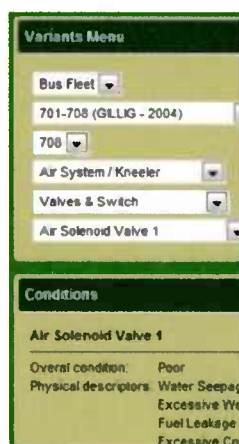




Office of Naval Research Grant N00014-07-1-0823

FINAL REPORT



R.I.T Defense Systems Modernization
and Sustainment Initiative

Center for Integrated Manufacturing Studies (CIMS)
Rochester Institute of Technology (RIT)
111 Lomb Memorial Drive ■ Rochester, NY 14623-5608
phone: (585) 475-6091 ■ fax: (585) 475-5455 ■ www.cims.rit.edu

20150922113

REPORT DOCUMENTATION PAGE					Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.						
1. REPORT DATE (DD-MM-YYYY) 31/03/2014		2. REPORT TYPE Final		3. DATES COVERED (From - To) 29-Mar-2007 to 31-Mar-2013		
4. TITLE AND SUBTITLE Defense Systems Modernization and Sustainment Initiative				5a. CONTRACT NUMBER N/A		
				5b. GRANT NUMBER N00014-07-1-0823		
				5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Nasr, Nabil, Z. Thurston, Michael, G. Haselkorn, Michael, H.				5d. PROJECT NUMBER		
				5e. TASK NUMBER		
				5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rochester Institute of Technology 7 Lomb Memorial Drive Rochester, NY 14623-5603				8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 495 Summer Street Suite 627 Boston, MA 02210-2109				10. SPONSOR/MONITOR'S ACRONYM(S) ONR		
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution is Unlimited. Copyright information included.						
13. SUPPLEMENTARY NOTES None						
14. ABSTRACT The objective of this work was to develop and demonstrate technologies that enhance the performance of defense weapons and support systems, while helping to monitor and control total ownership costs. Three different research and development areas were the focus: Sustainment Testbed; Platform Lifecycle Decision Support; and Remote Monitoring & Advanced Concepts. The outcomes of the study include technology to aid in the design of new military platforms, technologies to support efficient and effective platform operation, and technologies to extend the life of aging platforms.						
15. SUBJECT TERMS Military Weapon Platforms, Reliability and Maintainability Assessment, Sustainment Testbed, Platform Lifecycle Decision Support, Remote Monitoring, System Resilience and Survivability						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Nabil Nasr	
UNCLASS	UNCLASS	UNCLASS	U.U.	1	19b. TELEPHONE NUMBER (Include area code) 585-475-5106	



Office of Naval Research Grant N00014-07-1-0823

FINAL REPORT

Table of Contents

Executive Summary	ES1
Introduction to CIMS and RIT	IN1
Research Area 1: Sustainment Testbed	ST1
Research Area 2: Platform Lifecycle Decision Support	PLDS1
Research Area 3: Remote Monitoring and Advanced Support Concepts	RM&ASC1
Research Area 4: System Resilience and Survivability	SR&S1

R·I·T Defense Systems Modernization
and Sustainment Initiative

Center for Integrated Manufacturing Studies (CIMS)
Rochester Institute of Technology (RIT)
111 Lomb Memorial Drive ■ Rochester, NY 14623-5608
phone: (585) 475-6091 ■ fax: (585) 475-5455 ■ www.cims.rit.edu

2. Executive Summary

The Center for Integrated Manufacturing Studies (CIMS) at Rochester Institute of Technology was awarded the grant entitled *Defense Systems Modernization and Sustainment Initiative* (N00014-07-1-0823) from the Office of Naval Research (ONR) for the period March 29, 2007 to March 31, 2014.

CIMS' research focuses on developing and demonstrating technologies that enhance the performance of defense weapons and support systems, while helping to monitor and control total ownership costs. CIMS collaborates with a wide range of Department of Defense organizations, such as the Office of Naval Research, Naval Air Systems Command, Marine Corps Systems Command, the Naval Air Depots, Program Manager – Light Armored Vehicle, Marine Corps Combat Development Command, and Army Research Lab, in a wide variety of life-cycle engineering projects on several major weapons systems. In support of these diverse customers, CIMS applies programs to predict equipment health and failure, reliability and maintainability assessment, life-cycle technology insertion, material analysis technologies to predict the service life of systems and components, and data analytics. These efforts included technology to aid in the design of new military platforms, technologies to support efficient and effective platform operation, and technologies to extend the life of aging platforms.

The efforts conducted by CIMS are organized in terms of three different research and development focus areas: Sustainment Testbed; Platform Lifecycle Decision Support; and Remote Monitoring & Advanced Concepts. System Resilience & Survivability is another R&D area that was not pursued by CIMS due to the inability to find a partner for the proposed work. A number of projects were successfully completed during the contract period in the other three areas. Results in each focus area are described below.

Sustainment Testbed program is advancing technologies that will allow the Department of Defense to maintain vehicles beyond the end of their planned service life. CIMS performed multiple programs focused on understanding material aging and applying the knowledge to component restoration and material-based prognostics.

Through these efforts, CIMS has assisted in characterizing the aging of batteries both individually and in packs. As a result, strategies were identified that would allow individual cell replacement within a larger pack with only a slight degradation in pack performance, assuming the use of active cell balancing within the pack. These strategies can extend the life of packs with early cell failures. Additionally, CIMS evaluated methods of estimating cell capacity and state of charge. The techniques may be used to identify batteries that have considerable capacity remaining for use in secondary, less demanding applications.

CIMS also evaluated methods for remanufacturing military parts with needle-bearing contacts, such as steering boxes or pump shafts. These components are at risk of quick, catastrophic failures as the surfaces degrade. Currently, if these parts are inspected and

degradation is identified, the parts are replaced with new OEM components which can be costly. CIMS was able to show that a twin-wire arc thermal spray coating was a feasible and affordable approach to remanufacturing these contact critical surfaces.

Gear cracking is another potential catastrophic failure mechanism within many applications, such as planetary gear sets. CIMS was able to initiate cracks in gear teeth while utilizing an improved feature identification method that improved the ability to identify the crack initiation. Once the cracks have been detected, CIMS measured the propagation of the cracks in an effort to characterize the impact of microstructure on crack propagation. Utilizing this data, various models were tested in an attempt to estimate the time-to-death for all gears, ultimately resulting in identification of two models that could accurately predict time-to-death.

3D laser scanning can capture large quantities of point-cloud data which is useful for cast parts; however the accuracy is not fine enough for precision surfaces. CIMS was able to develop techniques for collecting 3D laser scanning model data and augmenting it with high measurement accuracy data from a CMM to reverse engineer complex parts with both critical and non-critical surfaces. Additionally, it was proven that the model could be utilized to generate a mold design for casting the component.

Finally, CIMS collaborated with the Program Manager for the Light Armored Vehicle (PM-LAV) to test a fuel catalyst that was advertised to decrease emissions and increase fuel efficiency. Utilizing an engine dynamometer, gas exhaust analyzer, precision fuel meter and a data acquisition system, CIMS was able to perform testing of the fuel catalyst at multiple engine test points (engine speed and torque settings). The tests showed that the catalyst produced no detectable changes in CO or NO, a significant increase in NO₂ was detected, and the efficiency increased by approximately 2.3%.

Platform Lifecycle Decision Support is a total system approach to optimizing life-cycle performance through planned modernization and step-by-step improvements in technological capabilities. LEEDS® was initially developed for the remanufacture of the Surface Effect Ship-200 through a project with ONR. Through various ONR funded projects, the capability of LEEDS® has been extended to provide engineering support throughout the platform life cycle.

During the contract period, CIMS re-evaluated the capabilities and users that LEEDS would be applicable to. The result of this analysis was a list of features, users and interfaces that would define what future iterations of LEEDS should include. Each user interface identified included the identification of applicable features, as well as the users and possible secondary users of those features.

Additionally, as LEEDS would need a major update in features, LEEDS also required an upgrade to its architecture. The existing LEEDS architecture is outdated and does not support some of the newer browser capabilities. CIMS performed an extensive investigation of technologies and identified GWT and Portals as two contrasting but highly capable approaches. As a result, small proof-of-concept implementations were

made using each technology. It was determined that the portal technologies provide many features built into their framework that could be utilized to the benefit of LEEDS users, for example, roles, security, and content management. GWT, on the other hand, is very useful for developing a highly interactive, desktop-like application within the browser.

One of the key elements of the LEEDS system is the storage of data. Previously, the database for LEEDS was developed to meet specific needs and updated as system enhancements were performed. While reconsidering the functionality of LEEDS, the time was appropriate for evaluating available data models for the system. In particular, MIMOSA and the EIA-836 CM data exchange standards were evaluated. Although EIA-836 had broad capabilities, the model was more difficult to understand and did not cover all of the aspects we were looking for in a CM, maintenance tracking, and health monitoring system. MIMOSA, which the existing AHM data model is based on, was determined to be more appropriate for the needs of LEEDS.

Additionally, a demonstration of Case Based Reasoning for maintenance aiding was implemented as a standalone application. This was one of the features desired within LEEDS and was developed as a demonstration that could easily integrated into LEEDS. Through this demonstration, we were able to demonstrate the ability of a maintainer to be assisted in making maintenance decisions based on previously successful resolutions to similar maintenance problems.

Remote Monitoring & Advanced Support is the research of enhanced health assessment technologies and health management approaches for high-value equipment and fleets of equipment. The advancement of the remote monitoring includes the off-vehicle tools that help maintainers assess the system; an example is reliability centered maintenance approaches. During the contract period, CIMS performed multiple smaller programs that were divided into the following categories: Applied Remote Monitoring and Advanced Support Technologies; AHM 3rd Generation Hardware Improvements, AHM 3rd Generation Software Improvements; Maintainer Tools; and Advanced Anomaly Detection and Diagnostics Framework Technologies.

Applied Remote Monitoring and Advanced Support Technologies

CIMS implemented an AHM system on a UAV engine to demonstrate the capabilities of health monitoring on unmanned vehicles. The data that was collected allowed CIMS to develop methodologies for predicting glow plug failure.

In coordination with PM-LAV, an RCM analysis was performed on the LAV AT A2 turret design. As the turret was still under design, the RCM analysis provided a number of recommended preventative maintenance tasks.

A system for collecting health monitoring data was implemented on the engine dynamometer at CIMS. The system allowed for collection of LAV engine data under controlled conditions for development of algorithms. Additionally, the system provided a testbed for implementing and testing new AHM technologies. For example, CIMS

implemented a framework for predicting normal behavior of signals which allowed for identification deviation from normal. Specifically, fuel use was able to be predicted utilizing engine speed, engine torque, engine load, and intake manifold pressure. Deviations from the predicted value would indicate poor engine performance.

As large quantities of data became available from the Embedded Platform Logistics System program, it was beneficial to analyze the data, specifically alerts, to identify areas for improvement with AHM. Although many of the alerts were unable to be identified as actual faults or false alarms, it was determined that some of the alerts were caused by poor alert limits. As the data was collected and analyzed, limits could be updated and the occurrence of false alarms decreased.

CIMS equipped a Canadian Grizzly (similar to an LAV) vehicle with an advanced set of hardware as a platform demonstrator. The system included a more powerful processor, more memory, expanded storage capacity, and a broader set of sensors than utilized under the EPLS program. As new software was developed, it was pushed to the demonstrator for a final performance analysis.

AHM 3rd Generation Hardware Improvements

In an effort to expand the capabilities of AHM, CIMS implemented a prototype data acquisition controller (DAC) capable of collecting vibration data. The goal of the program was to develop a DAC that could run MATLAB generated code while collecting the vibration signals. The prototype was unable to achieve the desired 100 kHz sampling rate due to contention for resources within the hardware. Additionally, generating C code from MATLAB required significant understanding of the project.

In order to demonstrate the vibration DAC, CIMS developed a protocol for transferring waveform data across the J1939/Can bus. This protocol was implemented and tested on the AHM system on the UAV. Although the protocol was designed to be capable of continuous waveforms, the bus speed limits the actual data rates. In the demonstration, the vibration data was collected over a short period based on a trigger.

