

Decision Forgetting and Decision Smoothing for Diagnostic Decision Fusion in Systems with Redundant Information

Kai Goebel*

GE Corporate Research & Development, 1 Research Circle, K1-5C4A, Niskayuna, NY 12309

ABSTRACT

This paper introduces techniques to deal with temporal aspects of fusion systems with redundant information. One of the challenges of a fusion system is that individual information is not necessarily announced at the same time. While some decisions (or features or data) are produced at a high sampling frequency, other decisions are generated at a much lower rate, perhaps only once during the operation of the system or only during certain operating conditions. This means that some information will be outdated when the actual information fusion task is performed. An event may have occurred in the meantime leading to a decision discord. We tackle this challenge by introducing the concept of “information or decision forgetting”. In other words, in case of an information discord, more recent information is evaluated with higher confidence than older information. Another difficulty is distinguishing between outliers and actual system state changes. If tools perform their task at a high sampling frequency we can employ “decision smoothing”. That is, we factor out the occasional outlier and generally reduce the noise of the system. To that end, we introduce an adaptive smoothing algorithm that evaluates the system state and changes the smoothing parameter if it encounters suspicious situations, i.e., situations that might indicate a changed system state. We show the concepts introduced in the diagnostic realm where we aggregate the output of several different diagnostic tools.

Keywords: Diagnostic Information Fusion, Decision Forgetting, Decision Smoothing, Diagnosis, Classification, Decision Making

1. INTRODUCTION

Classification is a field that spans a wide range of disciplines such as pattern recognition, modeling, diagnosis, clustering, etc. Often times, problems not typically associated with classification tasks can be re-phrased to allow use of tools from the classification toolbox. One problem that makes classification a hard task is the dynamic nature of many systems. As their signature changes, they escape the carefully crafted partitioning domain. Efforts such as adaptive classification¹ try to copy with this challenge. An added level of complexity comes through the integration and aggregation of information of redundant information which has to be ranked based on their integrity². In case of dynamic systems, the aggregation scheme has to deal with temporally disjoint information. That is, although relevant, information is produced at different sampling frequencies. Some classification tools may operate at a millisecond sample period while others give only one estimate at a specific phase of operation of the system. This is a problem if a significant event occurs between the estimate of one classifier and the estimate of the other classifier. The fusion tool does of course not know whether an event occurs. As far as it is concerned, either one estimate is bad (and it does not know a priori which one that is) or something has happened. To illustrate, consider two classification tools A and B. We start with a base case where both tools render their information at the same time. Tool A can only establish whether or not event x is present while tool B can detect events x and y. Tool A can identify event x very reliably while tool B's performance is only mediocre. If event x occurs, there is a high likelihood that tool A indicates event x and also a lesser likelihood that tool B indicates event x. If both tools indicate event x one should be reasonably certain that event x did indeed occur. If tool B indicates event y one might be tempted to choose event x because of the good performance of tool A. However, should event y occur – of which tool A knows nothing about – tool A will always misdiagnose that event as either event x or nil. Tool B will perform its mediocre job by sometimes indicating event 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 event x (or nil) while tool B indicates event y (or worse, since it does not do a very good job, sometimes event x). Incorrect classification is the likely result with traditional averaging schemes and voting schemes.

* Correspondence: Email: goebelk@crd.ge.com; www: <http://best.me.berkeley.edu/~goebelk/kai.html>

Now assume that first tool A carries out its job, then tool B. Consider that first event x is present and event y occurs after tool A has completed its classification task but before tool B. An added problem is that both tools could perform correctly and that an a priori discounting of tool B's estimate is not appropriate. Rather, the temporal nature of the information has to be taken into account. In Sections 2-4 we will discuss approaches to deal with some of the issues outlined here.

Our work is motivated by a diagnostic task. 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. However, applying the data fusion principles of Hall and Garga³ to decision level fusion, there is no substitute for a good diagnostic tool. In general, multiple, marginal-performance tools do not necessarily combine to produce an improved result and in fact may worsen the outcome.

