

Diagnostic Information Fusion: Requirements Flowdown and Interface Issues

Kai Goebel, Michael Krok, Hunt Sutherland
GE Corporate Research & Development
One Research Circle
Niskayuna, NY 12309
{goebelk, krok, sutherland}@crd.ge.com

Abstract— In this paper we examine the implementation of a real-time, diagnostic, information fusion algorithm, which achieves a more accurate diagnostic estimate of system health than individual estimates produced by a heterogeneous collection of diagnostic tools. We discuss an approach that both decomposes performance requirements (through a flowdown) into the design and accommodates the differences among the interfaces of a heterogeneous collection of diagnostic tools. We focus in particular on issues which arise from an implementation which monitors system health in real-time and do not consider an off-line post-processing implementation. Through a systematic flowdown of performance requirements into the system design, the combined effects of all the diagnostic tools are considered in order to achieve the desired high level of accuracy for each fault of interest.

NOMENCLATURE

CDP	Customer Discharge Pressure
CLM	Component Level Model
D1	overall delta diagnostic approach
D2	delta-delta diagnostic approach
FD	Fault Detected
FP	False Positive
FN	False Negative
FADEC	Full Authority Digital Engine Control
FOD	Foreign Object Damage
GDL	Ground Data Link
GUI	Graphical User Interface
HPT	High Pressure Turbine
IFM	Information Fusion Module
IGV	Inlet Guide Vanes
IMATE	Intelligent Maintenance Advisor for Turbine Engines
LPT	Low Pressure Turbine
MEMS	Micro Electro-Mechanical Systems
MR	Management Related
NFD	No Fault Detected
RR	Requirements Related
VBV	Variable Bleed Valve
VSV	Variable Stator Vane

INTRODUCTION

Today, manufacturers and service providers tend to develop different tools to accomplish specific diagnostic detection needs. This patchwork approach achieves optimization at the

component level, but ignores benefits gained by taking a system-level view. In our system-level view, no one tool is required to deal with all faults of interest at a high level of accuracy, because no one method can do so. For example, some diagnostic tools are less sensitive to slight environmental changes than others are; some cannot easily be expanded to detect new faults; and still other tools are very good at detecting certain faults while being much poorer or inapplicable for other faults.

Recent advances in both hardware and software technology have made it possible to implement more sophisticated and reliable diagnostic algorithms in real-time systems. Therefore, it seems logical to take the next step and use a system-level scheme that gathers and combines the results of different diagnostic tools to maximize the advantages of each one while at the same time minimizing the disadvantages. Such a fusion scheme holds the promise to deliver a result that is better than the best result possible by any one tool used. In part this is achieved because redundant information is available that when combined correctly improves the estimate of the better tool and compensates for the shortcomings of the less capable tool.

When several tools agree on the diagnostic state, the task is straightforward and the resulting output will be done with high accuracy and confidence. However, issues arise when the information to be fused is different between the tools [1]. For example, if tools disagree, one has to decide which tool to believe and to what degree. In addition, when information is expressed in different design domains, such as probabilistic information, fuzzy information, binary information, or weights, the fusion scheme needs to map the different domains into a common one to be able to properly use the encoded data. The fusion scheme also has to deal with tools that produce estimates at different sampling frequencies. Some tools may operate at a millisecond sample period while others give only one estimate at a specific phase of operation of the system. We present here an approach that tackles the above topics and will focus in particular on the requirements flowdown and interface issues.

SYSTEM DESCRIPTION

The system under consideration is a gas turbine engine. The main task was to provide in-flight health monitoring capability for gas path faults. Key system components considered for this health monitoring scheme are the fan, the high pressure compressor, the high & low pressure turbines, and bearings. In addition, wireless micro electro-mechanical systems (MEMS) measure and process vibration data from the bearings. This sensing technology offers enhanced

turbo-machinery vibration diagnostics without an accompanying weight penalty. The information fusion module (IFM) demonstrates dual use capability by being designed, and tested on both a commercial and a military engine (CFM56 and F110, respectively). The faults considered are:

Fan fault – Fan blade damage, typically occurring due to bird strikes or other Foreign Object Damage (FOD) during takeoff.

