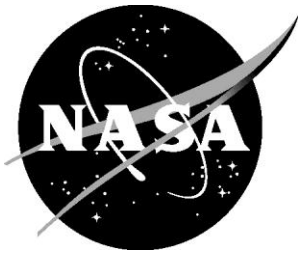
NASA/CR–2013-217972



# Vehicle Integrated Prognostic Reasoner (VIPR) Final Report

*Raj Bharadwaj, Dinkar Mylaraswamy, and Dennis Cornhill*
*Honeywell International, Inc., Golden Valley, Minnesota*

*Gautam Biswas, Xenofon Koutsoukos, and Daniel Mack*
*Vanderbilt University, Nashville, Tennessee*

March 2013

## NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

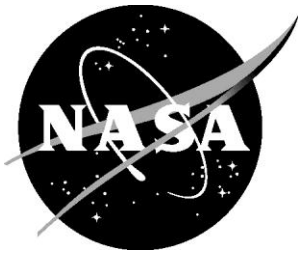
- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 443-757-5803

- Phone the NASA STI Information Desk at 443-757-5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/CR–2013-217972



# Vehicle Integrated Prognostic Reasoner (VIPR) Final Report

*Raj Bharadwaj, Dinkar Mylaraswamy, and Dennis Cornhill*
*Honeywell International, Inc., Golden Valley, Minnesota*

*Gautam Biswas, Xenofon, Koutsoukos, and Daniel Mack*
*Vanderbilt University, Nashville, Tennessee*

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NNL09AD44T

March 2013

## Acknowledgments

We would like to thank the Honeywell and Vanderbilt teams for conducting the work under this program. In particular, we wish to acknowledge Craig Schimmel, Howie Weibold, Darryl Busch, Lee Graba, Wayne Schultz, Gordy Nagel, Mick Young and George Hadden for their many contributions. We would also like to thank Hal Voges for his technical feedback and support, Craig Goodrich for helping us with contract related issues, and Savit Boyd for her patience and her lessons on program management.



Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

# Table of Contents

# Table of Figures

# Table of Tables

# 1. Executive Summary

A systems view is necessary to detect, diagnose, predict, and mitigate adverse events during the flight of an aircraft. While most aircraft subsystems look for simple threshold exceedances and report them to a central maintenance computer, the vehicle integrated prognostic reasoner (VIPR) pro-actively generates evidence and takes an active role in aircraft-level health assessment. Establishing the technical feasibility and a design trade-space for this next-generation vehicle-level reasoning system (VLRS) is the focus of our work.

In the first year of this program, we laid the technical foundations for a next generation Vehicle Integrated Prognostic Reasoner (VIPR) [1]. Specifically we: documented user requirements and illustrated the advanced features using scripted scenarios; formulated a three-tiered reasoning framework that could be implemented as extensions to existing aircraft VLRS; and identified historical data sets from a regional airline operator that could be used to show the effectiveness of the advanced reasoning system.

Year two focused on detailed design of VIPR, implementing and testing the necessary software code, and development of necessary prognostic monitors from the historic regional airline data set. In the second year we: implemented a prototype for vehicle integrated prognostic reasoning (VIPR) whose operations were defined in Year 1; documented the impact of data mining combined with an onboard VIPR to avoid four historically recorded safety incidents; and developed an approach to systematically study VIPR metrics such as accuracy and cost using a simulation testbed. We recommend hardware should be inserted into the demonstration system so that we can evaluate VIPR performance in more realistic environments.

Following successful completion of Years one and two, the third year focused on: a hardware-in-the-loop demonstration of the VIPR reasoner with Honeywell's LaserRef VI Inertial Reference Unit; integration of heterogeneous evidence such as on/off built-in-tests; and condition indicator trending from three aircraft subsystems (engines, APU, avionics) to establish technical feasibility of a next generation prognostic reasoner. We also defined the anomaly detection function as an extension of the aircraft condition monitoring and, using select examples, demonstrated the utility of anomaly detection for identifying precursors that may have evolve to a safety incident.

# 2. VIPR Accomplishments and Significance

Significant accomplishments for the VIPR program include:

1. Prognostic reasoning. We established the technical feasibility of a Bayesian prognostic reasoner for onboard detection of adverse safety events. The demonstration included working prototype software and design documentation for implementing the algorithms as expansion of the central maintenance computer functions. The results show the proactive role of VIPR in detecting three safety incident precursors from the regional airline historic data; and its ability in detecting a potential cascading avionics fault based on hardware demonstration with an inertial reference system.

a. Standardized definition and interpretation of advanced monitors that enable member systems to express complex evidence formats and hence enable prognostic reasoning.
b. A three-tiered architecture for deploying VIPR functions such as monitor generation, cascade handling, fault hypothesis generation, temporal reasoning, messaging and suppression.
c. A Bayesian framework for generating fault hypotheses that represent the aircraft health state by explaining all observed forms of evidence. It uses a noisy-or approximation along with merging and splitting rules to provide an upper bound on computational complexity without making a single-fault assumption.

2. Semi-supervised learning loop. The system reference model within VIPR encodes aircraft-specific data such as failure mode definitions as well as evidence that can be generated and their relationships. While this member-system, supplier-provided information drives onboard diagnostics and prognostics of known problems, detection of emergent/unknown events that may impact safety is enabled by a semi-supervised learning loop. We established the technical feasibility of such a learning loop using:
   a. Tree-augmented, network-based supervised learning. This supervised learning method generates an evidence-failure mode map when trained with annotated fault cases. This map is readily incorporated in the VIPR reference model as (1) new evidence nodes, (2) updating selective arc information in the evidence-failure mode bipartite graph.
   b. An anomaly detection function for generating outliers based on information entropy in time series parametric sensor values. We demonstrated a Kolmogorov-complexity metric to detect these outliers and download contextual bad actors data for expert analysis. Some of these outliers presented interesting cases for offline expert analysis as pre-cursors of incipient faults.

3. A design trade space. Establishing the design trade space is as important as demonstrating technical feasibility. Our focus was to design the VIPR functions as an expansion of existing Aircraft Diagnostics and Maintenance Systems (ADMS).
   a. We recommended the way in which prognostic functions can be implemented as extensions to the central maintenance computer function—an ARINC 624 based messaging protocol for implementing a distributed VLRS within existing aircraft computation and communication constraints, anomaly detection function as an expansion of the aircraft condition monitoring function.
   b. Using a combination of historic regional airline data, simulation and limited aircraft hardware, we demonstrated a base set of failure modes that VIPR can detect with relatively high degree of accuracy and confidence. While the confidence is a strict function of the number of fault cases, a handful of scenarios have a significant impact on safety and condition-based maintenance. Table 1 lists these impacts and a first step towards developing favorable cost-benefit trade studies.

4. Performance metrics. Advanced functions within VIPR such as generation of advanced monitors and more complex inference rules require increased communication and computation bandwidth. We developed a systematic approach for making these trades. Using designed experiments, we demonstrated that the extended ARINC 624 messaging protocol with VIPR reasoner can achieve lossless performance when the inferencing is distributed across available computation nodes within an aircraft. Using a standard communication model, we concluded that a 10kB/second bandwidth is sufficient to achieve the 10-second latency goal. Our simulation studies also showed a detection accuracy of 98% with one initiating fault and 76% accuracy when we had two or more initiating

faults. While this is a significant improvement over the state of the art (which assumes only one fault at any given time), there is definitely room for VIPR performance improvement.

5. Onboard inferencing. Simple on/off threshold-crossing evidence is insufficient for prognostic reasoning. VIPR standardized three formats to express more complex and heterogeneous forms of evidence such as: noisy evidence, prognostic vectors, and condition indicators. We extended the naïve Bayesian framework to handle uncertainty in evidence, causal cascades, and trend information. Standardization of the evidence enabled us to make these inference rules aircraft agnostic.
6. Prototype implementation of VIPR to establish the technical feasibility of these features as extensions to the central maintenance computer function (CMCF) on legacy Boeing, Bombardier, and Dassault aircraft. This prototype provides a cost-effective pathway for realizing VIPR as an extension to existing aircraft hardware and software.
7. Hardware integrated demonstration. The high quality of the LaserRef VI self-diagnostics allowed us to use the device's existing output for input to the VIPR software; consequently, integrating the LaserRef VI with VIPR added no overhead to the its operation. We integrated a test version of the Honeywell LaserRef VI Inertial Reference Unit and other aircraft fault evidence streams with the VIPR reasoner. We injected four faults into the LaserRef VI, and these faults were detected by the reasoner, three at the LRU level and one at the Area level.

Our program shows that VIPR has a favorable impact on safety and condition-based maintenance. This conclusion is summarized in Table 1. We envision VIPR primarily as a software enhancement, although the role of new sensors greatly assists fault detection and isolation.

**Table 1. Significance of VIPR on safety**

| A/C Systems | VIPR Accomplishments | Safety Impact | CBM Impact |
|---|---|---|---|
| Engines, APU, Bleed, Fuel Delivery | • Active query during engine start to generate a richer evidence set. | • Detect incipient faults in fuel control and turbine blade erosion that progress as uncommanded engine shutdown. | • For the fuel controller fault, avoid an unscheduled engine removal. |
| | | • Using available data demonstrated a 20-30 flights prognostic window for fuel control fault (overspeed shutdown) and 10—15 flight prognostic window for blade tip break (vibration shutdown) | • For the turbine blade break, avoid secondary damage and plan a scheduled borescope inspection. |
| | • Prognostic reasoning on richer evidence set discovered via data mining. | • Detect fuel manifold problems that may manifest as false engine-on-fire alerts. | • Reduce diagnostic trouble shooting by focusing on the manifold and avoid false engine removals. |
| | • Active query during APU start of the main engine. | | • Schedule condition-based borescope inspection and usage based APU removals. |

| A/C Systems | VIPR Accomplishments | Safety Impact | CBM Impact |
|---|---|---|---|
| | • Reasoning across engine, APU, bleed, and fuel delivery subsystems. | • Provide better situation awareness by identifying root cause and preventing nuisance messages. | • Differentiate among faults that manifest within the bleed duct, APU and engine bleed. |
| **Avionics (Lref6 IRU)** | • Utilized condition indicator-based on system knowledge to detect ADS fault. | • Fault detection of in-range air data faults and prevention of fault cascade. | |
| **Actuators (EMA)** | • Utilized condition indicator-based evidence developed under AFRL program FA8650-08-D-7803 0001 in the reasoning process. | • In addition to fault detection, VIPR can provide parametric values to enable control reconfiguration which can prevent loss of control. | |
| **Avionics (power supply and GPS)** | • Utilized condition indicator-based evidence developed under NASA program NNA08BA45C in the reasoning process. | • Some of these problems if undetected caused stall and overspeed warnings which in turn lead to undesirable pitch up and pitch down autopilot commands. | • Fault isolation between power supply and GPS. |
| **Anomaly detection using aircraft data** | • Developed an approach to hierarchical anomaly detection for the aircraft fleet. | • Provide the mechanism to capture the bad actor data related to anomalous events with unknown/unspecified safety significance.<br>• Anomalous flap settings could indicate multiple types of safety problems.<br>• The incorrect fuel level sensor could cause an incorrect alert and nuisance alarm to the flight crew. | • Provide mechanism for capturing data corresponding to the anomalous events that can be matured to add missing fault conditions to the reference model. |
| | • Demonstrated the feasibility of extending the above approach to on-board anomaly detection. | • Foundation for a smart ACMF function. Could help to capture data for unknown-unknown conditions. | • Foundation for a smart ACMF function. Could help to capture data for unknown-unknown conditions. |

# 3. Introduction and Background

An important challenge for aviation safety is safeguarding against system and component failures and malfunctions. Faults in one aircraft subsystem can propagate to other subsystems, and multiple faults could interact. The overall objective of VIPR is to detect adverse events that may be occurring in the vehicle that will lead to a safety incident in the near future [2].

The four functional modules of VIPR are: onboard inference engine, system reference model, learning loop and communication interfaces. The modules are numbered 1 through 4 in Figure 1 and described below.

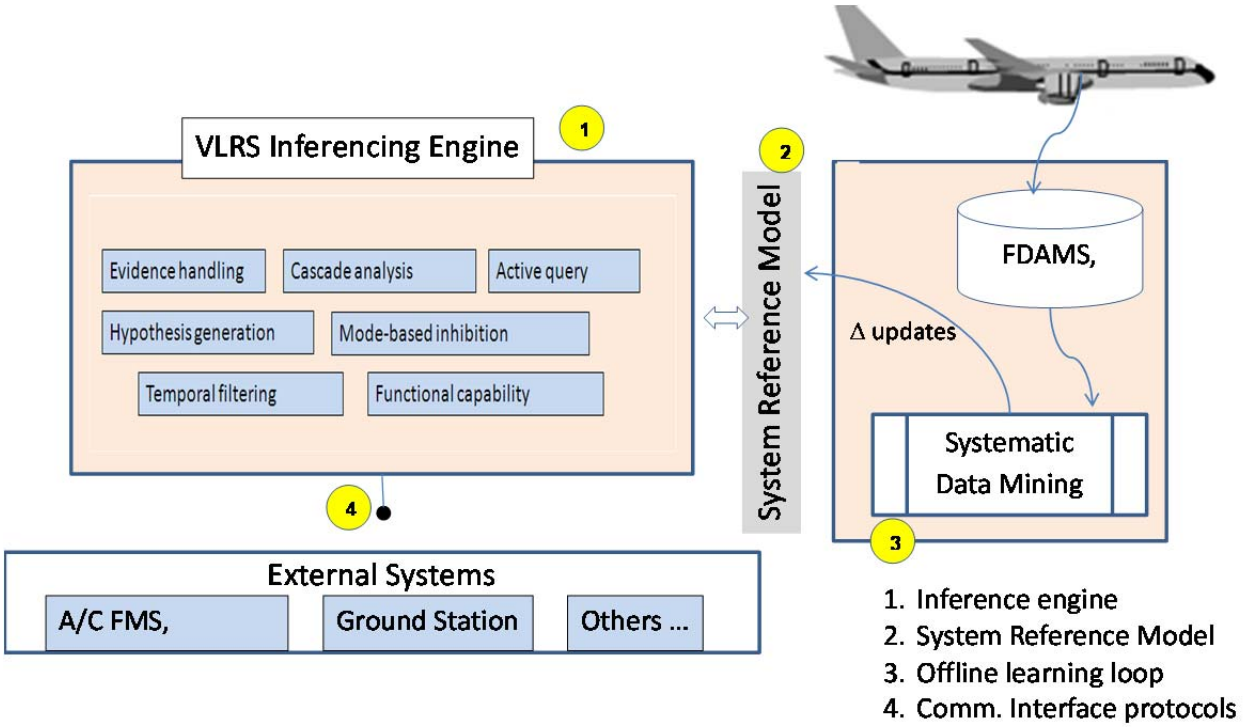



**Figure 1. Functional modules within VIPR**

1. Inferencing engine. This module uses health evidence generated from all components and subsystems within a vehicle (such as an aircraft) to produce the current diagnostic state or predict the future evolution of a fault. It produces the most plausible explanation for all the symptoms exhibited by various components; creates new hypotheses for tracking multiple faults; and deletes hypotheses that may have weak or no supporting evidence.
2. System reference model. The relationship necessary for inferencing is separated as a static system reference model. This partitioning allows the same inference engine software code to be reused on multiple vehicles and minimizes certification and qualification costs for deploying VIPR onboard an aircraft. The system reference model is an aircraft-loadable software module that describes the relationship between evidence generated at the component or subsystem level and failure modes that can be mapped to specific maintenance or corrective actions.
3. Data mining and learning loop. Fleet modeling, data mining, and knowledge discovery methods working on historical data can detect anomalies and precursors to critical failure modes.