2. METHODOLOGY FOR DECISION SMOOTHING

We consider decision smoothing a pre-processing step before the information fusion main modules (Figure 1). As outlined above, the challenge in dynamic systems is in providing a different reaction to situations where decisions agree and situations where decisions disagree. When decisions agree, and in the absence of evidence to the contrary, we postulate there is no reason for the fusion agent to change the collective opinion within that time step (there is still a minute chance of joint incorrect classification). However, if tools disagree, the fusion main module has to decide whether one tool is correct and the other is not (and which) or whether an event has occurred between the two tools. To help in this situation, we try to support the fusion main module by removing outliers and generally by smoothing decisions of individual tools in situations of agreement and by updating decisions quickly when a changed event is indicated. This necessitates the need to provide a system status recognition tool which allows the two different strategies to work side by side.

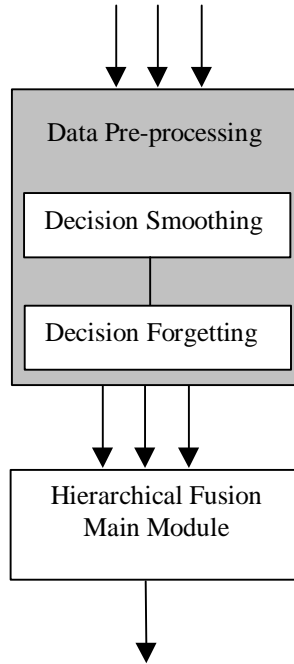


Figure 1: Decision Smoothing and Decision Forgetting as a Fusion Pre-Processing Module

We implement the concept of decision smoothing via an exponential averaging time series filter with adaptive smoothing parameter. The filter is expressed as

$$x(k) = \alpha x(k-1) + (1-\alpha)y(k)$$

where

$x(k)$ and $x(k-1)$ are the filtered value at time k and $k-1$, respectively

α is the smoothing parameter

$y(k)$ is the new incoming value

Changes of the smoothing parameter will allow weeding out noise and outliers when no fault has occurred but to react quickly to changes from one event to another. Therefore, if everything is calm, the smoothing parameter is large (weed out noise); if something is happening, the strategy is to be more cautious, and to reduce the smoothing parameter value. The reduction is accomplished with the help of an intermittency factor. This factor can be interpreted as a sentinel which monitors system changes. The sentinel checks if the status of a decision changes and keeps track of these changes. First, a change counter establishes whether there is a change of opinion compared to the last one. If the change counter is greater than zero, an intermittency factor is computed dividing the change counter by some user defined sentinel window. The sentinel window is a running window that is small initially to allow operation even when there is only one measurement. This implies that initially, the parameter will be more receptive to changes. When more values become available, the window increases and the decision smoothing becomes more temperate. This also means that in general, more information is better under this paradigm because longer and more refined smoothing can take place. The smoothing parameter α is calculated by

$$\alpha = (1 - \text{int ermittency})^{\text{constriction_exponent}}$$

where the `constriction_exponent` scales α and is a value > 1

Additionally, alpha is bounded between a lower and upper value to ensure that the filter becomes neither too sluggish nor too responsive. The filtered decision is then calculate by the exponential filter

$$\text{filtered_decision} = \alpha \cdot \text{filtered_decision} + (1-\alpha) \cdot \text{new_decision}$$

Finally, a forgetting factor is employed that has the effect of reducing the sensitivity of the parameter when no changes have been observed for a while thus allowing the algorithm to settle to its smoothing task. The forgetting factor is a value less than 1. It is computed over time as

