
LESSONS LEARNED 44
Allow extra time when integrating code into new hardware and operating systems 44
Recognize and address ITAR issues early in the project 45
Use existing hardware and software whenever possible 45
Plan extra time for travel, even if the location is nearby 46
NEXT STEPS 46
Algorithm Development 47
Development Directions for IMS 47
ISHM Technologies 48
ISHM Operations 50
F/A-18 Technology Accelerator 50
SUMMARY 51
ACKNOWLEDGEMENTS 52
REFERENCES 52

INTRODUCTION

PROJECT BACKGROUND

Integrated System Health Management (ISHM) is an essential capability that will be required to enable upcoming explorations mission systems such as the Crew Exploration Vehicle (CEV) and Crew Launch Vehicle (CLV), as well as NASA aeronautics missions. ISHM technologies will improve the reliability and availability of complex systems, reduce the amount of manpower required to prepare vehicles for both aeronautical and space flight, and reduce the number of induced failures caused by intrusive inspections. NASA Ames Research Center (ARC) and the Jet Propulsion Laboratory (JPL) have developed strong competencies in ISHM, particularly in the areas of data fusion, data analysis, monitoring, and diagnosis; however, the lack of flight experience and available test platforms have held back the infusion of these technologies into future space and aeronautical missions.

To address this problem, a pioneer project was conceived to use a high-performance aircraft as a low-cost proxy to develop, mature, and verify the effectiveness of candidate ISHM technologies. Given the similarities between spacecraft and aircraft, an F/A-18 currently stationed at Dryden Flight Research Center (DFRC) was chosen as a suitable host platform for the test bed. Given the availability of the aircraft, this test bed allows ISHM technologies to be developed more economically and mature more quickly.

Figure 1. F/A-18 High-performance test aircraft at NASA Dryden

This report describes how the test bed was conceived, how the technologies were integrated on to the aircraft, and how these technologies were matured during the project. It also provides the reader with a description of the lessons learned during the project and a forward path for the continuation of this important work.

AIRCRAFT AS SPACECRAFT PROXY

A high-performance aircraft provides many of the same relevant characteristics and challenges as a spacecraft and could effectively be used as a surrogate for developing new technologies for space flight. From a top-level prospective, a high-performance aircraft provides similar subsystems, including propulsion, avionics, environmental controls, and power distribution (see Figure 2). Software development requirements for data collection, data filtering, and interpretation are comparable. In addition, issues involved in modeling, integrating, and fielding ISHM systems are similar for both platforms.

A high-performance-aircraft can also provide similar environmental and integration challenges, including data bus, EMI, electrical noise, vibration, and thermal issues within a defined flight envelope. There have been previous efforts between the aircraft and space community to develop systems that were capable of supporting tests in both the atmospheric and space environment, but these were primarily point designs that are not relevant to today's challenges. We would like to engage these groups in a dialog to define general environmental operating envelopes for ground, aircraft, and sounding rocket tests that would support spacecraft technology development.

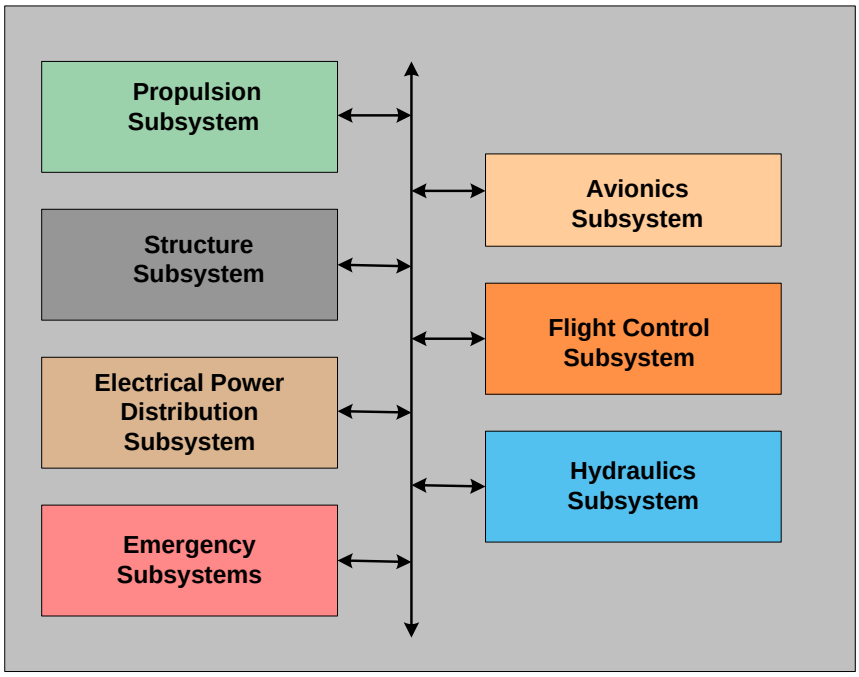

Figure 2. Generic aircraft/spacecraft block diagram, subsystem level

Health management software for an F/A-18 cannot be expected to perform the same functions for a CEV; however, the building blocks for creating such a health management system could be conceived and significantly tested on the F/A-18. By using a high-performance aircraft as a test bed, the technology can gain credibility, as ISHM in general has demonstrated that it can reduce the number of maintenance hours required to service a vehicle.

TEST BED ADVANTAGES

A high-performance aircraft also provides significant advantages to the ISHM technology development process: lower cost, more flights, and deeper integration experience. Spacecraft are currently few and expensive to operate in comparison to most aircraft. Test aircraft, such as those at Dryden, can fly frequently with a relatively low per-fight cost. Estimates for dedicated sounding rocket flights are $100K/flight, while flights of rockets capable of being controlled in flight are $1M/flight. In comparison, flights of the F/A-18 at Dryden cost approximately $8K/flight and are reusable vehicles. If anomalies occur during flight, they can be investigated on the aircraft's return.

With the aircraft's ability to fly often, software and systems can be developed faster. Frequent flights provide more chances to load and test new versions of the algorithms and software. An important aspect in the development of ISHM technologies—especially the detection software used in this project—is the need for flight data. Aircraft that fly frequently can provide more data sets, which can be used to better train and test the models.

Finally, the test bed accelerates the process of development by providing ISHM developers with valuable integration experience. The developers can move the software out of the laboratory and expose it to a real-time, real-world environment. They also gain experience with integration timelines, noisy data, and the "give and take" of development. This process of hardening and enhancing software effectively raises the Technical Readiness Level (TRL) of the technologies from 4 to 6. Flight demonstration on a relevant test bed should provide verification to the space flight community that the software is available as a candidate space technology.

OBJECTIVES AND SUCCESS CRITERIA

GOALS AND OBJECTIVES

The goals of the F/A-18 ISHM Test Bed Project were to achieve the following:

1. Establish a test bed facility for low-cost, high-return technology maturation using ground and aircraft resources
2. Use the test bed to mature and demonstrate the effectiveness of two existing ISHM technologies: Inductive Monitoring System (**IMS**) and Beacon-based Exception Analysis for Multi-missions (**BEAM**)
3. Develop new relationships with other NASA centers to create new and improved synergistic teams and partnerships

SUCCESS CRITERIA

The F/A-18 ISHM Test Bed Project had both minimum and complete success criteria.

Minimum success criteria:

- Validate BEAM and IMS correct function on spacecraft-like, ground-based hardware

- Achieve first test flight of algorithms by August 1, 2005

Complete success criteria:

- Verify that in-flight performance of BEAM and IMS matches performance predictions from laboratory analogues
- Achieve a minimum of three, and preferably ten, test flights by September 2005

TEAM COMPETENCIES

Three NASA centers—Ames Research Center (ARC), the Jet Propulsion Laboratory (JPL), and Dryden Flight Research Center (DFRC)—were involved in developing the F/A-18 ISHM host test bed. This technology maturation project required the integration of modeling, software development, hardware and avionics integration, aircraft systems, flight test, and project development technologies. Each center brought special competencies to this integrated team effort.

DRYDEN FLIGHT RESEARCH CENTER (DFRC) – AIRCRAFT AND HARDWARE SUPPORT

NASA Dryden provided significant experience in the development and testing of flight hardware. Dryden currently maintains several different flight platforms which are capable of accepting processors and instruments that can be tested in flight, including the F/A-18 used in this project. Dryden has specific expertise in the development and integration of small processors for flight testing and provided the PC/104 data processor for this project, as well as expertise and facilities for environmental and flight qualification of hardware.

Dryden also provided archived flight data from specifications that could be used for analysis or algorithm development. Dryden on-site engineers were familiar with the various aircraft and systems and could advise on their operation and limitations. Dryden maintains an F/A-18 flight simulator that was used to simulate flight conditions for testing flight software and 1553 interface and hardware communication. Finally, Dryden provided the flight test support, scheduling, and data download abilities crucial to the project.

JET PROPULSION LABORATORY (JPL) – BEACON-BASED EXCEPTION ANALYSIS FOR MULTI-MISSIONS (BEAM) SYSTEMS ENGINEERING SUPPORT

The Jet Propulsion Laboratory brought extensive expertise to the project in autonomous sensing and control, vehicle health management, and systems integration. In particular, JPL developed and integrated BEAM, a tool designed to detect and classify anomalies and degradation in a wide variety of complex systems. The BEAM algorithm provides a “front-end” system failure detector by establishing (learning) baseline system operations and detecting off-nominal conditions when they occur.

In addition, JPL has extensive experience in the development of space systems that incorporate new technologies in critical mission applications. Combined with DFRC’s unique knowledge of aircraft flight test platforms, JPL adapted BEAM for demonstration and validation using ISHM-relevant F/A-18 flight systems.

The overall system architecture, operating system, and algorithm integration responsibility resided with the project System Engineer (SE) at JPL. The JPL SE had a dual role as the Dryden Project Chief Engineer, and this position carried the traditional burdens of responsibility for and authority over all aspects of system development and end performance. The SE position was complimented by a system technical lead at Dryden who was familiar with, and implemented, all of the safety, qualification, and review procedures required for avionics integrated into manned NASA aircraft.

AMES RESEARCH CENTER (ARC) – INDUCTIVE MONITORING SYSTEM (IMS) AND SYSTEMS INTEGRATION

NASA Ames Research Center brought expertise in the areas of model-based reasoning methods and development experience of real-time ISHM systems with several successful demonstrations and flight experiments.

ARC used its experience to develop the Dispatcher program that read applicable data from the F/A-18 MIL-STD-1553 data bus and served this data to the ISHM algorithms (IMS and BEAM) for processing. Ames also provided and enhanced the IMS detector that monitored the data stream for anomalies based on data-derived models and data fusion. IMS had been previously used in several engineering domains, including space shuttle post-flight and helicopter real-time data analysis.

ARC also provided project management and organization expertise. Software configuration management was performed at ARC using the free, open source Concurrent Version System (CVS), which was accessed by the NASA centers to control the current version of all software.

X-WORKS TEAM CONCEPT

Using the competencies and resources of three centers, the project organized along a new program architecture that could potentially reduce project risk. Using industry as a model, the X-works tenets include:

- Keep the team small to minimize communication delays; use experienced leads and senior engineers wherever possible.
- Use the best facilities and people wherever they exist rather than forcing certain work to a certain lab (or center).
- Leverage existing technologies and facilities wherever possible so that development can focus on development needs (ISHM software, in this case).

The promise of X-works is to help address technology maturation and risk reduction requirements for a wide variety of ongoing and future NASA projects.

TEAM ORGANIZATION

DFRC was the lead center for the project. The Dryden project manager was responsible for the coordination of scheduled project activities with the other managers at both Ames and JPL. Project management also included systems engineering led out of JPL, team leads at Ames and JPL, and a JPL systems engineer.

Table 1 summarizes the facilities and expertise provided by each center, along with their development responsibilities on the project.

Table 1. Facilities/Expertise and Project Responsibilities

Facilities and Expertise		
DFRC	**JPL**	**ARC**
F/A-18 Aircraft	Software integration laboratories	Software and hardware integration laboratories
Data Processor Hardware	Hardware and operating system expertise	Real-time ISHM system development
F/A-18 Flight Simulator	Systems Engineering	1553 simulation
Environmental Testing		
Development Responsibilities		
DFRC	**JPL**	**ARC**
Simulation and Engineering Support	BEAM algorithm development	Dispatcher data server development
Archived data and data reduction	Systems Engineering	IMS algorithm development
Hardware Integration	Test and integration	Engineering, test, and integration
Flight Operations		

PROJECT COMMUNICATIONS

Given six-month duration and a very limited budget, the project did not follow a traditional PDR, CDR format. Instead, a project plan was used to document technical decisions, and technical milestones were used to drive the schedule and assess progress. Weekly teleconferences were also used to monitor and focus development, and technical breakout meetings were conducted to resolve issues.

Technical documents and data were provided on a NASA secure web access system (PBMG). A CVS software configuration site was used to manage software and flight data. A Flight Readiness Review and a One-Design Review were also required.

TECHNICAL APPROACH

A minimum number of initial meetings were required to develop a feasible project plan that included a technical approach, work breakdown structure, resource allocations, budgets, and a realistic schedule. The following is a high-level summary of our development process.

- **Use an existing Dryden F/A-18 as test vehicle:** Our test aircraft was active on the flight line and would require minimal work to integrate new ISHM hardware.

- **Use an existing flight-certified hardware design:** We used an existing PC/104 data processor and a Linux OS that had been previously flight tested, rather than try to integrate a new processor and operating system. We borrowed a processor from another Dryden group while we purchased additional systems as flight and test units.

- **Interface with a 1553 bus to collect data and serve it to IMS and BEAM:** In parallel with purchasing and integrating the flight hardware, we developed a reliable data server that read data from the 1553 bus, stored the data to disk, and served the data to the two ISHM algorithms. We used the 1553 bus because of its simple interface and parallel space bus systems.

- **System selection, model development, and data gathering**: We chose to model and monitor the two aircraft engines based on data available on the 1553 bus for those systems and team expertise in engine systems. We also chose to gather system input (e.g., throttle position) and state data that could be used to better understand engine operation under differing flight conditions.

- **Develop enhanced versions of BEAM and IMS:** While the Dispatcher was being developed, we used archived data to build initial versions of the IMS and BEAM systems. As flight data became available, additional models and algorithm modifications were made, leading to the versions that flew on the aircraft.

- **Verify operation of the hardware and software using the DFRC F/A-18 simulator:** We used the F/A-18 flight simulator in several modes. Initial tests of the data processor and software were performed by interfacing to the simulator via the 1553 bus. We also used the simulator to further test the IMS and BEAM software by exercising the simulator through several engine and control scenarios.

- **Qualify, install, and test the data processor and software:** Established Dryden environmental and safety tests were performed on the data processor in parallel with preparing the aircraft for hardware integration. Several ground tests were performed to ensure the proper operation of the software.

- **Repeat for a minimum of at least three data collection flights:** We planned on a minimum of three successful data collection flights to verify proper flight operation, make changes to the software, and run a final version. We actually were able to greatly exceed this number. The following processes were repeated for each flight.

    - **Install current software, fly and collect data:** We froze the version of the Dispatcher, BEAM, and IMS software the weekend before the available flight. This version of the software was installed on the data processor and tested using the flight simulator. The data processor was installed on the aircraft and the software “rode along” to monitor the operation of the engines during the flight. Flight data and IMS and BEAM data was recorded on a Flash disk.