Discovering new patterns and updating old relationships in the system reference model can continually improve aircraft safety to a higher level. Information from this learning loop, which results in a $\Delta$-change in the reference model, enables VLRS to provide an accurate health assessment of a component, subsystem, or system and support condition-based equipment maintenance and replacement.

4. Communication interfaces. VIPR is designed to take a system-wide view of the adverse event detection problem. While the input interfaces define how VLRS receives health information from various member components, the output interface defines how it communicates its outputs to the flight crew (displays), ground maintainer (ground station), or a flight management system for automatic fault accommodation.

# 4. Approach

VIPR incorporates new concepts and technologies that reason about data captured from multiple subsystems to detect a potential adverse event, diagnose its cause, and predict the effect of that event on aircraft safety. Validating these concepts early is important not only to increase the likelihood of technical success, but also for early adoption within the community. Broadly speaking, our activities were organized in three phases that correspond roughly to Year 1, Year 2, and Year 3 of the program.

In Year 1, we gathered user requirements and then used an animated concept of operations (ConOps) model to visualize the VIPR operation. This work helped us gather customer expectations and systems requirements and make course corrections. In Year 2, we implemented a prototype VLRS that can be deployed as an extension to existing aircraft hardware and software; demonstrated how VIPR helped to avoid aviation safety incidents; and outlined a systematic approach for cost-benefit tradeoff studies. In Year 3, we integrated the LaserRef VI IRU as a hardware member system in a VIPR-demostrated system to evaluate VIPR performance in more realistic environments. We also explored the use of anomaly detection to augument the event data gathering of the auircraft condition maintenance function.

We realized the VIPR objectives by performing these major tasks:

1. VIPR detailed design. VIPR defines three new formats to express more complex and heterogeneous forms of evidence (besides the three that exist today). These formats are: noisy diagnostic monitors, prognostic monitors, and condition indicators. In this task, we described specifications associated with each type of evidence. We documented the extension of the naïve Bayesian framework to reason with this evidence in a detailed design report and presented the theory of VIPR during a technical interchange meeting.

2. VIPR implementation. In this task, we encoded advanced VIPR functions to illustrate its ability to handle heterogeneous evidence, hypothesize and disambiguate competing fault conditions, and fuse multiple prognostic and trend evidence. Our secondary objective was to simulate VIPR in a more relevant aircraft environment by using a common library to support ARINC 624-based messages. Prognostics implementation in the VIPR demo showed the proactive detection of three safety incident precursors from the regional airline historic data.

3. Data mining:

a. Safety case study analysis. Using publically available safety reports, we identified ten cases that could be analyzed with available historical data. In each case, a combination of supervised and unsupervised learning, we discovered precusors that manifested across subsystems such as engines, navigation, actuators, bleed, APU, and fuel delivery. Our objective was to incorporate this discovered knowledge within VIPR and compare the outputs. The before and after results illustrate the net safety impact.

b. Anomaly detection. Using the regional airline data, we developed anomaly detection approach for both off-line and on-line anomaly detection.

4. Performance metrics and cost benefit analysis. While historical, recorded events can illustrate effectiveness, they are insufficient for verficaition and validation. In this task, we developed an approach to systematically study VIPR performance using a simulation testbed. Using a Monte Carlo analysis, we documented summary statistics such as communication and computation cost to support the advanced functions within VIPR.

5. Integrated demo. We demonstrated VIPR with LaserRef VI hardware as a member system. We simulated LaserRef VI faults including software and sensor faults. The sensor faults included both internal sensors such as accelerometers and gyros, and external sensors such as air data sensors. We demonstrated the VIPR prototype with a sample aircraft reference model. This model included representative failure modes and evidence generated from key subsystems such as propulsion, navigation, 24V power supply, GPS receiver, APU, fuel manifold, and an electro-mechanical actuator. The objective of this activity was to provide a simple graphical tool with which the user could introduce one or more failures and visualize VIPR's internal working.

# 5. Progress Summary

Table 2 summarizes the key VIPR proposal concepts, Year 1-3 status, and progress.

**Table 2. Accomplishment summary with respect to proposal concepts**

| Proposal Elements | Year 1 Status | Year 2 Status | Year 3 Status |
|---|---|---|---|
| Fault condition construct and operations | Concept of Operations document described in CDRL 4.1.04. | Naïve Bayesian framework extended to handle heterogeneous evidence. Internal peer review of theory, available to NASA on request. Software implemented and tested. | |

| Proposal Elements | Year 1 Status | Year 2 Status | Year 3 Status |
|---|---|---|---|
| System Reference Model | Definitions and requirements described in User Requirements, CDRL 4.1.05. | Created a reference model to include representative failure modes and evidence generated from subsystems such as propulsion, navigation, 24V power supply, GPS receiver, APU, fuel manifold, and an electro-mechanical actuator. | Created a reference model to include representative failure modes and evidence generated from subsystems such as LaserRef VI IRU, propulsion, navigation, 24V power supply, GPS receiver, APU, fuel manifold, and an electro-mechanical actuator. |
| Monitor and evidence abstraction | Recommendations captured in CDRL 4.1.02. | Completed specification and implementation for three new forms of monitors (noisy diagnostic monitor, prognostic monitor, and condition indicators). | Implemented monitors, including LaserRef VI hardware. Imple-mented the trend monitor for the propulsion engine. Completed exercising all forms of monitors (noisy diagnostic monitor, prognostic monitor, and condition indicators). |
| Tiered and distributed architecture | ARINC 624-based messaging protocol to support distributed reasoning defined in CDRL 4.1.03. | Implemented the message encoding and decoding protocols. Demonstrated the protocol over a TCP/IP transport layer. | Demonstrated the tiered VIPR architecture using data from multiple systems: LaserRef VI hardware, propulsion (regional airline data), APU, actuator and power supply (simulated). |
| SMART Process | Steps 1, 2 & 3: User requirements, animated ConOps & architecture flow. | Steps 4 & 5: detailed design and implementation. | Step 6 & 7: Validation of concepts through demonstration and simulation. |
| Data mining and learning loop | Identified ten incidents recorded in the Aviation Safety Information Analysis and Sharing (ASIAS) database and matching operational data. | Applied tree-augmented network for discovering precursors. Analyzed four case studies to tune existing monitor thresholds and discovered new monitors. Demonstrated that newly discovered knowledge can be used to selectively update probability values in an existing system reference model. Delivered as CDRL 4.2.01. | Applied anomaly detection to regional airline data. Discovered evidence, such as high energy take-off and abnormal engine temperatures, of future failures. Anomaly detection can be incorporated in the ACMF. Results are documented in CDRL 4.2.08 and CDRL 4.3.03. |
| Metrics: performance and cost | Defined a set of performance and cost CDRL 4.1.07. | Implemented a simulator to inject failure modes and generate noisy heterogeneous evidence. Developed Monte Carlo analysis software to run this simulator with VIPR and collect performance accuracy and cost metrics. Results documented in CDRL 4.2.05 | Updated the results with metrics from HIL. Results are documented in CDRL 4.3.05. |

| Proposal Elements | Year 1 Status | Year 2 Status | Year 3 Status |
|---|---|---|---|
| Metrics: impact on safety | Scripted scenarios to illustrate how VIPR would address select safety incidents. | Demonstrated how an onboard VIPR can detect an incipient problem before it manifests as a safety incident (such as in-flight engine shutdown, engine on fire and hydraulic fluid leaks adverse events). Results documented in CDRL 4.2.05. | Demonstrated how a VIPR systems approach can detect and avoid fault cascade. Results for an air data fault are documented in CDRL 4.3.01-4.3.05. |

## 5.1 VIPR Detail Design – Functional Decomposition

The concepts of failure modes and evidence are central to the VIPR design. Failure mode is an abstract entity that can be mapped to specific corrective or mitigation actions; for example, a failure mode could map one-to-one with physical failures like those defined by the component manufacturer or a condition that has a well-defined corrective action (such as remove and replace) defined in the aircraft maintenance procedures. Evidence, or symptom, is also an abstract entity and represents observations regarding the aircraft. Figure 2 is an example of a bipartite graph that lists the failure modes and evidence for an engine subsystem.

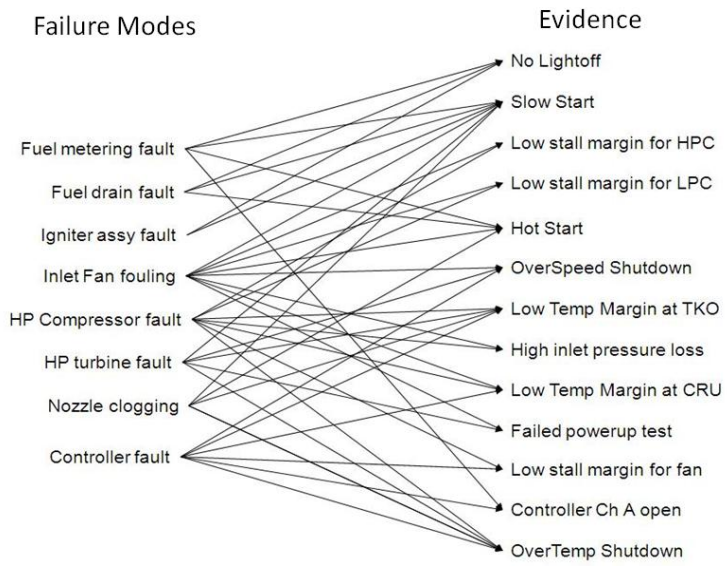



**Figure 2. Failure modes mapped to evidence for an engine subsystem**

The i'th evidence within VIPR denoted as $e_i$ has three states: indict, exonerate, and unknown. In the indict state, evidence signifies the presence of one or more failure modes. The exonerate state signifies the absence of one or more failure modes. The unknown state indicates that we have not yet made this observation. Evidence is generated by analyzing sensor measurements. A monitor is an expression of evidence. In the current state of the art, member systems within an aircraft provide on/off or pass/fail indications.

Typically, the on/fail indicator implies an indict state, and off/pass implies an exonerate state for the i'th evidence. This information is insufficient to support prognostic reasoning, so we expanded the monitor set. While the inference engine should not depend on how the monitor was generated, it must have a uniform interpretation of monitors generated from various aircraft subsystems; hence standardization of the monitor definition was a critical step in our design. Our design needed to provide enough flexibility for member systems to express heterogeneous forms of evidence besides the simple on/off format. To support prognostic reasoning, VIPR defines the new forms of monitors: (1) diagnostic

monitor, which is a probabilistic indication for the state of the i'th evidence; (2) prognostic monitor, which allows the member system to express a probabilistic state for the i'th evidence in future; and (3) condition indicators-based monitor, which provides a time series signal together with an indict/exonerate region.

The i'th evidence can have multiple monitors—some of which can be diagnostic, some prognostic, and some condition indicators. Further, monitors can be issued at different timestamps. Each of these monitors assigns a probability distribution to the three states of the evidence. The **Evidence handling or monitor function** within VIPR combines the information provided by heterogeneous monitors and assigns a combined probability for the i'th evidence state. Reference [1] provides details of this fusion approach.

A failure mode manifests as one or more pieces of evidence; a given piece of evidence can be triggered by more than one failure mode. The **fault isolation** function analyzes all the monitor observations to generate fault condition (FC), which *explains* all observed monitors. More specifically, this function takes three actions:

- First, it generates a fault condition that can explain some of these observed monitors. More than one fault condition may be needed to explain all observed monitors. New failure modes may be added or deleted to an existing fault condition to modify the assertions about the current state of the subsystem. Eventually, it deletes fault conditions whose monitors are no longer in the system.
- Second, it calculates the normalized strength of various fault hypotheses asserted by a given fault condition. This value must be derived using the current and future values of the evidence state as reported by various monitors, which provides a forward-looking assertion that a failure mode may be occurring in the system—this is VIPR's prognostic feature.
- Third, it applies several tests such that weak fault hypotheses can be rejected and strong ones can be reported as the prevailing or future state of the subsystem. Details of a fault condition and its operations are described in [1]. The fault isolation function is also called the **hypothesis handling** function since the fault condition encodes VIPR's prevailing hypothesis about the aircraft health state.

"Cascade" refers to interactions between subsystems. From VIPR's point of view, it describes failure modes in one subsystem that can trigger a monitor in another subsystem, as well as failure modes in one subsystem that can cause a failure mode to occur in another subsystem. The **cascade function** within VIPR uses these subsystem interactions to modify existing FCs generated by individual subsystems. VIPR supports two forms of cascade analysis: fault and symptom cascade. The difference between them is best explained using simple examples.