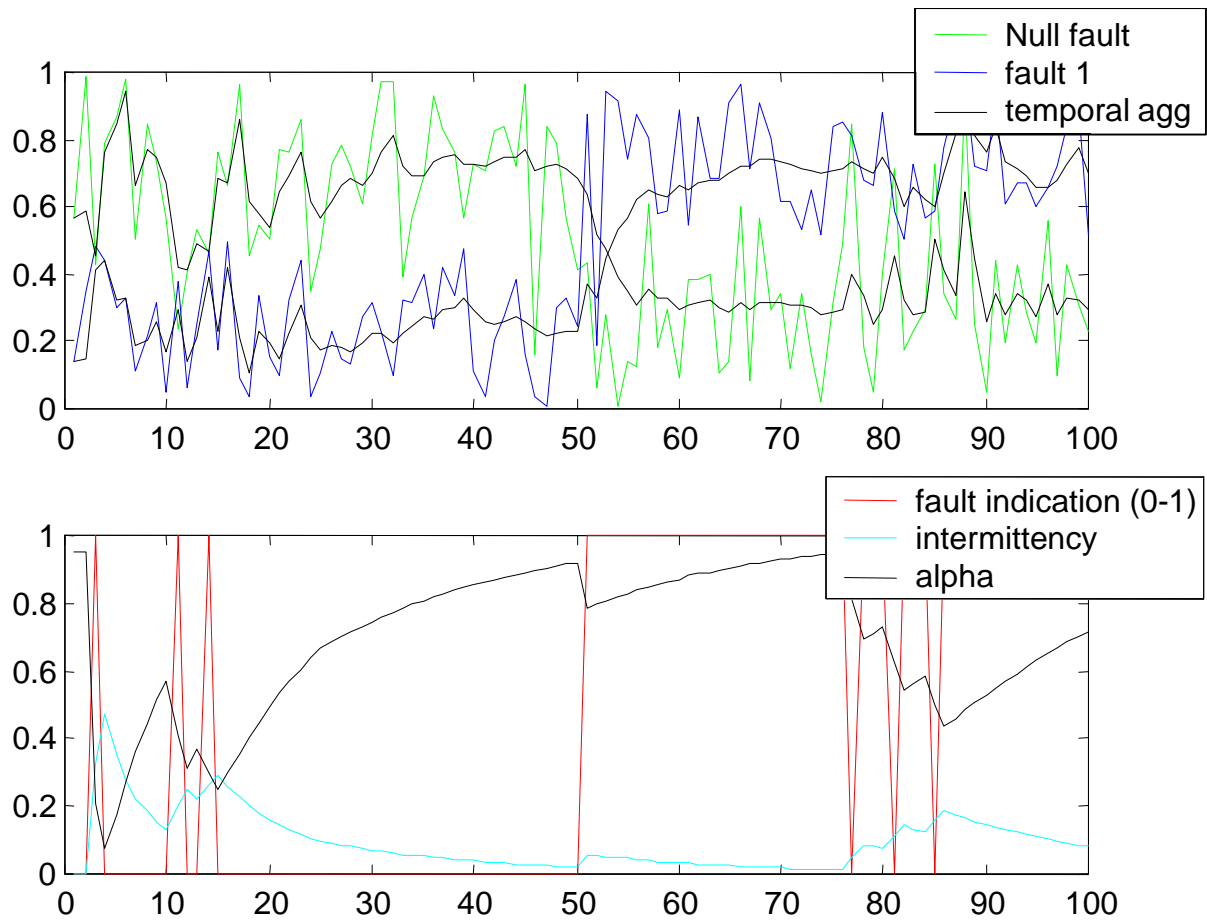
$$\text{change_count} = \text{change_count} \cdot \text{forgetting_factor}$$

Put together, the algorithm is as follows:

```
BEGIN
  IF there is a change in the status of a tool THEN
    Increase the change_count
  END IF
  IF change_count > 0 THEN
    intermittency=change_count/window
  ENDIF
  alpha=max(lower_bound,min(upper_bound,(1-min(1,intermittency))^constriction_exponent))
  filtered_decision=filtered_decision*alpha+new_decision*(1-alpha)
  change_count=change_count*forgetting_factor
END
```

Figure 2 shows an example for the averaging where two decision states (null fault and fault 1) were simulated with a change of decision state at step 50. An underlying assumption is that decisions are scaled between 0 and 1 and are interpreted as confidence or likelihood for a fault³. One can see the initial responsiveness of alpha in Figure 2(b) and also a more sensitive value during the change of fault states. The smoothing parameter mirrors the intermittency's behavior. Also to be seen in Figure 2(b) is the fault indicator which is the classification result. As can be seen, the individual classifiers have a number of misclassifications if the likelihood of 0.5 is used as a classification threshold or when the "maximum wins" strategy is used. However, the smoothed values have a greatly reduced misclassification with only 2 misclassified decisions close to the crossover point where the fault state changes. Clearly visible in Figure 2(a) is how the averaging weeds out noise and outliers and responds quickly to the change of fault state.