    - **Download flight data:** The recorded data was downloaded from the data processor using a portable PC and made available to the developers.

- **Enhance BEAM and IMS based on current data**: The algorithms and system models were enhanced, based on their operation as compared with the data that was recorded. The operation of the Dispatcher was also modified as required.

PROJECT BUDGET AND SCHEDULE

The project budget was initially limited to $500K, with $200K, $150K, and $150K allocated to Ames, JPL, and Dryden respectively. Another $200K was added during the project to cover additional procurement and personnel, resulting in a final cost of $700K.

The project schedule is shown in Figure 3. The overall duration of the project was limited to six months, from April of 2005 to October of 2005. In spite of the short duration, we were able to meet our projected goal of a first flight of the software the first week of August 2005, and all major project goals and objectives were achieved prior to October 2005.

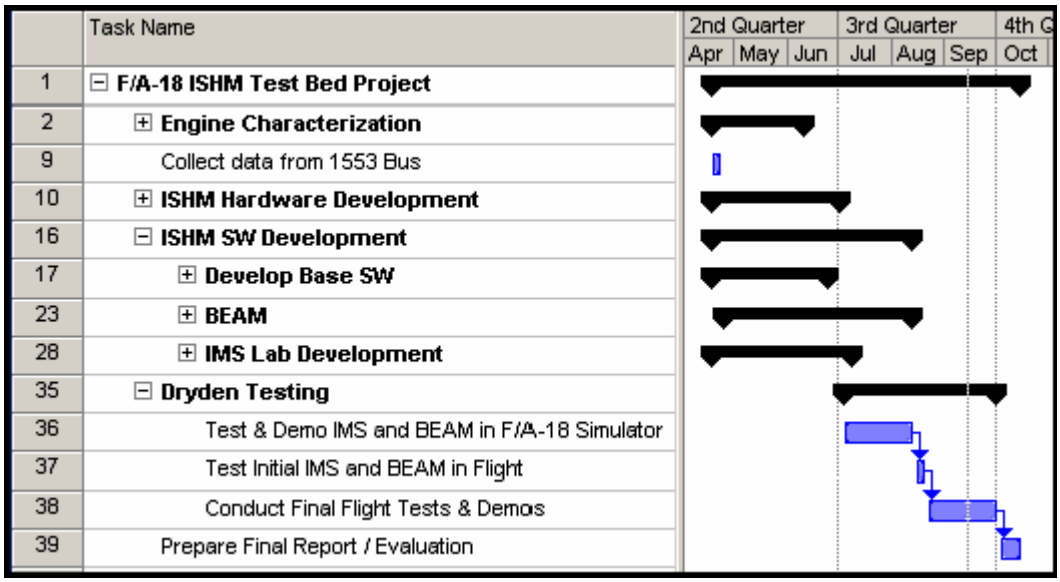

Figure 3. ISHM Technology Accelerator project schedule

Much of the credit for this achievement should be awarded to a highly talented and efficient project team; the development of a realistic project scope; a sound and realistic technical approach; and outstanding coordination, cooperation, and communication within and among all centers involved.

TECHNICAL PROJECT COMPONENTS

In addition to the development of the software components, several technical decisions, processes, and laboratories were required to realize the development process described above. The following sections describe individual components of the development process.

AIRCRAFT SUBSYSTEM SELECTION

The F/A-18's engines were selected as the target subsystem for the project for several reasons. The most important was the fact that a relatively large amount of data associated with both the left and right engines, including engine state and engine control input information, was accessible on the 1553 data bus. Experience with engine systems on prior projects by the JPL and ARC PIs and the legacy of DFRC support in this area also entered into the decision.

Another reason for selecting the engines for detection monitoring was related to the significant amount engine data that DFRC has accumulated over the years supporting F/A-18 aircraft. This data was invaluable in supporting early model development. In particular, BEAM model development and IMS learning (to be described in later sections of this report) were greatly accelerated, given the access to engine flight data early on in the project. Without this preliminary data, several flights would have been dedicated strictly to data collection and characterization.

Figure 4. F/A-18 engine in repair cradle

DATA SELECTION AND ACCESS

Given the selection of the engine subsystem, a large amount of associated data was required to develop preliminary BEAM models and to train IMS. Initially, 45 parameters were selected; however, not all of these parameters were included in the final configuration.

Table 2 shows the variables used. The same variables were selected from the left and right engine (15 state and 30 total engine variables). Most of the data parameters selected fall in one of three categories: engine control input, aircraft status, and engine status. This combination of parameters allows the ISHM algorithms to effectively determine if the engine exhibits consistent behavior for a given input and aircraft state. The data parameters used represented only a subset of all the parameters available on the engine (and the 1553 bus itself). The data required supporting the initial BEAM model development and the IMS learning processes originated from DFRC data archives. When building the models, we expected that this data would differ from what we might expect from the aircraft used for the flight. Note that X-band Center Freq. was later excluded, as the signal was completely flat in simulator and flight.

Table 2. Data Parameters Selected for Monitoring

Left and Right Engine Data Variables	Aircraft State Data Variables
X BAND CENTER FREQ	STATIC PRESSURE
ENG NARROW BAND VIBRATION	LOCAL ANGLE OF ATTACK
ENGINE BROAD BAND VIBRATION	MACH NUMBER
EXHAUST GAS TEMPERATURE	TRUE AIRSPEED
COMPRESSOR DISCHARGE PRESSURE	BAROMETER CORRELATED PRESSURE ALTIMETER
TURBINE DISCHARGE PRESS	TRUE SIDESLIP
LOW PRESSURE ROTOR SPEED	LOCAL SIDESLIP
HI PRESSURE ROTOR SPEED	DISCRETE AIRCRAFT STATES
ENGINE INLET TEMP	NORMAL ACCELERATION
ENGINE OIL PRESSURE	INDICATED STATIC PRESSURE
ENG NOZZLE POSITION	LONG STICK POSITION
MAIN FUEL FLOW	LAT. STICK POSITION
FUEL INLET TEMP	RUDDER PEDAL FORCE
TURBINE DISCHARGE TEMPERATURE	AMBIENT TEMPERATURE
POWER LEVER ANGLE	VERTICAL VELOCITY

The format used for the data was comma-separated text values. The data was distributed to JPL and ARC using approved International Traffic in Arms Regulations (ITAR) procedures. For the most part, BEAM and IMS used the same set of parameters to support their respective algorithms. Some of the data received from the data archives required preprocessing to remove data records perceived to be corrupted or not indicative of realistic system behavior. Much of this data could be found in the first few hundred records of the files; however, a significant number of records within the archived files

contained discontinuities that had to be removed in order to realistically characterize engine behavior. Because these files contained hundreds of thousands of records, they could not be processed using standard tools such as Excel. Instead, specific filter routines were produced in C++ to remove data discontinuities and data dropouts, or to remove incomplete data records. We also expected to need the same filtering algorithms for flight tests, under the assumption that the data discontinuities were a “real-world” property of the F/A-18 aircraft.

Because of the way that the 1553 data bus operates, the data required to support the engine Health Monitoring algorithms arrived from several different remote terminals. These remote terminals communicated on the bus asynchronously, and as a consequence the records received from the bus contained data from different times. If the delays are significant, the state of the vehicle may be misrepresented in each of the records. Additional filters were developed to discard those data vectors whose timestamps differed by more than a specified amount. This routine ensured that no vectors were processed that contained stale data. Timestamp information was available in both the archived files used to train the original applications for flight and in real time during flight from the 1553 bus. This filter was therefore useful in the post-processing algorithms and in the final flight application.

DATA PROCESSOR HARDWARE

The data processor and configuration used for this project had been used previously by DFRC for the C-17 program. This platform was adopted for this project because it provided sufficient resources at a low cost, had already been certified for flight, and DFRC had a previous track record on a prior project.

The data processor was composed of the following hardware components:

1. Power Supply
2. PC/104 Computer
3. RAM
4. Memtech AT1830-2048 2-GB Flash drive (mode PIO 4)
5. Condor Engineering Q104-1553 MIL-STD-1553A data bus interface card
6. Ethernet, Keyboard, SVGA, serial and parallel I/O ports.

DFRC provided four of the PC/104-based computer systems similar to that shown in Figure 5 for use in this task. These included two systems for use as ground test beds at Ames and JPL and two ruggedized and flight-qualified systems (flight and flight spare) suitable for use on the F/A-18. All other equipment was identified by the team as existing at their center, or was procured for use in the ISHM task.

In addition to the development systems, all handling fixtures, shock mounting hardware, and 24 VDC power supply filtering was supplied by DFRC. DFRC also procured, assembled, and qualified the hardware.

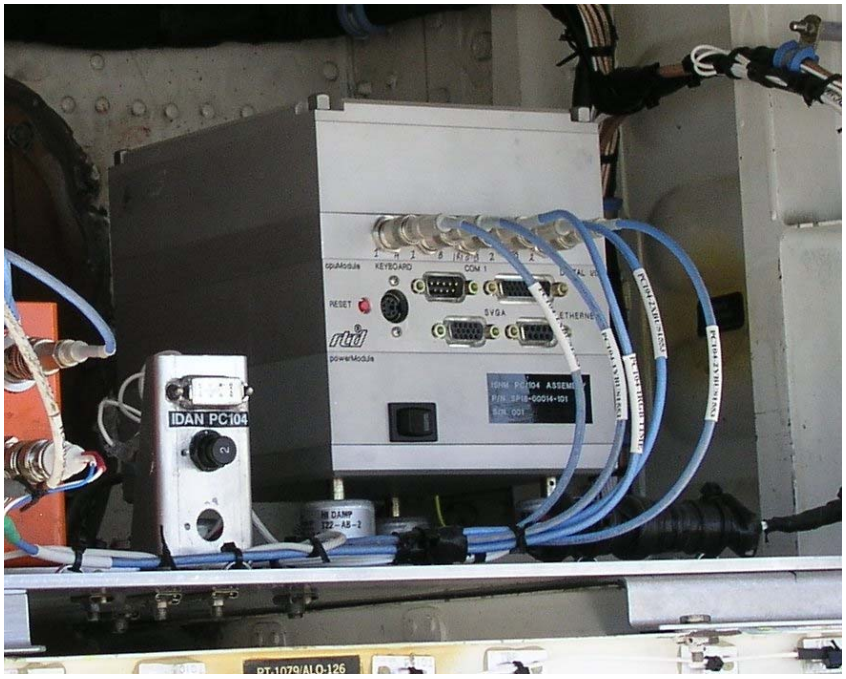

Figure 5. PC/104-based computer installed in project aircraft equipment bay

SOFTWARE AND ALGORITHM DEVELOPMENT

With the engine system selected, data available, and a data processor, the development team was able to begin building models. The major software modules in the system were the Dispatcher, BEAM, and IMS. This section will describe the functionality and capabilities of these algorithms.

Figure 6 shows the high-level software architecture and hardware interface for the flight processor. The system contained three software components: The Dispatcher (data server) and two detection algorithms: BEAM and IMS. The Dispatcher gathered data from the system by interfacing to the 1553 through an interface card and driver. The Dispatcher recorded the raw data to disk, converted the data to engineering units, and then served the data to the IMS and BEAM components. While processing the data, those components each saved their results to disk.

DISPATCHER

The Dispatcher was created as a standalone application to read raw data from the 1553 bus and to serve (dispatch) this data to IMS and BEAM. This approach has the advantage of avoiding duplication of code and minimizes code changes required to the BEAM and IMS. The Dispatcher was capable of reading from a file or from the MIL-STD-1553A bus. A command-line argument controlled selection of file playback or reading from the bus. This multiple role of the Dispatcher enabled both post-processing and real-time processing. Data could be received from another source as a file and then read in with the Dispatcher. IMS and BEAM would train and test themselves on this data before being deployed on the aircraft with real-time data.

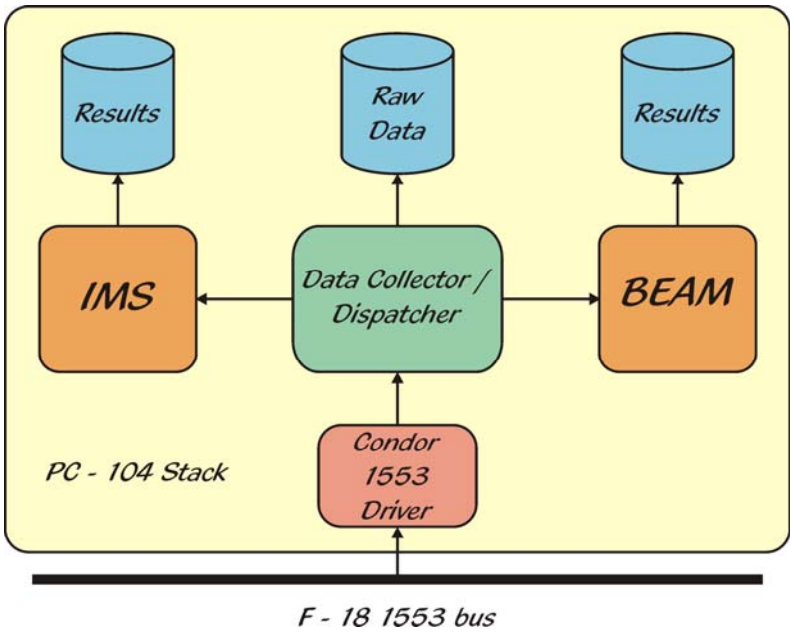

Figure 6. High-level component architecture

Each of the four flight hardware boxes was configured with a Debian Linux 2.4.25 operating system at DFRC. Ames received the first hardware box and configured it to allow developers from ARC and JPL to connect to it over the Internet using PuTTY and WinSCP. These applications allow users to log into the box via a remote terminal from either a UNIX or Windows operating system environment.

The manufacturer of the MIL-STD-1553A/B bus card, Condor Engineering, provided much of the software needed to configure the card to communicate on the 1553 bus and store raw 1553 data to disk. The code received from Condor was modified and integrated into the Dispatcher. Linux message queues were used for Inter Process Communication (IPC), as opposed to Pipes. If Pipes were used, the Dispatcher would potentially have to wait for IMS and BEAM to process data from the message queue before the Dispatcher could read additional data from the bus. As a consequence, data from the MIL-STD-1553A bus could be missed if the BEAM or IMS processes delayed the Dispatcher too long. Our use of message queues eliminated this problem because the Dispatcher could either wait for the other application to process data in its queue or continue on without waiting.

The Dispatcher receives real-time 1553 data from several remote terminals on the bus. These terminals communicate on the bus asynchronously as requested by the other aircraft data processors (not related to the ISHM PC/104 box). The Dispatcher timestamps the 1553 data as it is received from the remote terminals, based on the clock from the PC/104 CPU. The Dispatcher then converts the data into engineering units and stores the data in data packets as it is received from the remote terminals. Engine, control, and state data were transmitted over the MIL-STD-1553A bus as 16-bit signed integers. This raw data needed to be scaled by some factor to be converted into engineering units (e.g., altitude in feet, aircraft velocity as a Mach number, engine temp in Fahrenheit, etc.).

This saved time because only one conversion of raw data to engineering units was performed for both health monitoring applications.