> *A failure mode in the 24V power supply could trigger monitors in the radio subsystem (symptom cascade). A broken blade failure mode in the APU can cause an erosion failure mode in the propulsion engine (fault cascade).*

"Suppression" refers to filtering out inconsequential information that may distract the flight crew or a maintainer. If filtering does not occur, the crew may address a secondary problem rather than the

primary one. The **suppression function** within VIPR uses this information to inhibit these monitors. The need for suppression arises when two or more subsystems share the same resource, such as fuel or a power supply. In these cases, once the fault isolation function establishes an FC that contains a failure mode for the shared resource, then we need to suppress the monitors that may have triggered in the other subsystems that are sharing this common resource. This example explains the suppression function:

> *In the previous example, a leaking capacitor in the power supply was sending a high-frequency ripple current and causing a crackling noise in the radio. Once we establish a "leaking capacitor" fault hypothesis within the 24V power supply subsystem, we would like to suppress the ''crackling noise'' monitor so that the maintainer/crew can focus on the root problem.*

The term "active query" refers to an intelligent data collection function. As we saw earlier, monitors allow us to express the current and future state of evidence elements. Linking this evidence to failure modes enables us to isolate, perform cascade analysis, and suppress them. What if we are yet to discover this relationship? One way to do this is to collect sensor measurements that may help us build an appropriate monitor, then through a supervised learning process, create a new element in the evidence set and establish the relationship between the newly discovered evidence and existing elements in the failure mode set. We call this the **active query** function.

Within VIPR, an active query can be triggered when a monitor is issued or an FC containing a specific fault hypothesis is established. These are called monitor and FM queries respectively, and are explained with the following example.

> *Consider a two-engine aircraft. Suppose we observe a "slow start" monitor from engine one. Once this monitor fires, we would like to collect an engine speed sensor measurement 30-seconds before this monitor fires at the 10Hz sampling rate. We would like this data from both engines. We define this as a monitor query.*

The reasoning process within VIPR is initiated by the arrival of a new monitor that changes the probability distribution for the evidence state space, which then changes the probability distribution of the fault hypothesis state space. The **temporal analysis** function controls the rate at which these changes are made to these probabilities. Two forms of temporal analysis are supported within VIPR: intermittent monitors and FC timeout.

A monitor is intermittent if it changes the evidence state assignment faster than a pre-defined time interval. Since each firing of the monitor will update the evidence state space, and the fault isolation function will update the fault hypothesis probabilities, we want to perform these calculations only when the monitor state is persistent. Intermittent monitors are handled through a simple latching mechanism. The FC timeout mechanism within VIPR provides a means by which an FC can be put on the back burner for lack of sufficient monitor evidence; the timeout is specified as a simple number.

While the overall objective of VLRS is to detect and diagnose ongoing failure modes accurately, the net impact of this prevailing fault on mission completion or safety is the **function capability** function.

## 5.2 VIPR Detail Design – Software Implementation

Figure 3 illustrates VIPR's functions, which are described above. All these functions are needed for VIPR to achieve its overall objective of detecting events before they escalate as safety issues.

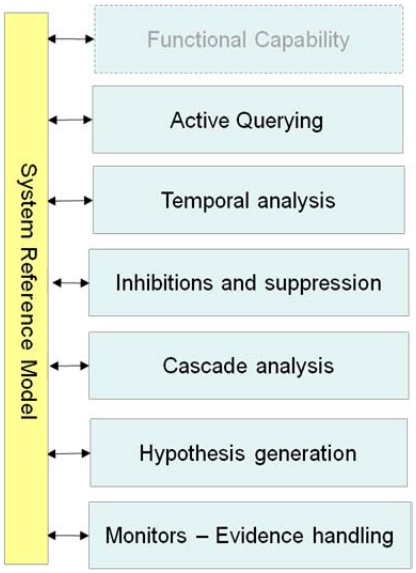



**Figure 3. Function modules within VIPR**

VIPR functions are data driven—where "data" refers to aircraft-specific information. For example, in an abstract sense, the fault isolation function operates on a bipartite graph: failure mode set, evidence set E, and the links between them. The exact mechanism by which the elements in the evidence set states were assigned is irrelevant to the fault isolation function. It starts with a state space distribution for the elements in the E set and generates the state space distribution for the elements in the F set. Clearly, the monitor function assigns the state values for elements in the E set. To make this assignment, it needs to know the relationship between the E set and another set called the monitor set M. The cascade function, on the other hand, operates on two F sets and links between them. The suppression function operates on the F and E sets using two bipartite graphs to achieve its objective.

Clearly, the functions operate on abstract sets, such as F, M, and E. Since VIPR is solving real aircraft problems, we need to "instantiate" these sets with aircraft subsystems. This instantiation is captured in the system reference model. A given aircraft can be composed of several reference models, each provided by a member subsystem.

For specific instantiation of the abstract sets such as F, M, and E, the data is specified as an externally loadable data image (LDI). This specification allows the member subsystems to continually update these sets as new knowledge is discovered. The underlying calculations do not change, the isolation function merely "receives" an instance of a reference model as an input argument along with other items.

The VIPR functions are loosely connected and can thus be distributed across multiple computation hardware. However, they are interlinked. For example the suppression function receives its inputs from the monitor function. Similarly, the cascade function operates on the outputs generated by the monitor and the fault isolation functions. A messaging protocol describes how the information is encoded so that the sender can send the data and the receiver can receive the data. VIPR defines an ARINC 624-based message protocol. Reference [3] describes messages that encode various entities such as fault condition, fault hypothesis, prognostic vector, and condition indicators.

VIPR is coded in MATLAB® using an object-oriented paradigm. For convenience, we defined a container class called `viprSystem` in the demonstration. Key classes and their interactions are illustrated in Figure 4. The implementation is summarized below.

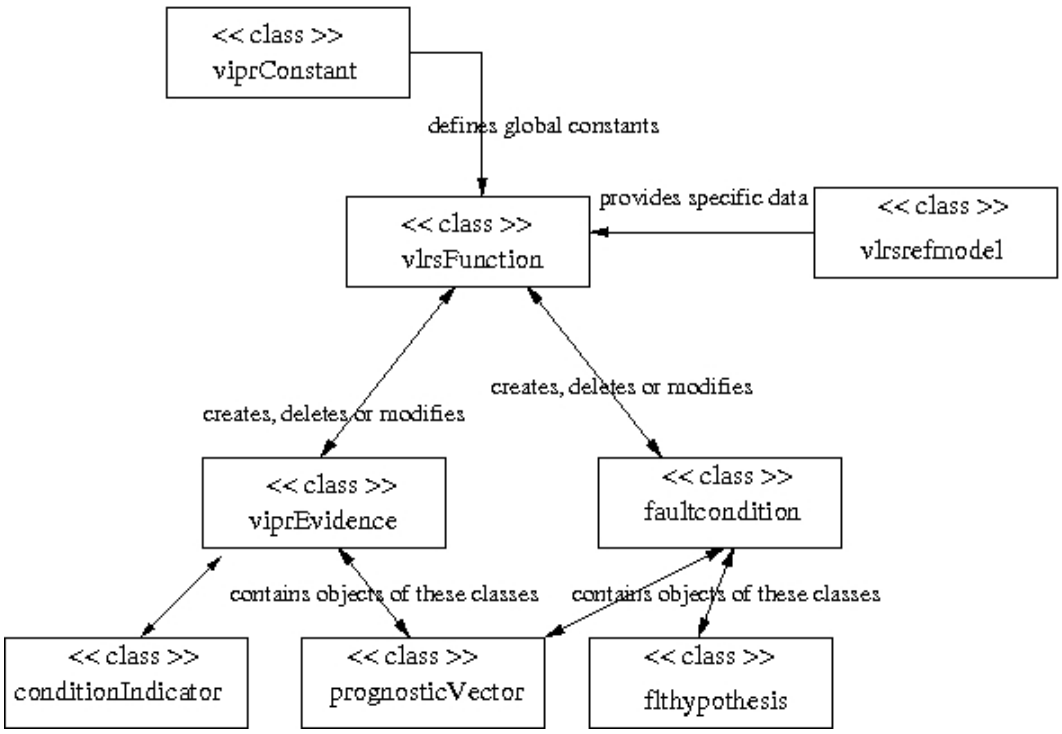



**Figure 4. VIPR demonstration classes and their interactions**

1. There is only one instance of the `viprSystem`. This object is a *container* that consists of several *nodes*. A node is a physical computation medium within an aircraft that has certain non-volatile memory, CPU time, and RAM.
2. VIPR can execute on a dedicated onboard computer, in which case, we have only one node. VIPR can also be distributed, in which case there are several nodes. If we choose a distributed architecture consisting of several nodes, the communication links between them must be defined. Currently, only a three-level (LRU, Area and Vehicle) tree configuration is allowed. In this case, the root (Vehicle) node communicates with the aircraft display system and provides the maintainer portal.
3. The nodes communicate among each other using an expanded ARINC 624 messaging protocol.
4. Each node executes one or more VIPR functions (those described in Section 5.1 and Figure 3). These functions are implemented as specialized objects of an abstract class called `vlrsFunction`.
5. Each function is data-driven. Aircraft-specific data for each function is given by a static reference model. Exactly one static reference model is associated with each `vlrsFunction`. This static reference model is implemented as specialized objects of an abstract class called `vlrsrefmodel`.
6. As various functions execute, several data objects are created, deleted, or modified. Two such entities are `viprEvidence` and `faultcondition`. These objects may contain objects of supporting classes such as `fltHypothesis`, `prognosticVector`, and `conditionIndicator`
7. The implementation also uses enumerated constants to support its operations. These enumerated constants are defined by a static class `viprConstant`.

8. The reasoning process is triggered by the arrival of a new monitor. This evidence generation is outside the reasoning process. A helper class, `evidenceStream`, simulates the arrival of new monitor to initiate the reasoning process.
9. Aircraft-specific information contained in various static reference models is saved in a TXT file. A single file contains all the necessary information to populate all the reference models. It is a mandatory input to the creation of the `viprSystem` instance.
10. A utility class, `viprUtils`, defines several utility functions that are used by various VIPR functions. This class can be interpreted as a utility library that is part of the code at each node. This class is not shown in Figure 4.

**Discussion about new evidence**

Prognostic monitor definition enables member systems to encode futuristic evidence. It standardizes a probabilistic expression (in the form of a prognostic vector) that can directly participate in the Bayesian reasoning process. However, not all member systems can produce a prognostic monitor. In this case, VIPR design allows the member system to periodically report a condition indicator (CI). The member system can then configure options for generating an equivalent prognostic monitor from the CI.

Basic terms are defined in Figure 5. Generating a prognostic monitor includes of a trend window containing $B$ number of historic sample points, a prediction window containing $W$ samples. The condition indicator is a time-series signal $x(t_O - k), k = 0, 1, 2, \ldots B - 1$. Here the current time is denoted as $t_0$.

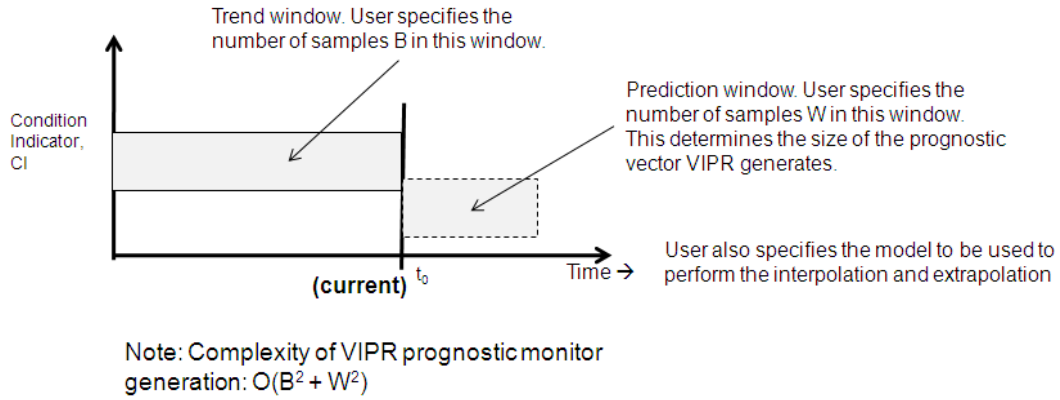



**Figure 5: Terminology definition**

Consistent with the data-driven VIPR design, the mechanism for generating a prognostic monitor from a given member system CI is configurable using an externally loadable data image (LDI). In this LDI, the member system specifies (a) the trend window size, (b) the prediction window size, (c) the failure mode set associated with this CI, and (d) the interpolation and extrapolation method. This method depends on the CI. VIPR handles both continuous and Boolean CI's. Corresponding methods are described next.

Note that prognostic monitor generation is a computationally expensive step. Specifically, the complexity scales as $O(B^2 + W^2)$. Hence, the number of calculations is proportional to the square of trend window and prediction window size.

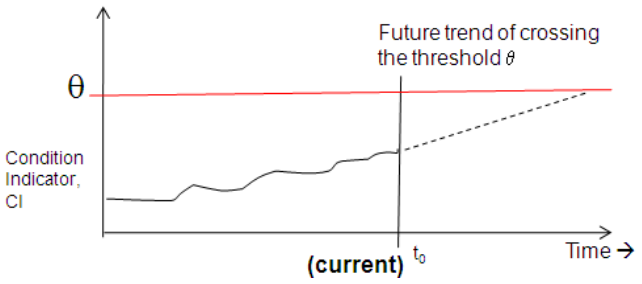



Figure 6: A continuous valued condition indicator.

**Continuous**. Mathematically, the samples take real values. In addition to N historic values, we are also provided a threshold $\theta$ such that $x(t) > \theta$ provides indicting evidence towards a pre-defined set of failure modes. Conversely, $x(t) \leq \theta$ provides exonerating evidence towards the same pre-defined set of failure modes.

In this case, a prognostic monitor is generated by a suitable extrapolation method that predicts threshold crossing $x(t) > \theta, t > t_0$ in the future as show in Figure 6.