CIMS evaluated the feasibility of using a USB based interface to an existing vehicle databus. By locating the databus interface outside of the system health node, the node may be located inside the vehicle with complete disregard for where the databus interface is. Although it is feasible to collect data through a USB interface board, the USB bus is very susceptible to noise and would need significant shielding to avoid errors.

CIMS also evaluated the ability to run the existing AHM software on low-cost ARM processors, as opposed to all previous x86 processor implementations. The conversion to ARM was fairly seamless as many of the existing packages were available for ARM. Once the system was up and running, a baseline test was performed that showed AHM used only 30% of the CPU, even at high load.

OBD-II is the protocol typically used for diagnostics on light vehicle fleets. As the

military does own a significant number of light vehicles, CIMS developed an OBD-II interface capable of collecting data from light vehicles and storing it in AHM. The OBD-II protocol differs slightly from previous protocols as it requires the connected system to request data, requiring changes within the AHM application. The system was demonstrated by collecting vehicle data from multiple manufacturers and multiple model years, spanning pre- and post-CAN format OBD data.

AHM 3rd Generation Software Improvements

A limitation on the existing AHM software architecture was the utilization of MySQL, which is limited by licensing, for the onboard database. To remove this limitation, CIMS developed a flat file data storage mechanism which is more compact, faster and reliable. Development also included a receiver which would parse files and upload the data to the master database.

J1939 allows for multiple ECUs to report the same signal, however, previous versions of AHM assumed a signal came from one ECU. To fix this problem, CIMS implemented a signal selection mechanism allowing the ECUs to be prioritized. Data would then be stored from the highest priority ECU sending the signal. Additionally, the system keeps track of which ECU the stored data was received from.

During development of the Vibration DAC, it became apparent that a tool for visualizing waveform data was necessary. CIMS developed a simple tool for visualizing the waveform data and allowed the data to be stored as a CSV file.

In an effort to make set-up of AHM easier to perform by an engineer, CIMS developed a system for programming data acquisition nodes without writing code. The system allows an engineer to use a graphical user interface (GUI) to assign the databus conversions and any additional calculations to incoming channels. The resulting outputs can be assigned to an appropriate J1939 packet.

Expanding on the existing AHM capability on J1939 or J1708 equipped vehicles, CIMS implemented the collection of Diagnostic Trouble Codes (DTCs). The DTCs are collected and stored into new data tables in the AHM database.

CIMS also developed a simple tool for parsing J1939 and J1708 databus packet captures. The tool would parse the packets, create a listing of all available signals, and provide a graph of each signal. This system allows an engineer to determine what signals exist on a platform, thus enabling the proper set-up of AHM.

Maintainer Tools

The Maintainer's Graphical User Interface (MGUI) tool developed by CIMS provides a mechanism for interrogating the vehicle or master database and visualizing signal data. During this program, CIMS enhanced the capability of the MGUI tool by implementing dynamic downsampling of large datasets, allowing annotations on graphs, allowing

selection of multiple missions, providing mission filtering capability, allowing the user to change the missions being viewed from the graph screen, and allowing the viewing and updating of real-time data.

Advanced Anomaly Detection and Diagnostic Framework Technologies

CIMS performed multiple projects in efforts to better utilize the available data to reduce downtime. In one project, CIMS utilized a regressions-based model to identify an oil pressure anomaly prior to it being detected as a DTC. Additionally, CIMS evaluated regressor models for identifying relationships between signals, such as predicting oil pressure based on oil temperature and engine speed.

In efforts to ensure valid data is being utilized in AHM algorithms, CIMS implemented multi-sensor fault detection (MSFD). Dempster-Shafer was utilized to identify a noisy battery shunt current measurement and cross-correlation was utilized to identify temperature faults in planetary wheel hubs. CIMS also utilized multiple regression and classifier models to improve prognostics around reactant ignition in a solid oxide fuel cell.

In efforts to speed up the processing of AHM data in a typical data mining operation, CIMS evaluated the use of a traditional data warehousing system which would require a one-time translation of asynchronous AHM data into a synchronous data set. Once the translation was complete, typical data mining operations saw a drastic performance improvement.

Prior diagnostic and prognostic work in CIMS focused heavily on mechanical systems. CIMS chose to test the feasibility of prognostics on a switched-mode power supply on a HALT test fixture. The study proved successful, showing the real-time correlation predicted the failure 30 minutes early under accelerated conditions.

3. Introduction to the Center for Integrated Manufacturing Studies at Rochester Institute of Technology

Since 1992, the Center (CIMS) at Rochester Institute of Technology (RIT) has joined analytical research practices with applied technology to help manufacturers remain competitive in the global marketplace.



CIMS building on the RIT campus

CIMS is a dynamic collaboration of dedicated engineers and technicians supported by RIT faculty and students, as well as academic, industry and government resources. Over the past two decades, CIMS has worked with 160 sponsors on 300+ separate research, development, and application initiatives.

The center is led by Dr. Nabil Nasr, an international expert in sustainable design and environmentally conscious manufacturing who has spent over two decades assisting companies and government agencies in enhancing profitability while also reducing their environmental impact.



One of several CIMS research bays

In response to the significant impact the center's research has achieved, Dr. Nasr and CIMS has received several national honors, including the Excellence in TBED (Technology-Based Economic Development) Award (2011), U.S. Department of Commerce's Economic Development

Administration's Excellence in University Led Strategies Award (2009), National Center for Advanced Technologies' 2004 Defense Manufacturing Excellence Award and the National Pollution Prevention Roundtable's 2006 MVP² award.

Golisano Institute for Sustainability (GIS)

Dr. Nasr also serves as Director of the Golisano Institute for Sustainability, a multidisciplinary academic unit of RIT that grew out of CIMS with a mission to undertake world-class education and research missions in sustainability. GIS academic and research programs focus on sustainable production, sustainable energy, sustainable mobility, and ecologically friendly information technology systems. These programs are led by a multidisciplinary team of faculty and researchers who collaborate with organizations locally, nationally, and internationally to create implement able solutions to complex sustainability programs.

Academic activities at GIS are complemented by five state-of-the-art research units:

- Center for Remanufacturing and Resource Recovery (C3R): an internationally recognized center for applied research in remanufacturing.
- Center for Sustainable Production (CSP): dedicated to enhancing the environmental and economic performance of products and processes.
- Center for Sustainable Mobility (CSM): focused on developing viable technologies for sustainable transportation systems and the support of complex equipment systems.
- New York State Pollution Prevention Institute (NYSP2I): a research and technology transfer center funded by the New York State Department of Environmental Conservation to provide a state-wide, comprehensive and integrated program of environmental research, technology development and diffusion, outreach, training and education.
- NanoPower Research Labs (NPRL): dedicated to the development of new materials and devices for power generation and storage for microelectronic components and micro-electromechanical systems (MEMS).

In late 2012, GIS began transitioning into its new headquarters building on the RIT campus. This new 75,000 sq. ft. LEED certified research building (targeting LEED Platinum certification) is a state-of-the-art laboratory for scientific discovery and experiential learning consisting of: 5 systems integration test beds; 7 sustainability technology support labs; and computing and collaborative spaces to support research.



New GIS headquarters building

Rochester Institute of Technology (RIT)

Founded in 1829, RIT is an internationally recognized leader in professional and career-oriented education enrolling more than 18,000 students. RIT is coeducational and has one of the oldest and largest co-op programs in the world. .

RIT, the 11th largest private university in the nation, offers 350 programs of study in eight colleges including the Kate Gleason College of Engineering, the E. Philip Saunders College of Business, and the B. Thomas Golisano College of Computing and Information Sciences. The university is internationally respected for its research and educational programs in imaging and

color science, photography and remanufacturing as well as its work in experiential learning and cooperative education.

RIT's modern 1,300-acre campus is located in Rochester, the third largest city in New York State.



RIT campus

Sustainment Testbed



4. Sustainment Testbed

4.1. Description of the Sustainment Testbed

The Sustainment Testbed research area is designed to develop new technologies that address the unique challenges of maintaining current Department of Defense equipment within or beyond its normal service life in a cost-effective way. Material reliability and material availability are key metrics in the effort to extend the life of current inventory equipment. New technologies have been evaluated and developed to provide unique sustainment tool sets for application to selected validation and demonstration platforms.

The Sustainment Testbed consists of several related research initiatives applied to the common purpose of platform readiness improvement, life-cycle cost reduction, and useful life extension. The three research initiatives undertaken that support the goals of the Sustainment Testbed are: Material Aging & Prognostics; Material Restoration; and Rapid Reverse Engineering.

Material Aging and Prognostics

Material aging research includes the analysis, evaluation and modeling technologies for identifying and quantifying the amount of aging experienced by a platform or component and for predicting the remaining useful life. Critical to the performance of this research is the incorporation of physics of failure for identifying the type of material aging of a component. Through this research, the testbed was able to develop inspection methods and simulation models for surface wear and fatigue of metals.

Material Restoration

Material restoration research includes the development of processes for repair or restoration of a component to its original function or dimensions at the material level. Material restoration is an extension of the work performed in Material Aging and is thus the next logical step in the sustainment of platforms. This area focused on the utilization of coatings and coating processes for surface repair and life extension of worn or cracked metal components.



Metal flame spray

Rapid Reverse Engineering

Rapid reverse engineering is the process of utilizing laser scanning technologies to enable design capture for mechanical components. Research work in this area focused on the development of technologies for converting a solid model from a laser scan into a computer model capable of being utilized for replication of parts and development of technical data packages.

4.2 Sustainment Testbed Research Objectives, Progress, and Results

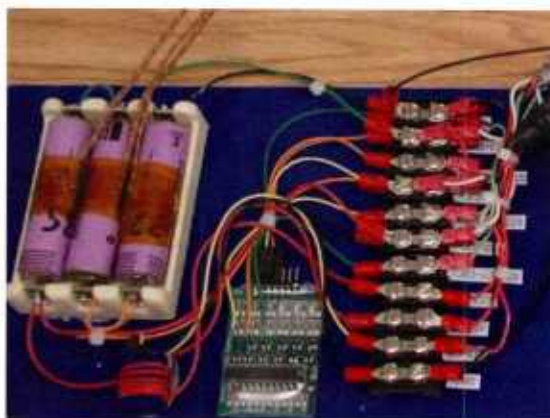
A primary objective of the Sustainment Testbed research is to develop advanced platform sustainment technologies that may be applied for U.S. Marine Corps vehicles. Several different

research objectives supported this end goal: crack detection in spur gears; development and evaluation of a remanufacturing process for contact surfaces of military parts that have needle-bearing contacts; analysis of battery pack cell replacement strategies; laser scanning development; and evaluation of a fuel additive for increased fuel efficiency in the Light Armored Vehicle (LAV).

During the contract period, CIMS continued to evaluate and model material surface degradation and fatigue failures. Crack detection in gears is critical to the prevention of catastrophic failures. In order to understand the physics of failure of gear breakage, the program set out to analyze both stages of the gear service life: crack initiation and crack propagation. The initial study of gear crack initiation allowed for the identification of cracks at the earliest phase, subsequently providing samples for further study during the crack propagation stage. Analysis of the gear under controlled conditions allowed for study of the propagation of the crack, as well as assessment of the effects of the material microstructure on the crack. The study resulted in identification of condition indicators that may be utilized for crack detection and damage estimation.

Previously, CIMS developed a process for removing a damaged surface coating from an LAV driveshaft and utilized a process for a new flame spray coating to bring the driveshaft back to Original Equipment Manufacturers (OEM) specifications. Under this contract, CIMS continued the evaluation of coating technologies for surface wear specifically for surfaces in contact with needle-bearing contacts. Currently, if a contact surface is inspected and identified as degraded, the part must be scrapped. Rather than scrapping a part that is structurally sound except for minor surface degradation, a solution was sought to repair the materials surface and extend the life of the component. A process was developed utilizing a twin-wire arc thermal spray solution and tested for surface hardness, thus proving that the surface coating may be a feasible alternative to scrapping the entire part.