The data contained in the data packets are listed in Table 2. All or part of each data packet could then be used by the health management applications. Regardless of the amount of data or the timing of data received from the terminals, the Dispatcher put a new data packet in two message queues every 20 milliseconds to be consumed by IMS and BEAM. If data is not received from any given terminal within a 50-millisecond interval, data from the last sample is used to fill the packet. (This data is referred to as "stale data"; BEAM and IMS applications have been configured to discard data packets containing timestamps that differ more than a specified amount.)

The Dispatcher also was programmed to store raw 1553 data to disk. Storing data to disk provided insurance that results from any given flight would not be lost in the event that the IMS or BEAM code should fail during flight: the data stored to disk could be played back on the ground using the Dispatcher's playback feature. The second reason behind storing the raw data was to further enhance the testing and debugging of the applications on the ground post flight. As more and more flights were conducted, the pool of available data was used to improve the fidelity of the BEAM models and to further train IMS. Each of the algorithms was responsible for writing its respective output data to file.

Engineers from JPL wrote the startup scripts that started the Dispatcher, IMS, and BEAM applications during boot-up. If an application died or quit, the start scripts would launch the application again automatically. This process was automated so that no human intervention was required to launch the software when the hardware box was powered up on the aircraft. It also provided insurance against software resets or failure to execute if there were any unexpected brownout conditions on the power bus—a possible occurrence during aircraft start.

INDUCTIVE MONITORING SYSTEM (IMS)

IMS was developed to provide a means to monitor the health of a complex system without the aid of a manually developed model. IMS is essentially composed of two parts: a *learning algorithm* and a *monitoring algorithm*. The learning algorithm characterizes typical system behavior by extracting general classes of nominal data relationships from archived data sets to build a knowledge base. The monitoring algorithm compares real-time operational data with the classes of nominal data contained in the knowledge base to detect anomalous behavior. This system, of course, is applicable for only those ranges of operation captured in the knowledge base. A more detailed description of IMS can be found in [Iverson, 2004].

A number of modifications were made to enhance the performance of IMS under this project. These modifications involved separating the knowledge base into operating state-based databases, adding statistical weighting to the control and status deviations, adding alert logic to help better identify system anomalies, and scaling the knowledge base. This section will briefly describe these modifications.

State Space Recognition
State space recognition was implemented to increase the speed and precision of IMS. It provided focused monitoring and decision thresholds for specific operating conditions. The Learn algorithm was modified to develop clusters for each aircraft operating state. In the case of the F/A-18, engines states were grouped according to throttle control settings, Mach number, and turbine fan speed as shown in Figure 3.

Alert Logic
IMS outputs for unusual events often consist of a series of peaks in the status distance over a short time period rather than a continuous high reading. This alert logic looks for time periods with several instances where the status distance is significantly higher than the controls distance. If this pattern occurs, we suspect there's something unusual in the data that is more significant than just a data spike and may indicate a developing system fault. This event can be recorded for later investigation or used to alert the pilot of the situation.

Scaling of Database
IMS needed to work within the capabilities of the on-board processor and keep up with the data in real time. IMS is capable of processing data for each engine at 20 Hz (effectively 40 Hz real-time monitoring). Most of the IMS processing is used to search the knowledge bases for the best match with the current data vector. Larger knowledge bases usually take longer to search. The IMS process was limited to 20%–30% of the CPU or less to be sure the other processes had sufficient CPU processing time and no process fell behind.

The method used to ensure IMS didn't spend too much time searching the knowledge bases was to reduce their size. This was done by increasing the maximum cluster size during the learning process so each cluster encompassed more of the training data, resulting in fewer clusters in the knowledge base. The trade-off is somewhat looser monitoring tolerance, but the results with the smaller knowledge bases didn't seem to be substantially different than with larger, more precise knowledge bases.

Control and Status Vectors
The data vector was partitioned into a control vector and a status vector. The Control partition included parameters that could be considered inputs to the engine, including environmental parameters (temperature, altitude, etc.) and pilot input (power lever angle, side slip). Some of these parameters were selected based on Iron Bird simulation information. The status partition included parameters that were indicative of engine state and engine performance. The control partition was used to relate current system state to system states covered in the nominal training data. If there was a low deviation on the control parameters, then IMS had good training data for that condition and consequently the IMS results could be regarded with a higher degree of confidence. On the other hand, if the control deviation was high, then any results from IMS could not be regarded with as high a degree of confidence. So a low control deviation combined with a high status deviation is a strong indication that the observed system performance is not nominal, while a high control deviation indicates that IMS lacks sufficient training data in that

respective region of operation and any associated high-status deviation in that same region of operation is not indicative of a system anomaly with a high degree of confidence.

Statistical Weighting

Statistical weighting on the control and status vectors provided a measure of how much deviation from nominal training data is to be expected in the control or status parameters. IMS was trained on most of the nominal data, with the remainder of the nominal data run through the monitoring routine for validating and statistical characterization. The deviation distances for the control and for the status parameters were collected, and a standard deviation calculated for each. These standard deviations were used to weight the IMS deviation distances during monitoring to determine how many standard deviations the monitored data was from the training data. A distance of 2 or 3 standard deviations indicated unusual behavior. If the status distances were several standard deviations above the controls distances, then something of interest was probably occurring.

Engine State Matrix and Logic Partitioning

Both IMS and BEAM used engine state information to focus their analysis on specific operating modes. Table 3 shows a derived state matrix that was independently applied by each algorithm. The operating modes of interest were originally determined by the BEAM engineers and implemented in SHINE. The IMS engineers rehosted this code, using engine state to determine which specialized knowledge base to query.

Table 3. Engine State Matrix

	Engine State	Throttle	RPM	RPM Transient	Mach Number
0	Off	Low	N/A	No	N/A
1	Startup	Med	Low	No	< 0.8
2	Non After-burn / Subsonic	Med	High	No	< 0.8
3	After Burning / Subsonic	High	High	No	< 0.8
4	Non After burn / Transonic	Med	High	No	> 0.8
5	After Burn / Transonic	High	High	No	> 0.8
6	Non After-burn / Subsonic / Transient	Med	High	Yes	< 0.8
7	After Burning / Subsonic / Transient	High	High	Yes	< 0.8
8	Non after burning / Transonic / Transient	Med	High	Yes	> 0.8
9	After Burning / Transonic / Transient	High	High	Yes	> 0.8

BEACON-BASED EXCEPTION ANALYSIS FOR MULTI-MISSIONS (BEAM)

BEAM is a software technology that analyzes system data to detect anomalies, classify faults, and track degradation in physical systems. BEAM consists of multiple components and requires the input of system experts and example datasets in order to develop models of nominal and/or faulty behavior. BEAM was originally designed to provide a generic system analysis capability for deep space probes and other highly automated systems. Such systems are typified by complex and unique architectures, high volumes of data collection, limited bandwidth, and a critical need for flexible and timely decision abilities. Since its original inception, BEAM has been matured and proven on many separate applications, both on-board and off-board. In this experiment, we began validation of BEAM in a flight-like environment, including modeling and testing issues.

The complete BEAM architecture is described in [James et. al, 2001], and contains several components that can be summarized approximately into three categories. The first category explicitly treats discrete information, comparing telemetry to a simplified state model to estimate system mode and detect obvious faults, such as mismatches between state indicators and out-of-range parameters. The second category, sometimes referred to as "detectors," examines quantitative sensor data for unexpected behavior, such as transients, unexpected features, or unknown interactions between sensors. Finally, the third category interprets and fuses results from the other components into a single actionable interpretation. Not all components are present in all BEAM deployments, providing some flexibility to meet different system requirements.

During this experiment, we begin with a minimal initial BEAM build, planning to gradually include additional components in follow-on efforts. The initial build described in these results contains three components, described below in additional detail. This configuration of BEAM is the minimum required to cover its intended functions.

Symbolic Data Model: The leading component of BEAM accepts raw data and applies a state model of the target system expressed in the form of rules. In this particular case, the state model examined throttle, aircraft speed, and engine RPM to establish the state of the engines. The Symbolic Data Model also, in this case, applied maximum rate-of-change filters to the incoming data in order to remove erroneous sensor spikes, discovered in early testing and thought to be due to aircraft power interruptions. Both of these functions are handled through an inference engine, namely SHINE (Spacecraft Health INference Engine), developed previously at JPL. In a complete build of BEAM, spike removal and other data correction is usually migrated to a different component that includes a partial physics model. We may have the opportunity to include this component in the future.

Coherence-based Anomaly Detector: Corrected sensor data and system mode pass into this detector, which examines the numerical data for anomalous features. The detector functions by computing cross-signal statistical moments for every signal pair in real time, constructing an evolving estimate of the transfer functions present in the physical system. This NxN matrix is compared to a library of nominal matrices computed from training data. The choice of nominal matrix is driven by the system mode, as provided by the Symbolic Data Model. Each nominal matrix is also associated with a companion

weighting matrix, reflecting the repeatability of individual signal pair relations. The comparison reveals departures from expected physical relationships in the system, indicating the presence, strength, and propagation of anomalies. This detector is described in additional detail in [Mackey, 2001].

Interpretation Layer: The final component is a combination of data summarization and data logging. In a fully developed system, the interpretation components combine results from multiple detectors and compare anomalous signal reports from the detectors to anomalies predicted by discrete indicators. The components then collect signal groups into distinct episodes and suspected sources versus secondary effects. For our experiment, we have only a single detector and no summarization is necessary—indeed, we want to access the full range of internal results to verify correct execution. In this experiment, we output the complete set of BEAM conclusions, updated on every signal sample as a rectangular matrix. This list includes sensed mode, global anomaly flag (yes/no), anomaly indicators for each individual signal, and normalized distance from ideal for each individual signal, as well as timing information to permit precise synchronization with the captured input data and reconstruction of results post-flight.

At the start of this experiment, the most mature software build of BEAM was configured for post-flight or ground controller analysis, with major focus on user interface rather than speed and real-time operation. Most software components existed in C# to enable rapid GUI development; however, the core algorithms were all designed to be easily portable, and we produced a C++ build, suitable for real-time operation and interfacing with the Dispatcher software, in a matter of days.

Associated with the run-time software are the associated tools required to configure BEAM. This validation experiment is as much a validation of these tools as it is the algorithms operating in flight, as all elements of the process were exercised. In this experiment, we constructed two separate models—one in the Symbolic Data Model, the other in the Coherence-based Anomaly Detector—required for operation. These models are also interdependent. The first model was coded and tested using support software developed for SHINE, in a process similar to entering individual program statements in other computer languages. The second model was generated in a nearly automatic process, wherein a set of numerical training data is passed through an algorithm similar to the Coherence-based Anomaly Detector, but with the raw library of matrices as its output rather than a differential. During the course of this experiment, both models were updated and loaded into the BEAM software without other software changes, and in all cases the software performed as expected after receiving the new models.

The rules that make up the Symbolic Data Model are somewhat ad hoc, as is the notion of a system state. For purposes of this discussion, we define a system state as a range of behavior that is distinguishable from others, but make no further requirements. Boundaries between states are open to some interpretation, but in general, choices of different modes should be as easily separable as possible on the basis of sensor data. We also desire as small a number of states as possible that provides the required level of performance. Too many states increases the complexity of state models and memory

requirements, but more importantly also increases the amount of training data required. Too few states may mean lumping together of dissimilar behaviors, leading to less determinism within a state and higher thresholds of detection.

BEAM Data Coverage

In this experiment, BEAM processed a slightly different set of data than IMS. The list of signals is given below in Table 4. As described in the following section, BEAM used only a small subset of aircraft state information to establish engine mode. Conversely, BEAM was applied to every available sensor parameter from the engines themselves.

The choice of sensors monitored is governed in part by performance limitations of computing hardware and timing requirements. As system complexity increases, the dominant factor in BEAM computation time scales as the square of the number of continuous parameters—for example, if applied to only one engine and its 13 signals as opposed to both engines and 26 signals, BEAM would execute roughly four times as fast. There is no significant limitation on discrete signals, since the discrete reasoning engines are inexpensive compared to the numerical algorithms. For relatively small numbers of signals, such as the 26 signals used here, algorithm time is comparable to data acquisition and reporting load. In the flight experiment, BEAM computational strain was measured to be comparable to that of the Dispatcher, affirming that I/O operations rather than algorithm complexity were driving resource usage.

Table 4. Data Parameters Monitored by BEAM

Left and Right Engine Data Variables	Aircraft State Data Variables
ENG NARROW BAND VIBRATION	POWER LEVER ANGLE, LEFT
ENGINE BB VIBRATION	POWER LEVER ANGLE, RIGHT
EXHAUST GAS TEMPERATURE	MACH NUMBER
COMPRESSOR DISCHARGE PRESSURE	
TURBINE DISCHARGE PRESS	
LOW PR ROTOR SPEED	
HI PR ROTOR SPEED	
ENGINE INLET TEMP	
ENGINE OIL PRESSURE	
ENG NOZZLE POS	
MAIN FUEL FLOW	
FUEL INLET TEMP	
TURBINE DISCHARGE TEMPERATURE	

Building the BEAM State Model

We began our experiments with a preliminary state model, following with a refinement after a few flights' worth of data was collected and analyzed. Following the guidelines above, we wanted a state model that was a simple and as easily separable as possible. Our insight into engine state was guided by perusal of the F/A-18 Flight Operations Manual as well as previous experience with aircraft propulsion systems.

There are numerous environmental parameters that contribute to engine performance. The most important of these is the throttle, or Power Lever Angle, as it defines thrust request from the pilot. Additional parameters include ambient pressure, temperature, and humidity; oiling and fueling system state; altitude and airspeed. A thorough physics modeling of the engine is a complex undertaking.

At the opposite limit of perspective, an aircraft engine exhibits only a few distinct classes of operation, roughly summarized as "Off," "Startup," and "Running." The first two classes require no further elaboration, but the third incorporates a relatively wide range of behavior. We originally divided the "Running" category according to two other factors: whether the engine was afterburning or not, as afterburning uses an additional set of fuel injectors and is a fairly large change; and whether the aircraft was subsonic or in the transonic flight regime (Mach number = 0.8 was chosen as the boundary), as the fluid behavior at the engine inlet changes substantially.

We were then left with six possible modes for each engine. In the interest of simplicity, we simply determined each mode independently, for 36 possible system modes, understanding that some modes (e.g., left engine subsonic, right engine transonic) would be unreachable. Since the mode could be determined instantaneously from raw telemetry, we did not need to consider transitions between modes.

In practice, the four "Running" modes were still found to contain too much variety in signal behavior. In other words, the associated statistics model built for the anomaly detector (see below) included large margins of error as it attempted to encapsulate different behavior into a single mode. Study of the input data revealed that the engine behavior should be divided once again, into "transient" and "steady-state" behavior. This need was anticipated due to our previous work on aircraft propulsion. As a result, our second state model included this additional classification criterion, requiring several additional rules acting on the Power Lever Angle and Low-Pressure Fan RPM signals. In brief, a significant change in thrust request (equal to 1 degree or more) signals the start of transient operation, which ends once the RPM signal stabilizes, signified in this case by five seconds of continuous readings within a 500-RPM dead band. The low-pressure rotor signal was selected because of its lower noise characteristics; however, this signal is not available from the aircraft during startup, for reasons unknown to us.