Figure 2: Exponential averaging of information with adaptive smoothing component



3. METHODOLOGY FOR INFORMATION FORGETTING

So far we have discussed how to reduce variations in decisions during decision agreement and how the smoothing is automatically turned off when the decisions disagree. However, we still need a mechanism to deal with the disagreement where one tool makes a decision d_1 about a system state at time t_1 and another tool comes to a different conclusion d_2 at a later time t_2 . It is then necessary to account for the fact that d_2 may have occurred between t_1 and t_2 . We postulate that the later decision needs to be given more weight in case of decision disagreement to account for the possibility of event d_2 . The question is how much more weight d_2 (or how much less weight d_1) should have. We further propose that the discounting is a function of time passed between d_1 at time t_1 and d_2 at time t_2 . The reason is that there is a higher possibility for event d_2 to occur when the period t_2-t_1 is larger and vice versa. In addition, the tools must have information about their a priori performance. Again, we assume that the decisions are scaled between 0 and 1⁴ and propose to change the forgetting factor as the confidence value increases. Figure 3 shows the forgetting factor as a function of the confidence. The forgetting factor is close to 1 for small tool confidence and rises to 1.1 as tool confidence increases. The particular slope and upper saturation is system dependent.

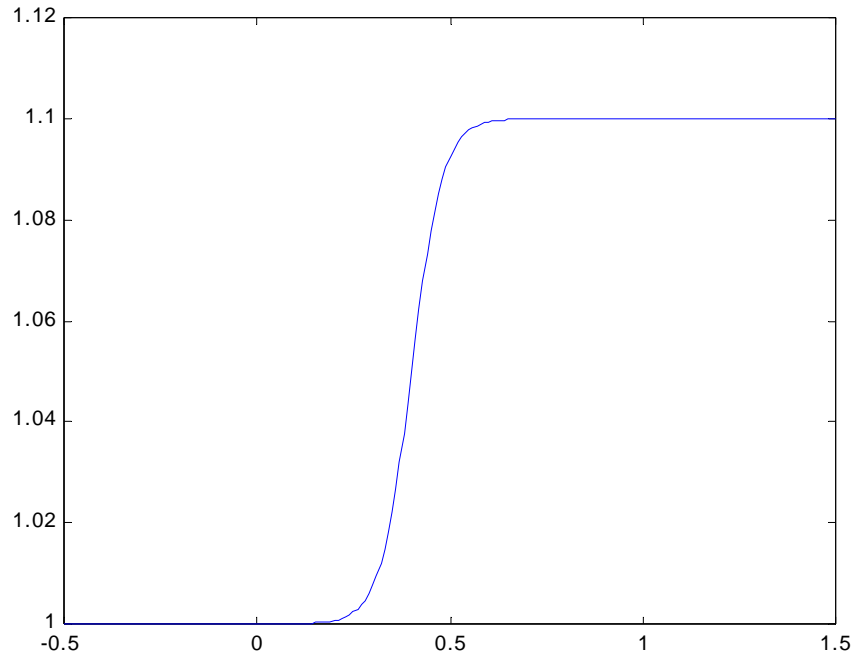


Figure 3: Fading exponent as a function of confidence value

The idea of decision forgetting is to discount information as it ages when tools disagree at different times (and no new update of both tools can be obtained). We force the older information to “fade” as a function of time passed. If the conditions outlined are met, the decision will be subjected to a forgetting exponent. The forgetting exponent must be tuned to the system at hand..