As the military makes a concerted push to reduce the use of fossil fuels and utilize renewable energy, new modes of failure begin to affect the capability of our military forces. For example, as the utilization of solar energy increases so does the use of batteries as a storage mechanism for that energy. In an effort to understand the far-reaching implications of battery replacement on the sustainment of military forces, CIMS developed a program for: evaluation of replacement strategies; evaluation of the potential to repurpose batteries for less demanding uses; and methods for increased accuracy of battery cell-age and state-of-charge estimation. The program determined that individual cells in a battery pack with an early-life failure may be replaced without any significant impact to the pack. However, replacement of cells causing a significant mismatch between cells reduced the efficiency of the pack. A strategy of active cell management may be employed to minimize the impact of mismatched cells.



Battery pack testing

As complex military systems age, it is increasing difficult to obtain components for these legacy systems because the OEM has stopped producing the component and the military does not have a

complete technical data package with which to obtain a quote from other vendors. Reverse engineering is a means of developing the technical data package, including but not limited to mechanical drawings, electrical specifications, circuit board designs, and component specifications. During the contract period, CIMS focused efforts on reverse engineering mechanical components through use of a 3D laser scanning system and a Coordinate Measuring Machine (CMM). CIMS was able to create solid models that included the overall part geometry from the laser scanner and features from the more accurate CMM.

As previously stated, the military is pushing for a major reduction in the use of fossil fuels to develop a less energy dependent military and to reduce the potential for loss of life in transport of materials to critical areas. The U.S. Marine Corps was provided a fuel catalyst for the Light Armored Vehicles that claimed to increase fuel efficiency and reduce emissions. CIMS performed a series of tests on an engine dynamometer to compare the efficiency of the catalyst enhanced fuel against regular fuel. After running multiple load and speed profiles, CIMS determined that the catalyst may provide a minimal fuel efficiency increase while significantly increasing nitrogen dioxide, a prominent air pollutant.

4.3. Sustainment Testbed Projects

The Sustainment Testbed program funded by this contract is enhancing CIMS' technical capacity and supporting numerous DoD programs and platforms. During the contract period, the program focused on understanding material aging and applying this knowledge to structural health monitoring, component restoration, and material-based prognostics.

Projects:

- Replacement Strategies for Maintaining and Repurposing of Lithium-ion Batteries
 - Cell Aging
 - Pack Studies
 - Estimation of Cell Capacity and State-of-Charge
- Remanufacturing Process Development
 - Twin-wire Arc Thermal Spray Coating of Metal Power on Steel
- Crack Detection in Spur Gears
 - Crack Initiation Studies
 - Crack Propagation Studies
 - Feature Analysis and Feature Fusion
- Reverse Engineering
 - 3D Laser Scanning Development and CMM Integration
- LAV Fuel Efficiency Testing

The projects are fully described in the next section.

Sustainment Testbed

PROJECTS

Sustainment Testbed Projects

Replacement Strategies for Maintaining and Repurposing of Lithium-ion Batteries

In many demanding applications (e.g., communications), rechargeable batteries are replaced at an early point in their lives while they still have considerable capacity left. These batteries can be used for less-demanding applications, such as storage in expeditionary power systems. A related issue is the maintenance of larger packs: there is considerable apprehension related to mixing new and old batteries. Methods need to be developed to understand the relative health of individual cells in those packs so that sustainable decisions can be made around repurposing the cells for less-demanding applications and maintenance of packs. Many issues are currently under debate, such as the need for cell matching when replacing cells within a pack, overall cell capacity, and matching numbers of cycles. The replacement strategies will address three distinct problems: 1) Cell aging and relationship between individual cell aging and aging of cells in packs; 2) Case studies of replacement strategies related to maintenance and repurposing; and 3) Estimation of state of health and state of charge of used cells. These three problems are described in more detail below.

Cell Aging

Problem

Original Equipment Manufacturers (OEMs) typically provide aging data only for early life aging which limits the ability to design longer life systems. There is a general perception that cells age faster when they are operated in a pack, possibly due to mismatched cell impedances, than when they are aged individually.

Goals

There were two goals: obtain cell aging data for different Li-ion chemistries beyond the point published by OEMs and their dependence on the operating conditions; and relate the pack cell aging to individual cell aging.

Approach

We conducted an empirical study of aging standard 18650 lithium-ion cells of different chemistries, including representatives from cobalt oxide and iron phosphates. The cells are first aged individually. Figure 1a shows the capacity fade of cobalt oxide cells. Representative impedance spectra of the aging cells from this family are shown in Figure 1b.

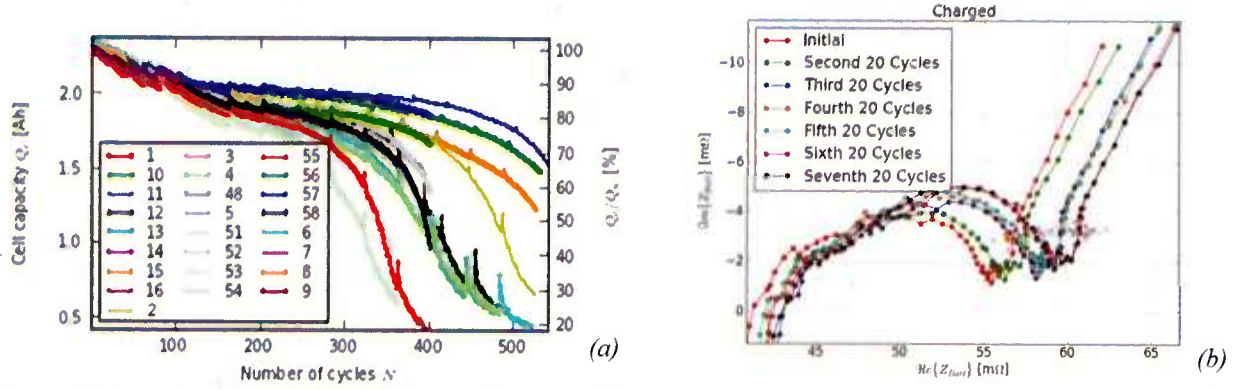


Figure 1. Individual aging of cells: (a) Capacity vs. number of cycles; (b) Impedance (limited to 140 cycles for clarity).

The cycling of cells was periodically interrupted to record impedance spectra. Traditionally, the industry uses the real part of the impedance at $f = 1$ kHz as the aging metric. Our measurements suggest that the real part of the resistance at low frequency, in the Warburg region, show more sensitivity to battery aging (Figure 2a)¹. The scatter plot of the resistance vs. capacity, shown in Figure 2b, confirms that the relationship persists as the cell age.

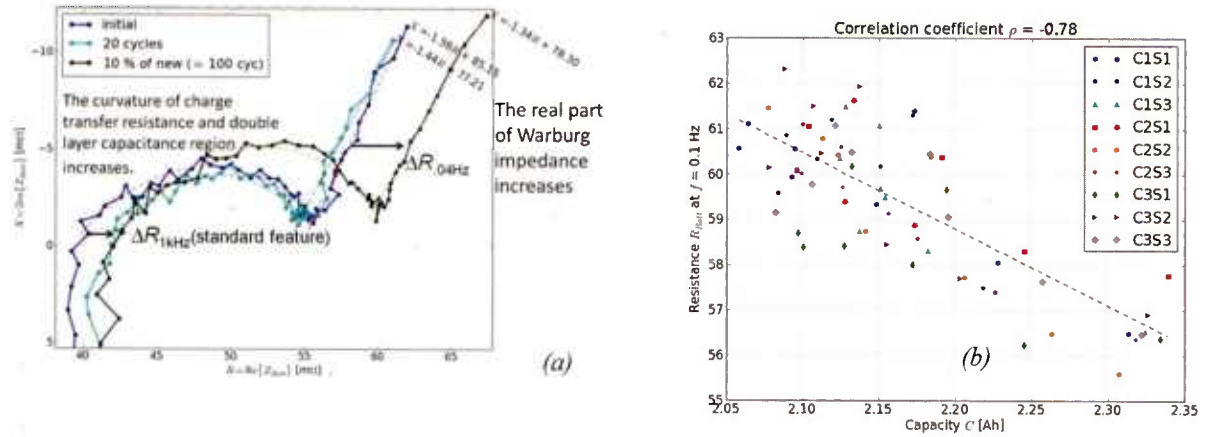


Figure 2. Individual aging of cells: (a) Capacity as the function of number of cycles; (b) Impedance (cut off at 140 cycles for clarity).

The cell capacity shows significant sensitivity to temperature. Figure 3 shows this dependence in two steps: the capacity strongly depends on the cell temperature, while the cell temperature is related directly to the ambient temperature. A few degrees Celsius will change capacity approximately by a percent, which is significant, considering that some of the cells are only used within 20% of their capacity.

¹ Chen-Mora model, described later (Figure 8), provides means for explaining better sensitivity at low frequency. At low frequency, the total resistance is the sum of all resistive parameter, i.e., $\text{Re}\{Z_{\text{Batt}}\} \approx R_{\text{Batt}} + R_{\text{TL}} + R_{\text{TS}}$, while at $f \sim 1$ kHz we have $\text{Re}\{Z_{\text{Batt}}\} \approx R_{\text{Batt}}$.

Temperature Effect

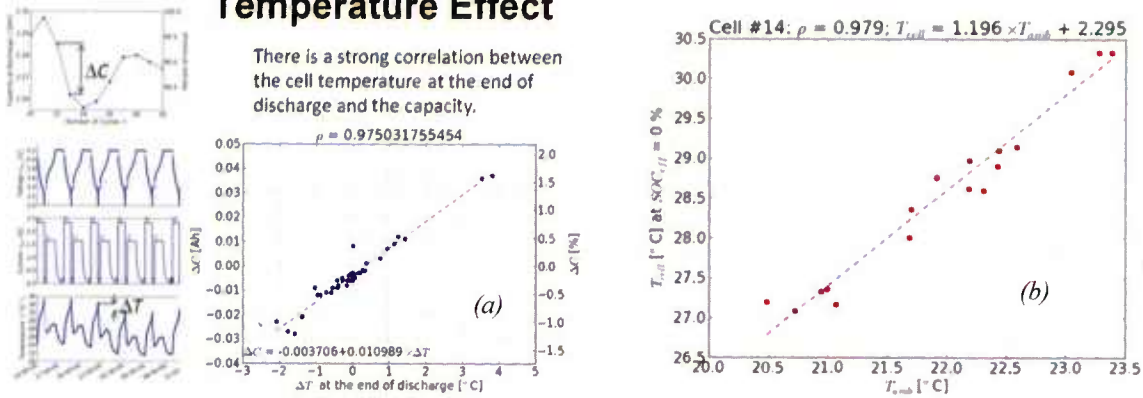


Figure 3. Temperature dependence of the capacity. (a) The temperature difference at the end of discharge for subsequent cycles vs. accompanied change in capacity. The Pearson correlation coefficient is $\sim 97\%$. (b) The dependence of the temperature at the end of discharge is strongly dependent on the ambient temperature.

To compare aging of individual cells to aging in a pack, a small 3×3 pack is built in 3s3p configuration, where three cells are connected in series to form a string and three such strings are connected in parallel. Figure 4a, Figure 4b, and Figure 4c show the photograph of the test stand, the pack configuration and the schematic, respectively.