VIPR provides two extrapolation mechanisms, linear and hidden state, which are shown in

Figure 7. Both mechanisms assume that the observed CI is a noisy realization of a hidden trend line; the noise follows a zero-mean, constant-variance, normal distribution. The confidence of the extrapolated line crossing the given threshold $\theta$ is a straightforward means for generating the corresponding prognostic monitor.

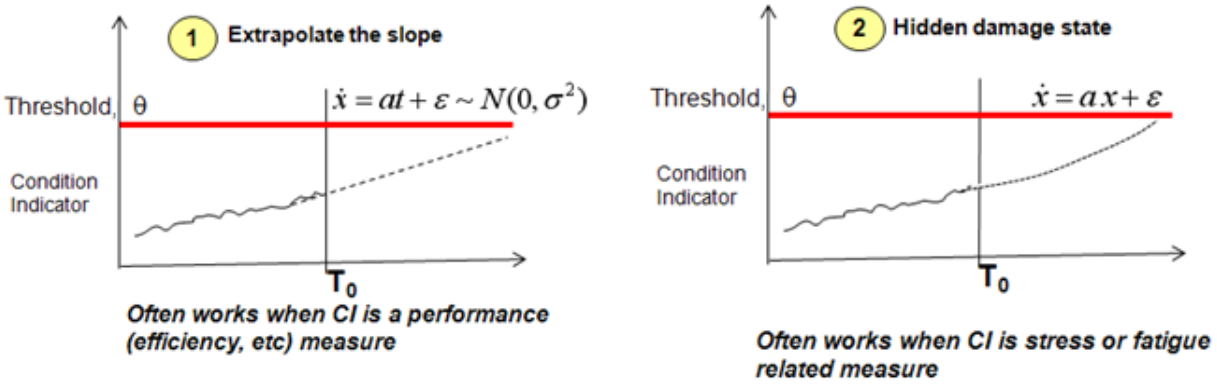



Figure 7: Generating prognostic monitors from condition indicators

**Boolean**. Mathematically, the samples take a 0/1 value. In this case, the member system may not provide the threshold $\theta$, but only information about exceedance. Also in this case, $x(t) = 1$ provides indicting evidence towards a pre-defined set of failure modes. Conversely, $x(t) = 0$ provides exonerating evidence towards the same pre-defined set of failure modes.

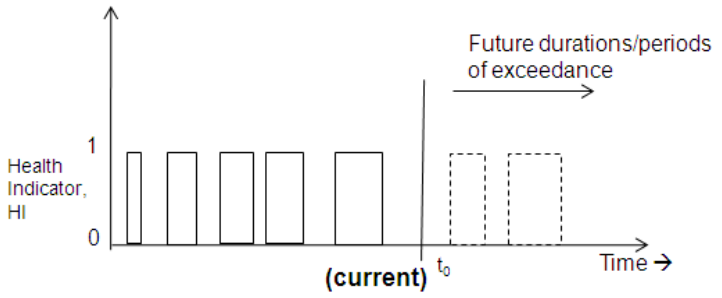



Figure 8: Boolean Condition Indicator (also called a health indicator)

In this case, a prognostic monitor is generated by a suitable extrapolation method that predicts the duration of obtaining a sequence of $x(t) = 1, t > t_0$ in the future as shown in Figure 8.

VIPR models the count of Boolean indicators (called health indicator) as a displacement of the random walk process. The displacement $L_n$ increments by 1 when $x(t) = 1$ and decrements by 1 when $x(t) = 0$. Let $L_n(t)$ denote the net displacement at current time $t$. During this time, assume that we have observed $X_n$ number of indicting evidence. To generate a corresponding prognostic vector, we are interested in calculating the probability $P(X_n|L_n(t))$ — that is the probability of observing $X_n$successes when the displacement is $L_n$ at time $t$. To calculate this probability, we must assume the underlying distribution for observing a success or a failure. VIPR provides two forms of correlation: (a) binomial distribution wherein the probability of getting an indicting evidence is $p$, (b) correlated binomial distribution where $\rho = \frac{1}{gL}$ is the correlation factor. Here $gL$ is the gap length and denotes the maximum number of samples for which an exonerating evidence can be ignored while observing a sequence of indicting evidence.

The transition matrix for this random walk model is given as follows:

$$\begin{array}{cc} HI(n) & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} q+\rho\ p & p(1-\rho) \\ q(1-\rho) & p+\rho q \end{pmatrix} \end{array}$$

where, $p + q = 1$ and $0 < \rho < 1$ for correlated binomial distribution and $\rho = 0$ for non-correlated binomial distribution.

Given a sequence of Boolean CI, the user can configure either a binomial or correlated binomial model to estimate the corresponding $p, \rho$ values. Generating a prognostic monitor is to calculate the probability of observing indicting evidence over the prediction window.

## 5.3 Regional Airline Data Mining

In Section 5.1 we described the fundamental elements for prognostic reasoning—evidence, failure modes, and the relationship between them and monitors that express the current state of evidence. Broadly speaking, the onboard inference rules operate on this bipartite graph (example shown in Figure 2) and calculate the probabilities for various failure modes using a Noisy-OR classifier, which is a simplified form of a standard Bayesian network. However, rather than addressing the problem as a traditional data mining problem, we approached it as an extension to the existing reference models. In other words, the data mining algorithms should be designed to provide information that supplements existing, expert-generated reference models, as opposed to providing different formulations and different reasoner structures. Verifying the model enhancements by experts is then relatively straightforward. Specifically, the data mining resulted in the following "surgical" updates to the existing reference model:

1. [DM_Up1]: Updating the relations between evidence and failure modes. Specifically updating the conditional probability that the $i^{th}$ evidence $e_i$ will be assigned an indicting state when the $j^{th}$ failure mode $fm_j$ is present. In other words: $P(e_i=1|fm_j=1)$.
2. [DM_Up2]: Updating indict/exonerate region specification for a condition indicators-based monitor. For example, the original reference model may specify that an indicting monitor for evidence $e_i$

should be issued when the aircraft climb rate is less than 200 ft/s. Following the learning step, we may change the threshold to 180 ft/s.

3. [DM_Up3]: Creating new monitors that combine $m_1$ and $m_2$ such that when both monitors are issued, their combined occurrence either asserts a stronger evidence for a specific failure mode $fm_j$. That is, the learning step can replace two conditional probabilities $P(m_1|fm_j=1)$ *and* $P(m_2|fm_j=1)$ with a new probability $P(m_1, m_2|fm_j=1)$.
4. [DM_Up4]: The learning process discovers a new monitor, hence, new evidence for an existing failure mode. For example, the learning step can discover a monitor when the aircraft climb rate is *greater* than 270 ft/s, which creates a new evidence $e_{new}$ in the reference model. In this case, the reference model update occurs in three-steps: creating new monitors, creating new evidence, and linking the model with existing failure modes.

Data mining activities and their role in the project are illustrated in Figure 9. Demonstrating and validating the proposed upgrades to the system reference model needs data. An important requirement for the success of data-driven techniques is the need for relevant and well-organized data. We call this the **data curation** step. To evaluate the ability of our data mining techniques to improve the reference models, we have conducted a set of experiments using the simulated data from CMAPS-S, which is a simulator developed at NASA's Glenn Space Center [4].

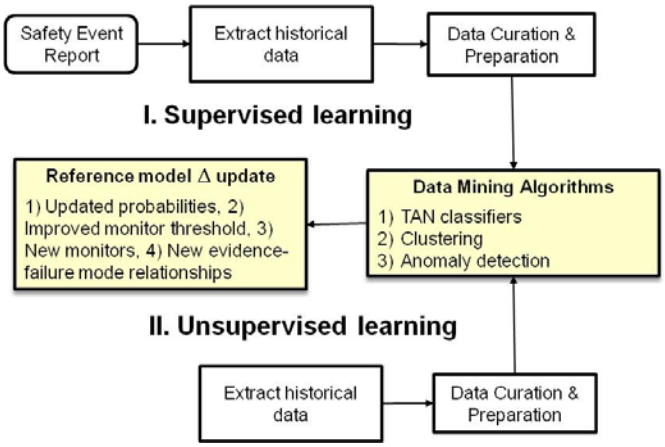



Figure 9. Data mining activities

The simulator parameters can be set to run in nominal and faulty modes of operation. We chose the following condition indicator (CI) monitors for improvement. The CIs were: (1) stall margin of the high pressure compressor: HPC CI, (2) stall margin of the low pressure compressor: LPC CI, and (3) stall margin of fan: Fan CI.

The CMAPS-S data was generated in a way that the fault(s) and their time of introduction were known, so it was easy to assign the nominal and faulty labels for each data stream. The CMAP-S data models three faults: (1) a fan fault (Fan), (2) a high pressure compressor fault (HPC), and (3) a high pressure turbine fault (HPT). We were hoping to generate three new pieces of evidence ($e_1$, $e_2$, and $e_3$) for the three failure modes listed above. The data mining results are listed in Table 3.

Table 3. Evidence discovered using the TAN supervised learning algorithm

| | Evidence $e_1$* | | | Evidence $e_2$* | | | Evidence $e_3$* | | |
|---|---|---|---|---|---|---|---|---|---|
| **Fault** | **Acc** | **FP** | **FN** | **Acc** | **FP** | **FN** | **Acc** | **FP** | **FN** |
| Fan fault | 94.4% | 0.4% | 0.7% | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| High pressure compressor fault | | | | 80.8% | 36.7% | 0% | | | |
| High pressure turbine fault | | | | | | | 94.7 % | 8.9% | 2.9% |

* All evidence used the three CIs listed earlier. Acc = accuracy, FP = false positive, FN = false negative

This knowledge was incorporated in the reference model as follows:

$P(e_1=1|fm_1=1) = 0.944$, $P(e_1=1|noise) = 0.004$
$P(e_2=1|fm_2=1) = 0.808$, $P(e_1=1|noise) = 0.36$
$P(e_3=1|fm_3=1) = 0.947$, $P(e_3=1|fm_1=1) = P(e_3=1|fm_1=1) = 0.015$, $P(e_3=1|noise) = 0.09$

With this confidence, we applied the same procedure to data recorded from a fleet of aircraft. This data set contained the statistical richness arising from aircraft-to-aircraft variation as well as heterogeneity of flight patterns. The fleet consisted of 30+ identical aircraft, each aircraft operating 2–3 flights each day. Data spanning three years was made available to support this work. Within this time period, the airline experienced several safety incidents that formed specific labels for the "supervised data mining" step. In addition, we also explored unsupervised learning to discover anomalous operations. Table 4 summarizes the $\Delta$-updates we made to a supplier provided reference model via supervised data mining. Additional details on performance of the Tree Augmented Network (TAN) algorithm used are described in [7 and8].

**Table 4. Summary of reference model $\Delta$ updates through data mining**

| Condition indicator | Before data mining | Post data mining |
|---|---|---|
| StartTime | An upper threshold to generate evidence called *no-start* | Discovered a lower threshold to generate a new evidence to indict a failing fuel metering unit. |
| IdleSpeed | A lower threshold to generate evidence called *hung-start*. | Updated the probability of this evidence to indict a failing fuel metering unit. |
| peakEGTC | An upper threshold to generate evidence called *over-temperature*. | Updated the probability of this evidence to indict a failing fuel metering unit. |
| N2atPeak | An upper threshold to generate evidence called *over-speed*. | -- |
| timeAtPeak | An upper threshold to generate evidence called *over-temperature*. | -- |
| Liteoff, prelitEGTC, | An upper threshold to generate evidence called *no-lightoff*, *hot-start* respectively. | Updated the upper threshold and this evidence to indict a failing fuel metering unit and turbine blade loss. |
| phaseTWO | No links were links defined for these CIs in the reference model. | Defined upper threshold and linked this evidence to indict a failing fuel metering unit and turbine blade loss. |
| tkoN1, tkoN2, tkoEGT, tkoT1, tkoPALT | No links were links defined for these CIs in the reference model. | -- |
| Stall margin, HPC margin, HPT margin | No links were defined for these CIs in the reference model. | Discovered a pattern among these CIs to generate three new items of evidence that indicted the fan, compressor, and turbine respectively. |

| | | |
|---|---|---|
| Rolltime, resdTemp, N2atDip, dipEGTC | No links were defined for these CIs in the reference model. | Defined upper threshold and linked this evidence to indict a failing fuel metering unit and turbine blade loss. |

The updated reference models were incorporated as an external loadable data image for onboard VIPR reasoning. The net impact on avoiding safety incidents are listed in Section 5.6.

## 5.4 Anomaly Detection

VIPR anomaly detection monitors individual aircraft in a fleet by collecting and processing onboard data and continuously seeking emerging patterns. These steps are depicted in Figure 10 and described below.

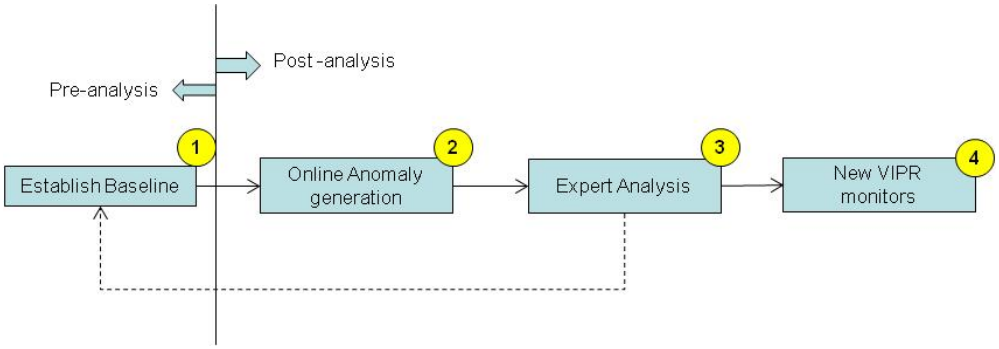



**Figure 10: Operational steps in VIPR anomaly detection**

- **Pre-analysis**: This step sets up the anomaly detection function within VIPR.
  1. Establish a baseline from historical fleet data.
- **Post Analysis**: These steps continually generate anomalies and translate significant ones as VIPR monitors to enhance prognostic reasoning.
  2. If a pattern is an outlier when compared to a pre-established baseline, it is downloaded from the airplane as an anomaly report for further analysis at a central location.
  3. An expert analyzes a series of anomaly reports and determines their significance with respect to operational practices, safety hazards, and/or equipment-related malfunctions.
  4. A subset of these cases deemed important to aircraft safety or operational efficiency are programmed and deployed across the entire aircraft fleet as new VIPR monitors.

