NASA/CR—2005-213813

AIAA–2005–4486

Performance Evaluation of a Data Validation System

T. Shane Sowers
Analex Corporation, Brook Park, Ohio

L. Michael Santi
Christian Brothers University, Memphis, Tennessee

Randall L. Bickford
Expert Microsystems, Inc., Orangevale, California

June 2005

The NASA STI Program Office . . . 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 Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published 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's counterpart of peer-reviewed formal professional papers but has 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 cosponsored 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 that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

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

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2005-213813 AIAA–2005–4486

Performance Evaluation of a Data Validation System

T. Shane Sowers
Analex Corporation, Brook Park, Ohio

L. Michael Santi
Christian Brothers University, Memphis, Tennessee

Randall L. Bickford
Expert Microsystems, Inc., Orangevale, California

Prepared for the
41st Joint Propulsion Conference
cosponsored by the AIAA, ASME, SAE, and ASEE
Tucson, Arizona, July 10–13, 2005

Prepared under Contract NAS3–00145

National Aeronautics and
Space Administration

Glenn Research Center

June 2005

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

This report contains preliminary findings, subject to revision as analysis proceeds.

This report is a preprint of a paper intended for presentation at a conference. Because of changes that may be made before formal publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

Trade names or manufacturers' names are used in this report for identification only. This usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Performance Evaluation of a Data Validation System

T. Shane Sowers
Analex Corporation
Brook Park, Ohio 44142

L. Michael Santi
Christian Brothers University
Memphis, Tennessee 38104

Randall L. Bickford
Expert Microsystems, Inc.
Orangevale, California 95662

Online data validation is a performance-enhancing component of modern control and health management systems. It is essential that performance of the data validation system be verified prior to its use in a control and health management system. A new Data Qualification and Validation (DQV) Test-bed application was developed to provide a systematic test environment for this performance verification. The DQV Test-bed was used to evaluate a model-based data validation package known as the Data Quality Validation Studio™ (DQVS). DQVS was employed as the primary data validation component of a rocket engine health management (EHM) system developed under NASA's NGLT (Next Generation Launch Technology) program. In this paper, the DQVS and DQV Test-bed software applications are described, and the DQV Test-bed verification procedure for this EHM system application is presented. Test-bed results are summarized and implications for EHM system performance improvements are discussed.

Nomenclature

DQV	Data Qualification and Validation
DQVS	Data Quality Validation Studio
EHM	Engine Health Management
GUI	Graphical User Interface
NGLT	Next Generation Launch Technology
VBA	Visual Basic for Applications

I. Introduction and Background

Online data qualification and validation (DQV) is a performance-enhancing component of a modern control or health management system. The quality of data used as the basis for control decisions can be affected by any measurement system component that converts physical system characteristics into quantitative digital information. Typical measurement system components would include transducers, signal conditioners, transmission hardware, and analog-to-digital conversion elements. The DQV system ensures that only valid information is provided to consumers of the data. To be effective for health management applications, a DQV system must be able to evaluate and qualify each measurement. Any erroneous sensor readings must be detected and isolated in a timely manner to prevent their consideration in health diagnosis, maintenance planning, and/or control decisions. This ensures the utilization of only good data and ultimately improves system availability and performance.

In order to establish confidence in a DQV system, its capabilities must be verified through a rigorous test program. Normal software validation procedures alone are not adequate. This is especially true for space propulsion health management applications with low to no tolerance for false alarms throughout the normal operating envelope. DQV system capabilities can be verified by characterizing the fidelity and timeliness of sensor fault detection and isolation.

An application referred to as the Data Qualification and Validation Test-bed (DQV Test-bed) was developed to provide a systematic, thorough, and quantitative method for assessing DQV system capability. To demonstrate this capability, the DQV Test-bed was used to characterize and evaluate performance of a specific DQV system termed

the Data Quality Validation Studio™ (DQVS), as applied to a reusable and throttleable liquid fueled rocket engine. DQVS is a software package developed by Expert Microsystems, Inc. of Orangevale, California and performance testing was conducted using simulations of a reusable engine system being developed as a boost stage platform under NASA's NGLT (Next Generation Launch Technology) program.

A. DQVS Software

The Data Quality Validation Studio™ software automates the production of online signal data validation modules that detect sensor failures and other data anomalies. The software consists of a data validation model development environment and a stand-alone runtime kernel. The development environment provides the workstation-based tools used to define the process model and decision strategy used by the runtime kernel to detect data quality problems in an online data stream. The development environment includes an autocode generator that automatically produces the model files used by the runtime kernel. The runtime kernel in conjunction with the autocode-generated process model and decision strategy files create an embeddable runtime module that can be integrated with a host control and monitoring system. The development environment is used for initial production and maintenance of the embeddable runtime modules that perform the real-time validation function in the specific process environment.