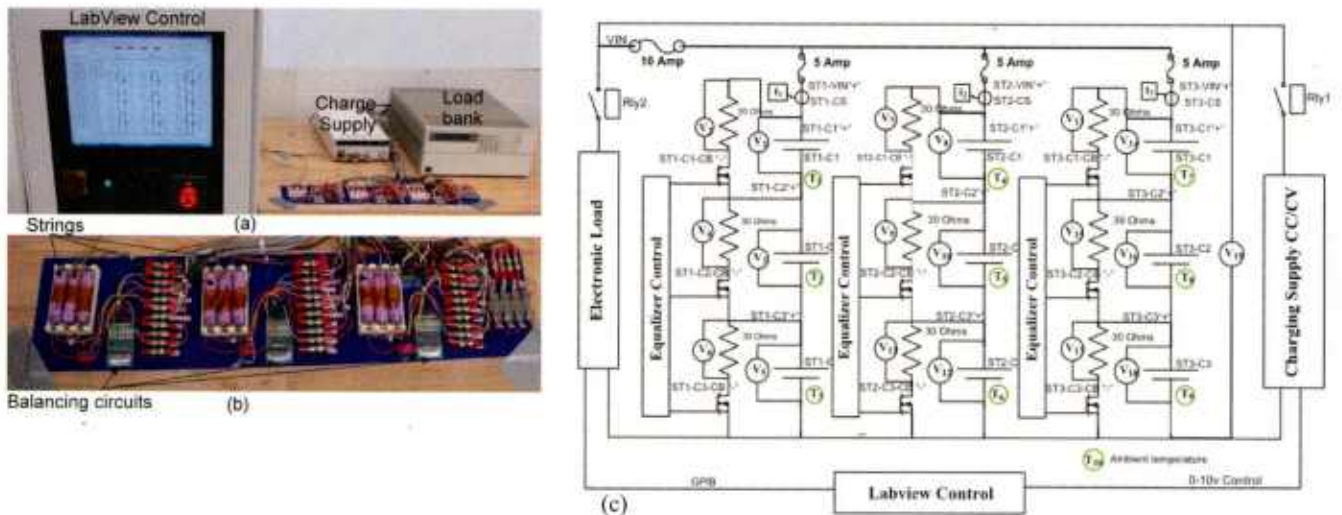


Figure 4. 3×3 battery pack in 3s3p configuration: (a) Test stand; (b) Zoom into the pack; (c) Schematic.

The experiments did not show any significant difference in aging cells individually and aging them in this pack. Figure 5a shows that the capacity fade of single cell aging from $\sim 100\%$ to $\sim 90\%$, with the average slope of capacity $\sim -0.11\%/cycle$. Figure 5b shows the same aging from the cell in the 3s3p pack. The average rate of change was approximately the same: $\sim -0.1\%/cycle$.

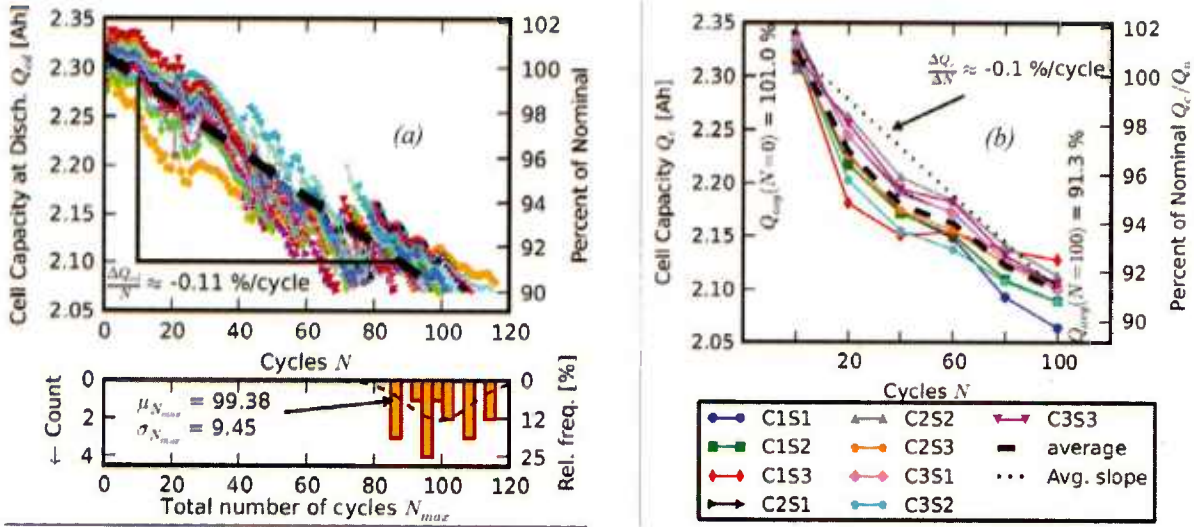


Figure 5. Capacity fade for cells for: (a) Individual cell aging; (b) Aging cell in a pack.

Results

Several types of cells are individually aged and characterized, resulting in a useful database for characterization of the used cells. Strong dependency of the cell capacity was found. The experiments did not find a significant difference between individual cell aging and aging of the cells in a 3s3p pack.

Pack Studies

Problem

Maintenance of large packs will require cell replacement in case of early life failures. Many issues are currently under debate, such as the need for cell matching when replacing cells within a pack, or overall cell capacity, or even matching numbers of cycles. Building packs from used cells is challenging due to mismatches of the cells.

Goal

The objective of the study is to investigate maintenance strategies in lithium-ion battery packs and rebuilding packs for less-demanding applications.

Approach

The 3s3p pack described above (and in Figure 4) was used to explore the cell-replacement strategies empirically. We examined two scenarios. The first scenario represents the case when one cell in a pack fails early with respect to the expected life of the pack. To simulate this situation we pre-aged cells in the individual pack tester to 90% of their initial capacity. In demanding applications, such as electric vehicles, the cells are considered usable only when the capacity is higher than 80% of the nominal capacity. Therefore, for these applications, cells aged down to 90% of their nominal capacity are at ~50% of their useful life.

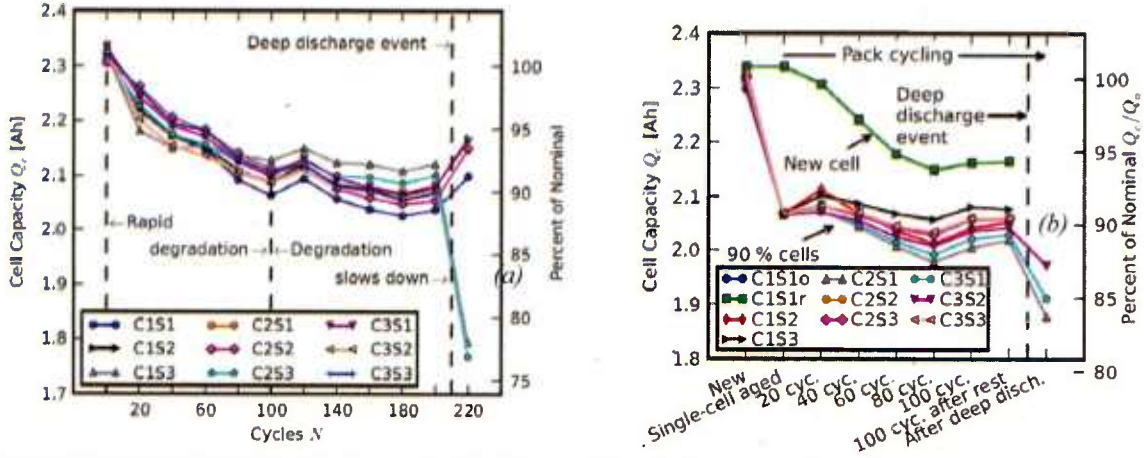


Figure 6. Capacity fade of cells in a pack: (a) All new cells; (b) Pre-aged cells with one new cell.

Comparing capacity fades of the cells of the pack comprised of eight aged and one new cell to the capacity fades of the reference pack, plotted in Figure 6b, shows that a new cell nicely coexists with the aged cells and did not introduce any obvious increase in the overall rate of capacity fade or decrease in efficiency. The slope of the capacity fade of the new cell is higher than that of the aged cells, but this behavior is consistent with cell aging in general. Capacities of the new cells in Figure 6a fade faster over the course of the first 100 cycles and the relative rate of capacity degradation slows down.

To create the second scenario for cell replacement strategy, two 3S3P packs were first subjected to deep discharges, as shown in Figure 6a and Figure 6b. The deep discharge events can be considered major pack failures and the cells recovered from these deeply discharged packs are good candidates for simulating cell repurposing processes. After the deep discharges, the individual cells were recovered by charging them at low current (100 mA). This process recovered 12 out of the original 18 cells. The remaining 6 cells had their current-interrupt devices triggered, which rendered them unusable for the Scenario 2 experiments.

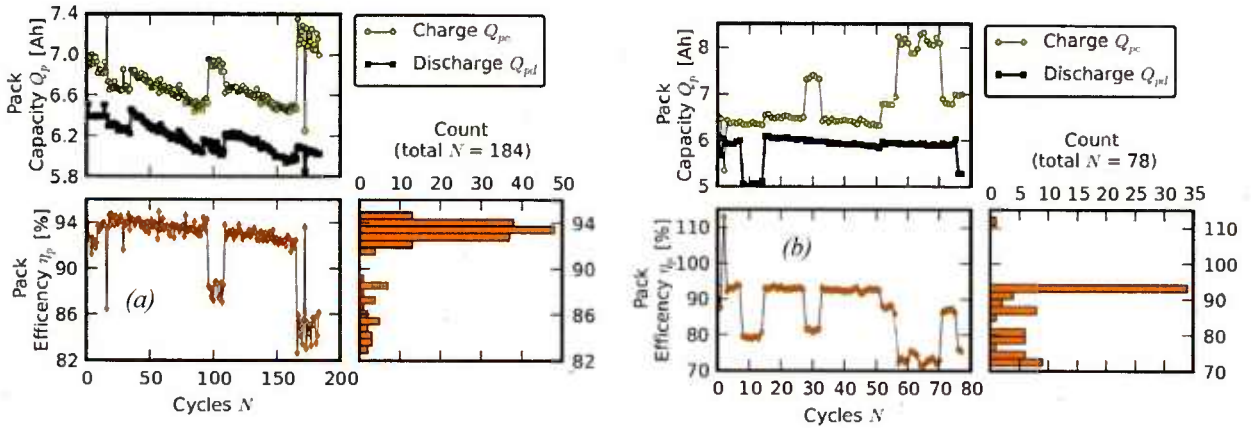


Figure 7. Capacity fade of cells in a pack: (a) All new cells; (b) Pre-aged cells with one new cell.

The cells are matched so as to balance the capacity of the strings. The new pack was successfully operated for ~80 cycles. Its efficiency was reduced compared to the new pack because of the energy loss associated with balancing of mismatched cells. Our experiments with active cell balancing showed that some of the efficiency loss can be recovered if an active cell-balancing technique is employed.

Results

A successful cell-replacement strategy that simulates pack maintenance activity associated with an early-life cell failure was demonstrated with no significant pack aging difference. A pack of cells with significant mismatches among cells was built and operated for some time. The efficiency was reduced compared to matched cells. The reduced efficiency is due to additional balancing and can be reduced by employing active cell balancing.

Estimation of Cell Capacity and State-of-Charge

Problem

Lithium-ion batteries in high-end applications, such as Electric Vehicles (EVs) and Plug-in Hybrid Electric Vehicles (PHEVs), are growing in numbers. After serving their useful life in high-end application, the cells have considerable capacity left (70-80% of the original capacity). Moreover, many Li-ion batteries in consumer electronic applications also have considerable remaining life. To reuse Li-ion cells in less-demanding applications, it is crucial to be able to assess their state-of-charge and age. While the most reliable estimation is cycling a cell from its fully charged state to its fully discharged state, this process is prohibitively expensive and time-consuming.

Goal

Investigate methods for quick, cost-effective estimation of Li-ion cell age and state-of-charge.

Approach

Because direct capacity measurements are too expensive, the age of the cells is estimated indirectly. The two methods considered here are based on extensions to the well-known correlation between age and resistance: one is based on time-domain analysis; the other is based on the frequency analysis.

A popular battery model adopted in time-domain analysis is illustrated in Figure 8. The key parameters are open-circuit voltage V_{oc} , DC resistance R_{Batt} , and two parallel RC circuits $R_{TS}-C_{TS}$ and $R_{TL}-C_{TL}$ for modeling transients at different rates.