From a functional point of view, within the VIPR context [10], a diagnostic/prognostic monitor provides evidence about the presence of specific failure modes—called its ambiguity set. *Ambiguity set* indicates that the evidence provided by a D/P monitor may not map to exactly one failure mode. Nevertheless, all failure modes associated with a D/P monitor are actionable through appropriate maintenance or mitigation actions.

Anomaly monitors, on the other hand, do not have a pre-defined ambiguity group. In fact, the overall objective of the anomaly detection function within VIPR is to define this ambiguity group.

The functional element of Step 1 (establish baseline) is an offline analysis, described in Section 5.4.1. From a data mining perspective this is an unsupervised learning step. Steps 2–4 are online, semi-supervised learning steps; they are described in Section 5.4.2.

### 5.4.1 Establishing a baseline: Pre-analysis

The offline approach to deriving the nominal model that is the basis for anomaly detection is based on an unsupervised learning approach. The overall approach involves the following steps, which are marked 1–5 in Figure 11.

1. Data frames surrounding key flight phases such as taxiing, take-off, cruise, descent, and touch down are collected from individual aircraft in an operating fleet. Each data frame is a two-dimensional vector, and each flight defines a unique data point. Therefore, a set of flights, define a $P \times N \times M$ data cube, where P is the number of flight segments, N is the number of features associated with each flight segment, and M is the number of samples that define the time-varying characteristic of a feature. Here, the term "feature" is synonymous with an aircraft sensor parametric value. Pairwise feature distances between every pair of data points is computed using the Kolmogorov complexity measure [Kolmogorov, 1965]. This computation requires $O(P^2N)$ calculations, which can be computationally intensive, because P is typically of the order of $10^4$ to $10^5$ and N is typically of the order of $10^3$ (see Table 1).
2. The pairwise feature dissimilarities between flight segments are converted to a two-dimensional matrix of pairwise distances among flight segments.
3. The Euclidean metric is employed for building the two-dimensional dissimilarity matrix among flight segments.
4. A hierarchical clustering approach (in our case, we used the complete link clustering algorithm) generates the dendrogram used to define the nominal clusters of flight segments as well as the outliers and anomalous clusters.
5. Offline analysis bifurcates to: (a) extract a nominal model to be employed for on-aircraft anomaly detection (Steps 2–4 in Figure 11), and (b) an anomalous clusters that can be used directly to generate VIPR monitors as described in Section 3.2.2).

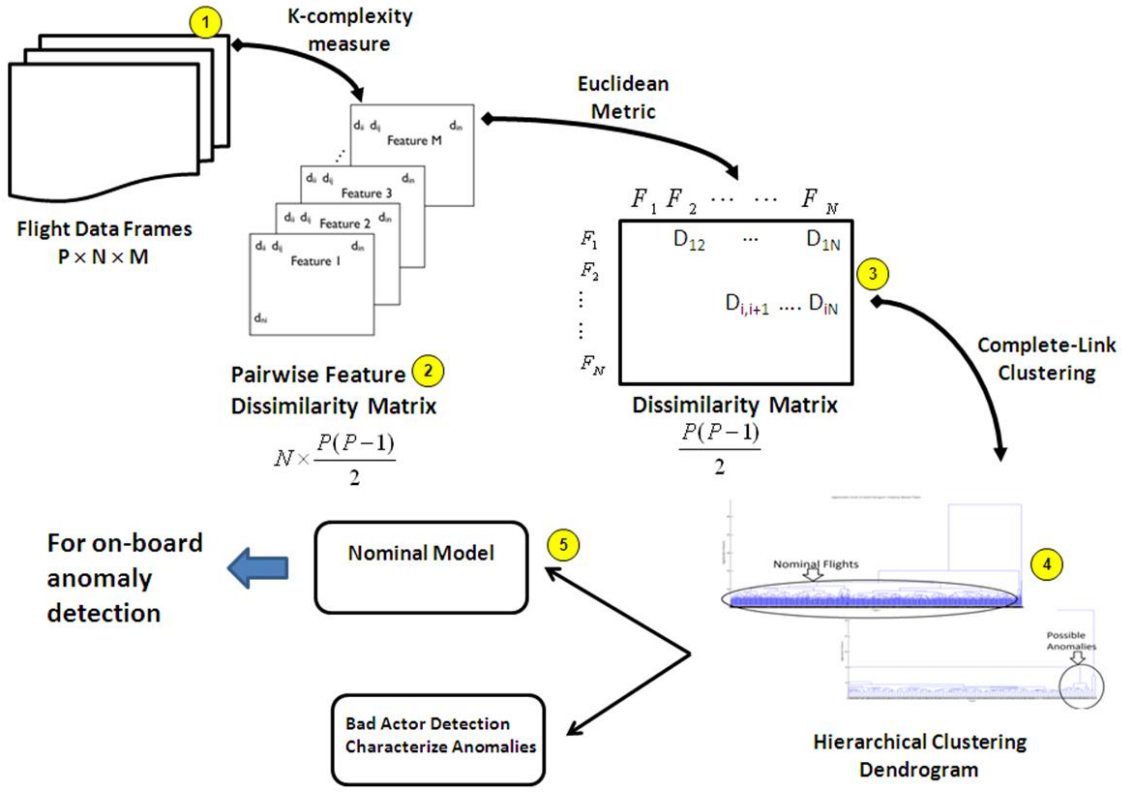



**Figure 11: Establishing a baseline—offline unsupervised analysis**

### 5.4.2 Anomaly Generation: Post Analysis

The on-aircraft element consists of an anomaly monitor generation element. Sensor data streams are continuously monitored by an on-board computing device for any new emerging patterns. If necessary, these patterns are sent over a low-bandwidth wireless or wired network to a central location for possible further analysis.

Off-aircraft analysis includes a human expert analyzing the data using numerous tools. These tools search for patterns in data for multiple flights; report key descriptive statistics grouped by phase of flight, airports, and weather data; and document the distribution of these patterns as they relate to standard operating procedures. Figure 12 illustrates the steps for online anomaly generation within VIPR.

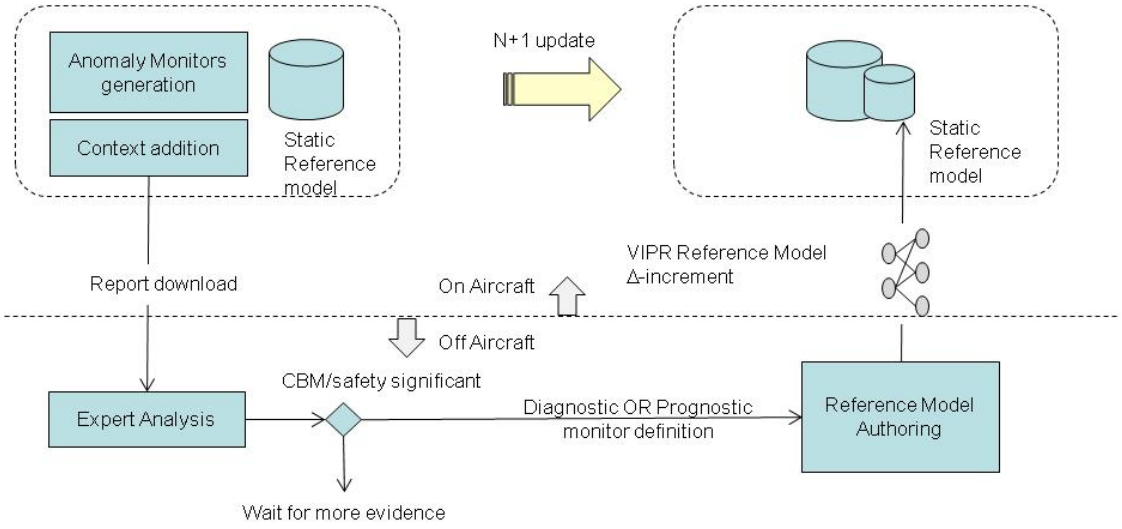



**Figure 12: VIPR Anomaly Detection: Functional View**

The result from the offline expert analysis helps determine the significance of the anomaly. From an operational point of view, significance implies that the anomaly affects operational, safety, or equipment maintenance, and most importantly, an appropriate mitigation or maintenance action can be defined for the next time this pattern occurs.

Functionally, within VIPR the failure modes provide this logical abstraction, which when asserted by a fault condition, enables the flight crew to avoid a safety incident or helps the maintainer execute a condition-based maintenance action. This failure mode set {F} is defined within the VIPR static reference model, which is an externally loadable data image (LDI). If an appropriate failure mode already exists in the VIPR static reference model, the newly-defined anomaly pattern can provide additional evidence as additional diagnostic or prognostic monitors. If an appropriate failure mode does not exist, then we may need to create a new failure mode node in the static reference model.

Procedurally, anomaly detection reduces to adding more nodes to the evidence set {E} or adding more nodes to the failure modes set {F}, adding more arcs to the bipartite graph, and assigning a detection probability. This step is called reference model authoring. Functionally (as shown in Figure 12), it creates a delta-increment to the existing static reference model. This delta-increment appends additional information to the VIPR LDI and can be uploaded to every aircraft in the fleet during a regular software update cycle.

These functional steps can repeat several times in response to the detection of novel anomalies. The process begins with the generation of anomaly monitors that are eventually translated to on-aircraft diagnostic and prognostic (D/P) monitors. These D/P monitors, viewed as increments to the on-aircraft VIPR reference model, are used by the reasoner algorithm to generate plausible hypotheses of fault conditions that may cause adverse safety incidents or trigger condition-based maintenance.

Table 5. Summary of anomaly detection through data mining

| Anomaly | Method of Discovery | Post data mining |
|---|---|---|
| Broken fuel quantity sensor | PCA-DBSCAN identified the initial sequence. K-complexity clustering isolated relevant features that included the fuel quantity sensor. Analysis of sensor indicated an empty tank. Further analysis confirmed that tank was indeed not empty. | Introduced the use of contextualization after detection to understand the nature of a collective anomaly. In this case, going from location based context in PCA-DBSCAN to Tail Number based context in K-complexity clustering. |
| High energy take offs | Online K-complexity identified several flights where the bad actors included altitude, altitude rate, and engine temperatures, which showed more energy than normal was used on take-off. | Can isolate from these features potential condition indicators for use in diagnosis. |
| Wind altered take offs | PCA-DBSCAN Identified the initial sequence. K-complexity clustering isolated relevant features that included glide slope and wind speed, which spiked on take-off. | Can isolate from these features to mark this as a warning for other aircraft at that location. In this case, the initial context of location was suitable for both methods. |
| Power lever angle inconsistencies | Online K-complexity identified a flight where the bad actors included glide slope, and a single power lever angle for the third engine. The PLA sensors for the three other engines were not ranked. Offline analysis confirmed that only this lever had anomalous behavior. | Online method found a potentially interesting anomaly that is fairly rare. |

## 5.5 Performance Metrics

The VIPR evaluation approach is shown in Figure 13. We used a regional airline data base to enhance the reference model and to generate monitors for testing VIPR. The reference model is also an input to the Failure Mode Simulator, which generates an evidence stream corresponding to a selected failure mode defined for the reference model. The evidence stream is then fed to the VIPR reasoner which produces outputs including the faults isolated, potential faults (faults detected with high probability but not isolated), time of isolation, isolating reasoner, etc. These outputs are logged and processed by metrics analysis scripts which compute the true detect rates, false alarm rates, average time to isolate, volume of reasoner messages, processing and communications latency, etc. The metrics analysis results can be used for improving the reference model and the reasoner.

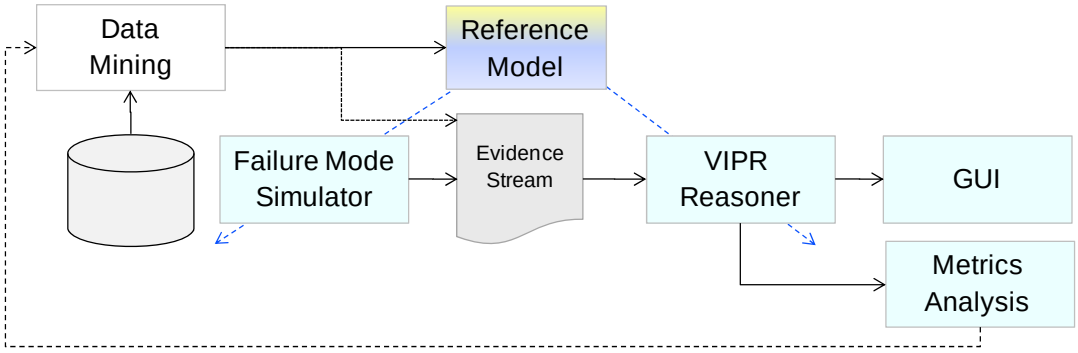



**Figure 13. VIPR evaluation approach**

The reasoner accuracy and communication metrics are calculated for two equivalent reference models—flat and hierarchical. VIPR utilizes the hierarchical reasoner where the reasoning is distributed between the LRU and the area- and vehicle- health managers. The flat reasoner model is utilized in the central reasoner framework, where all the reasoning is performed at the central location.

We measured the reasoner accuracy to the following outcomes:

- Inserted fault conditions isolated
- Incorrect fault isolations
- Inserted fault conditions detected with high probability (but not isolated)
- Incorrect fault conditions detected with high probability
- Missed fault conditions: inserted fault conditions that were not even detected

We developed the following notation for marking these conditions:

| | |
|---|---|
| **I** | Fault detected and isolated |
| **D** | Fault detected with high likelihood but not isolated |
| ***** | Denotes an inserted fault; for example *I indicates that the inserted fault was isolated |
| **+** | Denotes a fault that was not inserted; for example I+ indicates a fault that was isolated but not inserted |
| **M** | Missed fault condition |

The '*' and '+' annotations can be combined with the 'I' and 'D' annotations. For example, a test outcome with the annotation '*I+' indicates that the inserted fault was isolated, but so were other fault conditions.