1. Model Based Reasoning Algorithm.—The DQVS software's model based validation algorithm combines analytic redundancy with Bayesian decision logic. Analytical redundancy is a technique that predicts the value of a signal using values of other signals and known or empirically derived mathematical relations. System design relationships and signal redundancies provide many of these relations. Empirical relations derived using techniques such as statistical analysis, pattern recognition and neural networks are also employed.

A set of signals and a set of relations form a network of cross checks used to validate all signals in the model. The difference between a signal's predicted value and its directly sensed value is termed a residual. The probability of a relation holding, given that all contributing signals are valid, is determined statistically by placing a threshold value on the relation residual. Threshold values for the relation residuals are pre-computed using nominal system operating data. Threshold values can be adjusted for normal run-to-run variations in the process signals using a procedure termed relation biasing. Biasing is accomplished during a short user defined interval at the start of each new monitoring phase. The software automatically computes bias offsets for signal validation relations over this short interval using special threshold and bias limiting logic in order to prevent faulted signal characteristics from being "learned" by the biasing algorithm.

2. Limit Filtering Algorithms.—Limit filters provide fast, simple validation coverage for gross signal failures, such as open or short circuit sensor failures and excessive noise. Limit filters compare a signal value to a pre-established nominal value and alarm when a threshold value is exceeded. The software implements a hierarchy of range limits, statistical noise limits, maximum step size limits, and missing signal checks that serve as a data validation preprocessor for the more sensitive model based validation algorithm. Each limit filter may be enabled for one or more process operating phases. The software provides analytical tools to statistically determine the appropriate limit filter threshold settings based on nominal system operating data.

The limit filtering algorithms monitor different characteristics of the sensor signal. Range filters monitor for upper and/or lower limits on the value of a signal. The limit values expected for the signal and the number of consecutive fault alarms required to generate a failure indication provide range violation detection capability. Delta filters monitor for positive and/or negative limits on a step change in a signal's value. Noise filters monitor upper and/or lower limits on the standard deviation value of a signal. A localized line fit to the data may be used in order to remove value trends from the standard deviation calculation. Flat filters monitor for signals that have become non-responsive while continuing to return values within their valid range of operation.

B. Engine System Application

The objective of the DQVS application was to provide data validation/qualification for a liquid rocket engine system under development as a boost stage platform for NASA's NGLT program. The engine system was to be reusable with a service life of 100 missions. Staged combustion was to be employed with a throttling capability. The propellants were to be a mixture of liquid oxygen and RP-1 (kerosene).

II. Data Sources

To facilitate development of a DQVS model, training data that spans the normal engine operating envelope are required. These data are used to calibrate an empirical model of nominal engine system function. The empirical engine model is then used to identify sensors and/or engine components that exhibit anomalous behavior outside of

designated bounds. Because DQVS models are empirically derived, training data quality directly impacts DQVS performance capabilities. These models derive their learned definition of nominal system behavior entirely from the training data. Operating characteristics not represented in the training data will appear abnormal relative to the learned behavior. Several sources of training data may be available, each presenting advantages and disadvantages for engine health management (EHM) system development.

An excellent source of data is produced from actual firing of an engine, either on a test stand or in flight. This source provides engine data considered most consistent both in format and condition with the target system. Generally speaking, hot-fire data reflect the cumulative effects of physical characteristics not modeled well or at all in typical engine computer simulations. These effects include higher order spatial dependence and small time scale transients. Hot fire data are, however, very costly to obtain, which contributes to limited testing and restricted operating envelope coverage. Depending on engine complexity, test setup time may also be significant resulting in reduced coverage due to schedule constraints. Furthermore, hot fire data must be carefully evaluated to remove all errors or anomalies prior to use for model training.

Obtaining data for an engine component degradation or failure during a normal hot-fire test program can be challenging. Whereas failures of sensors not utilized by the engine controller can be simulated in conjunction with hot fire data, a similar mixed data approach is not viable for engine hardware faults. Engine anomalies are usually difficult to accurately simulate given their impact on sensed conditions throughout an engine system. In general, engine fault data are only observed when an anomaly coincidentally occurs during an engine firing sequence. Planned seeding of engine component failures is generally considered impractical due to the risk to the costly engine system and test facility. In addition, if a fault of sufficient criticality happens to occur during a firing, the engine system is typically redesigned to avoid the recurrence of that specific failure mode. Therefore, once a critical failure signature has been observed, it is unlikely to reoccur in the same manner. This reduces the value of the observed failure signature for training engine health management systems.

Engine model based simulations provide another source of data; however, they are burdened by the need to verify physical consistency. Engine model simulation results are questionable prior to model calibration using test results because the fidelity of the engine model is usually constrained by available resources and understanding of physical phenomena. However, with sufficient resource investment, engine models can provide a valuable data generation platform. Model based simulations designed to facilitate EHM development might further include the ability to insert sensor and engine component faults. This provides a method for estimating sensor suite signatures of well-defined engine component faults. As previously mentioned, this is not easily accomplished with hot-fire testing.