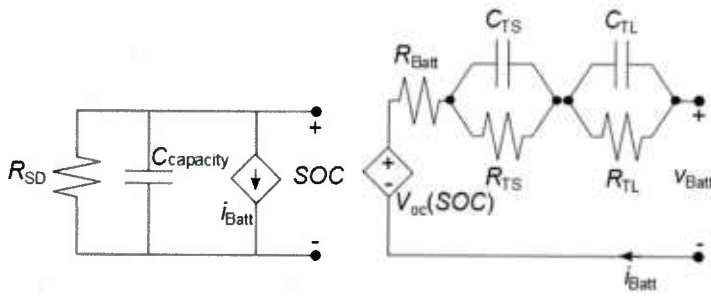


Figure 8. Chen-Mora model².

Time-domain analysis is based on a short-duration step charge (or step discharge). Step charge and discharge allow model parameter estimation during the discharge process. A single-step test takes a few minutes (compared to several hours for full charge/discharge). The model parameters are extracted from the transient waveform of the terminal battery voltage v_{Batt} and current i_{Batt} .

Figure 9a illustrates example parametric fits. The process is repeated for different levels of state-of-charge and at different ages (number of cycles) for a number of cells, which enables arriving at empirical curves of the model parameters as a function of state-of-charge and age as illustrated in Figure 9b.

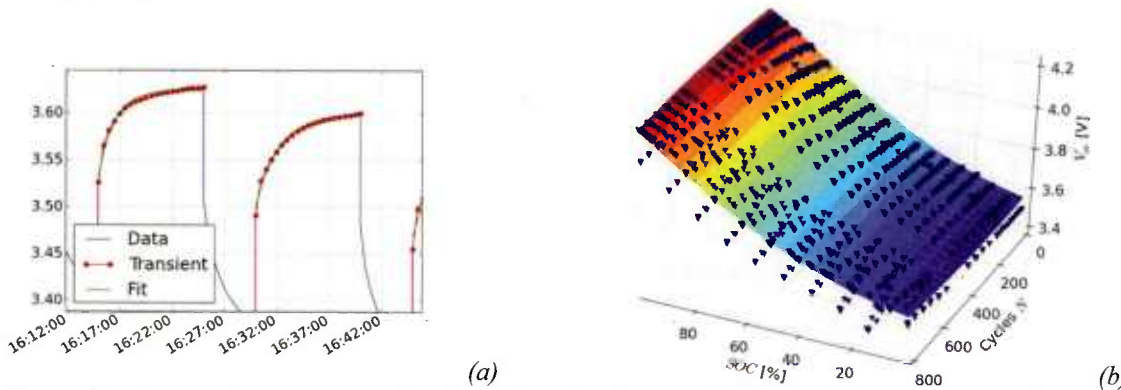


Figure 9. (a) Example parameter estimation of five-parameter model. (b) An example parameter V_{oc} estimated for different state-of-charge and different age (number of cycles).

Once these empirical relationships are available, they can be used for Bayesian inference. Figure 10a shows a posterior bivariate distribution of state-of-charge (SOC) and age (number of cycles N) after a cell's parameters were estimated from a single-step charge and compared to empirical parameter curves. Alternatively to Bayesian inference, cell parameters can be used directly to train a machine learner, such as Boosted Regression Trees (BRTs), to predict the age of a cell (see Figure 10b).

² M. Chen and G. Rincon-Mora, "Accurate electrical battery model capable of predicting runtime and I-V performance," IEEE Transactions on Energy Conversion, pp. 504-511, 2006.

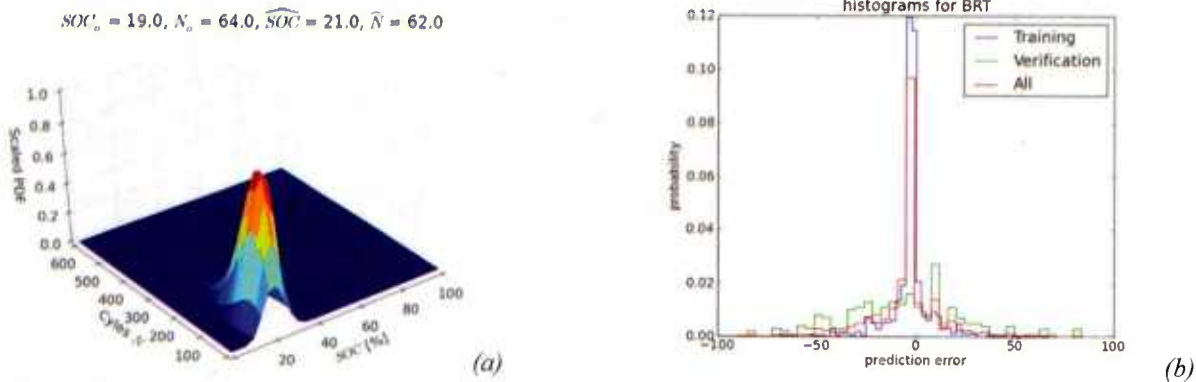


Figure 10. (a) Bayesian inference of cell SOC and age. (b) Histogram of prediction error on training, verification, and all data.

Frequency domain analysis is based on impedance spectroscopy. Figure 11 shows the average impedance spectra of new cells and cells aged to 90% of their original capacity. The standard aging feature employed in the industry is the real part of the impedance evaluated at $f = 1$ kHz, which closely approximates RBatt of Figure 8. The impedance data shows that more resolution is available at lower frequencies (1 Hz or less). The higher resolution at lower frequencies is because the change is $\Delta RBatt + \Delta RTS + \Delta RTL$, where at $f = 1$ kHz, the change is $\sim \Delta RBatt$.

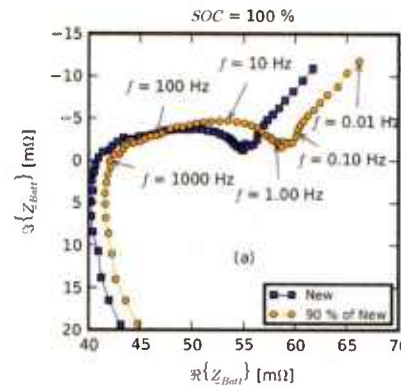


Figure 11. Impedance spectroscopy of new and aged cells.

Results

The main outcomes of the study are the approaches for cell age estimation with higher accuracy and precision than simple correlation with the battery resistance.

Twin Wire Arc Thermal Spray Coating of Metal Power on Steel

Problem

Mechanical systems degrade over time due to wear, corrosion, pitting, and fatigue. Contact surfaces of military parts with needle-bearing contacts, such as steering boxes or pump shafts, degrade after extended use in service because of rolling contact fatigue. Once the surface on which the needle bearings ride begins to degrade, the function of the associated component is at risk of failure. The effects associated with such a failure can be quick and catastrophic. To remedy this situation, parts should be inspected and tested during regular maintenance intervals. Fixing the damaged component typically requires replacement of the failed part.

Goals

Replacement of a component with rolling contact fatigue can be costly due to the material cost, labor to make parts with integral surface features, OEM sourcing, and qualification of new vendors. A better approach would be to remanufacture the component. The goal of this thermal spray project was to identify a material and set of process parameters that could be used to remanufacture a sector shaft from a typical large military or commercial vehicle that employees the R. H. Shepard® steering box or similar component (e.g., on the medium tactical vehicle replacement [MTVR]³).

Approach

A twin-wire arc remanufacturing method was selected for the remanufacturing of the sector shaft. Upon selecting the remanufacturing method, an extensive literature search was performed to determine if methods currently exist for similar thermal spray applications. Preexisting thermal spray solutions to this problem were not identified; for this reason, a statistical experiment was performed to identify the main parameters to be applied to this problem: electrical current level, gun stand-off distance, and air pressure. The study consisted of a twin-wire arc thermal spray gun applying the coating on plates that were then ground flat. Testing of many thermal spray samples within a steering box is prohibitive because it is costly, time-consuming, difficult to control repeatability, and hard to quantify differences in results with real-time feedback. Therefore, an Instron® test machine was used to simulate the load applied to a sector shaft from a needle bearing. A fixture was custom designed and fabricated to capture a needle from a steering box bearing and enabled it to be pressed against the thermal spray samples at force of 2000 lbf.

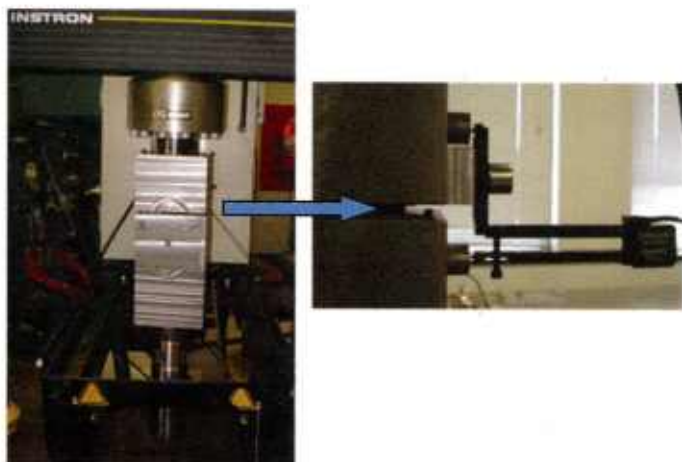


Figure 12. Thermal Spray Fixture and Extensometer

The fixture shown in Figure 12 incorporates stiff components to minimize displacement during testing to allow for more accurate measurement of the thermal spray samples. Even with the precautions taken, the

³ Oshkosh MTVR brochure; Available at: www.wenzlau.com/documents/Oshkosh_MTVR_brochure.pdf

compliance of the fixture was significant compared to the thermal spray sample and needle roller interaction. An extensometer was then installed across the halves of the fixture to measure only the points of interest, thus minimizing the background noise. The load was ramped to full force over two minutes with the final change in the extensometer equaling the value of interest.

Results

Previous development work showed that increasing the hardness of the thermal spray coating did not prevent the needle from denting the coating. Instead, the yield strength of the coating needed to be increased. Two methods were evaluated for improving the yield strength of the thermal spray coating. The first method consisted of using a Design of Experiments (DOE) to improve the density of the “as sprayed” coating. The variables in the DOE consisted of air pressure, gun stand-off distance from component, arc voltage and amperage. This DOE showed that reducing the stand-off distance, together with increasing the amperage and air pressure, at a constant voltage improved the density of the “as sprayed” coating. However, while increasing the density of the coating reduced the depth of the dents it did not eliminate the denting (Figure 13).



Figure 13. Stainless Steel Thermal Spray Shaft Denting

The second method evaluated incorporated a change to the composition of the thermal spray coating. A relatively new material from Praxair Surface Technologies Inc. was selected for the test samples. This new composition produced thermal spray coatings that did not show any denting from the needle as detected by eye; see Figure 14. The parameters identified as the most promising are lower standoff distance between the plate and spray gun, higher electrical current, and intermediate air pressure. This development project showed using thermal spray to remanufacture specific steering box sector shafts can be a feasible and affordable approach.

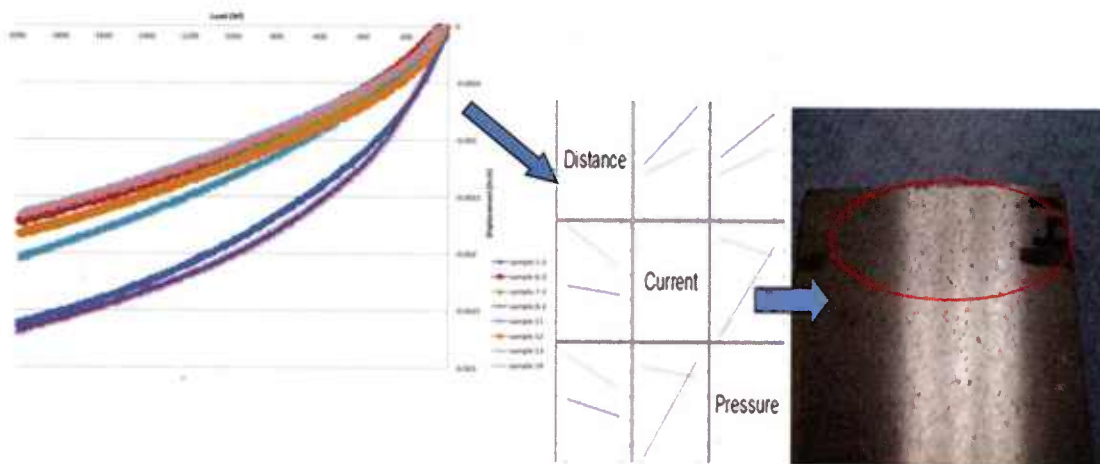


Figure 14. Thermal Spray Test Results, Statistical Matrix, and Sample Image.