To enhance the rigor of our testing, we simulated just the "complex faults" which are those faults implicated by more than one monitor that also each implicated more than one fault. Ten APU fault conditions and 12 engine fault conditions met this condition. Results of simulating the 22 complex faults, including all combinations of pairs of faults, are documented in the VIPR Metrics Report [7].

We ran the hierarchical and flat reference models for the same set of evidence streams for the set of complex faults and got very similar accuracy results for the two models. In summary, for the single fault case (see Figure 14), the reasoner provided an unambiguous and correct result (*I and *D results) in 75% of the test cases and a correct but somewhat ambiguous result (*I+ and *D+ results) in an additional 23% of the test cases, and isolated to an incorrect fault the remaining 2% of the test cases (I+ result).

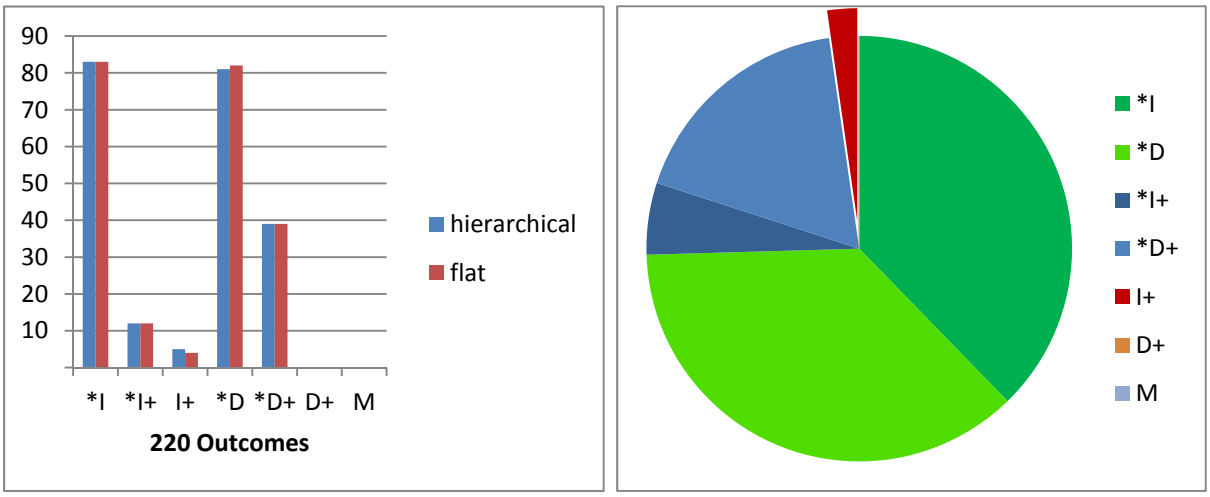



Figure 14: Results of single fault simulations

Outcomes for multiple complex fault cases are shown in Figure 15, below. The outcome was correct and unambiguous in 45% of the test cases (*I and *D results) and correct but ambiguous in 31% of the test cases (*I+ and *D+ results). In 14% of the test cases the reasoner detected the wrong fault (I+ and D+ results), and it failed to detect 5% of the fault insertions (M result).

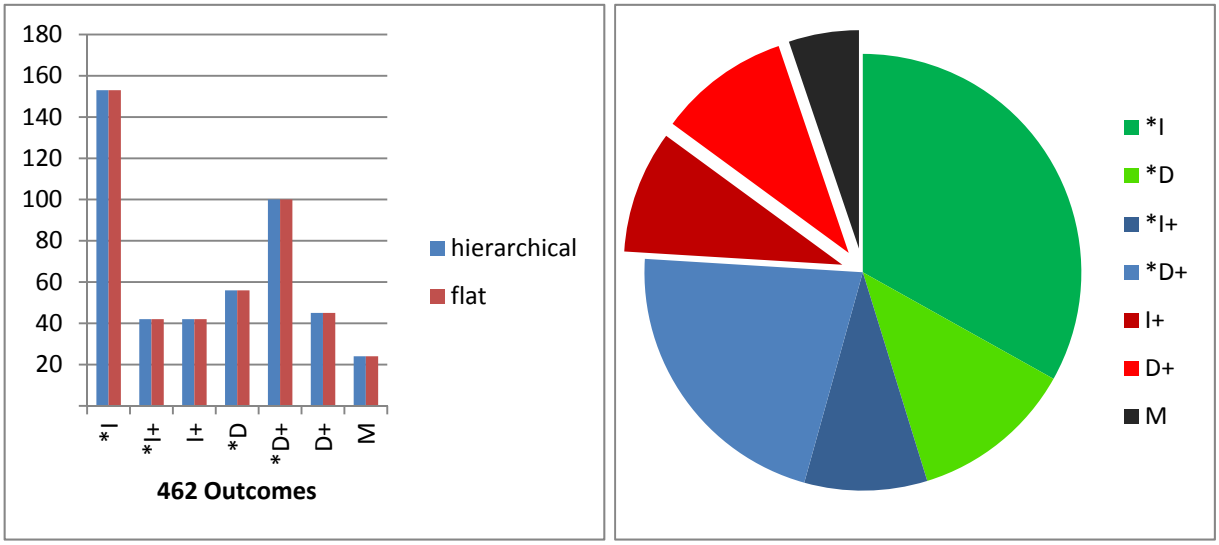



Figure 15: Results of multiple fault simulations

During VIPR Phase 2, we computed metrics that measure the reasoner accuracy, latency, communications bandwidth, computational cost, and the rate of false alarms. The summary and conclusions table from the metrics report [7] is included here for completeness in Table 6.

Table 6: Phase 2 metrics summary [7]

| Topic | VIPR Metrics Summary |
|---|---|
| Accuracy | The reasoner's ability to correctly isolate a failure from a given evidence stream depends on the quality of the evidence stream and the correctness of the reference model. When simulated with evidence streams that contained 0.1% erroneous data, the reasoner correctly and exactly identified the inserted faults 75% of the single fault cases and 45% of the multiple fault case. In addition, it correctly identified the inserted faults, but not uniquely, for an additional 23% of the single fault insertions and 31% of the multiple fault cases. Therefore, the reasoner correctly identified 98% of the single fault insertions and 76% of the multiple fault insertions.<br>Accuracy for the hierarchical and flat reference models were the same.<br>The prognostics have been shown to be accurate on three cases discovered from the airline database. The case studies on prognostics precursors have shown that the VIPR approach detects precursors to safety incidents multiple flights in advance of the actual event (in-flight shutdown). |
| Isolation time | For the single fault test cases, isolation typically occurred immediately after fault insertion (zero computation steps). The worst case time to isolate was 15 reasoner computation steps and the average time was 0.6 steps.<br>The time to isolate for the multiple fault scenarios was an order of magnitude longer than for the single fault insertions; the worst case time to isolate was 153 steps and the average case was 13.6 steps. |
| Communications bandwidth | For sending the ARINC 624 messages generated by the reasoner over a periodic safety critical communications system, we computed that 1 KB/second bandwidth would yield, on average, message latency of 1-3 seconds, depending on reference model and number of faults inserted. However, to reduce the worst case latency below the 10 seconds for all the simulated test cases would require a 10 KB/second communications bandwidth. |
| Isolating node | For the flat model, the isolating reasoner entity is always in the vehicle node. However, for a hierarchical model, reasoning can occur at any node. For our aircraft hierarchical model, isolation occurred at the LRU level 519 times, once at the area level, and never at the vehicle level. |
| Communications volume | The flat model required 22% fewer messages and 28% fewer bytes in total than did the hierarchical model. |
| Computation cost | When computation cost is the same at all aircraft levels (LRU, area, and vehicle), the computation cost for the flat model was lower than for the hierarchical model because the flat model requires fewer transactions to achieve the same results as the hierarchical mode. However, when computing at higher nodes is more expensive than at the lower levels, the distributed hierarchical model is less costly because so much of the computation is performed on the lower cost computing resources. |
| False alarms | The 0.1% rate of false evidence generated false alarms in only three of the 902 simulations run. |

We monitored the ATV communications cost for executing the HIL demonstration and found the cost to correspond approximately to the communications cost for a production LaserRef VI. For the demonstration, transmitting 14 IR output parameters at 25 Hz using a TCP protocol consumed about 4% of the device's CPU time. A LaserRef VI uses ARINC 429 (instead of TCP) to transmit about twice as many IR outputs and at double the rate (approximately four times more data than transmitted during the demo). The output process in the LaserRef VI is allocated 17% of the device CPU time.

Since the data used by the reasoner is already being transmitted by the device for consumption by other aircraft systems (principally the flight management system), we can assert that the reasoner did not place any additional processing requirement on the device.

## 5.6 Integrated Demonstration

### 5.6.1 VIPR Demonstration

The main objective of the integrated demonstration is to showcase the VIPR reasoner's ability to use heterogeneous data from different sources within the VLRS framework. It effectively demonstrates the integration of data from multiple aircraft subsystems, namely engines, APUs, avionics, flaps, and IRUs. Further, it integrates the input data collected from multiple systems at different sampling rates, which is representative of the diversity of data types and rates within the aircraft. Figure 16 shows the block diagram for the integrated demonstration in which the multiple evidence steams simulate two IRUs, engine data from the DAR files, and a fault simulator that can simulate evidence and false alarms for the other subsystems.

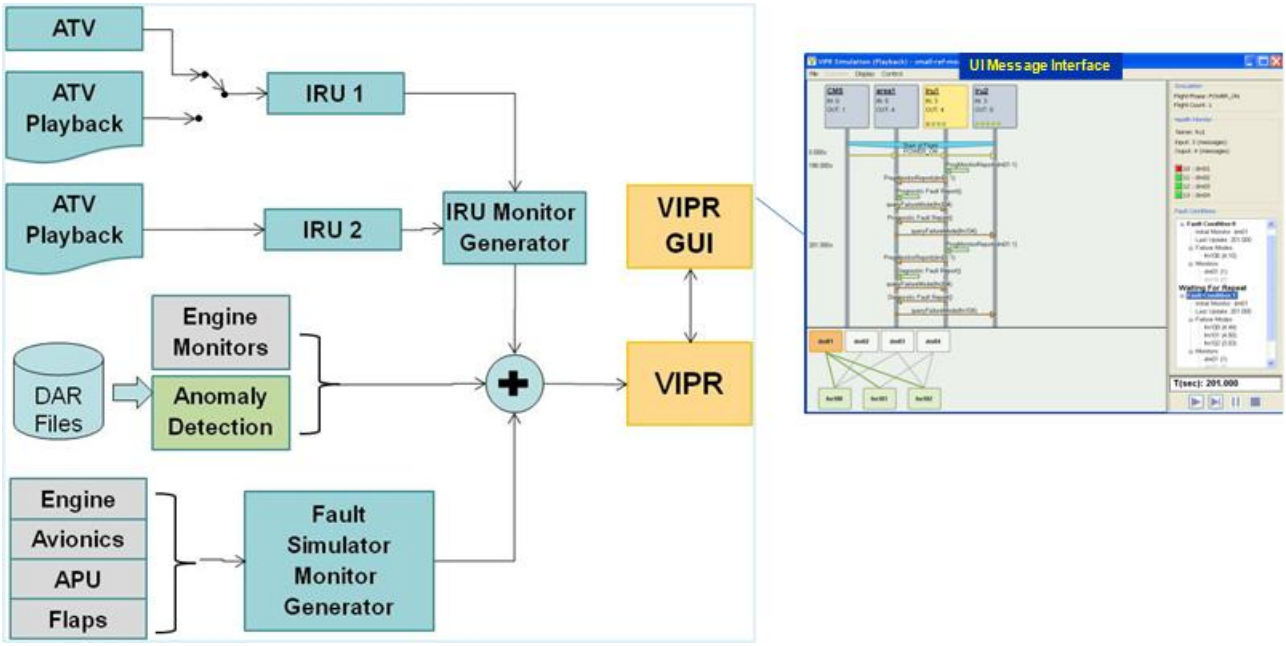



Figure 16: Integrated demonstration with multiple evidence streams.

The DAR files contain records of 181 parameters from the regional airline data in ARINC 717 format. It includes both continuous and discrete data. Different parameters within the DAR are recorded at 1, 4, 8 or 16 samples per second. The VIPR monitors are event driven (Table 7). The monitor generation

function uses the high frequency input from the DAR/IRU to generate the monitors (evidence streams) for the VIPR reasoner. The fault simulator directly generates the monitors for the simulated systems by using the system reference model. The false alarms for the simulated system monitors are generated using the false alarm rates specified in the fault simulator. The false alarms for the hardware (IRUs) and the DAR monitors are purely a function of the input data and the monitor quality.

**Table 7: Sampling rates for input evidence**

| System/Function | Sampling rate (samples/sec) |
|---|---|
| **ATV-IRU** | 25 |
| **DAR** | 1/4/8/16 |
| **Fault Simulator** (can be set to any other value) | 0.1 |
| **All Monitors** | Event Driven |

We built the software components shown in Figure 16 to graphically demonstrate VIPR's detection, isolation, and disambiguation of failure modes.

The reasoner, implemented in MATLAB with a Java library for ARINC 624 message formatting and TCP/IP communications, appears in the upper right portion of the figure. The reasoner communicates with a GUI implemented in Python that displays the step-by-step operation of the reasoner. The evidence simulator, also implemented in MATLAB, generates an evidence stream for a given fault set, which can be supplemented with erroneous evidence to simulate the generation of false alarms.

The demonstration is controlled from the GUI menu bar and the buttons at the bottom-right corner of the window. The demonstration operator begins by selecting a reference model, which the GUI forwards to both the reasoner and the evidence simulator. Next, the operator selects one or more faults to simulate, along with the fault insertion times. Finally, the operator starts the simulation. The "swim lanes" in the upper-left pane of the GUI list the individual messages sent among the reasoner elements; overall demo status is displayed at the top of the right-hand pane; and progress towards isolating the inserted faults also appears at the right and below the overall status. A hierarchical display of the indicting monitors and fault conditions is displayed in the window's bottom pane. In the bottom-right corner of the window are buttons for starting, stopping, pausing, resuming, and stepping the demo. A second GUI window (not shown in Figure 16) displays messages that could be displayed in an aircraft EICAS window to help the pilot manage failure conditions.