The number and types of faults that can be simulated with models is constrained by model fidelity, computational complexity and processing resources. Even the most complex models provide only limited failure mode coverage due to limited knowledge and representation of fault propagation phenomena and resulting system interactions. Therefore, fault simulations are typically restricted to those that are well understood and have the largest risk reduction potential. Fortunately, constraints related to computational complexity and processing resources continue to diminish as computer capability improves.

Because this study was conducted in the early stages of engine development, the only viable source of engine system performance data was computer simulation. With engine components at various stages of design and fabrication, hot-fire data were unavailable. An engine model constructed to support the development process was employed. This model provided the basic simulation platform from which nominal and off-nominal engine data were produced for DQVS model development and testing.

The engine model utilized knowledge from heritage engine programs. Without authentic hot-fire data, engine performance parameters in the simulator were not completely characterized. Research into heritage engine system programs provided the bridge for this information gap. Because requirements for the target engine system made it distinctive from any predecessor, information was gathered from a diverse set of heritage engine systems.

Simulated measurement readings from engine model output were categorized by sensor source into control, redline and/or health management groups. The control sensors were those utilized by the engine controller to infer the actual performance level of the engine system. Traditional redline sensors were designated to indicate hazardous operating conditions requiring an emergency engine abort command. A sensor selection study was conducted to designate the health management sensors.[1] The study selected a near-optimum sensor suite based on the ability to diagnose a targeted set of engine fault conditions. Some of the health management sensors were also members of the control or redline sensor groups.

Engine simulation output was augmented to reflect anticipated sensor suite configuration and signal characteristics. These considerations are outlined below.

1) Effects of random signal noise were added to each measurement. Signal noise was assumed Gaussian and noise statistics for individual sensors were estimated using heritage engine system data. A Gaussian random number generator reflecting these statistics was used to generate the sensor signal noise values.
2) Outputs from control and redline sensors were simulated independently for each redundant channel and/or sensor. Health management sensors were assumed to be non-redundant.
3) For individual simulation runs, a sensor bias value was applied to each redundant sensor channel. Sensor bias statistics were estimated using data from an ensemble of heritage engine system tests, and assuming Gaussian ensemble statistics. A Gaussian random number generator reflecting these ensemble statistics was employed to estimate the sensor bias value. Once computed, the bias value remained constant for each redundant sensor throughout a simulation run.

Other characteristics of the engine model included the following:

1) Dynamic system response simulation allowed effects of transient inputs such as changes to the commanded thrust level and engine fault conditions to be examined.
2) Engine-to-engine performance variation simulated by incorporating estimated build (i.e., manufacturing) variation in engine component performance characteristics.
3) Properties of engine component failure or degradation conditions scripted within simulations.
4) Commanded control profile defining engine control sequence scripted within simulations.
5) User defined engine propellant inlet conditions. The Mission baseline inlet propellant property profile was used without modification.
6) Uniform data sampling rate and unlimited data resolution assumed in the absence of specific guidelines.
7) Engine model executed in reasonable "wall-clock" time. The time to conduct a simulation was not a constraint.

III. The Performance Evaluation Process

The evaluation process was designed to meet three fundamental objectives. First, the capabilities and limitations of the DQV Test-bed as a comprehensive and cost effective verification and validation test platform were to be demonstrated. Next, the performance of the DQVS as a data qualification and validation model was to be characterized and evaluated. Finally, the robustness of the DQVS as a general engine health monitoring support component was to be characterized. This last evaluation objective required measures of DQVS ability to: discriminate sensor faults from engine system anomalies; and perform timely faulted sensor disqualifications relative to propagating engine system fault diagnostic timeframes.

It was considered essential to conduct the evaluation procedure independent of the DQVS model developer. Having an independent party conduct such an evaluation is a standard software industry practice. It was even more important in this situation because proprietary data concerns prevented the DQVS model developer from having access to an on-site data generation capability. Whereas it was feasible to transfer a relatively small number of nominal data files to the DQVS model developer for development, it was impractical to transfer a large number of data files for extensive performance testing. Conducting an independent evaluation where all of the necessary data spanning a large range of nominal and faulted operating conditions could be generated was both sensible and efficient.

An important factor in the evaluation was the expected range of operational conditions. Anticipated variances in hardware manufacturing and assembly processes were expected to admit considerable engine-to-engine variation in normal state measurements. Combined with typical run-to-run operational variation, this suggested that sensor values would vary greatly over the normal operating envelope. In addition, multiple operating states were anticipated for the normal command control profile. Taken together, these conditions suggested that simply recognizing nominal engine operation would be challenging.

As the test program was being constructed, four fundamental test categories were defined.

1) ***Nominal (No Faults)***.—Tests in this category examine resistance to false alarms due to normal engine-to-engine and run-to-run variations. A data validation system with a high false alarm rate given nominal data would not be acceptable.
2) ***Sensor Faults***.—Tests in this category examine how well a sensor fault is detected. Of course, this is the primary objective of a data validation system. Types of sensor faults include sudden shifts, gradual drifts, flat line signals, and signals containing excessive or abnormally low levels of noise.