Compressor fault – Compressor blade damage or abnormal operation

High Pressure Turbine (HPT) fault – Typically a partial loss of one or more blades, most commonly during high power conditions.

Low Pressure Turbine (LPT) fault – Typically a partial loss of one or more blades, most commonly during high power conditions. LPT blade faults are less frequent than HPT blade faults.

Customer Discharge Pressure (CDP) fault – Leakage in excess of the desired bleed level commanded by the aircraft and communicated to the Full Authority Digital Electronic Control (FADEC). FADEC does not have control over the CDP valve. The CDP valve takes air off the HP compressor for use in various aircraft functions such as air-conditioning, cabin pressurization, and anti-icing.

Variable Bleed Valve (VBV) fault – VBV doors not closing according to FADEC issued command, or one or more doors get stuck in a particular position. VBVs are intended to prevent low pressure compressor stalls.

Combustor Leak (Leak) fault - Holes burnt in liner and hot gas leaks into the bypass duct.

Variable Stator Vanes (VSV) fault – Manual errors in installation resulting in small misalignments in vane angles. The VSVs control the amount of air that goes through the high pressure compressor.

Inlet Guide Vane (IGV) fault – Manual errors in installation resulting in small misalignments in vane angles. The IGVs control the amount of air that goes into the fan.

The combustor leak, VSV, and IGV faults are applicable to the military engine, while the CDP leak and VBV faults are applicable to the commercial engine only; otherwise, the faults are applicable to both engines.

Several diagnostic tools as well as non-diagnostic information sources were selected for information aggregation.

INFORMATION FUSION PROBLEM DESCRIPTION

Current diagnostic and condition monitoring systems generate information that, while unambiguous in their specific intended application, will be less accurate as more fault coverage is demanded from the tool and less definite as new diagnostic tools are added to either enhance capability or address new faults. This may lead to: 1) ambiguity in troubleshooting, 2) maintenance personnel making uninformed decisions, 3) erroneous component removals, and 4) high operating costs. To illustrate, consider two diagnostic tools A and B. Tool A can only diagnose whether or not fault x is present while tool B can diagnose faults x and y. Tool A can diagnose fault x very reliably while tool B's performance is only mediocre. If fault x occurs, there is a high likelihood that tool A indicates fault x and also a lesser likelihood that tool B indicates fault x. If both tools indicate fault x one should be reasonably certain that fault x did indeed occur. If tool B indicates fault y one is still better off choosing fault x because of the good performance of tool

A. However, should fault y occur – of which tool A knows nothing about – tool A will always misdiagnose that fault as either fault x or no fault. Tool B will perform its mediocre job by sometimes indicating fault y correctly. The decision maker (maintenance personnel, fleet management, etc.) faces the quandary which tool to believe. He/she sees only that (otherwise reliable) tool A indicates fault x (or no fault) while tool B indicates fault y (or worse, since it does not do a very good job, sometimes fault x). Incorrect diagnosis leading to the aforementioned problems (1-4) is the likely result. Averaging schemes and voting schemes will not yield desired resolution of our dilemma. A more sophisticated fusion scheme is required.

The fusion effort is one part of an overall project which addresses these problems through the design and test of a condition-based Intelligent Maintenance Advisor for Turbine Engines (IMATE) system [2]. The overall goal of the information fusion is to combine the relevant diagnostic and other on-board information to produce a fault diagnosis estimate to mitigate each of the aforementioned problems. The vision is to achieve a more accurate and reliable diagnosis than any individual diagnostic tool.

REQUIREMENT ISSUES

The key role of requirements is to show users and developers what is needed from a system. When the requirements use a language which everyone can understand, all people involved, including customers, can see the whole system and their role within the system design and development. Therefore, the most basic role of requirements is defining what needs to be accomplished while the next most important role is communication. Changes will occur throughout the systems life cycle, and normally involve design or implementation changes, i.e., issues that disturb the requirements. However, unless we know the relationship between requirements and design, we have no rational basis for deciding on a change. Transfer functions denote relationships between the higher level and lower level requirements. A good set of requirements along with the transfer functions that flowdown the relationship of the higher level requirement to its constituent parts defines precisely what is wanted, but simultaneously leaves the maximum space for creative design.