After incorporating this new discriminant, we were left with 10 modes for each engine, or a possible 100 states for the propulsion system. As before, numerous states are unreachable, and in nearly all nominal flight the two engines are nearly in sync, or else started up according to a reliable sequence, resulting in 10 "common" modes and a few

dozen “rare” modes. Fortunately, virtually every flight contains the full range of behavior, meaning training data is readily available from actual flight in addition to simulation capabilities. Also, the number of modes impacts on-board memory requirements (in this case, approximately 5.4 kilobytes per mode), but does not adversely affect computation speed.

After updating the state model, all previous flight data sets were reevaluated on the ground to test the model’s impact on BEAM performance. The remainder of flights used this change, as no further modification was deemed necessary. While this mode mapping is not considered a unique separation of engine state, it proved to be sufficient. In general, we advise beginning with system models and architecture to construct preliminary state models, making only the bare minimum adjustments necessary to meet performance needs.

Building the BEAM Statistics Model

BEAM’s anomaly detectors contain statistical models of the system, best described as lookup tables of expected statistical moments indexed by mode. Because these models depend on mode as defined by the Statistical Data Model, and on upstream signal treatment, including filtering through physics models when available, the statistical models must be created last. The statistics models must also be recalculated after a change in either of the upstream models. Fortunately, we have automated this process to a considerable degree.

The BEAM statistics models are generated from training data. It is important to capture training data that resembles actual flight data as closely as possible (an analogue to the concept of “fly like you test, test like you fly”). To build a model, training data contained in one or more data files is submitted to a modified set of anomaly detectors, along with the mode signal as it would appear in flight. These modified detectors, typically run offline in the laboratory environment, perform most of the computations of the flight anomaly detectors, differing only in output. Instead of comparing computed statistics to a pre-existing model, they store the raw computed statistics for the training files, and then re-run the calculation to test the training files against themselves. The output model contains “nominal” (i.e., trained) statistical parameters, along with confidence measurements, as well as configuration parameters associated with those models. The configuration parameters include user-selectable sensitivity measurements and “tests for zero” that can be adjusted to improve performance, or to bias the detector towards higher sensitivity at the expense of false alarms, or vice-versa, if so desired.

If there is a need for frequent modifications to the statistics models, the process can be kept automatic so long as one retains a standard pool of training data. An adjustment to the Symbolic Data Model, for instance, that changes the anticipated mode signal—possibly adding or subtracting modes—requires a re-examination of the training data based on the new mode boundaries; however, since only the mode signal has changed from the statistics modeling perspective, the process of statistical model building is exactly the same.

In this experiment, we developed three separate statistics models for the Coherence-based Anomaly Detector. The first model, prior to first flight, used training data from the F/A-18 Simulator and the preliminary mode rules described above. We anticipated important differences between the simulator and the actual aircraft, which proved to be accurate. The first-flight performance was marked with several persistent anomaly detections, as the detector flagged signal dynamics that are simply not part of the simulation. In essence, BEAM was reporting that it was definitely not operating on simulated data!

Once we captured the first flight data set, we immediately calculated a new statistical model using the entire first flight as a training set, again according to the preliminary state model. This statistical model eliminated the "false alarms," but was deemed to be too insensitive. As recounted above, the state model combined too many variations in engine behavior into a single mode until we distinguished between transient and steady power requests.

The reason for this distinction is linked to the detector algorithms. Our detector functions by computing signal dependencies as a way to estimate physical relationships between parameters. While in steady-state operation, most sensor signals lie within a "dead band" where any sensed variation is dominated by sensor noise. The engine tolerates what little real variation there is, and so cross-signal behavior is extremely low. In contrast, when the engine is being spooled up or down, sensor signals are dominated by large structural changes and physical relationships, and cross-signal behavior is strong. As a result, we achieve much greater accuracy by considering these two types of operation independently.

The dichotomy between transient and steady operation is common to many mechanical systems. We addressed this through mode separation in this case. In other applications, there are several additional factors that help treat this problem correctly:

- *Sensor Performance:* In the limit of "noiseless" sensors, the concept of a steady-state case diminishes or disappears. This is rarely useful for mechanical systems but often applies to virtual systems, e.g., simulations and flight control or navigation computers.

- *Physics Modeling:* If the large-scale behavior of the system can be predicted, even in an approximate fashion, the transient case can be renormalized to bring it closer to the steady-state case. A full build of BEAM includes incorporation of physics models to perform this step. The F/A-18 engine simulation can be adapted to this purpose.

- *Multiple Detectors:* Depending on their underlying algorithms, different anomaly detectors may perform better in one or the other mode. The Coherence-based Anomaly Detector relies upon statistical moments computed over long time periods, and therefore is most sensitive in steady-state cases (unless the engine can be held in transient for a long period of time, which is rare but not impossible). The other detector frequently used in BEAM has opposite properties, susceptible to long-term drift but highly sensitive to transient behavior; however, it should be noted that detectors cannot be combined haphazardly through voting

schemes or other contrivances, as the overall sensitivity or false-alarm rate will suffer. BEAM integrates multiple detectors through a reasoning path involving the Symbolic Data Model.

The third statistical model reused the same training data, viz. the entire first flight, but relied upon the improved Symbolic Data Model and therefore contained many new modes. Performance with the third model was markedly improved, visible in the results as well as the statistical model itself. A researcher can examine the model directly to check for inaccuracy, much as an experimenter may inspect autoregressive results for a "bad fit" to data.

Software Configuration

A base software load consisting of the Linux 2.4.25 kernel and the Debian distribution was installed on all units. This kernel is not a real-time Linux variant but is rather a vanilla Linux distribution. In the early stages of the project, it was determined that a true Real-Time Operating System would not be required for this effort. In fact, budgetary and schedule constraints played a significant part of this decision, given the time and expense required to obtain and configure the real-time OS and make the necessary changes to port over the IMS and BEAM applications. The Debian software load was generated and delivered by JPL; the integration of the real-time OS has been left for future effort.

Throughout the project, the following workflow was used for project-specific software:

- The applications would be modified / tested by the developers.
- When working to the satisfaction of the developers, the code would be placed under configuration management (CVS) and tagged for release.
- The release would be installed on the JPL test bed and taken to Dryden for testing with the F/A-18 simulator.
- When working satisfactorily, the release would be installed on the flight box under QA supervision.
- Flights would be performed and data collected.
- The data would be distributed to the PIs for analysis.

This was done on a weekly cycle, with changes resulting from data analysis being integrated and flown the following week.

The configuration management repository is resident on a system at Ames that is also available to JPL personnel. In addition to code storage, the repository is also used for archiving flight data, both for safety from accidental deletion and so that it is available to all those who need it. The PBMA (Process Based Mission Assurance) Secure Working Groups system is used for documents and information exchange.

SIMULATION TESTING

Simulation was an important component to the development of this project. Simulators used included an F/A-18 simulator at NASA Dryden and a 1553 Bus analyzer at NASA Ames.

F/A-18 Flight Simulator

The F/A-18 flight simulator at NASA Dryden uses six-degree-of-freedom equations of motion with an oblate spheroid earth model and non-linear aerodynamics. The simulation dynamics are computed at 160 Hz. The control laws alternate between longitudinal and lateral-directional, resulting in an 80-Hz frame rate. The simulator is capable of multiple modes of operation (batch, real-time, Hardware-in-the-Loop (HIL), and Ironbird modes).

Figure 7. NASA Dryden F/A-18 flight simulator laboratory and cockpit

For this project the simulation was run in real time (HIL) on a multiprocessor computer in the F/A-18 simulation laboratory (Figure 7). This simulation is connected to a fixed-based cockpit with a control stick, rudder pedals, throttles, multiple analog gauges, switches, and lights. In addition, the cockpit contains an aircraft Multipurpose Display System, including two Digital Display Indicators (DDIs) and the Up-Front Control (UFC). Aircraft Mission Computers are also integrated into the cockpit and serve as the MIL-STD-1553 bus controller. The actuators, a single channel of the flight control system, and other systems are modeled in software.

While in HIL mode, the F/A-18 simulation executes in real time and integrates data from aircraft Flight Control computers located in a test bench in the Test Bay next to the simulation laboratory. The actuators are modeled by hardware in the test bench which can enable testing of failure modes and redundancy management.

The ISHM flight software was tested by connecting the flight PC/104 box directly to the simulator's 1553 bus. The simulation was enhanced using test scripts and the autotest function to modify bus parameter values. Modifications to the simulator software for this project included adding engine exhaust temperature (T5) through an autotest script. Noise was also added to several parameters to more closely simulate the flight environment.

The F/A-18 simulator was used to support the project in several stages of development. For initial testing of the flight processor 1553 interface, the simulator was used to provide realistic data to the computer. Once the interface and the Dispatcher software were capable of reading the 1553 data, the simulator was used to test IMS and BEAM software, including the generation of several test scenarios (for example, shutting down

one engine or pulling back power on both). The simulator was also used for acceptance testing of the software builds prior to installation on the data processor.

Ames 1553 Bus Analyzer

The team also made use of a 1553 bus analyzer at NASA Ames in the initial development and debug of the Dispatcher Software. The bus analyzer was easily configured to provide the same data interface as would have been expected at Dryden. The analyzer has the capability to generate bus traffic as well as the capability to replay bus traffic. The latter capability was used to replay data that was collected on the simulator as needed to resolve issues at Ames. The Rotorcraft Branch at Ames made the analyzer available to the team. The use of the analyzer saved at least a week of schedule time and at least one trip to Dryden.

F/A-18 FLIGHT TEST AIRCRAFT

The test aircraft used for this project was an F/A-18 tail number 852 (Figure 8). This aircraft serves multiple uses, which include chase, proficiency, and experimental flights. The aircraft provided us with multiple chances to collect data.

Figure 8. Dryden F/A-18 Tail Number 852 used in study

The 1553 data bus on the aircraft allowed easy integration with the flight processor and software. One challenge to the project was that the aircraft does not have the controls to create or simulate minor engine faults (other than engine shut downs) while in flight. Because DFRC does an outstanding job maintaining its aircraft, we also did not experience any faults that would have been written up in maintenance. We used the flight simulator to explore faults of this nature.

PROCESSOR ENVIRONMENTAL TESTING

The PC/104 flight research computer was subjected to environmental acceptance test procedures based on requirements specified in DCP-O-018, the Dryden Centerwide Procedure (DCP) entitled “Environmental Acceptance Testing of Electronic and Electromechanical Equipment.” This document provides the basic environmental test specifications for acceptance of electronic, electrical, optical, and electromechanical equipment for use in DFRC aircraft and flight vehicles. DCP-O-018 specifies that equipment to be installed on an F/A-18 support aircraft falls into environmental

acceptance test category II, which is applicable to all turbojet-powered vehicles. According to DCP-O-018, the basic category II test requirements are as follows:

- **Altitude:** Subject the unit under test to an altitude environment from ground level to 75Kft for pressurized compartments, or ground level to 100Kft for non-pressurized compartments, unless evidence for waiver of these requirements is shown. In the case of F/A-18/852, with the pilot not wearing a pressure suit, an altitude limit of 50Kft is enforced. For this reason, an environmental waiver was submitted and approved allowing the altitude environmental acceptance test requirement to be relaxed to ground level to 75Kft for a non-pressurized equipment bay.

- **Temperature:** Subject the unit under test to a cold-soak temperature of 0 Deg F and a hot-soak temperature of +160 Deg F, if installed in a temperature-controlled compartment. If the unit under test is to be installed in a non-temperature-controlled equipment bay, the Flight Systems and Operations Engineering groups shall determine the appropriate temperature test after considering other installation factors such as: proximity to cryogenic fluids, effects of aerodynamic heating, protective devices (heaters, insulators, etc.), proximity to heat-generating devices within the vehicle, temperature extremes encountered in prolonged ramp or hangar operations, and the extreme cold encountered at high altitude. DCP-O-018 states that in no case shall the cold soak temperature test be less severe than -65 Deg F and the hot soak test be less severe than +160 Deg F. In the case of F/A-18/852, other DFRC support F/A-18s with temperature instruments installed in equipment bays revealed that the operational temperature range of the PC/104 equipment bay was measured to be more in line with zero to +160 Deg F. For this reason, an environmental waiver was submitted and approved allowing the temperature environmental acceptance test requirement to be relaxed to 0 to +160 Deg F.

- **Vibration:** Subject the unit under test to vibration levels determined by the specifics of the installation. The vibration shall be applied in separate tests to each of three mutually perpendicular axes of the test unit. The unit shall be monitored throughout the vibration tests and performance shall be recorded. If the test unit is to be mounted on vibration isolators, it shall be so mounted for the vibration test when possible. Vibration shall be random, and shall be for a minimum of twenty minutes in each of its mutually perpendicular axes. In the case of F/A-18/852, the random vibration was per DCP-O-018, Curve B, which is a random vibration level of 12.2 G–RMS, over a frequency range up to 2000 Hertz.

PROCESSOR-AIRCRAFT INTEGRATION

For F/A-18 PC/104 installation, the box was installed into equipment bay 14L (left side of the fuselage). This is a non-pressurized and non-temperature-controlled equipment bay, forward of the engine inlets. The configuration changes to the F/A-18 aircraft required the approval of the NASA DFRC support aircraft Configuration Control Board (CCB).

The integration of the PC/104 flight research computer into the 852 support aircraft involved the mechanical installation of the shock-mounted base plate and PC/104 box into an available fuselage equipment bay, and supplying both electrical power and signal interfaces from the aircraft 1553 data bus. Aircraft power from the 28-volt DC non-essential power bus was provided via circuit breaker protection, and was integrated in such a way that PC/104 power was relay coupled to the existing RQIDS (Research Quick Instrumentation Data System) equipment. The RQIDS equipment was an existing flight data system also installed in equipment bay 14L, with power controlled via a switch at the pilot station. This relay coupling to the RQIDS system power gave the pilot control of the PC/104 in the event that the PC/104 was suspected of causing problems on the data bus.

The 1553 data bus interface was accomplished by via transformer coupling. The use of transformer coupling protects the aircraft bus from interference from additional bus loading and from any potential electrical malfunction such as shorted wires downstream of the transformer coupling. The existing RQIDS data system's single-stub 1553 data bus couplers were replaced with dual-stub 1553 data bus couplers, so that the identical 1553 data bus signal was paralleled out to both the RQIDS data system and the PC/104 flight research computer. The PC/104 flight research computer was configured strictly as a passive bus monitor, which would not transmit any data out on to the aircraft 1553 data bus.

All hardware and software modifications to the aircraft and to the flight box had to be performed without causing significant interference to the basic role of the aircraft as a research mission support aircraft. That meant that the aircraft could not be taken out of service for PC/104-related configuration changes unless other scheduled maintenance was required. The flight box placed very little burden on the flight operations. The pilot powered up the data processor before flight (so that data on engine startup could be collected) and shut down the data processor at the end of the flight, using a switch located in the cockpit. Because the system handled all boot-up and configuration setup internally, the ground crew was able to handle all other operations.

Other challenges pertained to post-integration PC/104 flight software configuration changes. The software changes originated at both Ames and JPL, but the software had to be loaded and tested at the NASA DFRC F/A-18 simulation lab prior to being integrated into the F/A-18 for subsequent flight operations. All software configuration changes had to be approved by the NASA DFRC support aircraft CCB, and then installed via NASA DFRC work orders. It was challenging to work through all of the associated paperwork, approvals, and associated testing to accomplish the nearly weekly software configuration changes dictated by the program.