3) ***Engine Component Faults***.—Tests in this category examine the behavior of the data validation system in the presence of engine component faults. This was used to characterize the ability to discriminate an engine component failure from a sensor failure. This discrimination is challenging for instances where a component failure and a sensor failure have a similar effect on the sensor data.
4) ***Sensor and Engine Component Faults***.—Tests in this category examine the complicated situation when both sensor and hardware faults are present. This was used to characterize the ability to discriminate sensor faults from other abnormal operating conditions. Once again, this discrimination is challenging for instances where a component failure and a sensor failure have a similar effect on the sensor data.

For initial development of the test-bed, a simplified test program was conducted using the first three fundamental fault categories. The fourth category was to be examined in later test sequences as experience was gained. In the condensed program, faulted function inputs were based upon statistics of the sensor signals. Sensor failures were implemented as a drift of one standard deviation of the signal per second. Engine component faults were implemented as shifts in hardware performance sufficient to produce a change of five times the standard deviation of at least one sensor signal. In the test sequence, all faults were initiated at the same point in time relative to engine start, which was early in the simulation run during steady state operation. This allowed the fault time to progress and provide insight into behavior during transients in the control profile. All input parameters were specified to be as "flight-like" as possible to be compatible with the DQVS training data.

The test program was organized into three sections for each test category:

1) Ten nominal cases based on normal engine-to-engine and run-to-run variability.
2) Two fault cases for each sensor not in the control group using a positive and negative drift in the sensor signal overlaid on engines exhibiting normal engine-to-engine and run-to-run variability.
3) One fault case for each engine component manifested as a sudden step degradation in performance, again overlaid on engines exhibiting normal engine-to-engine and run-to-run variability.

After determining the resources required to manually conduct the simplified test program, a decision was made to develop an automated evaluation capability. With more exhaustive test sequences being planned, the need for automation of the evaluation procedure was clear. Therefore, a computer program environment was developed that integrated the engine simulation and the DQVS model into a software test-bed environment. This was the genesis of the application herein termed the Data Qualification and Validation Test-bed.

IV. The Data Qualification and Validation Test-bed

The DQV Test-bed was designed to enable a comprehensive performance review of a data validation/qualification system. Its primary design feature is the automation of the evaluation process enabling an extensive and meticulous test program. An object-oriented design was applied to further the utility of the Test-bed by allowing easy modification of the evaluation process. This modular approach makes possible the incorporation of various health management technologies with a minimum of development effort. An overall diagram of the functionality flow of the Test-bed is given in figure 1. The Test-bed operates on a single Windows/PC platform and utilizes data files for data transfer between the integrated components.

The test sequence to be conducted is described by the Test Plan file. This file is formatted in ASCII text for readability and characterizes properties of each test case in the overall test series. It describes how the data will be generated, and what, if any, failures are to be considered. The general format of the Test Plan file is given in table 1. File format can be modified to accommodate changes in the evaluation process.

A Microsoft Excel spreadsheet referred to as DQVT.XLS functions as the graphical user interface (GUI). It permits the user to create and populate the Test Plan file, and then to execute the test protocol and evaluation procedure. At the conclusion of the evaluation procedure, an overall results summary is displayed in a separate Excel spreadsheet with a title consistent with the test sequence name given in the Test Plan. The results summary spreadsheet is formatted to be concise, informative, and easy to read. Functionality of DQVS.XLS is provided by macros written in VBA (Visual Basic for Applications) within Excel.