For the system at hand, the subsystem level output of the diagnostic tool is specified, but not the details as to how to achieve the required performance level. To address the two customer concerns, life cycle cost and safety, the system level performance requirements identified in this project were:

- Fault Detection (FD) $\geq 95\%$, (and reduced false classifications (FC))
- False Positives (FP) $\leq 1\%$.

Life cycle cost is addressed by reducing erroneous component removals through substantially lower false positive fault indications. Both life cycle cost and safety are addressed by reducing secondary damage through more accurate fault detections at small or incipient levels of component damage. Figure 1 shows the condition assessment matrix where we consider four possible outcomes to the condition assessment:

NFD: No Fault Detected; declaring a good system good.

FP: False Positives; declaring a good system faulted.
 FN: False Negatives; declaring a faulted system good.
 FD: Fault Detection; declaring a faulted system faulted.

		Estimated Condition	
		No Fault	Fault
Actual Condition	No Fault	NFD	FP
	Fault	FN	FD

Figure 1: Condition Assessment Matrix

In the case of a multi-fault system, this matrix extends to size (nxn) where n is the number of faults (including no fault). In that case, the FPs occupy the top row excluding the NFD entry, the FN occupy the first column (excluding the NFD). Furthermore, the FD can be broken down in correctly and incorrectly classified faults where the correct classifications are found on the diagonal and the FC are found in the off-diagonal entries [2].

Requirements are used throughout the life cycle - from concept to delivery. Although some projects use a dedicated requirements phase, we take an alternative approach in which there is no "requirements phase" since requirements have to be kept up-to-date throughout the project life cycle. They become the whole basis of technical management. We advocate the spiral design methodology for a new technology, as it is the best match to the way engineers work in a creative process [3].

Table 1: Ranking of Reasons for Project Failure

Reason	Relation	Failure Percentage
Incomplete requirements	RR	13.1%
Lack of user involvement	RR	12.4%
Lack of resources	MR	10.6%
Unrealistic expectations	RR	9.9%
Lack of executive support	MR	9.3%
Changing requirements/specs	RR	9.1%
Lack of planning	MR	8.7%
Didn't need it any longer	RR	7.5%

where RR: requirements related
 MR: management related

A majority of problems within projects are typically caused by systems engineering, not software engineering. QSS, Inc. and the Standish Report [4,5] identify five of the eight major reasons why projects are unsuccessful as requirements related (~52%), and the others as managerial (~29%), (see Table 1). None of the key reasons are technical in nature. Most of the time the problems are caused by poor or incomplete requirements, poor or minimal risk management, and failing to relate the software requirements and design to

the system requirements. To improve the chances of a successful project, the focus of interest has to be shifted away from software to systems engineering.

Requirements are the basis of modeling the problem, then the product. These requirements models are managerial in nature and are meant to increase the chance of the final product satisfying users, i.e., of being a quality product. Therefore, requirements management is the basis of quality. The systems engineer should be and typically is solely responsible for assuring that the system will satisfy the customer's needs. This accountability implies that the systems engineer or equivalent must possess several key skills: 1) an understanding of the customer's needs (or critical to quality characteristics); 2) the ability to derive and to decompose technical requirements from those needs; 3) the ability to allocate those requirements to design (both hardware and software); 4) the ability to communicate those requirements to designers and subcontractors; 5) the ability to manage and control the system development; and 6) the ability to assure the resultant design satisfies the requirements through verification (typically via testing).

Continuing with the requirement definition for the IFM, the primary requirements of the IFM are:

- combine relevant diagnostic and other on-board information available to produce a fault diagnosis, in real-time, that is more reliable and robust than any of the individual diagnostic module outputs;
- report and/or combine estimates of engine deterioration using all available information sources;
- accommodate discontinuities in diagnostic module fault coverage, i.e., each diagnostic module may only be able to detect faults in certain portions of the flight regime;
- accommodate temporal discontinuities in diagnostic fault coverage, i.e., different diagnostic modules produce their diagnoses at different rates during the various flight regimes;
- accommodate varying fault coverage of the diagnostic modules, i.e., different diagnostic modules can only diagnose certain fault subsets;
- accommodate varying fault diagnostic reliabilities, i.e., different diagnostic modules have different a priori performance credibility associated with particular fault diagnoses;
- resolve conflicts in fault diagnoses from individual diagnostic modules