The algorithm is as follows:

```

BEGIN
  IF decisions of tools not coherent AND time since last information sampling not the same THEN
    FOR EACH instance of the time difference
      new_decision=old_decision^fading_exponent
      old_decision=new_decision
    END FOR LOOP
  ENDIF
END

```

Figure 4 shows how the information of faults at two different confidence levels ($\text{confidence}_{\text{upper}}(1)=0.8$ and $\text{confidence}_{\text{lower}}(1)=0.2$) is discounted over time

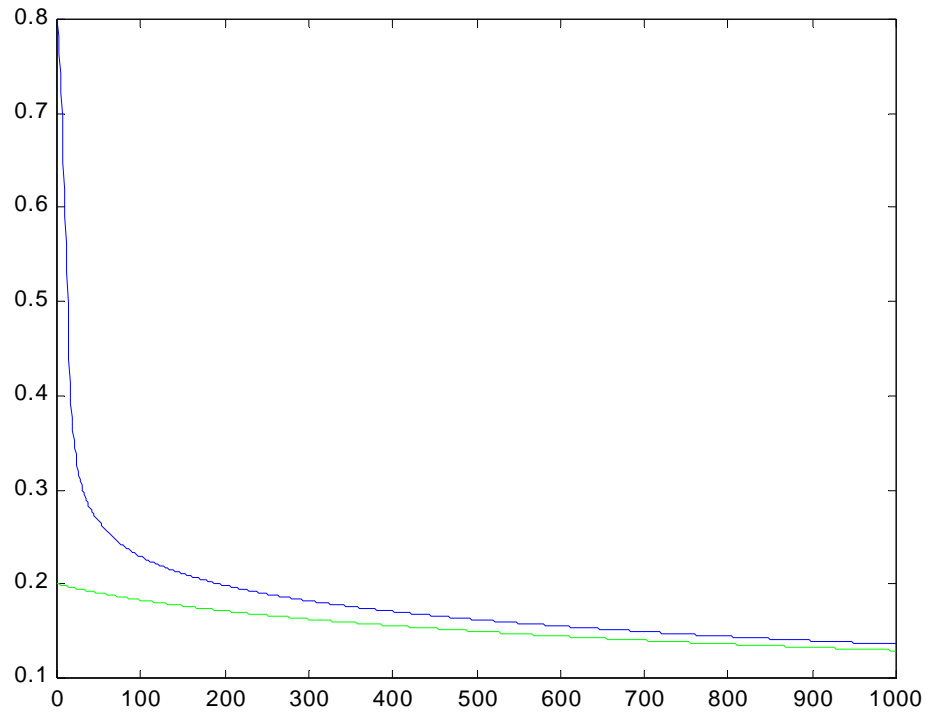


Figure 4: Fading over time

4. APPLICATION TO DIAGNOSTIC INFORMATION FUSION

We applied the principles of decision smoothing and information forgetting to the diagnostic information fusion task for a gas turbine engine. The main goal 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)⁴. 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.

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. The fusion effort is one part of an overall project that addresses these problems through the design and test of a condition-based Intelligent Maintenance Advisor for Turbine Engines (IMATE) system⁴. The overall goal of the information fusion was 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.

Several diagnostic tools (Model based diagnostic tools, neural nets, etc.) as well as non-diagnostic information sources (vibration, fault codes, etc.) were selected for information aggregation. The functional architecture of IMATE is shown in Figure 5. We carried out extensive Monte Carlo simulations to validate the tools as well as the fusion module and currently are instrumenting rig tests. The overall classification task of the integrated main fusion module with the pre-processing tool (decision smoothing and decision forgetting) improved the result by about two orders of magnitude compared to any individual tool alone from an average of 10% misclassifications to 0.1% misclassifications. The effect of the pre-processing modules was estimated to contribute about 10% performance improvement to the overall fusion scheme.