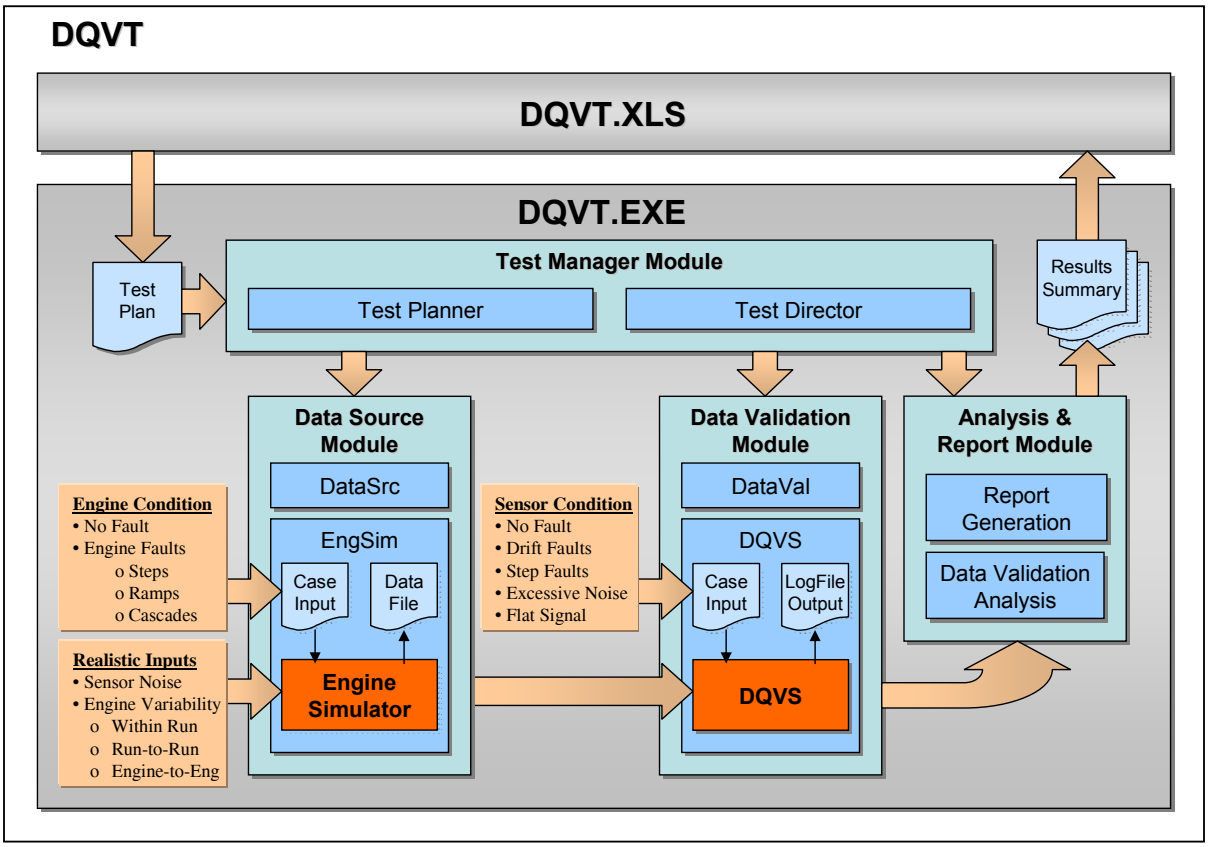

Figure 1.—DQV Test-bed Software Diagram *Functionality flow of the DQV Test-bed*.

TABLE 1.—TEST PLAN FILE FORMAT

Field Name	Description
Test Series Name	Name of the test series
Number of Test Cases	Total number of test cases
(For Each Test Case)	
Data Generation Needed	Binary value to indicate if new data needs to be generated. If not, then the previous data file will be used. This allows the same data file to be tested with different conditions.
Number of Faults	Total number of failures in the test case
(For Each Fault)	
Fault Category	Either a sensor or engine component failure
Fault Identifier	Object to be failed
Fault Mode	How the object fails (e.g. shift, drift, etc)
Fault Time	When the object fails during the simulation
Fault Magnitude	Magnitude of the failure
Fault Location	Location within an object where the failure occurs (not always applicable)
Fault Duration	Time necessary for the failure to reach full magnitude (not always applicable)

The evaluation procedure, as defined by the Test Plan, is performed by the component termed the DQVT.EXE. It is the software environment that integrates the engine simulator and the DQVS model. DQVT.EXE was written in C++ and is comprised of four major software modules.

A. Test Manager Module

The Test Manager controls the overall execution of DQVT.EXE. It consists of the Test Planner and the Test Director. The Test Planner uses the Test Plan to provide information on the conduct of the test series as defined by the user. The Test Director executes the test series.

B. Data Source Module

The Data Source Module provides performance data to the DQV Test-bed. An interface layer, termed DataSrc, was created to permit a variety of data sources to be employed. In this application, the data source was time dependent sensor output from an engine model simulation and a specialized interface layer was created to manage it. All engine component hardware faults were simulated in the engine model.

C. Data Validation Module

The Data Validation Module connects the DQVS model with the Test-bed environment. An interface layer, termed DataVal, was used to manage the DQVS model. Modification of the interface layer provides the ability to integrate other health management algorithms. All sensor faults were simulated using embedded features of the DQVS model. It should be noted that control sensor faults were not simulated. Doing so would require a feedback channel to the controller in the engine model that was not implemented.

D. Analysis and Report Module

The Analysis and Report Module compares the DQVS model results to the test conditions described in the Test Plan. At the end of each test case, a report is produced. When all of the test cases have been completed, an overall summary report is generated. An example of the format of each report can be found in tables 2 and 3.

TABLE 2.—TEST SERIES SUMMARY REPORT FORMAT

Format of report generated at the conclusion of a test series. This report includes the results of all test cases in the test sequence.