Differences among diagnostic tool output must be resolved to arrive at an unambiguous overall output. Such differences can occur for example due to dissimilar fault coverage because not all individual diagnostic tools deal with all faults of interest. This is typically the case because the tool's underlying reasoner prevents it from performing a particular diagnostic task. A diagnostic tool may also not have the necessary sensor measurements to make a judgment about a fault. However, when different diagnostic tools have overlap in the fault coverage, the performance of the diagnostic tools will most likely not be identical. Rather, some diagnostic tools will do better at recognizing certain faults than other tools. To complicate matters, some tools operate only at particular operating conditions during the flight, for example at quasi steady state conditions during cruise while other tools operate after the flight when the aircraft has landed, and yet other tools may operate throughout the flight. This has implications for the case when a fault occurs late in the flight after some tools have already given their diagnostic opinion about the health status (and found that everything is

fine). Obviously, even when the diagnosis was right at the earlier time, it is wrong later on. Does that mean that one trust only the later tool? But then why do an early diagnosis at all? One might argue that some faults show up, say, only during an early flight phase. Finally, differences in the operating rate at which the diagnostic opinion of a tool is expressed will vary. Some tools perform the diagnosis only once per flight, other do it at a fairly high rate of, say, 30Hz. Discarding all but the last reading probably does not make too much sense either because it may happen that just the last reading is an outlier while all the other ones were fine.

Besides integrating diagnostic information, the fusion module is also charged with considering peripheral information. That is, information that in itself does not constitute diagnostic information but can be used to support a diagnostic finding.

Finally, the IFM also needs to provide a flexible interface to ground station GUI, a pre- and post-fusion diagnostic display information, algorithm parameter change capability, and support drill-down to diagnostic modules. These issues are dealt with in the section "Interface Issues"

Information Fusion Interface Definition

The output of the diagnostic tools may be expressed in different domains such as weights, percent, probability, fault magnitude, etc. This poses a problem when different input types need to be mapped into a unified domain. Consider, for example, how a probability would relate to a weight of, say, a neural network. To solve this issue, we imposed a requirement on the providers of the individual diagnostic and non-diagnostic information sources. In particular, the output of the diagnostic tools – which is the input to the information fusion – had to take the shape of a confidence level for each individual fault. The confidence level is scaled between zero and one. The interpretation is that a level closer to one means that the fault is increasingly more likely while a confidence level less than 0.5 is increasingly not likely. It was then the task of the designer of the diagnostic (and non-diagnostic) tools to provide the mapping from individual diagnostic tool output to diagnostic confidence level. The mapping is performed in a different fashion for each tool, using an appropriate function which could be arbitrary but is expected to be monotonically increasing in shape. Figure 2 shows an example for a diagnostic output to information fusion input mapping function.

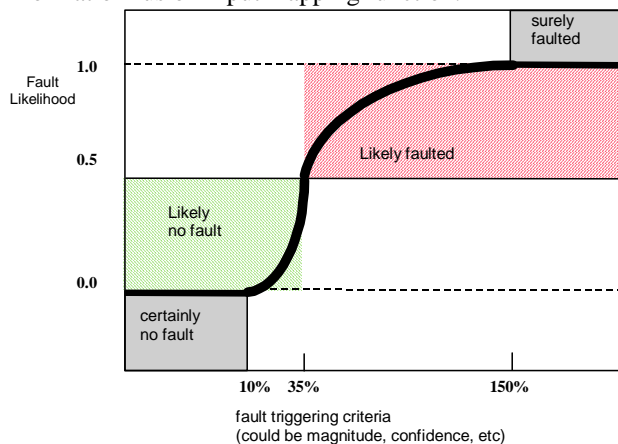


Figure 2: Example mapping from diagnostic domain to fusion input domain

Requirements Flowdown

The requirements flowdown had the purpose to provide the required performance for all faults. The underlying idea was that relaxation of recognition capability for some faults which have ample coverage from some tools might allow the tightening of recognition for other faults where one tool's performance is critical for the overall output. This is the case for example where only one tool is able to diagnose a particular fault. In general, false positives (incorrectly issuing an alarm) and false negatives (missing a fault recognition) have to be balanced to the needs of the customer. It is typically not possible to drive both false positives and false negatives down to zero. Therefore, the more critical component has to be identified first and a reasonable value for the second one needs to be established. For the two systems, it was found that two different strategies were employed. For the commercial engine more emphasis was placed on reducing false negatives to make sure that impending problems could be found. For the military engine more emphasis was placed on reducing false positives. In the latter case, this stems largely from the need to take down the engine if a fault was declared. In that context, false positives were very undesirable. Therefore, the system could be tuned differently for the two engines.