Crack Detection in Spur Gears

Introduction

Of four dominant gear failure modes (breakage, wear, pitting, and scoring), breakage is the most precipitous, which can lead to catastrophic failures. While gear fracture has been studied for many years, methods to predetermine gear condition remain elusive. This research contributes additional empirical studies and understanding towards the goal of gear prognostics. To detect a gear tooth breakage failure in practice, the gearbox is equipped with one or more accelerometers that sense vibration signals. These vibration signals are processed and condition indicators (CIs) – acceleration trends and features – are extracted. Several different gear research studies are described below: enhancing the process of accelerated crack initiation; understanding the properties of crack propagation under controlled conditions; and statistical analysis of CIs.

Crack Initiation Studies

This work builds on work that was performed under Award No. W911NF-09-2-0022. The prior work included development of the test fixtures, development of initial procedures for crack initiation and propagation, and a first pass at algorithms for analyzing the cracks. The gears being studied are NASA-designed spur gears, however, the methods and algorithms may be applied to other spur gears to identify their crack indicators. Figure 15 provides the dimensions of the NASA gear.



Figure 15: NASA-designed spur gears

Problem

Gear service life can be divided into two phases: crack initiation and crack propagation. From the viewpoint of fracture mechanics, nearly the entire service life of a gear is covered under crack initiation. The long crack initiation process must be accelerated to allow conducting empirical studies in a reasonable time. For testing, an initial crack in a gear tooth is made in order to specify the place where failure will most likely occur and to reduce the amount of time required for the crack to propagate. The crack must be large enough to verify and small enough to allow the propagation process to be observed in the early stages when there is time for corrective actions. Traditionally, cracks are seeded by imparting a notch via wire electrical discharge machining (EDM). Recently, fatigue-based approaches to accelerated crack initiation have been considered^{4,5}. Material and manufacturing variations result in cracks that start at different locations and depths, complicating the identification process. To control the crack size, methods for initiating and evaluating initiated cracks have to be further perfected.

⁴ N. G. Nenadic, J. A. Wodenscheck, M. G. Thurston, and D. G. Lewicki, "Seeding Cracks Using a Fatigue Tester for Accelerated Gear Tooth Breaking Rotating Machinery, Structural Health Monitoring, Shock and Vibration, Volume 5," vol. 8, T. Proulx, Ed., ed: Springer New York, 2011, pp. 349-357.

⁵ D. B. Stringer, K. E. LaBerge, C. J. Burdick, and B. A. Fields, "Natural Fatigue Crack Initiation and Detection in High Quality Spur Gears," presented at the 68th American Helicopter Society Annual Forum, FT Worth, TX, 2012.

Goal

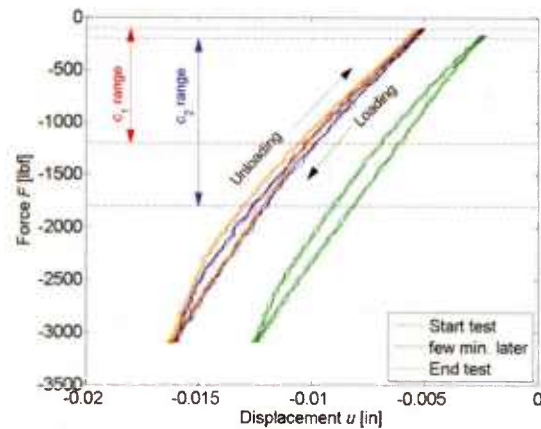
The objective of this study is to improve the resolution of a fatigue-based crack initiation.

Approach

Accelerated crack initiation in a fatigue tester is based on applying cyclical load, above the normal operating load. Load and displacement measurements are used to extract an initiation-related CI, also known as a feature. For example, the ratio between displacement and force is known as compliance, $c = \Delta u / \Delta F$. Our original compliance-based feature, c_1 , used all load-displacement data in -100 to -1100 lbf range. When the load was > 1100 lb, compliance estimates had more variation as the unloading and loading sides of the fatigue cycle diverged. A buffer of data was collected and processed by a linear regression function to get a compliance estimate.

Compliance Feature	c_1	c_2
Min Load Compression, lbf	100	200
Max Load Compression, lbf	1100	1800
Loading/Unloading	both	Unloading
Linear regression to calc compliance	Single buffer	Sliding window

Figure 16. Compliance feature enhancements. The table summarizes the main differences. The plot shows three characteristic cycles of load-displacement curves.



The resulting enhanced feature, c_2 , used only data points from the unloading side of the cycle over the extended range for force (refer to Figure 16). In addition to computing the feature, one must specify a threshold to stop the propagation of the initiated crack. This threshold is computed dynamically, and controlled via a user-adjustable parameter, θ , which roughly specifies the number of standard deviations of local feature distribution tolerated.

Results

Figure 17 shows example crack initiation data and performance comparison between the original and the new, enhanced feature. The new feature exhibits smaller standard deviation before the crack starts to grow: σ is reduced by ~28 % (from $\sim 4.2 \times 10^{-2}$ to $\sim 3.0 \times 10^{-2}$).

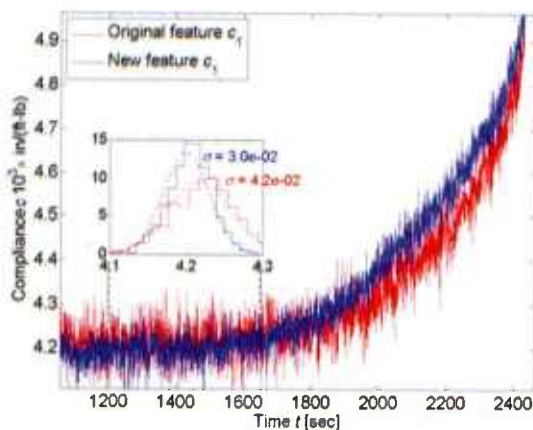


Figure 17. Performance comparison between the original feature c_1 (the red trace) and the new feature c_2 (the blue trace).

Table 1 shows a subset of the series of tests that compared the threshold parameter to the resulting initiated crack. The threshold parameter, θ , is expressed in terms of a number of standard deviations (σ) of the baseline distribution. The blue traces in the plots indicate the features, while the red traces indicate the thresholds. The associated images show the tooth side of the crack surfaces, e.g., the surfaces on the gears from which the teeth broke off. Note that the crack measurements, indicated in the plots, are the normal projections, not the actual continuous crack lengths (typically slightly longer). As this subset shows, while the compliance feature has some general indication of the crack size, the initiated cracks still vary considerably in size and shape. Some cracks are even discontinuous and initiate at different planes, which is unaccounted for in the feature.

Table 1. Excerpt from detailed comparison of performance of crack-initiation method.

Threshold $\theta = 4.5 \times \sigma$	Compliance (lb.in/lb.in) vs Time (sec)		
	Speed = 15, Torque = 10, Run = 1, $\sigma = 4.5$	Speed = 15, Torque = 10, Run = 1, $\sigma = 4.5$	Speed = 15, Torque = 10, Run = 1, $\sigma = 4.5$

Crack Propagation Studies

Problem

After a crack has been detected, the extent of gear damage has to be assessed both at its current state and, based on certain assumptions on the operating conditions, its expected growth in time. Crack propagation in a gearbox is difficult to observe directly. The rate of crack propagation in a gear depends upon operating conditions, but also upon material properties at the micro-scale.

Goal

Our intent was to investigate crack propagation on a number of teeth under well-controlled, fixed operating conditions to assess the effects of microstructure.

Approach

The fatigue tester employed for accelerated crack propagation was chosen for this propagation study because of its ability to control the load on the tooth accurately throughout the test. The anvil applies sinusoidal force in the range of 100-1400 lb at the highest point of single tooth contact (HPSTC). This is the same setup used for initiating the crack (refer to Figure 18a); the only differences are lower load levels. All cracks were first initiated using the same fixture and the compliance feature.

After the crack was initiated and verified, the cracked tooth is equipped with two multi-wire crack propagation (CP) sensors, one at each gear face (Figure 18b). The number of broken wires of the CP sensor provides a measure of crack size. Figure 18c illustrates one such propagation: the blue traces indicate voltage levels of the two CP sensors (the left y-axis) and the red traces (the right y-axis) indicate the estimated tooth crack.

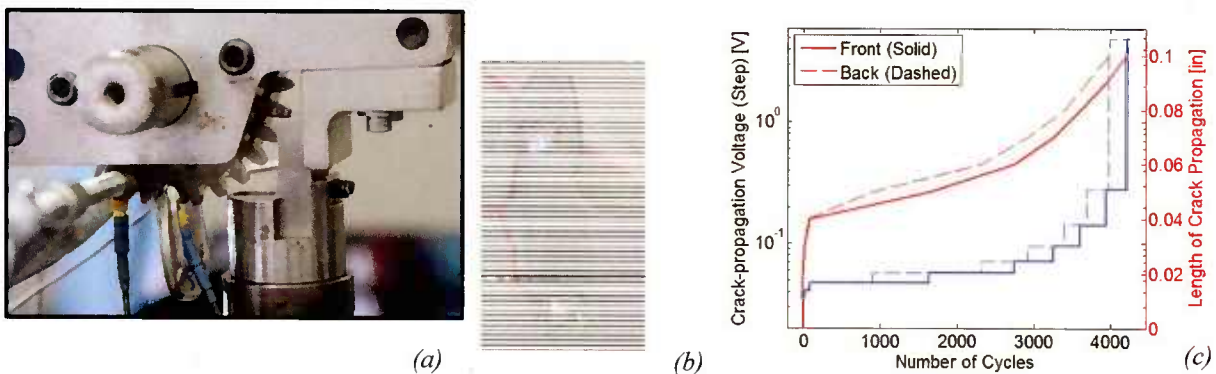


Figure 18. (a) Fatigue tester that applies the load on a single tooth. (b) The tooth is equipped with crack propagation sensors on each face. (c) Example crack propagation.

Results

Figure 19 summarizes the results of crack propagation on the fatigue tester. Figure 19a shows a histogram of the number of cycles spent for 11 different teeth subjected to the same loading conditions. The histogram employs a log scale on the x-axis to better show the large variance, spanning over a few orders of magnitude. Figure 19b depicts how the spread narrows as the crack propagates. Each gear tooth has its distinct marker. The blue markers indicate the “faster

propagating” CP, and the red the “slower propagating” CP. Note that the scale of the y-axis is logarithmic. Normal distributions were fitted at each stage, e.g., for a given number of broken strands on CP, and indicated with blue (faster propagating CP) and red solid line. All distributions are scaled by the same magnitude to enable their visibility. In the beginning, the distributions are so flat that the peaks do not show up for the chosen scale. The blue dotted line connects the means of the “blue distributions” and the shaded area indicates $\mu \pm \sigma$.

The microstructure strongly affects the crack propagation. In a few most dramatic instances, we observed a crack that starts propagating relatively steadily but then crack propagation seems to be halted. A better understanding of this phenomenon will be needed to understand uncertainty in prognostic assessments, or to improve gear prognosis.