Name	Description
Test Series Name	Name for the test series.
(For Each Test Case)	
Test Case Number	Represents the test case number within the sequence of the test series.
(For Each Inserted Fault)	
Name of Fault	Object that was failed during the simulation.
Fault Mode	Type of failure (e.g. shift or drift) inserted into the simulation.
Fault Time	Simulation system time when the failure occurred.
Fault Magnitude	Failure magnitude inserted into the simulation.
(For Each Detected Fault)	
Fault Name	Object that was detected as failing during the simulation.
Detection Time	Simulation system time when the failure was detected.
Baseline Value	Reference baseline value of the object that was failed.
Standard Deviation Value	Reference standard deviation value of the object that was failed.
Magnitude Value at Detection	Value of the failure at the time when it was detected.
Test that Detected the Failure	Method by which the failure was detected.
Value at Detection as Percentage of Standard Deviation	Detection value expressed as a multiple of the nominal standard deviation value.
Total Count of Correct No Fault Detections	Count of test cases that did not have any faults inserted and did not have any faults detected.
Total Count of Detected Faults	Count of faults inserted and correctly detected across all test cases.
Total Count of Missed Faults	Count of faults inserted, but not detected across all test cases.
Total Count of False Alarms	Count of fault alarms not explained by faults inserted in the test data.

TABLE 3.—TEST CASE SUMMARY REPORT FORMAT
Format of report generated at the conclusion of each test case.

Name	Description
Test Case Name	Name of the test series.
Number of Faults Inserted	Number of faults inserted into the test case.
(For Each Inserted Fault)	
Name of Fault	Object that was failed during the simulation.
Fault Mode	Type of failure (e.g. shift or drift) inserted into the simulation.
Fault Time	Simulation system time when the failure occurred.
Fault Magnitude	Failure magnitude inserted into the simulation.
(For Each Detected Fault)	
Fault Category	Indicates if a sensor or engine component failure was detected.
Fault Identity	Object that was perceived as failing.
Detection Time	Simulation system time when the failure was detected.
Detection Magnitude	Failure magnitude at the time of detection.
Correct No Fault Detection	Indicates if no faults were inserted and no faults were detected in the test case.
Correct Fault Detections	Count of faults inserted and correctly detected in the test case.
Missed Fault Detections	Count of faults inserted, but not detected in the test case.
False Alarm Detections	Count of fault alarms not explained by faults inserted in the test data.

V. Results

The DQV Test-bed was used to evaluate the behavior of two versions of the DQVS model each designed for a specific diagnostic sensor suite. Because the optimal diagnostic sensor suite was modified during engine development, a second DQVS model was generated to accommodate these changes. Each of the DQVS models was evaluated using the DQV Test-bed test protocol given in table 4. This test protocol was executed four times. The first pair of runs was performed with engine component variability initially enabled and then disabled. This provided insight into the effects of engine-to-engine variability on the DQVS model. It is worth noting that with engine component variability disabled, the engine model produced results representative of the anticipated baseline production engine and neither of the DQVS models were trained with data from this baseline engine. These results allowed a focus on the effects of run-to-run variability without engine-to-engine variability. Finally, the pair of test protocol runs was executed a second time (Test Series 2 in tables 5 and 6). The second set of runs was performed as a consistency check on the first set of results.

TABLE 4.—TEST PLAN TEMPLATE EMPLOYED

Category	Description
Category 1	Ten nominal runs with random engine component performance variation.
Category 2	One sensor drift in positive direction by one standard deviation per second. One sensor drift in negative direction by one standard deviation per second.
Category 3	One engine component performance degradation shift sufficient to produce a change of five times the standard deviation in at least one sensor signal.

During development of the DQVS models, two methodologies were employed to generate training data. The first method, employed in development of the original DQVS model, used training data generated by selecting engine component characteristics for each simulation run randomly within the accepted performance range. A modified method using a hybrid approach was employed to generate training data for the second DQVS model. In the modified approach, a portion of the training data was generated initially by selecting engine component characteristics randomly within the accepted performance range. These initial training data were augmented by data from runs in which each individual engine performance parameter was first set to the positive limit of its normal variation range and then set to the negative limit of this range with the other performance parameters held at their baseline values. The objective of this method was to include extreme variation corners of the nominal engine performance envelope in the training data.

The DQV Test-bed provided a much more thorough and insightful characterization of the DQVS models than was possible during developer testing. During developer testing, both models performed well, but the test procedure was restricted. The developer did not have an on-site data generation capability and was therefore constrained by the number of nominal simulation data sets provided. Consequently, all developer tests were performed using data from

a single simulation run deemed typical of the testing data and only sensor failures that could be simulated in the DQVS were examined. When each model was challenged more thoroughly using the DQV Test-bed, localized weaknesses were exposed and areas for additional development were identified. This clearly demonstrated the importance and utility of the cost effective and comprehensive testing provided by the DQV Test-bed. In the future, model development and testing steps will be performed iteratively until diagnostic performance is demonstrated to meet system requirements for all test scenarios.

Further post-test analysis was performed to characterize the results into the following categories.

1) ***Correct Fault Detection and Isolation***.—A fault condition was detected and isolated to the one correct fault.
2) ***Fault Detection and Partial Correct Isolation***.—A fault condition was detected and isolated to a fault candidate list that included the correct fault.
3) ***Fault Detection, but Incorrect Isolation***.—A fault condition was detected, but isolated to a fault candidate list that did not include the correct fault.
4) ***No Fault Detection***.—A fault condition was not detected.
5) ***False Fault Detection***.—A fault condition was incorrectly detected when a fault was not inserted.