It must be noted that the purpose of the flowdown is not to relieve the diagnostic tools of performing as good as they can. Applying the data fusion principles of Hall and Garga [6] to decision level fusion, there is no substitute for a good diagnostic tool. In general, multiple, marginal-performance tools do not combine to produce an improved result and in fact may worsen the outcome. It is a misconception to think of the fusion module as a tool which will help poor performing diagnostic tools.

Primary and Secondary Diagnostic Inputs

After a rigorous down-select process, three primary diagnostic tools were chosen for implementation on the IMATE system: 1.) Overall Delta (D1) approach which compares pre-flight and post-flight conditions based on efficiency and flow scalar deltas. 2.) Delta-Delta (D2) approach which compares sensor measurements with modeled values. Deviations ("deltas") are indicative of faults. The comparison is referenced to pre-flight deltas to remove residual deltas. This makes it insensitive to engine sensor biases, but it cannot address a-priori faults such as misrigging. 3.) Neural Net (NN) approach which is a pattern recognition method that takes engine measurements as inputs, evaluates them as patterns and – if found suspicious – renders a fault message. It is an absolute approach that allows the detection of a priori faults that may occur, for example, due to incorrect maintenance. The other two approaches (D1 and D2) cannot address a priori faults. In addition, the NN method can be used during takeoff where no other technique operates but is somewhat sensitive to engine sensor biases.

Secondary diagnostic information is information that in itself does not constitute diagnostic information but may be valuable at supporting a diagnostic finding. This information was used only to support a diagnostic finding but could not be used to discount a finding. The information sources considered were: 4.) Model Based Engine Deterioration Estimator; 5.) Engine Component Model Deterioration Estimator; 6.) MEMS Vibration Analysis featuring

synchronous extraction of fan and core vibration levels; 7.) FADEC Fault Codes; 8.) Intermittent Fault Code Tracking; 9.) Thermal History; 10.) other flight related information.

Requirement Flowdown Results

Simulation requirements flowdown to the diagnostic modules was completed through two requirement flowdown-capability flowback cycles for each engine using a Monte Carlo approach. In order to provide a transfer function generator to aggregate IFM performance against the system level performance requirements, a first prototype fusion approach, which disregarded the reported FC interaction, was employed. The distribution of a priori tool reliability inputs was varied across the full range with the majority of the inputs in the range 70% – 90% starting with a balanced design and then changing the reliability parameters based on reported capability flowback. Table 2 and Table 3 show the required capability flowdown to each of the diagnostic tools considered (CFM56 and F110, respectively).

For the CFM56 in Table 2, we see that, in general, no one tool is assigned the entire fault detection problem in isolation. For the VBV fault, although the two Delta algorithms can support the diagnosis if the fault occurs after take-off, we do not rely on this atypical situation to occur. We require the Neural Network algorithm (designed to operate only through take-off) to be capable of meeting the system level requirement alone. In the unlikely case that the VBV fault occurs during a flight after take-off, the VBV fault detection will still possess a fault detection probability greater than 0.90 due to the combined contributions of the two Delta algorithms.

For the F110 in Table 3, the Neural Network is designed for full flight operation. It is the only algorithm that currently detects the misrigging faults (VSV, IGV), and therefore is assigned the entire fault detection problem. This implies that the NN carries the burden for these faults from a requirements standpoint. Although the F110 is operated in a significantly more “nonlinear” fashion than the CFM56, the requirements are not substantially different (for non-VSV/IGV related faults). For the leak fault, the required performance reflects capability flowback from the D1 algorithm.