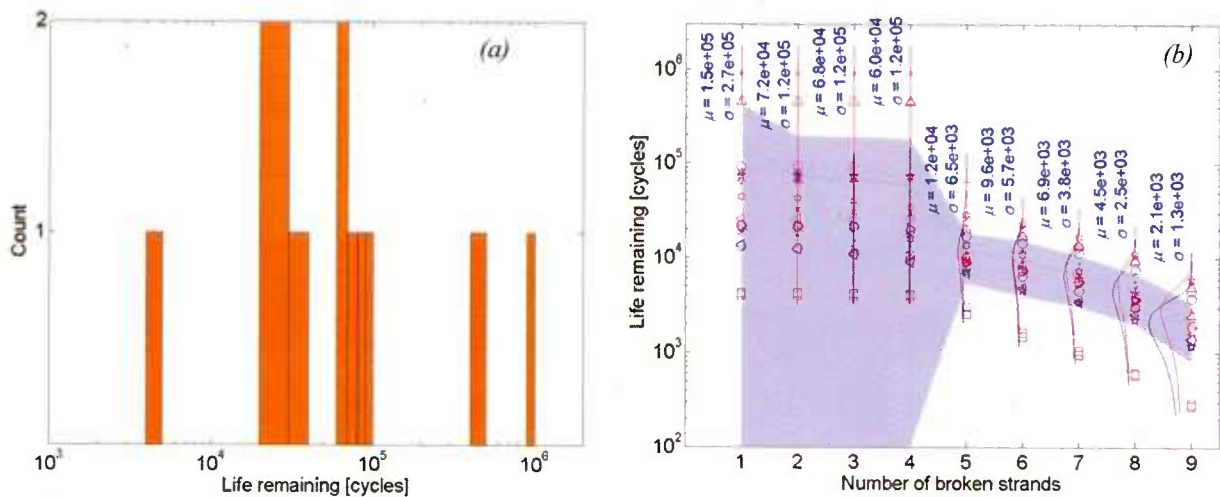


Figure 19. (a) Histogram of cycles for crack propagations on a logarithmic scale. (b) Life remaining in cycles distributions vs. the crack size as measured on gear faces with crack-propagation sensors.

Feature Analysis and Feature Fusion

Problem

Helicopter gear failures remain a problem for continued force maintenance and safety. One primary failure mode under study is the presence of gear cracks. Existing methods for in-place detection of spur gear cracks rely upon the use of Condition Indicators (CIs), which are thresholded or otherwise analyzed with low-order models to produce a decision. To date, CIs have not yet been compared and contrasted on a statistically significant data set.

Goals

We produced a novel CI test framework aimed at improving this situation. Our study was aimed at more reliably quantifying: 1) the ability to detect cracks prior to failure; 2) the ability to estimate damage levels; and 3) sensitivity to sensor placement. The overall intent was to provide a holistic study of CI performance as cracks progress from initiation to full spur failure.

Approach

CI is computed values originating from accelerometer readings. Thirty-five different CIs are computed, including Root Mean Square Value, Kurtosis, FM0, NA4, NP4, Autoregressive Prediction Error Kurtosis, and Autoregressive Prediction Error Sum (Sliding Window). The Autoregressive CIs were dubbed LP#Kur and LP#Sum where “#” is the order of the predictive model, given their expansion on Linear Predictive (LP) modeling. All CIs were computed for four accelerometers placed to simulate opportunistic deployment. CIs were computed over “bursts” of accelerometer data: 1 second of recording time at 1 kHz. Each CI was computed for five test gears on data prior to crack initiation (“pre”) and onwards to failure (“post”). CI performance was then evaluated in terms of classification-based crack identification (including class separability, threshold performance, fusion performance, and augmentation performance) and regression-based damage estimation.

Results

CI usefulness is bounded by the separability of values from healthy (pre-crack) conditions and those from damaged gears. We computed distribution statistics (e.g., mean, variance) for each CI on all gears and accelerometers. From these, we concluded that separability is sufficient for crack/no-crack classifiers. Further work was built upon this, aimed at finding a best classifier and its component CIs. Baseline performance was evaluated in (threshold-based) differentiating between pre-initiation data and early expansion (up to 10% surface propagation); Receiver Operating Characteristic (ROC) curves are shown in Figure 20. Initial conclusions support the continued use of NA4 and NP4 as primary CIs, with other CIs used supplementarily.

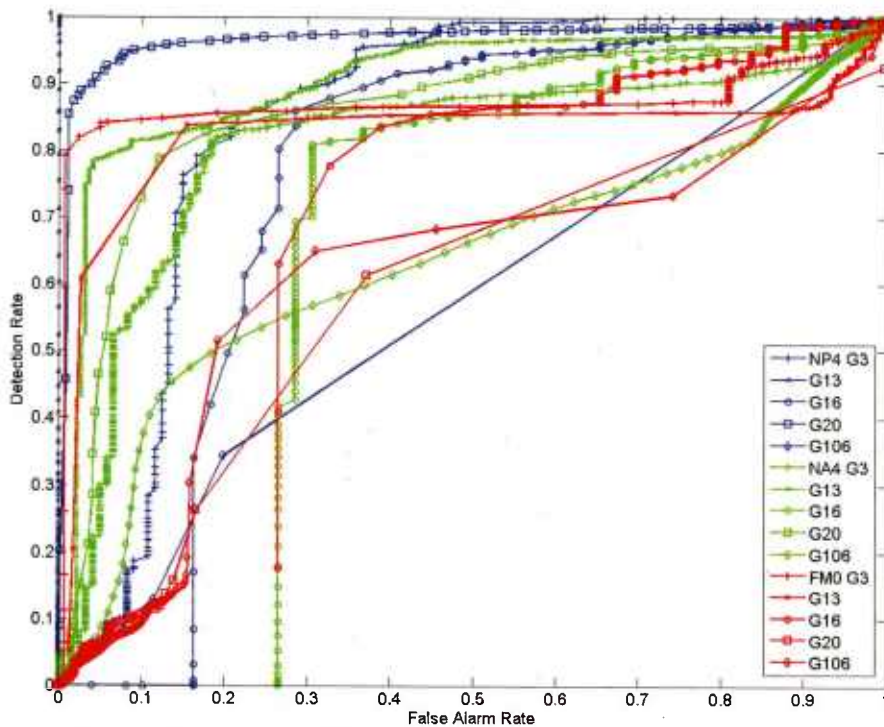


Figure 20. Crack Detection ROC Curves for CIs.

We also studied Logistic Regression and Boosted Regression Tree (BRT) CI fusion, with top performance occurring when using all CIs and BRT fusion. We then experimented with Detrended Fluctuation Analysis (DFA) as an augmentation on all CIs that were not based upon local trending (meaning the exclusion of kurtosis and autoregressive predictive CIs), and then compared ROC curves via delta (difference before/after) Area Under the Curve (AUC) for the 3 CIs. From our results, we discounted the possibility of DFA as an augmentation for NP4 and NA4 (poor performance), and performed additional study for FM0. These tests indicated that DFA should be used as a standard modification of FM0 for widespread use, tempered with spot checks to confirm specific viability (as poorly placed sensors saw no DFA improvement, although performance was never worse).

For the estimation of damage, we studied the use of a variety of methods, including Linear Regression, Support Vector Regression, Boosted Regression Trees (BRTs), Regression Hidden Markov Models (RHMMs), and Deep Belief Networks. We were able to produce accurate estimates of time-to-death for all gears using both RHMMs and BRTs; see Figure 21 for an illustration of BRT performance.

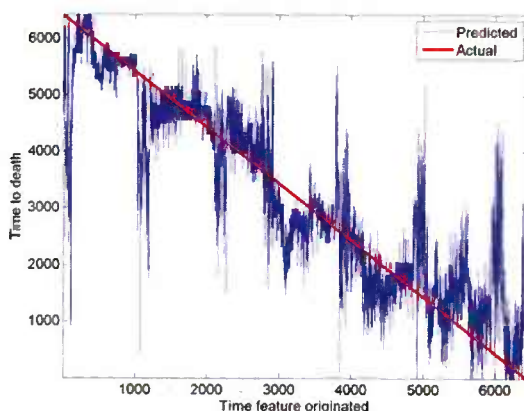


Figure 21. Estimating Time-To-Death using Boosted Regression Trees (all CIs).

Ultimately, our experimental framework proved successful in analyzing the performance of CIs in crack detection and estimation of damage, spanning a number of gears, with top- and side-mounted sensors and full CI fusion performing the best. Future work will focus on expanding the scope of the study to confirm statistical relevance and better differentiate between sensor placements.

Reverse Engineering: 3D Laser Scanning and CMM Integration

Problem

As noted earlier, as complex military systems age, it becomes increasingly difficult to obtain components for these legacy systems because OEMs have stopped producing the component or have ceased operations. In addition, when the military purchases a system they often do not purchase the complete technical package; as a result they do not have prints for the components. For this reason, there is a need to recreate the prints for these legacy components. One option is

3D laser scanning, which uses laser technology to produce detailed three-dimensional images of complex environments and geometries in only a few minutes. However, 3D laser scanning cannot capture high-precision surfaces.

Goals

The objective of this project was to develop and demonstrate a methodology for generating a 3D model of a complex component consisting of cast and machined surfaces using a 3D laser scanning system to capture the overall geometry of the component and then integrate the higher-precision surfaces and features of machined surfaces, obtained by a Coordinate Measuring Machine or other similar systems, into the point cloud.

Approach

The methodology developed consisted of first scanning the component using the 3D laser scanner using a scanner-arm combination that has an accuracy of ± 0.051 mm (± 0.002 inches), which is sufficiently precise for measuring “as cast” surfaces. This probe-arm combination was also used to provide point registration for accurately merging separate scan files.

The actual process of scanning is similar to spray painting. The operator must slowly sweep the laser over all surfaces to collect point data. Figure 22 shows a point cloud generated by the 3D laser scanner.



Figure 22 Point cloud of the top surface of a transmission tailpiece.

Next, the coordinate measuring machine (CMM) was used to measure and capture the machined surfaces. The CMM uses a physical contact probe that records single-point data and provides measurement accuracy of ± 0.005 mm (± 0.0002 inches). The CMM was also used for computing the dimensional relationships between measured features on a part, enabling the accurate dimensioning of physical relationships such as the holes angle and bolt holes pattern. In addition, other systems with similar or greater accuracies as the CMM, such as an optical gauging system or handheld measuring devices, also can be used to capture these surfaces.

Results

The laser scanning data was collected through the scanning-arm and probe-arm system by the Kube software program. This program was used for the point collection and initial point processing and was able to merge all the individual laser scans. However, Kube cannot take the data all the way to a solid model or a CAD software package.

Two approaches were used to transfer the point cloud data from Kube to a CAD software package. One was the curves approach, which uses Planar and Radial B-spline curves created when a defined datum's planes intersect collected points. These curves are then imported into CAD software for reference for developing a dimensional, 3D representation of the component.

Once this is complete, Geomagic Studio software was used to repair gaps in the scanned data and smooth the surfaces creating watertight surfaces. These surfaces were then used to create a NURBS (non-uniform rational B-spline) solid. The NURBS file was then used to produce the solid model. However, the model produced would not allow the production of a clean drafting file of the tailpiece. This issue is being worked out by the software suppliers.

In the second approach, Geomagic was used to qualify the accuracy of the 3D graphical comparison between digital reference models using scans and probes of as built parts. Final mechanical drawings were then created from the CAD-generated surfaces and features.