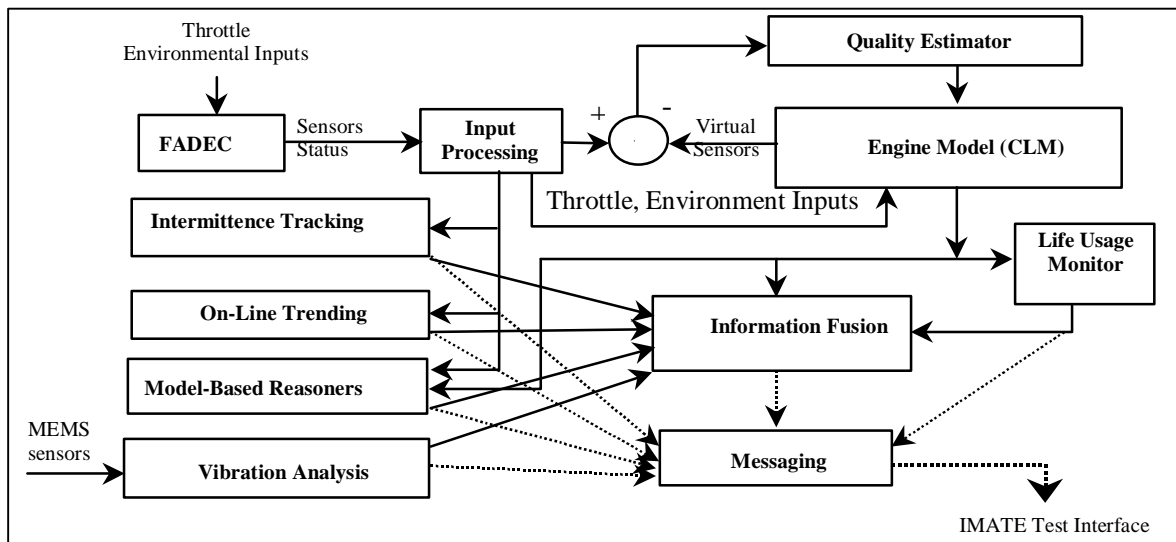


Figure 5: IMATE Functional Architecture⁵

5. SUMMARY AND CONCLUSIONS

We introduced a system for decision forgetting and decision smoothing. Both are considered pre-processing tools for a fusion main module. These pre-processing tools allow partial conflict resolution and improved fusion performance through the concept of averaging of information when no conflicts exist and aging of information where conflicts are present. We tackle in particular situations where information arrives at different times. Disagreement in these situations can be the result of either poor tool performance or the result of correct performance with an event occurring in the meantime. A system status sentinel determines whether smoothing can commence or whether increased responsiveness to changed states is required. The two tools have been tested in a diagnostic setting and contributed substantially to a reduction in misclassifications.

We view these techniques as applicable not only to decision fusion. Rather, they are equally useful for processing on the feature of data level. In this paper, we considered the two techniques decision smoothing and decision forgetting as pre-processing techniques, mostly because we designed a strictly hierarchical fusion main module. If a hierarchical model is not used, decision smoothing and decision forgetting can also be part of an integrated fusion tool.

6. ACKNOWLEDGMENTS

This research was in part supported by DARPA project MDA 972-98-3-0002. The author also gratefully acknowledges the comments of Malcolm Ashby, Kiyong Chung, Vivek Badami, Michael Krok, and Hunt Sutherland.

7. REFERENCES

1. P. Bonissone, Y. Chen, K. Goebel, P. Khedkar, "Hybrid Soft Computing Systems: Industrial and Commercial Applications", *Proceedings of the IEEE*, Sept 1999, Vol 87, No 9, pp. 1641-1667, 1999.
2. K. Goebel, A. Agogino, Fuzzy Sensor Fusion for Gas Turbine Power Plants, Proceedings of SPIE, Sensor Fusion: Architectures, Algorithms, and Applications III, pp. 52-61, 1999.
3. 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.
4. K. Goebel, M. Krok, H. Sutherland, *Diagnostic Information Fusion: Requirements Flowdown and Interface Issues*, Proceedings of the IEEE Aerosense Conference, 2000.
5. M. Ashby, W. Scheuren, *Intelligent Maintenance Advisor for Turbine Engines (IMATE)*, Proceedings of the IEEE Aerospace Conference, 2000.