Next, relative to currently reported diagnostic tool performance, the IFM design was updated to account for reported misclassification. The resulting IFM performance significantly exceeded the system level requirements during Monte Carlo simulation testing - fault detection (FD=99.8%), false positives (FP=0.01%), and false classifications (FC=0.2%).

Simulation testing of IFM has been completed and overall system performance is encouraging. A large number of engines with different component qualities, deterioration levels, and sensor bias errors were simulated at multiple operating regimes. Each flight regime was defined in terms of a range of altitude and Mach (e.g., 5k-40k feet, 0.4-0.95 Mach number), throttle resolver or power level angle (depending upon engine type), and ambient temperature variations to encompass cold through hot day operation. These engines were initially run unfaulted and then rerun with specific gas path faults of random magnitude introduced. Typically, the range of fault magnitude was limited to a 1.0%-3.0% change in component efficiency and/or flow.

INTERFACE ISSUES

A number of practical implementation issues were considered in the construction of the information fusion software for real-time operation. First, a step-wise development process was conceived to facilitate moving the design from an entirely off-line simulation environment to final implementation on the target platform. Second, an extensible interface to any of the diagnostic tools was defined with the goal of facilitating an easy migration from simulation to the real-time operation. Further, a flexible external interface was defined for logging and presentation of testing results on off-board computers. Finally, adjustment and configuration of the information fusion software was provided through use of a file based interface that is used both during the simulation and in real-time operation.

A three step development process was used to go from pure simulation of the system and algorithms to a final implementation suitable for execution on the target platform. In the first step, design of the software was accomplished entirely through use of a standalone simulation environment based entirely upon matlab®. During this phase, the design was developed using a non-physics-based simulation that permitted a convenient and efficient means to do extensive Monte Carlo tests without the strict limitations of the real-time platform. After the design was perfected the software was re-implemented in C code for integration into an off-line simulation of the real-time system. Use of the off-line simulation permitted integration of the information fusion with the various diagnostic tools in an environment which still permits a convenient means to test with a validated, physics-based simulation of the subject engines without the limitations of the real-time operation. The simulation runs approximately three times faster than real-time, and contains all the user aids for examination in the context of the full flight regime. Also accomplished during this phase was generation of test cases to be used for verification of the real-time operation. In the final step the code as implemented in the off-line simulation is transferred to the target platform for real-time operation. During this step we are concerned with accomplishing correct execution within the processing and memory constraints of the target platform. This step is complete when the test cases, as produced off-line, are successfully handled.

On the target platform a flexible interface was defined to communicate messages to an off-board GUI and data server repository that execute on ground station computers. We chose to transmit messages to avoid burden with storing data to local disk on the target platform. This facility not only supports system integration testing during real-time operation, but also is required in limited form when the system is deployed. Use of messages that are broadcast in a distributed computing environment permits easy extendibility to as many off-board computers as may be required during testing. Therefore, a set of messages were designed to permit capture of pre and post fusion conditions, as monitored by the information fusion, as well as sampling of more detailed drill-down data from the individual diagnostic tools. In our implementation we used an object oriented description of 11 message types, implemented in C++. A library of messages is maintained for use on the two target platforms and for the various Unix and PC ground station computers used during testing.

We implemented the library of data logging messages by means of a Unified Modeling Language (UML) editor [7]. As shown in Figure 3 we designed messages suited to capture data at the three processing levels in the system:

- 1) Sensor and preprocessed data;
- 2) Diagnostic tool outputs; and
- 3) System level reports.

The first two levels provide drill-down detail should there be the need to analyze the results from the information fusion. The system level messages also provide information on the operational status of the target platform. We used an editor that had the capability to generate C++ code ready for compilation and avoided having to directly code the message library. It was possible simply to define the content of the messages based upon the interfaces already defined at the three levels and generate the code with the appropriate methods used to load and retrieve data from each message type.