We used a version of the LaserRef VI, known as Acceptance Test Vehicle (ATV), for the demo because it allows accelerometer and gyro sensors to be simulated. Otherwise, the ATV hardware and software is the same as for a LaserRef VI.

The ATV in Figure 16 provides the evidence from the hardware IRU. The IRU has two sets of sensors—internal and external. The internal sensors include a set of three accelerometers and three gyros to measure the accelerations and rotations along the three orthogonal axes in the aircraft frame of reference. The external sensors include the inputs from GPS receivers and air data systems (ADR). The IRU uses the internal and external inputs to generate a pure inertial solution and a hybrid-inertial solution that is the inertial solution enhanced with the GPS data. The IRU also employs the BIT,

reasonableness and freshness tests to the inputs (both internal and external) to validate and qualify the output solution. Note that in the current system architecture, the IRU depends on the ADR and the GPS to detect their faults and provide a validity flag.

Figure 17 shows the timeline for the scenario shown at the VIPR hardware-in-the-loop final demonstration.

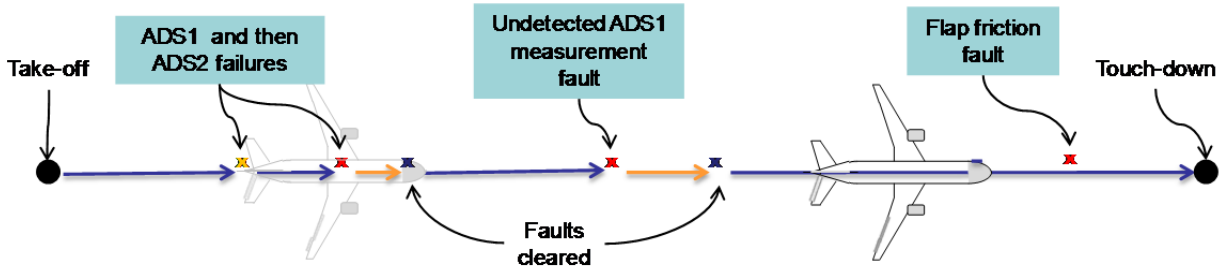



**Figure 17: Final demo scenario**

The demonstration illustrated a complete flight from take-off to touch-down with multiple subsystem injected faults. The 'x'es on the timeline mark the injection (orange and red 'x'es) and removal of faults (black 'x'es). The orange arrows show flight leg sections where the ADR inputs are not usable. Note that "ADS 1 undetected measurement fault" on Figure 17 represents an ADR fault that is not detected by the IRU (i.e. the ADR data flag is not set as invalid) but is instead detected by a VIPR Area level monitor. The monitor detects this ADS fault by comparing the air data values produced by the two air data receivers, but cannot isolate the fault to a specific ADR; hence both ADRs are marked as failed (the second red 'x' in Figure 17).

We simulated two ADR fault scenarios. In the first case (yellow 'x' shown in Figure 17), ADR 1 faulted; the IRU detected the fault and automatically switched to ADR 2. Subsequently, ADR 2 also failed (first red 'x' in Figure 17); the IRU detected the second air data failure and began producing a subset of the pure inertial solution feasible in the absence of air data.

In the second scenario, the ADR passed invalid data to the IRU that was marked as valid. An area level VIPR reasoner detected the fault and marked both ADRs as invalid, which tells the pilot that the airspeed, altitude, and vertical speed for the IRU cannot be trusted. Since the air speed is no longer accurate, any stall and overspeed warnings are meaningless. Even without the isolation, this knowledge will help the pilot by providing better situation awareness. If we enabled the hybrid IRU solution, we could use the GPS output to help isolate to the bad ADR [8].

The fault cases used to test the integrated demo are listed in Table 8, and also includes a flap excessive friction fault that occurred at time 500.

Table 8: Faults simulated in the HIL Integrated Demo.

| Time | Fault | Diagnostic Monitor Activity | Reasoner Response |
|---|---|---|---|
| 150 | ADS 1 fault | Monitors for the two IRUs report ADS receiver 1 faults | The reasoner reports ADS 1 failure to EICAS at Time 150.48 |
| 200 | ADS 2 fault<br>(ADS 1 and ADS 2 faults cleared at time 250) | Monitors for the two IRUs report ADS receiver 1 faults | The reasoner reports ADS 2 failure and No ADS Data condition to EICAS at Time 250.16 |
| 400 | ADS 1 and ADS 2 values are not consistent with each other<br>(inconsistent data fault cleared at time 430) | Monitors for the avionics area manager report ADR altitude and ADR airspeed mismatch conditions | The reasoner reports avionics – invalid air data condition to EICAS at time 400.80 |
| 500 | Flap friction fault | Monitors for the flap area manager report surf free and position limited conditions | The reasoner reports a flap area – excessive friction condition to the EICAS at time 500.48 |

We exercised two other ATV fault scenarios using the integrated demonstration configuration, but without evidence streams from the second IRU, the DAR files, or the fault simulator.

For the first scenario, we added a small bias to one of the gyro values. This error accumulated over time, and at 670 seconds into the scenario, the ATV reported three conditions, which triggered the Attitude Invalid, IR Fault, and Critical ATV Fault diagnostic monitors. The reasoner used these inputs to generate an IR Fault failure condition at time 670.2.

In a similar scenario, we added a small bias to the values for one of the accelerometers. After 1500 seconds, the ATV had not reported any faults, but the IR solution had diverged from the actual path. This fault would have been detected had we configured a second IRU and built area-level monitors that compared the two IR solutions, as we did for checking the values from the two air data receivers. This fault can also be detected at the IRU level with GPS inputs using the hybrid solution as illustrated in Bharadwaj et al. [8].

The final ATV-related fault we simulated was a clock mismatch between ATV processing and the sensors. In this scenario, the sensor values are correct but the clock mismatch causes them to be used in the IR solution at the wrong time. The fault was inserted at the beginning of the flight. After 180 seconds of simulated flight time, the ATV reported conditions that caused the Attitude Invalid, IR Fault, and Critical ATV Fault monitors to fire. It then took the reasoner 0.52 seconds to report an IR Fault condition.

Using data mining techniques, we were able to predict the occurrence of three faults:

- Fuel metering fault with 95% accuracy and less than 1% false positives from 30 flights before the occurrence of the actual fault
- Blade break/nozzle damage fault with better than 90% accuracy and less than 3% false positives from 20 flights before the fault's occurrence
- Fuel manifold rupture with better than 90% accuracy and false positives at 22% from four flights before the fault's occurrence

## 5.7 Impact on Safety

We provide a detailed case study to illustrate the potential impact of VIPR on a safety issue. The example surrounds an in-flight engine shutdown (IFSED) incident recorded by the ASIAS database. An in-flight engine shutdown is a highly undesirable, adverse event that significantly impacts aviation safety. The flight crew immediately responded and, per the operating procedures, turned back and safely landed the aircraft. Investigation by the maintenance crew indicated a faulty fuel metering unit, although no exceedances had been reported for that engine in the recent past. If detected earlier, correcting such a problem would have been a routine line maintenance activity that would have prevented the IFSD and potential safety impact.

In Section 5.3, we described the reference model provided by the engine supplier together with the condition indicators. Message traces from the reference model indicated a fuel metering failure and a hypothesis was formulated by the onboard VIPR. However, the assigned probability was too low and competed with another failure hypothesis that indicated an engine turbine nozzle failure.

We used the condition indicators from the last 50 flights leading up to an engine shutdown event to improve the reference model as described in Section 5.3. The $\Delta$-updates to the reference model and the results are summarized in Figure 18. On the left hand side of Figure 18, Item #1 summarizes the causal sequence of events discovered by the data mining step—starting from the fuel metering actuator fault that initially manifested as sluggish engine start. The controller compensated by aggressive schedules. At some point, the controller saturated, which resulted in lower idling speeds. Eventually, the speed dropped below its allowed threshold, while the engine exhaust gas temperature (EGT) remained high, triggering the adverse IFSED event.

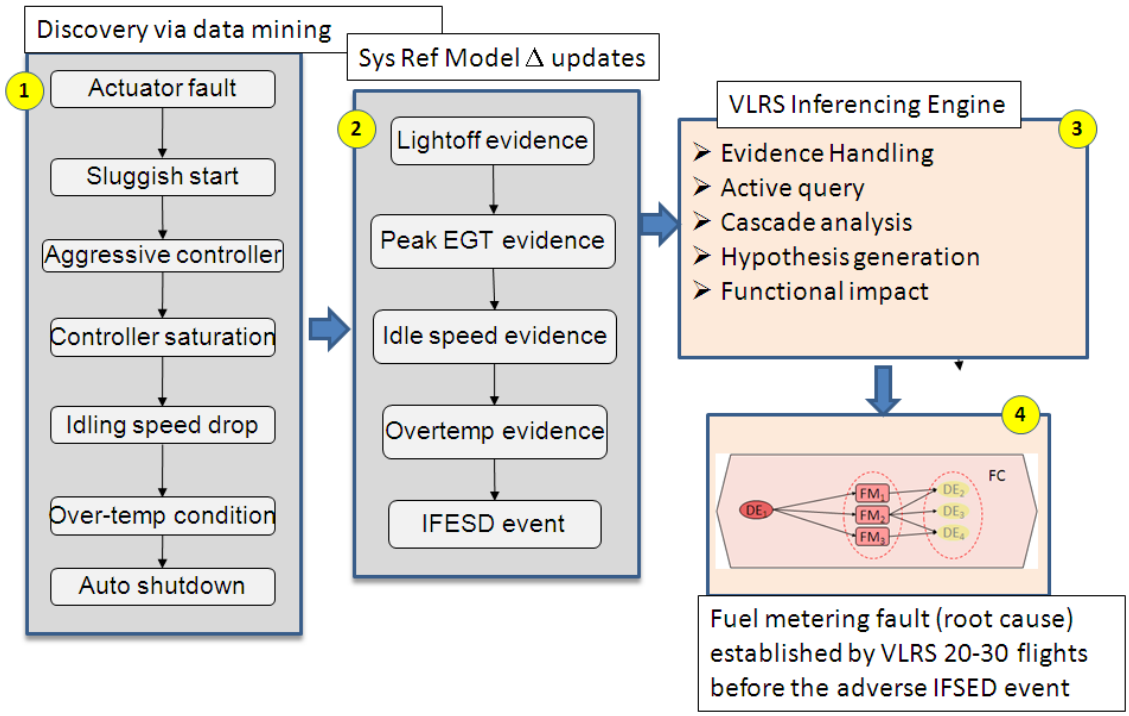



**Figure 18: VIPR combined with Δ updates made to the reference would detect the fault prior to an in-flight engine shutdown safety incident**

Second, we encoded this newly discovered knowledge as a Δ-update in the system reference model. These updates are marked #2 in Figure 18. This knowledge implied the addition of three new pieces of evidence to the system reference model: (1) time when the engine started, called the lightoff evidence, which looks for abnormally long engine start times, (2) peak exhaust gas temperature evidence that looks for abnormally high exhaust gas temperature during engine startup, (3) the idling engine speed evidence that looks for abnormally high and abnormally low speeds when the engine is idling before aircraft takeoff. The two existing pieces of evidence—namely over-temperature exceedance and INFSED are shown in Figure 18.

Third, we built the VLRS such that the inference engine exercises all seven functions we described in Section 5.1. The actual functions exercised by the VLRS are marked #3 in Figure 18. Finally, we re-ran VIPR with the updated reference model using the data from the last 50 flights before the IFSED event occurred and monitored its outputs—the fault conditions as described in Section 5.1 and marked #4 in Figure 18. The most plausible fault state of the aircraft was isolated to a fault condition that contained exactly one element in its ambiguity group—the fuel metering unit. Repeated experiments with varying notification threshold on the fault condition hypothesis we concluded that the VLRS would have established the fuel metering root cause anywhere from 20–30 flights before the IFSED event. This discovery would (in theory) allow the maintainer to fix the faulty component, eliminate the source of the fault, and completely avoid this safety event.

While this single event may be statistically insufficient for machine-learning validation metrics, the clear explanation discovered by the data mining method and the Δ-change to the reference model, together with a VIPR, seemed sufficient for the engine domain expert to give his approval.

We analyzed two more safety incidents and the impact VIPR would have in avoiding them. The second incident also involved an inflight engine shutdown caused by a high vibration event. In this case, VIPR correctly identified a broken blade that eventually led to high vibration and a safety engine shutdown. The third case was an engine-on fire safety incident. In this case, the data mining discovered a rather "noisy monitor." However, VIPR was able to handle this noisy monitor and, through active query and cascade analysis, conclude that the problem was a leaking fuel manifold, which led to an engine-fire incident. A leaking manifold affects engines #2 and #3 simultaneously, while failing fuel metering (HMA) affects only one engine. These three cases not only showed the prognostic accuracy of VIPR but also its impact on avoiding safety incidents. The prediction window produced by the VIPR approach is summarized in Figure 19.