PRE-FLIGHT PROCEDURES AND FLIGHT TESTING

HARDWARE AND SOFTWARE GROUND TESTS

Ground tests were performed to test the operation of the hardware and software prior to the initial flight. The first ground test involved exercising the flight controls while the flight box was powered using APU power (no engines running). This test enabled a limited checkout of the Dispatcher software to ensure that the Dispatcher was able to receive at least a portion of the 1553 data. This test also improved confidence that the software configured on the simulator would work with the flight vehicle. The second ground test involved powering up the engines for static test. The results of this test showed that all of the 1553 aircraft data expected was received correctly by the Dispatcher.

Pre-flight procedures were also developed to verify that the PC/104 flight research computer and software were configured correctly and ready for each flight. This was accomplished by interfacing the flight processor to the F/A-18 simulator using the current software load. The preflight tests procedures verified that the appropriate software modules were installed by checking for appropriate checksum values for each of the software modules. The pre-flight process also eliminated data files generated by previous computer activity, whether it was ground or flight test related, in order to allow maximum data storage capacity for the next test activity.

FLIGHT APPROVAL AND INITIAL TEST FLIGHTS

Our flight experiment had the advantage of being able to piggy-back on other existing programs at DFRC and as a consequence was spared the cost associated with directly requested flights. During the project, aircraft 852's flights included: proficiency flights (flights that the pilots use to brush up on skills or stay familiar with flights), test flight duty (including sonic boom creation flights), and as a chase/photo plane (among our monitoring flights, the aircraft was used to provide chase for the departure of Discovery on the 747). In addition to reducing our cost per flight, having our hardware installed on an aircraft out on the flight line provided us a variety of flight regimes.

Three flights were initially approved based on the successful Dryden Flight Readiness Review (FRR). These flights were completed within the first two weeks of August 2005. Details on the ground tests and early flight tests meeting the success criteria are shown in Table 4. Additional flights were approved at two follow-up FRRs. In total, the ISHM hardware and software was flown on 25 flights, with 23 of them successful in collecting data throughout the entire flight. These flights have produced a significant amount of data that can be used to further enable continued software development on the ground and should fuel further interest in its capabilities in the air. The wide variation of flight data that was recorded turned out to contain many more cases of interest than could have been achieved (or imagined) in just 2–3 test flights.

Table 4. Details of flights meeting initial project success criteria

ISHM Engine run test, Test Day 8/1/05 (Monday) – **PC/104 software version 1.0.** Normal engine start without ground power, so RH engine was started first, then PC/104 power was turned ON with 2 minutes allowed for computer boot-up, then LH engine was started. Includes various individual and dual throttle (PLA) steady state points, then individual engine shutdown.
ISHM Engine run test 2, Test Day 8/1/05 (Monday) – **PC/104 software version 1.0.** Normal engine start with ground power, expect engine start data for both engines. Includes various two throttle (PLA) steady state points. Both engines shutdown simultaneously.
ISHM FLT 001, Flight day 8/2/05 (Tuesday) – **PC/104 software version 1.0.** Nominal proficiency flight. Non ground power engine start, so RH engine was started first, then PC/104 power was turned ON with 2 minutes allowed for computer boot-up, then LH engine was started. Includes supersonic flight conditions.
ISHM FLT 002, Flight day 8/3/05 (Wednesday) – **PC/104 software version 1.0.** Proficiency flight. Ground power start was planned, so should include engine start data for both engines. Individual engine shut-downs planned. Each engine to be shut down individually. Expect data for both LH and RH engine shut downs and restarts
ISHM FLT 003, Flight day 8/4/05 (Thursday)) – **PC/104 software version 1.0.** F/A-18/852 chase flight for another F/A-18 research aircraft (845). No special test points planned for ISHM. Ground power start was planned, so includes engine start data for both engines.
ISHM FLT 004, Flight day 8/10/05 (Wednesday**)** – **PC/104 software version 1.1.** Sonic boom flight, no special ISHM test points will be performed. The flight plans include a steady accel-decel from mach 0.8 to 1.4 to 0.8 at 38Kft pressure altitude. This will be followed by 49K pressure altitude with 180 degree roll to inverted entry into supersonic dives with 180 degree roll back to upright attitude with pullouts at approx. 38Kft pressure altitude. This maneuver will be repeated as many times as fuel permits.

The project benefited greatly from being on the flight line, given the wide range of operational conditions observed. Since the project did not fund the F/A-18 program directly, all flights were flown under the understanding that they could not be dedicated to ISHM research. Some of the maneuvers that were requested (such as an engine shut down) to support ISHM development were not part of normal operations, therefore these maneuvers had to be approved prior to the flight; however, after discussing our research with the pilots, some of these non-standard maneuvers "found their way" into the proficiency flights.

DATA DOWNLOAD AND ANALYSIS

Our post-flight procedures consisted of basic status checks of the processor and downloading the flight data files, which had been written to the PC/104 Flash drive module. The download was accomplished using Windows laptop computer, an SSH network connectivity protocol application (PuTTY.exe), and a secure FTP program called Sftp.exe. The secure FTP application was then used to copy all flight-generated data files from the IMS, BEAM, and Dispatcher modules to the laptop hard drive. After downloading was complete, the associated data files were compressed and uploaded to a secure data server that was approved for ITAR-controlled data. Both the JPL and NASA Ames program participants then redeployed the data to a secure server.

RESULTS AND CONCLUSIONS

SUCCESS CRITERIA EXCEEDED

The minimum success criteria set for the project required that IMS and BEAM be tested on the ground simulator and on three flights by the end of August 2005. By mid July, the software had been tested in the simulator and the first flight was conducted in the first week of August. By the end of August, IMS and BEAM tests were conducted successfully on ten flights. Table 5 lists the milestones and their dates of achievement.

Table 5. ISHM Technology Accelerator Milestones

Project Milestone	Date Achieved
Dispatcher tested on 1553 simulator at Ames	5/20
Dispatcher demonstrated on Iron Bird Simulator at Dryden	5/26
IMS and BEAM prototypes using archived data	6/07
IMS and BEAM prototype on Processor	6/22
IMS and BEAM prototype on F/A-18 Flight Simulator Min Success	7/22
Dispatcher run on Flight processor during environmental testing	7/15
IMS and BEAM Ground Test	8/1
IMS and BEAM Flight Test	8/2
Data Flight #5 Success Criterion Met.	8/12
Data Flight #10	8/30
Data Flight #25	11/01

PROBABLE ANOMALIES DETECTED

Fifteen flights were conducted at DFRC between August and September of 2005. Most of the off-nominal behavior reported by the monitoring programs during these flights was due to incomplete training data sets and unrefined aircraft mode characterizations; however, the algorithms detected four classes of anomalies: Transient Effects, Repeating Intermittent, Unanticipated Operating Mode, and Seeded Faults.

Transient Effects

During the August 10, 2005 flight, a low oil pressure reading was detected with respect to the right engine. The graph shown in Figure 9 begins about 62 minutes 38 seconds after right engine start. The chart shows a deviation of oil pressure from what was expected by IMS. The yellow line in the chart shows recorded real-time oil pressure data. The band of expected oil pressure values for the operating conditions was determined by IMS based on analysis of previous flight data. The blue and pink curves show the degree of deviation from expected values relative to other monitored parameters as calculated by IMS. Essentially, the blue line indicates how well the control and environment inputs collected in real time match the conditions contained in the training data. The pink line represents the IMS monitored system status parameters and is an indication of how well the real time system data matches the system behavior represented in the training data.

Higher values indicate a larger deviation. Theoretically, off-nominal system performance is indicated during time periods where the magnitude of system behavior deviation significantly exceeds control input deviation.

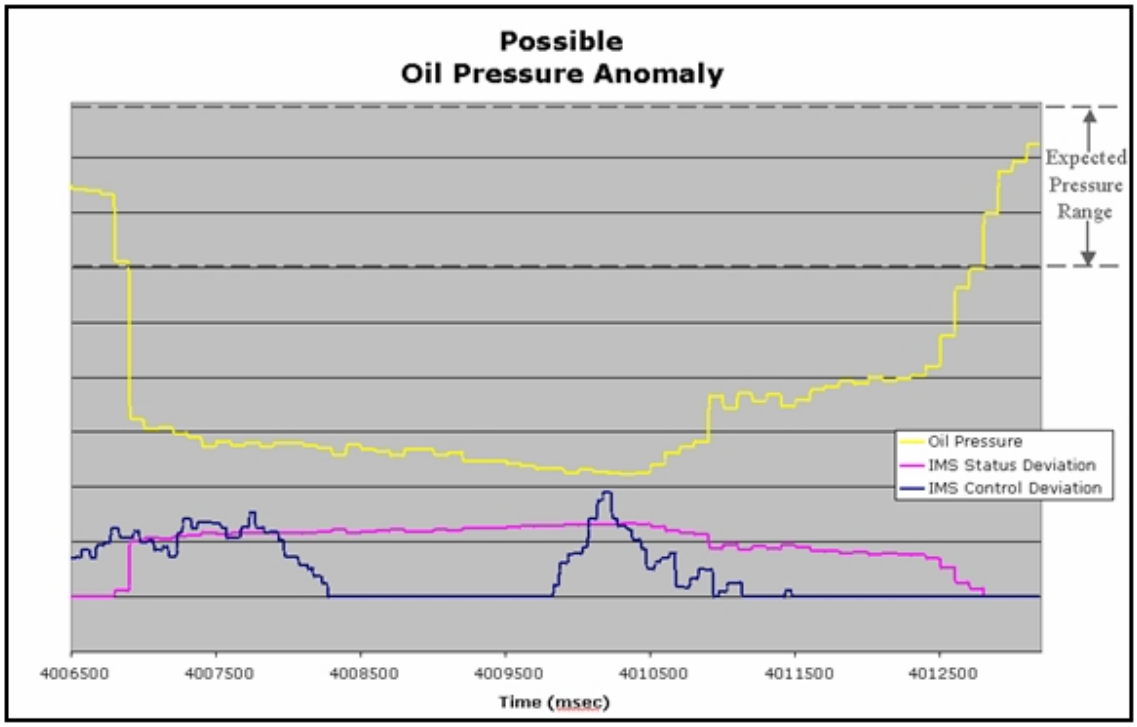

Figure 9. IMS plot of oil pressure anomaly

The low-pressure reading lasted approximately five seconds. Note that this particular low-pressure reading probably would have been detected by the pilot if it had exceeded the red line. In this case, the pressure drop was seen by IMS in context with 18 other parameters, not on absolute pressure values. IMS effectively established a red line for the current system state.

After further review of the flight data, it was found that the aircraft dove rapidly from approximately 49 Kft to about 3.2 Kft; and then executed a loop starting at about 3.2 Kft, peaking at about 10Kft, and ending at about 3.2 Kft. The aircraft then performed a near level (starts at 3.2 Kft, peaks at about 3.4 Kft, and ends at 3.1 Kft) 360-degree roll. The aircraft also was inverted during the low-pressure observation.

In view of the maneuvers performed, the low oil pressure was most likely a normal consequence. Aircraft ground personnel provided charts that showed engine oil pressure versus engine core RPM (N2) and oil type. These charts indicate that oil pressures below the minimums are allowed during to certain types of maneuvers, including climbs, dives, turns, rolls, or negative G's.

Given that the low pressure observed during the maneuvers described above is normal, the fact that IMS identified this condition as a potential fault is not a failure with respect to IMS. In fact, it's encouraging because IMS demonstrated it is capable of identifying

conditions not previously observed. In the future, training data should be incorporated into the knowledge base that represents these types of maneuvers or IMS must be configured to differentiate these extreme operating states from more typical aircraft maneuvers. Both of these features were incorporated into subsequent IMS builds.

Repeated Intermittent

As described earlier in the document, the BEAM output includes the following results:

- *Global anomaly flag:* This is an unambiguous on/off measurement that indicates whether current data is significantly outside the prescribed nominal envelope. This flag does not latch, but is recalculated at each sample. The flag is fault-positive; i.e., if insufficient data exists to confirm or deny an anomaly, the flag defaults to "off."

- *Channel-specific flags:* Similar to the global flag, the channel-specific flags indicate the presence of an anomaly on each individual sensor channel. Since we considered 26 sensor signals, we reported 26 flags at every sample. A global anomaly result implies a channel-specific anomaly on one or more signals.

- *Channel Distance:* In addition to the flag, each signal also reported its distance from the nominal ideal. Small distances do not trigger anomaly flags, while large distances do. "Small" and "Large" refer to statistical significance, which is a function of the amount of data considered and our confidence in the original nominal estimate.

All of the anomalies seen in actual flight to date were of a relatively mild nature, which is consistent with the observation that at no time were any existing F/A-18 alarms activated, nor did the test pilots themselves indicate any sign of faulty behavior. The global and channel-specific anomaly flags output by BEAM occasionally alarm, but usually at the peaks of anomalous behavior, whereas consideration of channel distance typically reveals a much longer episode lurking in the gray areas between nominal and faulty. For this reason, it is most useful to consider the Channel Distance results in our discussion.

Channel Distance, as described above, is output as a 26xN rectangular matrix of values normalized from 0 to 1. The dimension of 26 reflects that we are considering 26 sensor quantities. The other matrix dimension represents Time, in units of samples, with each column representing results at a particular instance of time, in this case sampled at 10 Hz. The distance magnitude represents the absolute value of the *unweighted* discrepancy between signal moments and the nominal case, which for the Coherence-based Anomaly Detector is averaged over all signal pairs involving a specific signal. To put meaning to the numbers, a channel distance of zero means that the cross-correlation of that signal with *all other signals* is precisely equal to the expected result, whereas a distance of one means the cross-correlation with all other signals is maximally different from nominal. In practice, values of zero are rare as there is usually some variation somewhere in the system, whereas values of one are never practically attainable. As a rule of thumb, distance values above 0.3 are significantly unusual, and values above 0.5 indicate a strong anomaly.