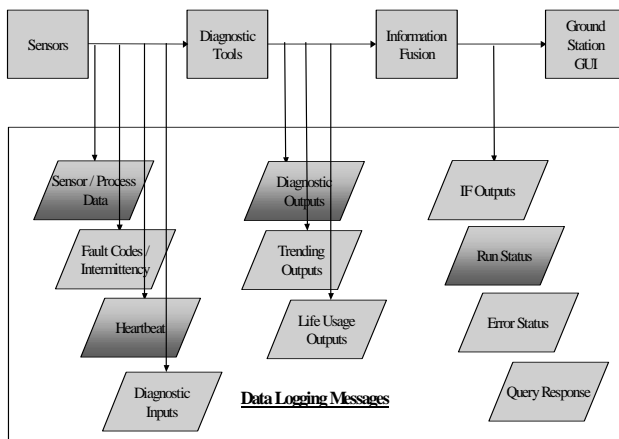


Figure 3: Types of messages

SUMMARY & CONCLUSION

In this paper, we examined the implementation of a real-time, diagnostic, information fusion algorithm, which achieved a more accurate diagnostic estimate of system health than the individual estimates produced by a heterogeneous collection of diagnostic tools. The system under consideration was a gas turbine engine, with dual use (commercial & military) applicability. Through a systematic flowdown of performance requirements into the system design, the combined effects of all the diagnostic tools were considered in order to achieve the desired high level of accuracy for an in-flight health monitoring capability for gas path faults. We focused in particular on issues which arose from a real-time implementation of system health. We solved the issue of mapping different input types into a unified domain by imposing an interface requirement on the providers of the individual diagnostic and non-diagnostic information sources. Namely, the input to the information fusion must be the shape of a confidence level (scaled between zero and one) for each individual fault. A three step development process was used to go from pure simulation of the system and algorithms to a final implementation suitable for execution on the target platform. On the target platform a flexible interface was defined to communicate messages to an off-board GUI and data server repository. Simulation testing of IFM has been completed and overall system performance is encouraging. The resulting IFM performance significantly exceeded the system level requirements. We

conclude the following: 1) the IFM is ready for engine testing for final proof-of-design; 2) the IFM design approach is applicable and will lead to successful implementations for a wide variety of gas turbine engines; 3) the IFM design approach is applicable to a wide variety of health monitoring problems (or more generally, classification problems) since the algorithm is not tied to gas turbine engine domain knowledge.

There are a number of avenues for potential future work: 1.) The highly modular design of the information fusion module allows the addition of other diagnostic and non-diagnostic tools. This would dramatically widen the scope of the tool from gas-path faults to component level faults and perhaps airframe faults. Related to this scope increase is the use of more advanced sensor technology which is able to capture a wider range of faults and perhaps at a higher level of accuracy, such as exhaust debris sensing and monitoring, oil quality monitoring, blade monitoring, etc.)

2.) The current architecture requires user-provided tool performance characteristics. Through the addition of on-line learning capability, some of the characteristics could be acquired over time, thus avoiding the need for extensive a priori testing.

3.) Finally, a multi-criteria decision making unit could be employed which would balance risk, cost, confidence of the fault, severity, impact on schedule or mission, etc. to aid the decision-maker in what to do about an existing fault situation.

APPENDIX

Table 2: CFM56 requirement flowdown to the diagnostic modules

Number	Type	Comments	Overall Delta	Delta-Delta	Neural Network
1	No Fault	Typical engine operation	0.850	0.950	0.800
2	Fan	Fan blade damage due to FOD	0.850	0.925	0.800
3	Compressor	Blade failure – both HP & LP	0.900	0.925	0.800
4	HP Turbine	Partial loss of one or more blades, typically during take off	0.900	0.925	0.775
5	LP Turbine	Partial loss of one or more blades; potential secondary damage with loss of upstream blade	0.875	0.925	0.700
6	CDP	Duct connection fails and causes a leak	0.775	0.925	0.850
7	VBV	VBV door(s) fail to close	0.400	0.900	0.950