Condensed tables of results are provided in tables 5 and 6, and a summary of findings is provided below.

1) The use of redundancy networks in the DQVS model improved sensor failure detections. The average time required to detect and isolate a fault was enhanced over the sole use of sensor signal limit filters. It is worth noting that all of the sensor failures were inserted as sensor value drifts at a rate of one standard deviation per second. Therefore, the results in table 5 can also be considered as fault detection magnitudes in terms of standard deviations of the sensor signal.
2) The performance of the second DQVS model showed some improvement in timing and fidelity over the first model. There was an improvement in the time for fault detection and isolation, and in the number of correct fault detections and isolations. The improvement can be attributed to differences in the training data generation methodologies as well as the number and locations of sensors employed. Further study is required to determine the individual contributing effect of these changes.
3) Enabling engine-to-engine variability had an influence on the performance of the DQVS model. Primarily, it reduced correct fault isolations while raising partial correct and incorrect fault isolations. It also increased the number of faults detected before fault insertion. These results suggest that engine variability will inhibit fault discrimination. They also indicate that additional training data will be needed to counter variability effects and better characterize the variability envelope.
4) The large number of missed detections for engine component faults indicate that the DQVS model was not sensitive to engine component fault signatures at the threshold fault levels employed. This result implies that threshold engine component fault signatures did not consistently rise above the expected engine-to-engine and/or run-to-run variability in the DQVS training data.
5) The DQVS model was not expected to identify specific engine component faults. A partial correct fault isolation in the case of an engine component fault was considered a positive result. A result of this type effectively supports diagnostic indications from other active health monitoring components specifically charged with detecting and isolating engine component faults.

TABLE 5.—AVERAGE TIME FOR SENSOR FAULT DETECTION AND ISOLATION PER METHODOLOGY

Time from sensor fault insertion to correct fault detection and isolation in seconds. Time is classified by detection methodology and averaged across the test series. Missed detections and incorrect fault identifications are not considered. Because the inserted faults were sensor value drifts at a rate of one standard deviation per second, time in seconds can also be considered as fault magnitudes in terms of standard deviations of the sensor signal.

Seconds Equivalent to Standard Deviations	**Engine Variability Disabled**				**Engine Variability Enabled**			
	DQVS Rev. 1		**DQVS Rev. 2**		**DQVS Rev. 1**		**DQVS Rev. 2**	
Sensor Fault Detection and Isolation Method	**Test Series 1**	**Test Series 2**	**Test Series 1**	**Test Series 2**	**Test Series 1**	**Test Series 2**	**Test Series 1**	**Test Series 2**
Threshold Limit Exceeded	25.35	24.12	18.90	19.12	23.90	25.10	20.67	20.62
Relationship Failed	10.92	10.88	8.05	8.07	10.57	10.78	7.84	8.05

TABLE 6.—DQV TEST-BED RESULTS WITH ENGINE COMPONENT VARIABILITY DISABLED AND ENABLED

	Engine Variability Disabled				Engine Variability Enabled			
	DQVS Rev. 1		DQVS Rev. 2		DQVS Rev. 1		DQVS Rev. 2	
	Test Series 1	Test Series 2	Test Series 1	Test Series 2	Test Series 1	Test Series 2	Test Series 1	Test Series 2
Category 1.—Nominal Runs (No Fault Inserted)								
Percent of Runs with No Fault Detection	100%	90%	90%	100%	80%	60%	90%	90%
Percent of Runs with False Fault Detection	0%	10%	10%	0%	20%	40%	10%	10%
Category 2.—Sensor Fault Inserted								
Percent of Runs with Correct Fault Detection and Isolation	83%	86%	91%	93%	67%	63%	75%	71%
Percent of Runs with Fault Detection & Partial Correct Isolation	8%	5%	3%	1%	22%	26%	15%	19%
Percent of Runs with Fault Detection, but Incorrect Isolation	1%	1%	0%	0%	5%	3%	5%	5%
Percent of Runs with No Fault Detection	8%	8%	6%	6%	6%	8%	5%	5%
Category 3.—Engine Component Fault Inserted								
Percent of Runs with Correct Fault Detection and Isolation	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Percent of Runs with Fault Detection & Partial Correct Isolation	26%	20%	29%	29%	29%	32%	32%	35%
Percent of Runs with Fault Detection, but Incorrect Isolation	10%	3%	3%	3%	6%	13%	13%	10%
Percent of Runs with No Fault Detection	64%	77%	68%	68%	65%	55%	55%	55%

VI. Summary

The DQV Test-bed proved to be an effective tool for cost effective and comprehensive testing of health monitoring systems. It provided an efficient avenue for assessing the strengths and weaknesses of a prototype data validation and fault detection system. In the future, model development and testing steps will be performed iteratively until performance of the health monitoring system is demonstrated to meet system requirements for all operating scenarios. By using the DQV Test-bed, an improved understanding of capabilities and trade-offs was achieved and incremental improvements were initiated. Unfortunately, the project was abruptly halted when the NGLT program was terminated and these initial test results could not be used immediately as a guide for additional development of DQVS models. As the result of this work, the DQV Test-bed is being upgraded to serve as a general-purpose testing tool for a variety of future NASA engine health monitoring development programs. The DQV Test-bed provides a potentially useful and important element of the necessary infrastructure to iteratively develop and flight qualify future engine health monitoring systems.