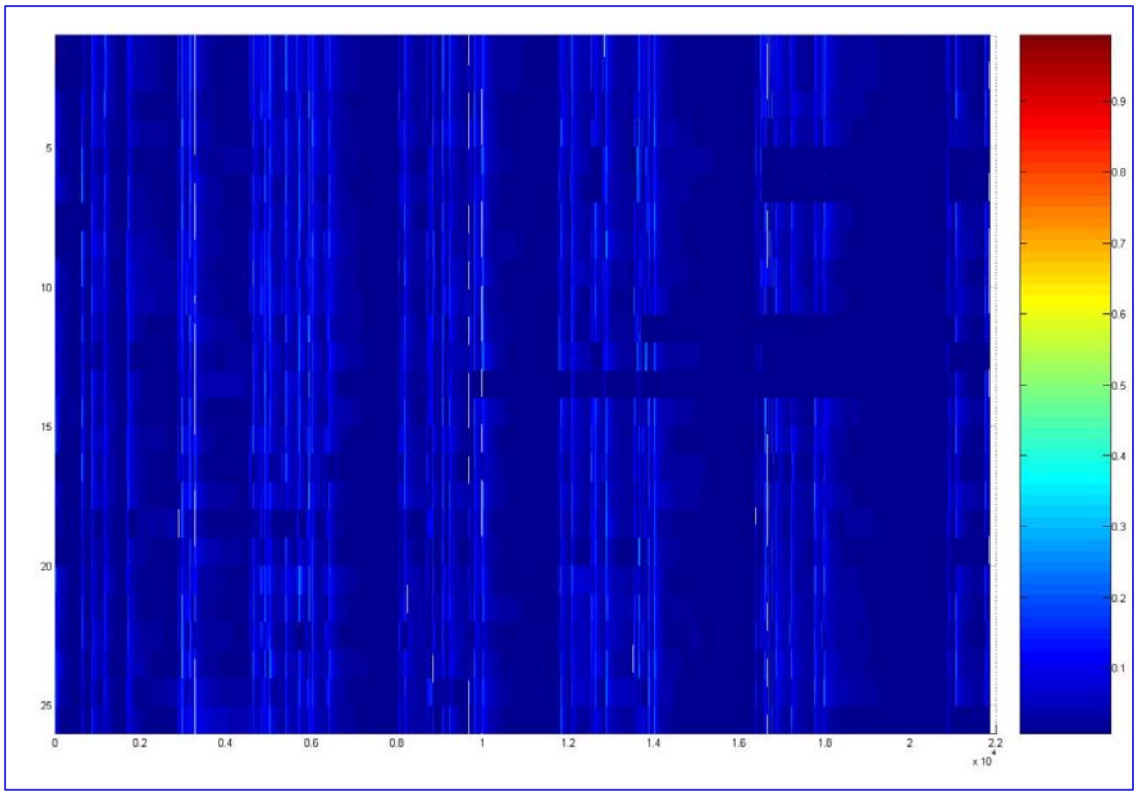

Figure 10. BEAM channel distance output, Flight 1 results

Figure 10 demonstrates an example of Channel Distance for a completely nominal run. The x-axis represents time in units of individual samples, taken from engine startup at left to shutdown at right, and spans approximately 36 minutes of operation. The y-axis indicates different signals, arranged in order of appearance in the data dictionary. The input data is the first flight itself, which was also used for training purposes; therefore, this output represents the minimum case.

Notice in Figure 10 how the deviations fluctuate from time to time, but always stay within bounds, and always return to zero, signifying convergence with the statistical model. These fluctuations are normal and represent sensor noise, environmental variation, interaction with other aircraft systems, and aircraft usage that moves within but does not cross mode boundaries.

While we were not able to inject a proper series of fault cases in flight for cost and schedule reasons (not to mention the potential to imperil the aircraft), we observed a number of unusual effects in the data. One of these effects is seen in Figure 11, from Flight 17.

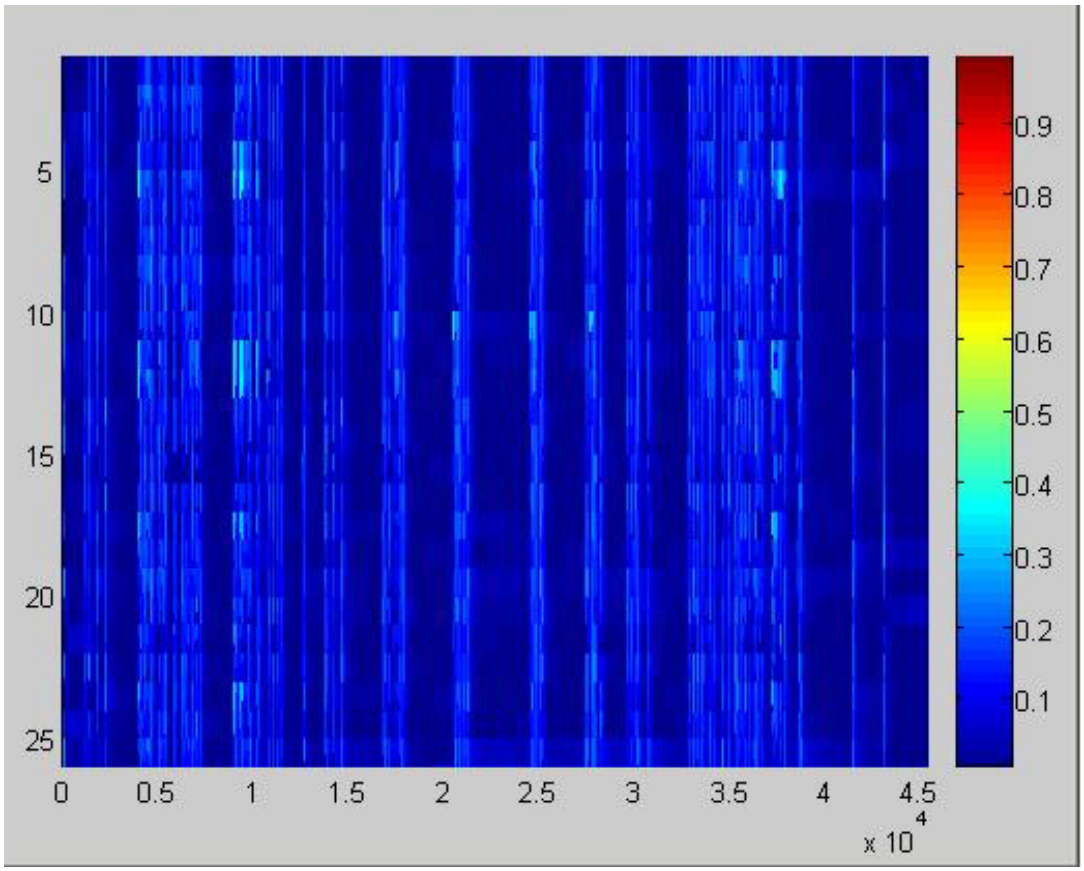

Figure 11. BEAM channel distance output, Flight 17 results

There are two distinct features present in this result. The first is the two brief episodes in channels 4, 5, 11, and 12, one at about 9000 samples, the other at approximately 37000 samples. This effect will be discussed below. The other noticeable effect is the series of four high readings on channel 10 in the middle of the figure.

Channel 10 is the Right Engine Exhaust Gas Temperature (EGT). All four of these blips occur in periods of engine transient after operation at high power. These blips can be traced back to features in the raw data for this signal, where high and unusually sudden jumps in EGT can be found. To date, this effect has been seen in one and only one other flight, namely Flight 19 as shown in Figure 12.

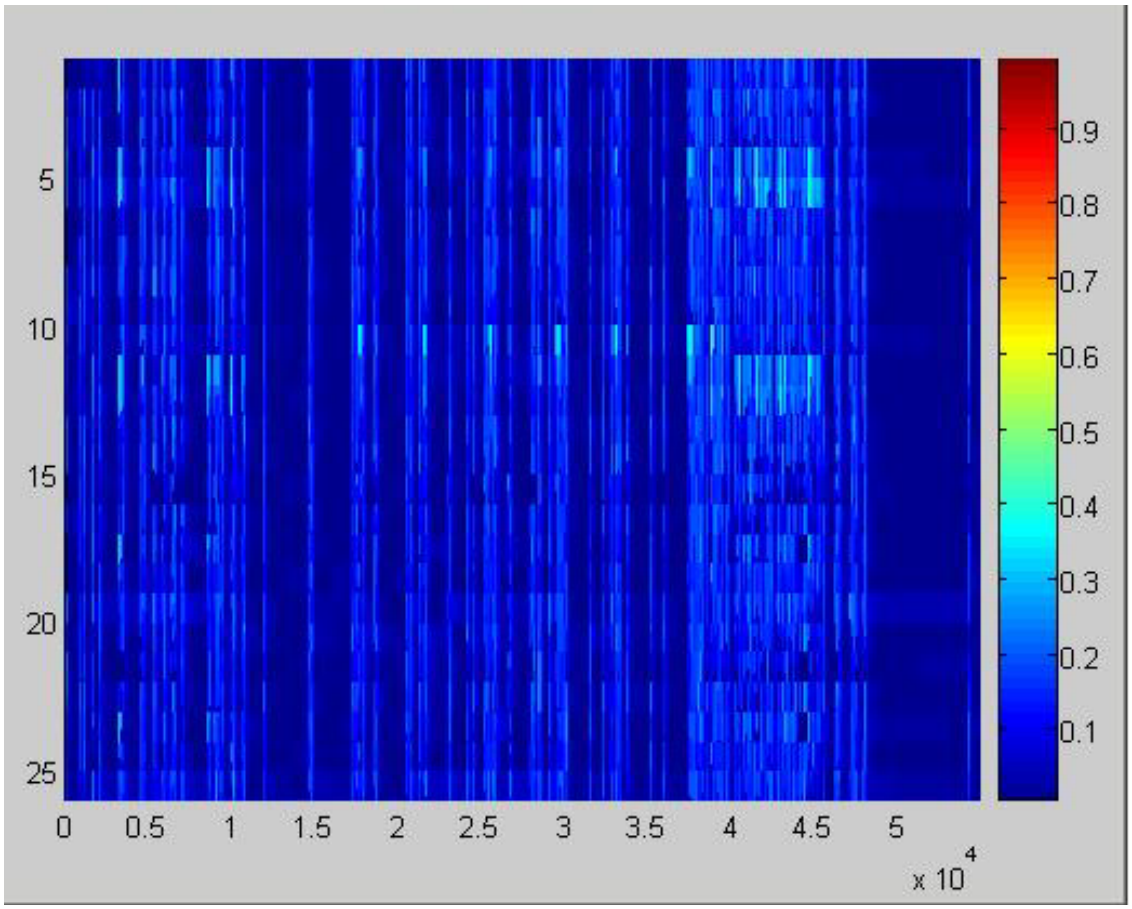

Figure 12. BEAM channel distance output, Flight 19 results

As before, the feature only affects the right engine EGT. It does not seem to affect the left EGT, nor is it correlated to any other signal except possibly channel 4, which is the left engine high-pressure rotor RPM (see below). It is also relevant to note that the effect has only been seen on two *non-consecutive* flights.

Regarding the other period of relatively high discrepancy, affecting signals 4, 5, 11, and 12, this anomaly reflects a feature discovered in approximately half of the F/A-18 flights. The signals are the high-pressure rotor RPM and low-pressure RPM for the left engine (signals 4 and 5) and the right engine (signals 11 and 12) respectively. This effect typically persists for long periods of a flight, and is most clearly seen in engine transients. A good example of this behavior is seen in Figure 13.

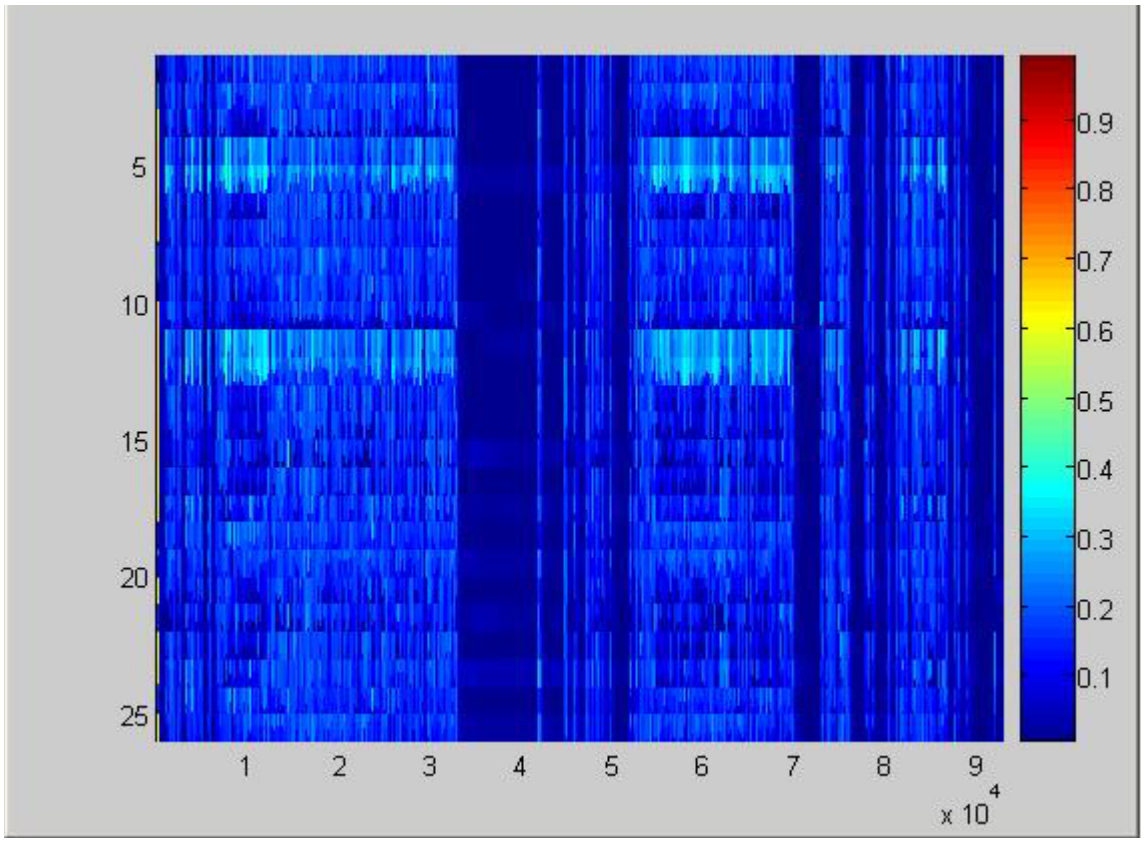

Figure 13. BEAM channel distance output, Flight 6 results

We should remark that our statistical model was trained with a relatively short duration of transient operation, resulting in lower accuracy compared to steady state; however, this effect is seen in some but not all flights. It also does not always persist from start to finish—as it did here—but sometimes emerges towards the end of a flight, as it does in a very limited sense on flights 17 and 19 above. It's also worth noting that IMS also detected these conditions, corroborating the BEAM detection.

Even though the effect is repeatable and strongly localized to these four signals, it is difficult to see what is causing the effect, even through hand examination of the raw data. We have, however, established that the effect is due at least in part to an increase in delay between each pair of RPM measurements, and also reflects a delay increase relative to the thrust request. This delay is probably not significant from the standpoint of aircraft safety, but does merit further investigation. Possible mechanisms responsible for this change include oil quality, fuel delivery, bearing health, or significant changes in the atmosphere or engine itself that could affect combustion efficiency. One key point in pursuit of this anomaly is that it is not persistent from one flight to the next. This seems to rule out the possibility of wear or gradual damage such as bearing problems, while emphasizing day-to-day variables such as fuel, oil, or interactions with other (unsensed) systems that could be intermittent. It is also relevant that the effect is usually stronger at the end of flights than at the beginning, suggesting a shift in behavior due to heating, contamination, or perhaps simple expenditure of consumables.

Unanticipated Operating Mode

A third class of anomaly, sensed strongly on Flight 8 and weakly elsewhere, also implicates the fuel system, but does not have the persistent character needed to explain the results above. This anomaly is illustrated in Figure 14. Note that in this graph, the vertical (color) scale has been magnified by 2.5 times for additional clarity.