Table 3: F110 requirement flowdown to the diagnostic modules

Number	Type	Comments	Overall Delta	Delta-Delta	Neural Network
1	No fault	Typical engine operation	0.850	0.950	0.800
2	Fan	Fan blade damage due to FOD	0.850	0.925	0.800
3	Compressor	Blade failure – both HP & LP	0.900	0.925	0.800
4	HP turbine	Partial loss of one or more blades, typically during take off	0.900	0.925	0.775
5	LP turbine	Partial loss of one or more blades; potential secondary damage with loss of upstream blade	0.875	0.925	0.700
6	Combustor leak	Holes burnt in liner	0.400	0.925	0.850
7	Inlet guide vane (positive direction)	Misrigging error that affects amount of air through fan	NA	NA	0.950
8	Inlet guide vane (negative direction)	Misrigging error that affects amount of air through fan	NA	NA	0.950
9	Variable stator vane (positive direction)	Misrigging error that affects amount of air through fan	NA	NA	0.950
10	Variable stator vane (negative direction)	Misrigging error that affects amount of air through fan	NA	NA	0.950

REFERENCES

- [1] K. Goebel, V. Badami, A. Perera, *Diagnostic Information Fusion for Manufacturing Processes*, Proceedings of the Second International Conference on Information Fusion, Fusion '99, vol. 1, pp. 331-336, 1999.
- [2] M. Ashby, W. Scheuren, *Intelligent Maintenance Advisor for Turbine Engines (IMATE)*, Proceedings of the IEEE Aerospace Conference, April 2000.
- [3] T. Kelliher, *Development Life Cycle Mobile*, Proceedings of the 8th Annual International Symposium of the International Council on Systems Engineering (INCOSE), July 1998
- [4] R. Stevens, G. Putlock, "Improving Requirements Management" (HTML), Quality Systems & Software, Inc., 1993-1999; <http://www.qssinc.com/library/paperlist.html>
- [5] The Standish Group, *"The Scope of Software Development Project Failures,"* 1995
- [6] D. Hall, A. Garga, *Pitfalls in Data Fusion (and How to Avoid Them)*, Proceedings of the Second International Conference on Information Fusion (Fusion '99), 1999.
- [7] S. Albir, *UML in a Nutshell*, O'Reilly & Associates, Inc, 1998.

Kai Goebel received the degree of Diplom-Ingenieur from the Technische Universität München, Germany in 1990. He received the M.S. and Ph.D. from the University of California at Berkeley in 1993 and 1996, respectively.



Dr. Goebel joined General Electric's Corporate Research and Development facility in Schenectady, NY in 1997 as a computer scientist after working as a visiting postdoctoral fellow at UC Berkeley from 1996 to 1997. He has carried out applied research in the areas of artificial intelligence, soft computing, and information fusion. He has worked on using soft computing techniques for real time monitoring and diagnosis of aircraft engines, power plants, and manufacturing processes, and for both data fusion and decision fusion in automated vehicle control systems, aircraft engines, and manufacturing systems.

Dr. Goebel is an adjunct professor of the CS Department at Rensselaer Polytechnic Institute (RPI), Troy, NY, since 1998 where he teaches classes in Soft Computing.

Michael Krok received a Master of Engineering and a B.S. in Electrical Engineering from Rensselaer Polytechnic Institute in 1980 and 1979, respectively.



Michael Krok is a Systems Engineer at the General Electric Corporate Research & Development Center (GE CR&D). He has participated in and has been responsible for the conceptualization, analysis, design, development, & evaluation of several, large scale, complex systems. He has provided technical and project leadership for the System Requirements/Concept /Design, and Integration & Test activities, and for the application of real-time control and diagnostic theory in many GE products in both military and industrial applications. His research interests include methods and tools for the application of Systems Engineering, and the study of real-time, nonlinear, electro-mechanical/hydraulic control systems and related diagnostic techniques. He has published technical papers and has several patent applications under review. He is a member of INCOSE.



Hunt A. Sutherland received a M.S. in Computer and Systems Engineering from Rensselaer Polytechnic Institute in 1978 and a B.S. in Electrical Engineering from Worcester Polytechnic Institute in 1974.

Hunt Sutherland is a Computer and Systems Engineer at the General Electric Corporate Research & Development Center (GE CR&D). He has participated in and led project teams responsible for the application of embedded controllers in many GE products and industrial applications. In 1995 he received the Willis R. Whitney Technical Achievement Award for leading a team which developed the firmware in the Spectra Electronic Control product. His research interests include methods and tools for design of real-time embedded systems and the study of control system architectures and analysis of system reliability. He has received four patents and authored technical papers, and articles in the field of design and implementation of embedded computer systems. He is a member of the ACM professional society.