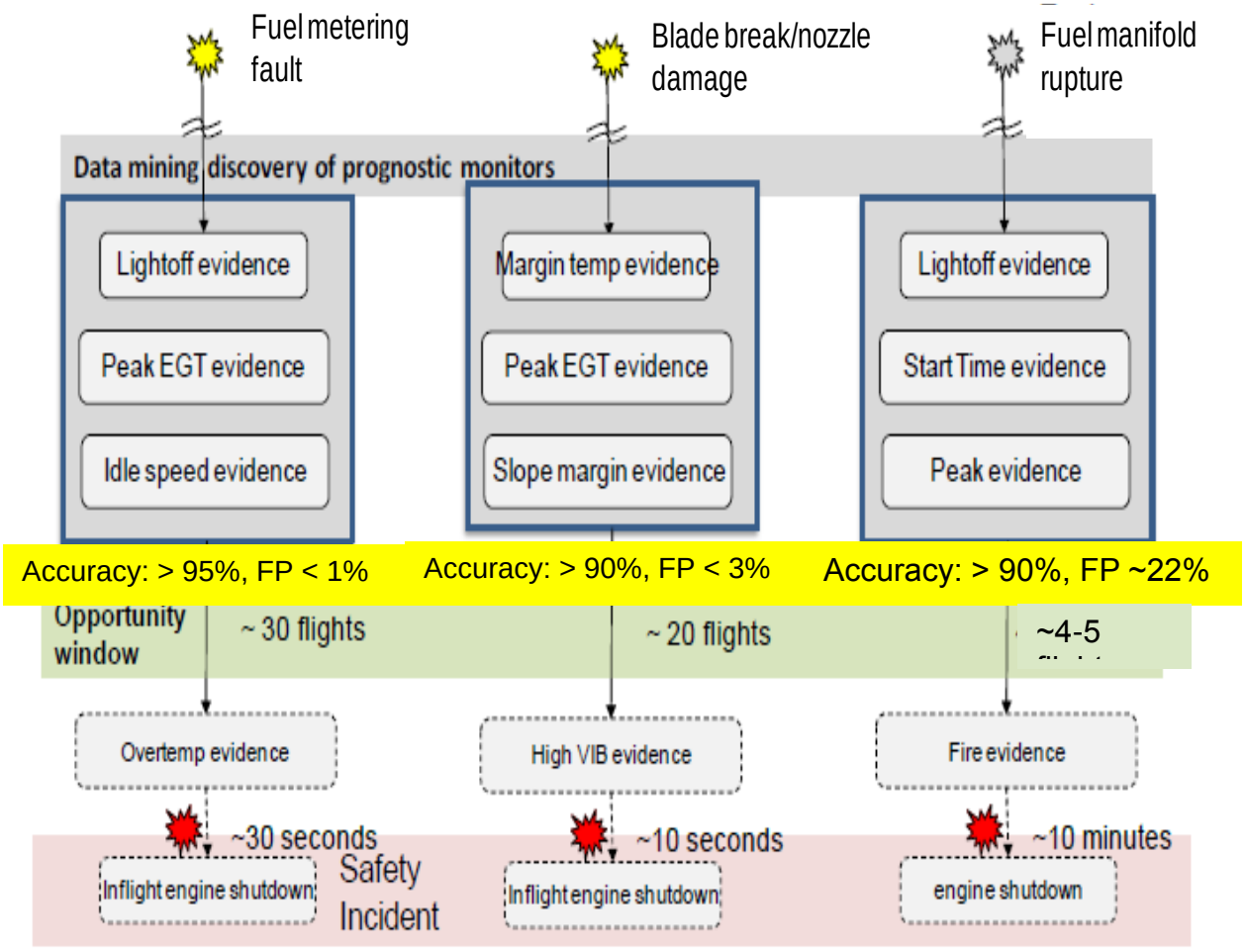



Figure 19: VIPR results for three safety incidents

## 6. VIPR Software Certification

To be resident with operational software in the aircraft, the VIPR reasoner software would need to be certified to DO-178B requirements up to the level of safety implied by the intended use of the reasoner. The table below discusses the expected level of safety certification needed for three potential uses of the reasoner.

| DO-178B Safety Certification Level | Reasoner Usage |
|---|---|
| Level D | Level D certification enables the reasoner to operate as part of the current CMC and be present in the aircraft for compiling information that could be used by an airplane mechanic to diagnose and repair failures. At this level, the information produced by the reasoner cannot be depended upon for making decisions that could impact safety. The pilots and maintenance personnel retain complete responsibility for safety. |
| Level B or C | Level B or C certification (depending on how responsibilities are assigned in the overall aircraft safety plan) is required to use reasoner conclusions to guide pilots or maintenance personnel on matters that could affect safety. Examples could be to suggest a course of action to a pilot following the detection of a failure, or to recommend deviations, based on operational parameters monitored by the reasoner, from the regular maintenance schedule. |
| Level A | Level A certification would be required to use reasoner results for automatically reconfiguring redundant systems after detecting a failure. |

There are two parts to the reasoner's safety certification. The first is to certify the reasoner software. From a certification perspective, there is nothing exceptional about the reasoner software. Consequently, existing processes for developing software that meets DO-178B requirements should be sufficient for the VIPR reasoner.

At Levels A, B and C (but not D), the second part is to certify both the reference model and the overall technical approach (reasoner algorithms, etc.) to ensure that a correctly functioning reasoner will, with suitably high probability, generate conclusions that are safe. Because the reference model is not an ordinary piece of software, the certification procedure will likely require a period of education and negotiation with the designated engineering representative (DER) community to define an acceptable process for certifying reference models.

## 7. Suggested Future Direction

VIPR embodies three core concepts, all of which support pro-active detection of conditions that may eventually manifest as safety incidents and/or enable condition-based maintenance. These three concepts—expression of complex evidence generation such as prognostic monitors and condition indicators, platform-agnostic Bayesian inferencing rules that take an active role in the diagnostic process, and offline data mining for continual improvement of the onboard reasoner reference model—form the core of the VIPR technical solution.

At the end of the three-year program, we have established the technical feasibility of these concepts and defined a design trade-space for practical implementation of prognostics within a vehicle level reasoner. Although this is an important step for business decision making, we believe the future of VIPR can further benefit by continuing activities in the following three areas:

1. **Continually building safety and CBM case-studies for practical acceptance.** We demonstrated a Bayesian inferencing process and generation of enriched evidence as an expanded central maintenance computer and aircraft condition monitoring function. All such expansions must have a cost-benefit analysis for practical acceptance. We clearly demonstrated VIPR's ability to detect failures related to fuel metering unit, turbine distress, and manifold leak all of which led to inflight engine shutdown. While these examples clearly demonstrate the effectiveness of VLRS, our examples were limited by availability of historical data and FAA recorded safety incidents. Validating the VIPR system on other airline data will start building the necessary examples for a favorable cost-benefit analysis. Since VIPR uses data that is commonly available on most modern aircraft, we believe applying VIPR to a larger fleet of aircraft is relatively straightforward and can be a programmed as a back-off server application that does not interfere with airline operations.
2. **Fleet-wide anomaly detection.** Specifically, we evaluated a Kolmogorov-complexity based measure to detect outliers against a nominal baseline data set. Within VIPR, this baseline data set is encoded as the static reference model that captures the relationship between failure modes and evidence. While one can investigate other statistical methods for outlier detection, we believe activities in this area should focus on expanding the reference model to include pilot check-lists under nominal non-normal (such as inflight shutdown, loss of air-data, etc.). Encoding this knowledge as additional elements in the reference model will enable the same data mining and/or inference rules to detect anomalies that arise due to pilot and aircraft interactions. Some of these anomalies can be mapped to incipient aircraft problems or airport-related problems. This is the first step towards understanding emergent events that neither the aircraft OEM nor component supplier can anticipate, and hence positively impact aviation safety.
3. **Onboard reasoning and evidence handling.** The definition of prognostic monitors enables the member systems to encode futuristic evidence. It standardizes a probabilistic expression (in the form of a prognostic vector) which can directly participate in the Bayesian reasoning process. We believe this expression is sufficient to handle incipient faults that arise in engines, electrical systems, APU, bleed systems and avionics. The gap one needs to fill is related to expressing evidence from a structural health monitoring system. This evidence has both space (location of damage) and time information. The prognostic fusion within VIPR needs to be expanded to handle this form of complex evidence. Based on our experience, we recommend pre-processing of such evidence before it enters the fusion step. In other words, we should address the challenge of how to fuse two monitors that provide the same damage-related evidence such that one is looking at the problem from the left and the other one is looking g at it from the right. A data-driven scheme for performing this spatial averaging is an open area of research.

# 8. References

## 8.1 Documents Referenced in this Report

Numbers on the following document references correspond to the cross-reference numbers in the body of this report.


1. G. D. Hadden, D. Mylaraswamy, C. Schimmel, G. Biswas, X. Koutsoukos, and D. Mack, "Vehicle Integrated Prognostic Reasoner (VIPR) 2010 Annual Final Report," NASA/CR–2011-217147 http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110012171_2011012606.pdf
2. D. Mylaraswamy, "Vehicle Level Reasoning and Data Mining," CIDU, 2010.
3. D. Mylaraswamy, D. Hamilton, and G. Hadden, "Information Protocols: NASA VIPR Program," Technical Report, submitted by Honeywell to NASA (CDRL Sequence Number: 4.1.03), December 2009.
4. D. K. Frederick, J. A. DeCastro, J. S. Litt, "C-MAPSS User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)," NASA/TM–2007-215026
5. D. L. C. Mack, G. Biswas, X. D. Koutsoukos and D. Mylaraswamy, "Using Tree Augmented Naive Bayesian Classifiers to Improve Engine Fault Models," presented at the 8th Bayesian Modeling Applications Workshop in Barcelona, Spain on July 14th, 2011.
6. D. L. C. Mack, G. Biswas, X. D. Koutsoukos, D. Mylaraswamy, and G. Hadden, "Deriving Bayesian Classifiers from Flight Data to Enhance Aircraft Diagnosis Models," to be presented at the Annual Conference of the Prognostics and Health Management Society, 2011.
7. R. M. Bharadwaj, D. Cornhill, and D. Mylaraswamy, VIPR Metric Report, CDRL 4.3.05, 2012.
8. R. M. Bharadwaj, K. Kim, C. S. Kulkarni, G. Biswas, "Model-Based Avionics Systems Fault Simulation and Detection," American Institute of Aeronautics and Astronautics, AIAA Infotech Aerospace 2010, April 2010, Atlanta, GA. AIAA-2010-3328.
9. T. Dodt, "Introducing the 787," ISASI, Sept 2011, http://www.ata-divisions.org/S_TD/pdf/other/IntroducingtheB-787.pdf .
10. D. Mylaraswamy, et al., "Vehicle Integrated Prognostics Reasoning," NASA Aviation Safety Annual Meeting, St Louis, 2011.


## 8.2 Published VIPR documents

The following documents were published under this contract:


1. D. Mylaraswamy, "Vehicle Level Reasoning and Data Mining," CIDU, 2010.
2. T. Felke, G.D. Hadden, D. Miller, and D. Mylaraswamy, "Architectures for Integrated Vehicle Health Management," AIAA-2010-3433, 2010.
3. G.D. Hadden, D. Mylaraswamy, C. Schimmel, G. Biswas, X. Koutsoukos, and D.L.C. Mack, "Vehicle Integrated Prognostic Reasoner (VIPR) 2010 Annual Final Report," NASA/CR–2011-217147 http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110012171_2011012606.pdf
4. R. Bharadwaj, D. Mylaraswamy, D. Cornhill, C. Schimmel, G. Biswas, X. Koutsoukos, and D.L.C. Mack, "Vehicle Integrated Prognostic Reasoner (VIPR) 2011 Annual Final Report".

5. G. Biswas, X. Koutsoukos, D. Mylaraswamy, and G.D. Hadden, “Benchmarking the Vehicle Integrated Prognostic Reasoner,” Annual Conference of the Prognostics and Health Management Society 2010.
6. D. Mylaraswamy, et al., “Vehicle Integrated Prognostics Reasoning,” NASA Aviation Safety Annual Meeting, St Louis, 2011.
7. D.L.C. Mack, G. Biswas, X.D. Koutsoukos and D. Mylaraswamy, “Using Tree Augmented Naive Bayesian Classifiers to Improve Engine Fault Models,” presented at the 8th Bayesian Modeling Applications Workshop in Barcelona, Spain on July 14th, 2011.
8. Srivastava, D. Mylaraswamy, R.W. Mah, E.G. Cooper, “Vehicle-level Reasoning Systems” in *IVHM Perspectives on an Emerging Field*, Ed. I.K. Jennions. Publisher: SAE International. 2011.
9. D.L.C. Mack, G. Biswas, X.D. Koutsoukos, D. Mylaraswamy, and G.D. Hadden, “Deriving Bayesian Classifiers from Flight Data to Enhance Aircraft Diagnosis Models,” presented at the Annual Conference of the Prognostics and Health Management Society, 2011.
10. D. Mylaraswamy, “Addressing Aviation Safety using Vehicle Level Reasoning,” 1st Indo-US Workshop on IVHM and Aviation Safety (WIAS), NAL Bangalore, 2012.
11. Srivastava, D. Mylaraswamy, R.W. Mah, E.G. Cooper, “Vehicle-level Reasoning Systems” in *IVHM Perspectives on an Emerging Field*, Ed. I.K. Jennions. Publisher: SAE International. 2011.
12. R. Bharadwaj, et al., “Vehicle Level Prognostic Reasoning System,” talk at SAE 2012 Aerospace Electronics and Avionics System Conference, Oct 30--Nov 1, 2012. Phoenix, AZ

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| 01-03-2013 | Contractor Report | |

**4. TITLE AND SUBTITLE**

Vehicle Integrated Prognostic Reasoner (VIPR) Final Report

**5a. CONTRACT NUMBER**

NNL09AA08B, NNL09AD44T

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Bharadwaj, Raj; Mylaraswamy, Dinkar; Cornhill, Dennis; Biswas, Gautam; Koutsoukos, Xenofon; Mack, Daniel

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

534723.02.03.07

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

NASA Langley Research Center
Hampton, Virginia 23681

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Washington, DC 20546-0001

**10. SPONSOR/MONITOR'S ACRONYM(S)**

NASA

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

NASA/CR-2013-217972

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

Unclassified - Unlimited
Subject Category 06
Availability: NASA CASI (443) 757-5802

**13. SUPPLEMENTARY NOTES**

Langley Technical Monitor: Eric G. Cooper

**14. ABSTRACT**

A systems view is necessary to detect, diagnose, predict, and mitigate adverse events during the flight of an aircraft. While most aircraft subsystems look for simple threshold exceedances and report them to a central maintenance computer, the vehicle integrated prognostic reasoner (VIPR) proactively generates evidence and takes an active role in aircraft-level health assessment. Establishing the technical feasibility and a design trade-space for this next-generation vehicle-level reasoning system (VLRS) is the focus of our work.

**15. SUBJECT TERMS**

Aircraft health; Anomaly detection; Data mining; Prognostics; Reasoner

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | STI Help Desk (email: help@sti.nasa.gov) |
| | | | | | 19b. TELEPHONE NUMBER *(Include area code)* |
| U | U | U | UU | 45 | (443) 757-5802 |