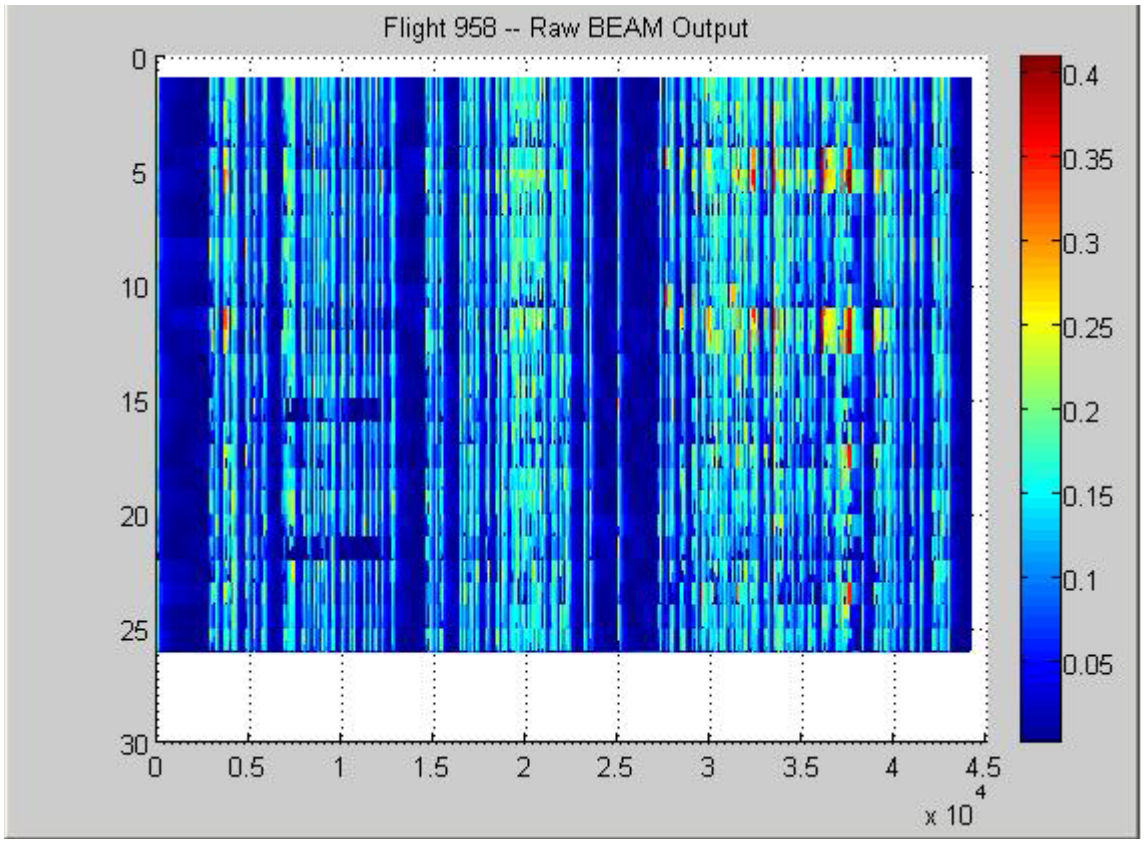

Figure 14. BEAM channel distance output, Flight 8 (vertical scale magnified)

This output also shows the RPM anomaly in the latter half of flight, as well as a brief episode at startup. More unusual are the coincident anomalies in signals 17 and 24 near sample 37000. These two signals are the fuel flow measurements for the left engine (channel 17) and the right engine (channel 24).

The combination of RPM and fuel anomalies was the strongest anomaly seen on any flight to date. This detection appears to overlap the result from IMS on this flight, and we will defer to the previous section for a discussion of anomaly implications.

Seeded Faults

Finally, we also simulated two faults early in the experiment, by shutting down the left and right engines at different times while in flight. Results are shown in Figure 15.

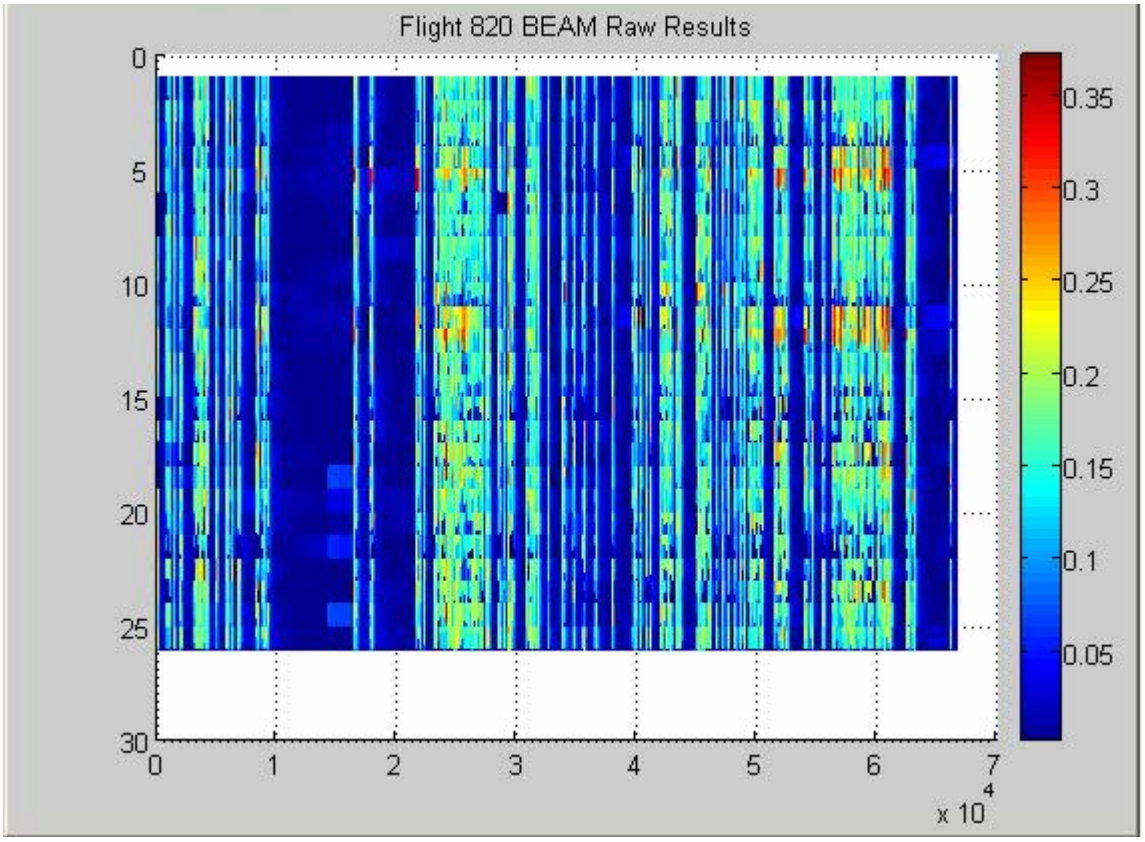

Figure 15. BEAM channel distance output, Flight 2 (vertical scale magnified)

Besides the now familiar RPM anomalies, seen most strongly at right, there is no obvious detection, with all other channel distances remaining below 0.2. Where are the engine shutdowns? This can be seen in the accompanying mode output in Figure 16, below.

The engine shutdowns are represented by the two "dips" before and after sample 30000. By aligning the time axis of this graph with Figure 15, one can see that there is no meaningful detection.

This result is actually correct. BEAM did not consider either engine shutdown to be an anomaly simply because its state model told it to *expect* an engine shutdown. In the case of the left engine shutdown, BEAM applied a model garnered from the training file, where the right engine was running but the left engine had not yet been started, and found it to be consistent. Neither engine was malfunctioning, the only difference was that the training file reflected data taken on the ground while this data was in flight, and both engines were behaving similarly in both regimes. In contrast, when the right engine was shut down, BEAM *had no* accompanying model, since the training file contained no example data for left-engine-only operation. In response, BEAM flagged the time period as novel, but did not compute a channel distance, reporting zeroes instead. It should be noted that IMS flagged the engine shutdowns as an anomaly because it considered static pressure and altitude in its calculation and therefore knew that aircraft was not on the ground (shutting down the engine was not appropriate). BEAM did not include these parameters.

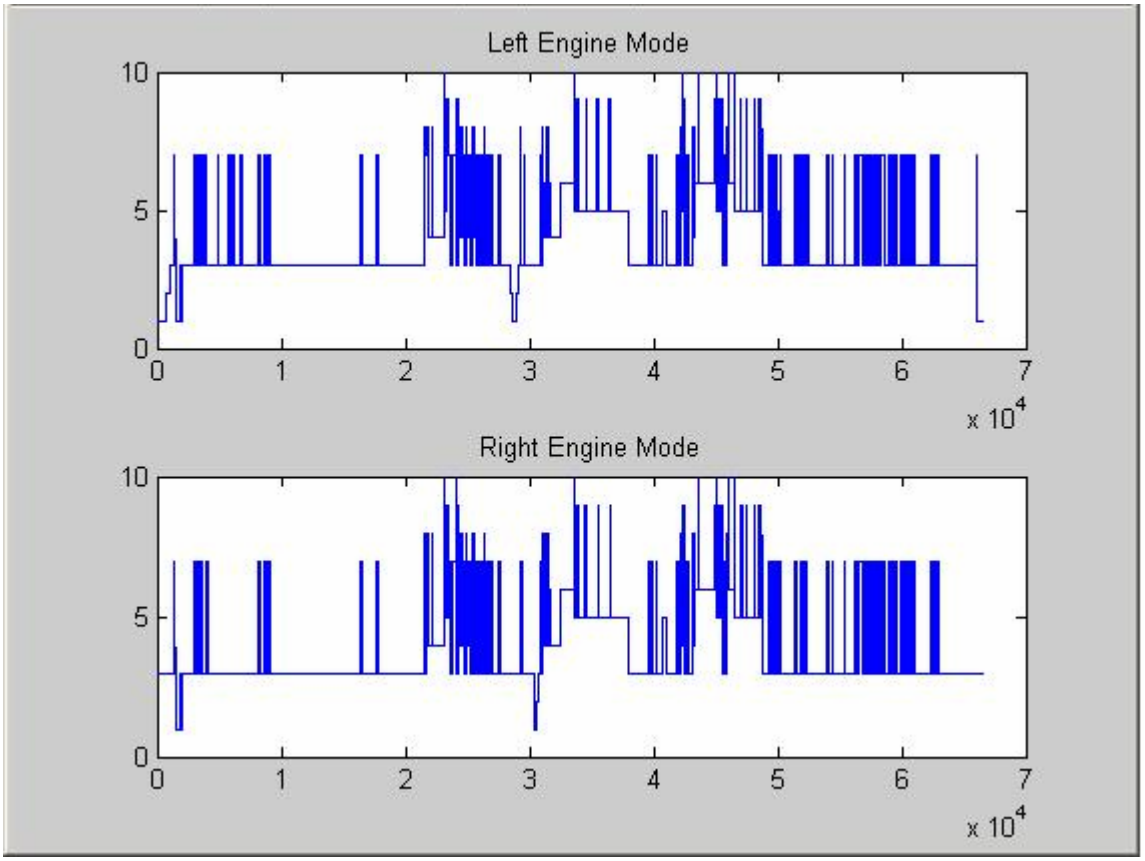

Figure 16. BEAM sensed mode, Flight 2 results

We plan to re-run the engine shutdowns using a false mode signal that instead informs BEAM not to expect the mode change. This test should indeed result in a sensed anomaly, specifically a mismatch between expected and detected mode. This and numerous other fault injection cases can also be tested using the F/A-18 simulator in the future.

F/A-18 TECHNOLOGY ACCELERATOR WAS A SUCCESSFUL TEST BED

The Dryden F/A-18 technology accelerator was an extremely valuable tool for developing new ISHM technologies. This includes not only the aircraft and the data processor, but also the larger system that includes the people, process, and data required to perform research. The Dryden team provided access to systems and shared knowledge about the aircraft. The F/A-18 1553 Bus provided convenient interface to the data and was easy to work with. The prior flights of the aircraft provided large samples of data for model development. The DFRC flight-rated PC/104 processor worked well for development and testing, as well as for the integration into the aircraft and flight testing. The F/A-18 flight simulator became a very valuable part of the process. Our procedures for testing the software prior to loading on the aircraft worked well, and the data download process instituted by DFRC helped continued refinement of the models and algorithms.

TEAM SYNERGY

The three-center collaboration worked well on this project. Given a fixed budget and a short timeline, the team made great progress while employing only a small group of part-time senior engineers (Specialists) and two experienced principal investigators. The Specialists were able to support this project as well as their own primary responsibilities at their respective centers with minimum schedule impact. This sharing of resources was beneficial to the project because it saved both time and money. Overall, fewer resources were required to support the project activities at each of the centers. Further, problems encountered at each center could be overcome quickly by using experts located at the other centers.

X-WORKS

The X-Works Program Concept holds great promise for the development and maturation of future ISM technologies. As more NASA centers and industrial firms join this effort, more expertise and resources can be dedicated towards creating technologies that are geared to improving safety, reducing operations costs, and increasing system availability. Once more ISHM technologies are proven in the field, they can then be made available as building blocks for future space transportation designs. This process will undoubtedly feed on itself in a progressive upward spiral, potentially creating additional opportunity both for those creating the technologies and those applying the technologies.

LESSONS LEARNED

Many technical and organizational lessons were learned during this project. This section attempts to briefly describe potential problem areas and how they may be anticipated and avoided in future projects.

ALLOW EXTRA TIME WHEN INTEGRATING CODE INTO NEW HARDWARE AND OPERATING SYSTEMS

The Dispatcher used a Flash drive to store real-time data. The Flash drive did not use DMA, so the CPU was used to transfer the data to the drive. The large number of writes to the Flash drive caused the Q104-1553 software driver to hang. This problem was encountered fairly late in the integration process during testing on the F/A-18 flight simulator.

Our investigation strongly indicated that the software hang was due to a subtle compatibility issue involving the CPU, Flash memory, and 1553 card. This resulted in 1553 hardware queue overruns. By the time that the problem was narrowed down, it was too late in the project to procure and replace the hardware before the start of the data flights. The problem was mitigated using a software watchdog that could detect the problem and restart the 1553 driver if the system was to hang, resulting in at most a few seconds of data loss. In 2 of the 25 data collection flights, we encountered a hardware fault that was not caught by the software watchdog and resulted in a partial loss of data. Although further ground tests using a different operating system, updated 1553 firmware,

and drivers did not resolve the problem, we feel that moving to an ATA-2 (fast ATA or EIDE) Flash drive with DMA should result in proper operation.

The lesson learned is that even proven systems employing similar combinations of hardware and software may not perform as expected in new applications. In the future, similar issues may be avoided if more time is spent up front exercising and thoroughly testing the platform earlier in the integration process.

RECOGNIZE AND ADDRESS ITAR ISSUES EARLY IN THE PROJECT

ITAR restrictions are applicable to all sensitive information. ITAR restrictions were underestimated early on in the project and resulted in significant delays. Each center has different requirements and a different team of people that may have to be consulted before sensitive data can be transferred. In addition, the process of releasing data is not a common task and is often misunderstood.

We found that many systems that have been developed at NASA (such as simulators, prior projects) have the process documented; however, finding the documentation or the set of people responsible be releasing data can be difficult. To avoid delays in the future, all data needs should be requested in writing well in advance so any issues can be worked out before the data is actually needed. The process of understanding applicable ITAR requirements can be streamlined if the data requirements are documented and all personnel who need to handle the data are identified early on in the project planning stages.