VII. Recommendations

From the beginning, the DQV Test-bed was designed to be modular. The principal reason was to make the Test-bed generic so that various health monitoring technologies could be quickly and easily “plugged-in”, exercised and evaluated. The structure of the Test-bed supports this objective and a more flexible architecture is now under development. This will allow the inclusion of data from other engine simulators and from hot-fire tests as they became available.

Whereas a graphical user interface is provided by an Excel spreadsheet, a traditional “Windows” look-and-feel is desired and should be developed. Requirements of this graphical application would include:

1) Allow the user to easily select, edit, save, and execute a test plan.
2) Provide reference information (e.g. baseline or standard deviation values) to the user during editing.
3) Allow the user to easily modify the evaluation procedure elements by selecting the source of test data and components to be tested.
4) Permit distributed computing where the elements of the evaluation procedure are executed on separate processors or on networked computers.
5) Allow the user to define the metrics used in the evaluation procedure.

The range and maturity of considered fault cases need to be advanced. Random time-variant sensor fault components (e.g. spikes and oscillations), and changes in sensor dynamic response characteristics (e.g. lag times) should be considered. Improved engine fault simulations are needed to realistically challenge and characterize DQV system performance and thereby support the development of EHM systems that are flight certifiable.

Additional trade studies to better define qualification test decision parameters are needed. This will support refinement of DQV processes and improve diagnostic performance evaluation.

A more thorough simulation plan for training is needed to characterize the normal engine operating envelope. Run-to-run engine variability must be well understood for effective multi-start engine data validation. The simulation test results and the large degree of freedom present in a health management system both suggest a Monte Carlo training approach. When challenging regimes are identified, a staged refinement process will likely be necessary.

Empirically-derived models depend on high-quality, comprehensive training data to produce accurate and consistent results. In-depth testing to identify shortcomings in the training or performance of empirically-derived models will be an essential task in the development of flight certifiable systems. The model training and testing tasks are likely to be iterative, which emphasizes the critical importance of developing highly efficient, automated tools for both model development and model testing.

References

[1]Santi, L. Michael, Sowers, T. Shane, and Aguilar, Robert, "Optimal Sensor Selection for Health Monitoring Systems," 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Paper No. AIAA 2005–4485, Tucson, Arizona, 10–13 July 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2005	Final Contractor Report

4. TITLE AND SUBTITLE

Performance Evaluation of a Data Validation System

5. FUNDING NUMBERS

WBS–22–794–20–12
NAS3–00145

6. AUTHOR(S)

T. Shane Sowers, L. Michael Santi, and Randall L. Bickford

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

Analex Corporation
1100 Apollo Drive
Brook Park, Ohio 44142

8. PERFORMING ORGANIZATION REPORT NUMBER

E–15170

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

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2005-213813
AIAA–2005–4486

11. SUPPLEMENTARY NOTES

Prepared for the 41st Joint Propulsion Conference cosponsored by the AIAA, ASME, SAE, and ASEE, Tucson, Arizona, July 10–13, 2005. T. Shane Sowers, Analex Corporation, 1100 Apollo Drive, Brook Park, Ohio 44142; L. Michael Santi, Christian Brothers University, 650 E. Parkway S., Memphis, Tennessee 38104; and Randall L. Bickford, Expert Microsystems, Inc., 7932 Country Trail Drive, Suite 1, Orangevale, California 95662. Project Manager, Edmond Wong, Research and Technology Directorate, NASA Glenn Research Center, organization code RIC, 216–433–8917.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 20, 61, and 65

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Online data validation is a performance-enhancing component of modern control and health management systems. It is essential that performance of the data validation system be verified prior to its use in a control and health management system. A new Data Qualification and Validation (DQV) Test-bed application was developed to provide a systematic test environment for this performance verification. The DQV Test-bed was used to evaluate a model-based data validation package known as the Data Quality Validation Studio (DQVS). DQVS was employed as the primary data validation component of a rocket engine health management (EHM) system developed under NASA's NGLT (Next Generation Launch Technology) program. In this paper, the DQVS and DQV Test-bed software applications are described, and the DQV Test-bed verification procedure for this EHM system application is presented. Test-bed results are summarized and implications for EHM system performance improvements are discussed.

14. SUBJECT TERMS

Systems health monitoring; Qualification; Diagnosis; Program verification; Propulsion; False alarms; Error analysis; Performance tests; Simulation

15. NUMBER OF PAGES

17

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	