USE EXISTING HARDWARE AND SOFTWARE WHENEVER POSSIBLE

Initially the Ames and JPL teams elected to procure and configure a Power PC running VxWorks because it was perceived that this type of platform would more closely match a system found in a state-of-the-art space transportation system design. After some discussion, DFRC helped the rest of the team understand that this was not the best approach, given the amount of time and effort required to certify a new system for flight. Instead the team agreed to pursue a PC/104-based system running a Linux OS, based on its previous flight and acceptance experience at Dryden. Although not truly a real-time system, a similar PC/104 system would only require minimum testing to pass flight certification.

To give the project a jump start, DFRC provided ARC and JPL with a spare PC/104 that had been used on another project. This spare unit gave ARC and JPL the ability to begin software development as soon as possible, further helping to reduce overall cost and speed up the schedule. Purchasing the replacement for the borrowed hardware as well as flight spares took more time than was originally anticipated because of the long lead times; however, the amount of time spent performing this task was small compared to the amount of effort that would have been required to start up a new system from scratch.

PLAN EXTRA TIME FOR TRAVEL, EVEN IF THE LOCATION IS NEARBY

Given the multi-center collaborative effort, the project recognized the need to get people together to ensure that everyone remained focused on the same page. Good planning, communications, and the use of electronic and telecommunication technology helped minimize the number of meetings required; however, some meetings were still necessary to transfer hardware and to resolve issues. Given that DFRC was the focal point of the project, most of the initial face-to-face meetings were conducted at DFRC.

Much of the effort required to travel to DFRC, however, was underestimated. The time to drive from JPL and ARC to DFRC is approximately three and six hours respectively. Commercial airports are not located close enough to DFRC to make flying on commercial airlines worthwhile and landing private aircraft at DFRC is not trivial to arrange. As an alternative, ARC utilized NASA-7 (a NASA twin prop plane routinely flown as a shuttle between the three centers); however, this form of transportation was not always reliable, given there was a minimum number of passengers required to initiate a flight. As a result the flights were often delayed or canceled for several days at a time.

As a second alternative, the project enjoyed the use of a team member's private aircraft for transportation to nearby airports. The project was fortunate to be able to test the hardware initially at Ames because only 2–3 follow up trips to DFRC were necessary after the major integrations. In addition, given JPL's proximity to DFRC and their willingness to travel, JPL was able to perform many of the details associated with short-term integration and testing. A lesson learned is to plan for travel funding and travel delays, even if the project is short-term in nature.

NEXT STEPS

The ISHM Test Bed Project demonstrated that aircraft can be used economically and effectively as a test bed to develop and mature ISHM technologies. The project also demonstrated how resources from three centers can be combined into a synergistic team that can effectively overcome technical and programmatic challenges and meet common goals. It is important that this work be continued and expanded to support the development of ISHM. With improved ISHM technologies, the space community would be better able to improve the overall reliability and effectiveness of newly designed space transportation systems.

This section provides an overview of follow-on work that is recommended to further expand the capability and usefulness of the two ISHM algorithms (IMS and BEAM) and ISHM technology in general. It discusses ways that the "technology accelerator" itself can be expanded to fulfill ISHM and other roles.

ALGORITHM DEVELOPMENT

Great strides were made in testing and advancing the readiness level of the IMS and BEAM algorithms using the flight test bed. The success generated many natural development directions for these algorithms that should be explored.

Development Directions for IMS

To date, IMS has been shown to be an effective tool for detecting anomalies in system performance. Given a sufficient amount of training data, IMS can determine if a systems behavior has changed, which may be indicative of a system fault.

Better Fault Detection and Isolation: To identify potential faults, the IMS output files must be scanned for consecutive records with large distances. A knowledgeable system engineer or service technician could then further investigate the validity and or nature of the fault by viewing the system state, vector elements, and timestamp information.

Future versions of the software could be augmented to automatically determine the nature of the fault without the need to manually scan IMS data files. These methods were demonstrated in the lab to identify the specific parameters in the data vector that are out of tolerance with respect to other elements. Additional routines could also be employed to match the system behavior anomalies against known fault signatures. If specific failure modes can be uniquely identified, then methods can be produced to isolate the fault to the line replaceable unit (LRU) level. These methods could be implemented either by developing additional knowledge data sets based on known fault data or by incorporating other diagnostic tools (such as TeamsRT) to perform fault isolation to the LRU level once a failure mode has been identified.

As a first step towards maturing IMS, additional data must be analyzed to develop a higher degree of confidence in the fault detection algorithms. Ideally, sufficient amounts of data should be collected over time (possibly using a simulator) to allow IMS the opportunity to identify a significant number of faults. Once the databases have been matured to adequately characterize the engine system behavior, IMS should be able to detect faults correctly with a high degree of confidence. One possible approach for improving the knowledge database would be to work with DFRC ground personnel to understand the overall engine performance envelope and to make sure that all possible engine states have been characterized.

Once the engines are characterized with sufficient data under all operation conditions, IMS engine monitoring can be optimized. Specifically, the minimum number of parameters can be identified and the size of cluster files can be reduced as much as possible to make the algorithms run at an optimal speed. The next step would be to develop a Man Machine Interface to allow ground personnel to review in-flight faults or anomalies. Ground personnel could then develop confidence that the system works and is able to make ground operations more efficient. After proving the ground system performance, a next goal could be to develop an interface in the aircraft that could be viewed by the pilot.

Monitoring Additional Subsystems: IMS is a scaleable technology and as such is suitable to be used to monitor faults across more than one subsystem. Additional IMS components could be configured to monitor other subsystems independently of the engines; examples include the avionics subsystem, structural mechanisms, and power distribution. Once IMS routines have been proven at the subsystem level across all subsystems, supervisory IMS routines could then be developed to detect faults at the system level. Such a capability would prove to be an invaluable source of information that would allow the integrated fault detection system the ability to confirm or dismiss potential failure reports using alternative and redundant information that is obtained from independent and multiple sources.

ISHM TECHNOLOGIES

There are several development steps for extending this study and to address some of the fundamental ISHM issues that will need to be resolved for CEV and CLV development.

- Continue to record data on this aircraft.
- Demonstrate the integration of IMS and BEAM.
- Use simulation to demonstrate fault detection coverage of IMS and BEAM.
- Expand BEAM and IMS to demonstrate their ability to individually monitor multiple subsystems.
- Integrate a diagnostic system and demonstrate that it can use the detection information that would be provided by BEAM and IMS.

Continue to record data on this aircraft and engine system: ISHM developers and researchers have made good progress recording flight data on an aircraft. So far, data has been successfully collected from 25 flights and the software has demonstrated the ability to identify system behavior that is not consistent with past behavior or behavior that has not yet been incorporated into system models and training data. Additional data is needed from the F/A-18 engines to further refine the engine models and to ensure that all areas of operation have been characterized in the training data. Furthermore, the software must be given the opportunity to prove over time (perhaps through several maintenance cycles) that it can correctly identify anomalous system behavior. Collecting data before and after major maintenance periods will help demonstrate the software's ability to effectively recognize any changes in engine performance or ability to correctly detect any faults in the system.

Demonstrate the integration of IMS and BEAM: BEAM and IMS are examples of failure detection systems that are configured to work independently. Currently, these applications each monitor a different set of input variables using different methods. Post evaluation of the output has shown that in many cases, IMS and BEAM are able to recognize common differences in engine performance; however, this does not occur in every case and the detected anomaly may not be synchronized in time.

As a solution, a next logical step would be to integrate the output of IMS and BEAM either through an interface in BEAM or through a higher-level monitor program that can, at a minimum, monitor the output of both systems and report whether an anomaly is

suggested by both systems. Such a system does not currently exist, and will be a crucial component of future monitoring systems that employ multiple detectors, including both hardware and software detectors. This could be a very good means for performing dissimilar redundant detection.

Use simulation to demonstrate fault detection coverage of IMS and BEAM:
IMS and BEAM are currently capable of detecting minor (subtle change in system performance) and major (engine shut down) problems; however, not enough data has been collected to ensure a high reliability for detecting the likely range of faults.

This next step proposes to use the simulator to collect a data set rich in faults. Fault-rich data can be obtained by simulating failure modes in the software. This data set can then be used to show that the algorithms can identify faults accurately or can be used to make further enhancements to the fault detection algorithms.

Expand BEAM and IMS to demonstrate their ability to individually monitor multiple subsystems: In this study, both BEAM and IMS have been configured to monitor the two engines on the aircraft; however, IMS and BEAM have not been configured to identify faults beyond the engine system.

To continue this work, IMS and BEAM should be extended to monitor multiple subsystems on the aircraft. Other subsystems could include the Navigation and Guidance, Hydraulics, Structures, or Power Distribution. Expanding monitoring to additional subsystems is a research area in ISHM that could not only provide the detectors with the capability to recognize additional faults from other subsystems, but would also provide additional data that may allow the detectors to understand and exclude faults across systems such as the engine subsystem.

Integrate a diagnostic system and demonstrate that it can use the detection information that would be provided by BEAM and IMS: An important challenge in future ISHM systems will involve the integration of other fault isolation algorithms. A diagnostic system could be integrated with BEAM and IMS to isolate faults to the LRU level.

This next step proposes to integrate a state-of-the-art diagnostic system in the flight box and use the output from IMS and BEAM as the input to that system. Examples of such diagnostic systems could include a commercial system such as TEAMS or a NASA-developed system such as Livingstone. In software development, this will require the development of interfaces between the detectors and the diagnostic system and also will require development of models to support the diagnostic systems. To accommodate these significant changes to the flight box, a new operating system may be required in concert with a processor upgrade and an increased memory capacity.

ISHM OPERATIONS

In addition to evaluating how the ISHM technology interfaces with the aircraft, it's also important to evaluate how ISHM will interface with the human operators and systems that will react to their output.

Display IMS and BEAM results to maintenance crews: This next step proposes to provide a platform and environment that displays IMS and BEAM results to the ground and maintenance personnel. These results could be used to show the status and state of the aircraft after every flight and could be used to demonstrate that IMS and BEAM would be of value to the maintenance processes. Although there have been some side questions and discussions, the observations and proposed results of the IMS and BEAM algorithms have not been shared with the flight and ground maintenance personnel. A future operational ISHM system should provide feedback to the pilots, monitors, and maintenance personnel.

Develop and integrate ISHM Cockpit Display capability: NASA Dryden has significant experience in the introduction of interfaces into the cockpit. A proposed next step would be to develop a display output that will provide an interface to the cockpit (and/or cockpit simulator) that would be capable of alerting the pilot to any ISHM issues. If done properly, this same interface could be used throughout the development process: in the lab, ground test facilities, F/A-18 simulator, and F/A-18 test aircraft.

F/A-18 TECHNOLOGY ACCELERATOR

There are also several other steps that should be pursued for extending capability and interfaces of the tech accelerator:

- Integrate a space-rated processor and operating system.
- Expand the capability of the F/A-18 flight Simulator.
- Provide access to other systems on the aircraft beyond the 1553 interface.
- Install multiple ISHM data processors.

Integrate a space-rated flight hardware and operating system: Provide the capability to allow additional hardware (as well as software) to be tested on board the F/A-18. The PC/104 processor has been sufficient for the current work; however, further work (such as that proposed in the section above) will most certainly exceed the performance and memory capacity of the current system.

This next step proposes to upgrade the hardware and software to a real-time system that can be rated for space flight, such as a Power PC and VxWorks platform. Testing the software on a space-rated system will provide project planners with more assurance that the software has evolved to a higher maturity level, will provides the researchers and developers hands-on experience using the advanced processor, and will save the space community development costs through the use of common hardware and software at all levels of development.

Expand the capability of the F/A-18 flight simulator: The F/A-18 flight simulator was an essential element in the development and testing of the 1553 interface, flight hardware, and software. This next step proposes to continue this use and extend the capabilities of the simulator by developing and integrating expanded F/A-18 system models that would provide enhanced realism to the simulation and would allow more tests to be performed. We could also introduce additional systems to the simulator that would represent prototype systems that might be on a space vehicle. This would allow both the models and the reaction of the system to be tested and developed. These models could also be used for the development of diagnostic systems that would interface with IMS or BEAM systems.

Provide access to other subsystems on the aircraft beyond the 1553 interface: The 1553 interface provides some but not all the data and information associated with the F/A-18's subsystems. Dryden has installed an RQUID (Research, Quick Interface to Data) data recorder that provides information from additional systems on the aircraft. This next step proposes to expand the data interface available to our researchers to include the RQUID interface along with other data systems that could be economically interfaced with the data processor. A further step would be to research the state of the art in space-based interface busses and develop a simulation of that bus that could be implemented on the F/A-18 aircraft, as well as the F/A-18 simulator and the ISHM ground test facility.

Install multiple ISHM data processors: Install additional hardware (as well as software) on board the F/A-18. ISHM processor hardware required for this study occupies only a small part of the equipment bay of the F/A-18 research aircraft at Dryden. There also is space available in a forward gun bay, along with external racks that could be used to carry hardware to altitude for testing.

In addition to testing ISHM software interaction with the aircraft, the existing data processor used to monitor the engines could also interface with other hardware (for example, other avionics, systems, electronics) that may be used in space. This approach would enable integrated system development testing of ISHM in a cost-effective manner.

SUMMARY

Achieving NASA's exploration vision will require high-performance flight test platforms that can be used to advance and access the technology readiness level of ISHM hardware and software technologies for use in space projects. In this successful technology accelerator project, a cross-center team composed of NASA JPL, Dryden, and Ames engineers developed such a flight test platform based on available F/A-18 test aircraft, flight simulator, bus, processor, and software technologies.

Developed over a span of five months with minimal funding, this platform was used to enhance, integrate, and test existing IMS and BEAM detection algorithms over 23 multi-purpose flights. BEAM and IMS individually monitored the performance of both engines throughout the aircraft's flight regime. In addition to demonstrating the ability to detect

overt anomalies such as engine shut down, both algorithms detected subtle differences in engine system performance between training and actual flight data.

The team utilized X-Works development and management methods to accelerate this project, which integrated well with Dryden's pre-flight hardware and software test requirements. In addition to advancing technology, the project also fostered the development of a new sense of partnership and synergy between these three NASA centers. This high-performance aircraft platform, along with the development hardware and processes, is currently available to advance ISHM and other cutting-edge technologies.

ACKNOWLEDGEMENTS

We would like to thank Joe Totah and Serdar Uckun at the NASA Ames Research Center, Jack Levine and David McBride at the NASA Dryden Research Center, and Stephen Prusha and Art Murphy at the NASA Jet Propulsion Laboratory for working to make this project a reality.

REFERENCES

Iverson, D. Inductive System Health Monitoring, in *Proceedings of The 2004 International Conference on Artificial Intelligence* (IC-AI'04), CSREA Press, Las Vegas, NV, June 2004.

James, M., Mackey, R., Park, H., & Zak, M. BEAM: Technology for Autonomous Self-Analysis, IEEE Aerospace Conference, March 2001.

Norlin, K.A. Flight Simulation Software at NASA Dryden Flight Research Center, NASA Technical Memorandum 104315, NASA Dryden Flight Research Center, October 1995.

Mackey, R., Some, R., & Aljabri, A. Readiness Levels for Spacecraft Information Technologies, IEEE Aerospace Conference, March 2003.

Park, H. G., Mackey, R., James, M., Zak, M., Kynard, M., Greene, W., & Sebghati, J. Analysis of Space Shuttle Main Engine Data Using Beacon-based Exception Analysis for Multimissions, IEEE Aerospace Conference, March 2002.