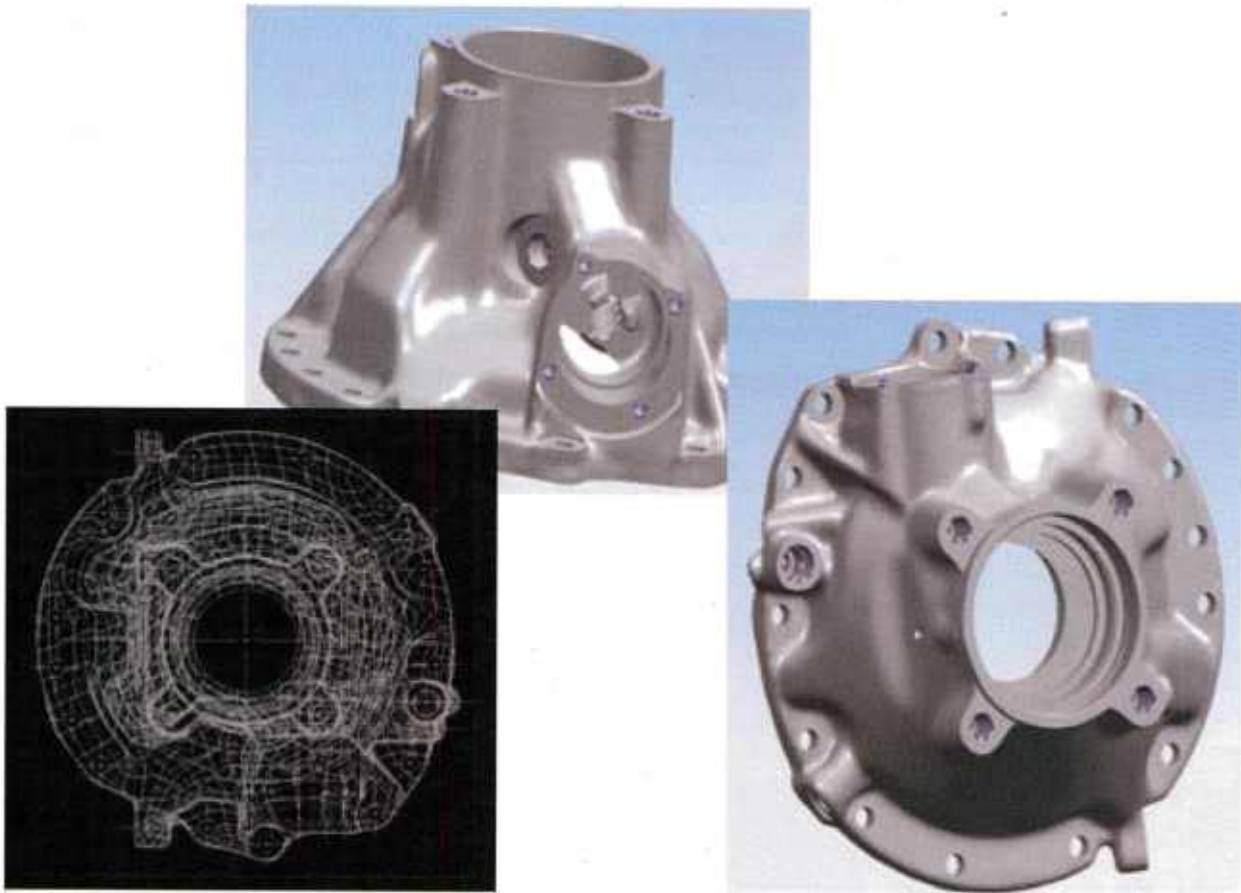


Figure 23. Completed NURBS model showing various views and model detail.

Conclusions

A methodology was developed and demonstrated to create an accurate CAD solid model of machined and non-machined contoured surfaces using a combination of 3D laser scanning and CMM measurements. It was demonstrated that this method, together with scanning a sufficient number of samples of the OEM component, would provide enough data to establish average dimensions and tolerances for the machined surfaces.

The outcome of the methodology development not only demonstrated a procedure but produced an accurate solid model that could be provided to a casting manufacturer. From the solid model the casting manufacturer was able to generate a mold design for casting the component.

LAV Fuel Efficiency Testing

Problem

The U.S. Marines were asked to try a fuel catalyst in their Light Armored Vehicles (LAVs). The catalyst was claimed to increase fuel efficiency by 12-23% and to decrease exhaust emissions by up to 82%. When the product was used in LAVs, operators observed increased engine exhaust temperature.

CIMS at RIT was asked to test the claims of a fuel catalyst manufacturer and monitor exhaust gas temperature to verify the Marines' observations.

Goals

1. Compare the efficiency, emissions, and exhaust temperature of a diesel engine operating with and without QuantumFire™ Fuel Catalyst.
2. Test the Marines' observations of increased exhaust temperature while operating with the catalyst.

Approach

CIMS' 350 HP Mustang Eddy Current engine dynamometer is housed in an Engine Dynamometer Lab. Also housed in the lab are an ECOM 5-gas exhaust analyzer, a precision fuel metering station, an Iotech Daqbook 2001 data acquisition system with thermocouple, transducer, frequency and digital inputs. A Cummins ISC-240 turbocharged, intercooled, common-rail injected engine capable of 240 hp and 660 lbf*ft of torque was coupled to the dynamometer and used to perform the requested tests.

Fuel efficiency, exhaust emissions and exhaust temperature measurements were made at four steady-state conditions as shown in Table 2 and represent a full range of engine operating conditions.

Table 2. Engine Test Point Settings.

Bin #	Torque (lb*ft)	Speed (RPM)
1	165	2110
2	330	1370
3	330	2110
4	495	1740

The engine was operated on untreated ultra-low sulfur diesel (ULSD) as a baseline test. It had also been run on this fuel as part of a 1700-hour study. Using the baseline test and the previous study test data, baseline fuel efficiency, emissions, and exhaust temperatures were recorded.

The four steady-state test points were repeated with ULSD treated with QuantumFire™ catalyst. John Dabels at QuantumFire, Inc., advised that the proper mix ratio is 1 part pre-mixed catalyst to 400 parts fuel. The fuel was prepared at that ratio. Measurements were made immediately and after every eight hours (approximately 30 gallons) of operation. The process was repeated until 96 hours of operation were completed and 130 gallons of fuel were consumed. After the treated fuel runs were complete, four 8-hour test segments were run on pure ULSD fuel with no catalyst.

Results

The measurements were tabulated and analyzed using Analysis of Variance (ANOVA) calculations in MiniTab™ statistical software. The ANOVA failed to discriminate the efficiency effect of the catalyst from random error. Figure 24 shows that the fuel catalyst effect is nearly non-existent compared to the engine loading conditions.

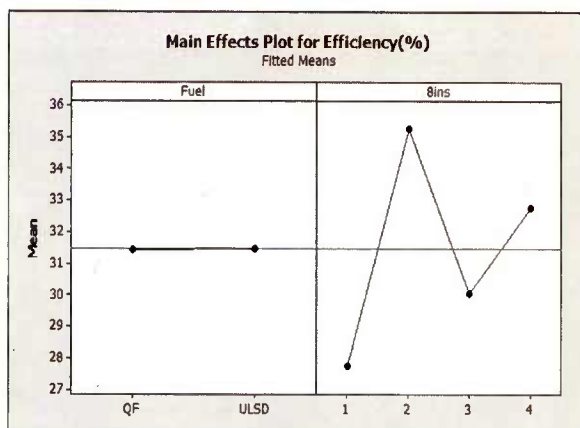


Figure 24. ANOVA results for efficiency.

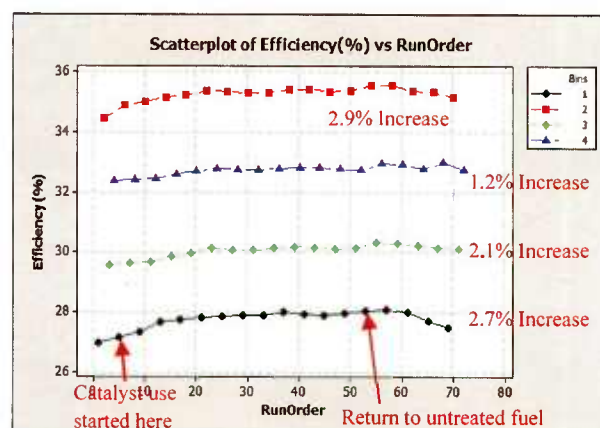


Figure 25. Run Order Trend of Efficiency.

Even though the statistical evaluation showed no significant effect, a plot of efficiency over time was analyzed. The time or Run Order Plot shown in Figure 25 indicates a gradual increase in efficiency over time that stabilizes after approximately 32 hours of operation. The average normalized efficiency change is 2.3% and varies with engine operating conditions.

Likewise, the exhaust temperature, throttle position, and emissions measurements were analyzed statistically and graphically. Table 3 contains a summary of the experiment results.

Table 3. Summary of catalyst test results.

Factor	ANOVA	Graphical Analysis
Efficiency	Not Significant	Slight improvement with catalyst
Throttle Position	Small reduction with catalyst	No visible change
CO	Slight increase with	Inconclusive – too much random

	catalyst	error
NO	Slight reduction with catalyst	Trend down then up with catalyst
NO2	Strong increase with catalyst	Large increase trend with catalyst
Exhaust temperature	No Change	No Change

In summary, the catalyst appears to have produced a 2.3% efficiency improvement. No detectable changes to CO or NO were observed. A significant increase in NO2 was observed while using the catalyst and a possible small decrease in throttle position was observed.

Platform Lifecycle Decision Support



Variant Manager

Bus Fleet

700

Air System / Kneeler

Valves & Switch

Air Solenoid Valve 1

Conditions

Air Solenoid Valve 1

Overall condition	Poor
Physical descriptors	Water Seepage
	Excessive Vibration
	Fuel Leakage
	Excessive Corrosion



5. Platform Life-Cycle Decision Support

5.1. Description of Platform Life-Cycle Decision Support

Platform Life-Cycle Decision Support (PLDS) is a set of processes and tools to aid in platform engineering decisions during the design, development, fielding, sustainment, and disposal phase of the platform life cycle. In particular, we are focused on research that facilitates cost-effective platform upgrades that allow the platform to adapt to changing operational needs and to utilize evolving technologies throughout its life cycle. With respect to modernization, the research focus will be on evaluating the impact and timing of technology upgrades (tech insertion planning).

The U.S. military ground and air vehicle fleet is composed of many aged platforms, some of which are well past their original design life. These older vehicles, together with newer vehicles, have undergone vastly accelerated aging over the last decade or more of military activity in Iraq and Afghanistan. Environmental factors such as heat, sand, and dust, in addition to operating these vehicles at a pace well in excess of normal peacetime service, has severely reduced their platforms' remaining life. The advanced use and extended service life of many military platforms requires the ability to effectively evaluate individual vehicles and fleets of platforms for repair, overhaul, technology upgrade, and/or replacement. PLDS tools are designed to provide aid in evaluating the operational effectiveness, reliability, and affordability of platforms, allowing Program Managers to better align their funding requests with related program strategies that will sustain, modernize, or replace existing legacy equipment systems.

5.2. Platform Life-Cycle Decision Support Research Objectives, Progress, and Results

A major objective of the Platform Life-Cycle Decision Support program is to determine what tools need to be provided to assist in making informed decisions over the course of a platform's life. Tools will be designed to target specific users, providing the information necessary to make decisions applicable to that user only. For example, a Program Manager may be interested in evaluating the increased effectiveness of a platform after a specific upgrade has been performed; but a Maintenance Manager is more concerned with the effectiveness of certain diagnostic procedures for identifying a particular failure. So as a first step toward implementing decision support systems, CIMS identified the users of the Lifecycle Engineering and Economics Decisions Support (LEEDS) system and the functionality that each user would utilize.

CIMS also performed an evaluation of the available technologies for the implementation of LEEDS in order to identify which one should be utilized. The existing LEEDS system was limited in its capabilities due to age, and was slowly becoming obsolete due to the increased

utilization of mobile devices. The objective of the evaluation was to identify features that the new implementation of LEEDS would require, such as the ability to display on a tablet or smartphone. Utilizing these features, an analysis of the available technologies was made to identify what technology should be chosen for the next implementation. It was determined that two technologies, Portals and Google Web Toolkit (GWT), were viable for meeting the implementation requirements.

The next step toward evaluation of the implementation technologies was to perform some initial development of basic features of LEEDS, such as the vehicle component hierarchical trees, mechanical drawing retrieval and display, and platform usage viewing. For the implementation, CIMS chose LifeRay Portal and Icefaces JavaServer Faces after an evaluation of the available portal technologies. The initial demonstration of LEEDS implemented using LifeRay allowed LEEDS to be utilized in all of the major browsers; in addition, users may configure the portlets to suit their needs and utilize roles to control what portlets different users may access.

An additional development activity was undertaken utilizing Google Web Toolkit (GWT). For this activity, a demonstration of a new maintenance decision support tool was developed. The system was designed to provide a library of solutions to maintenance problems based on the problem indicators on a platform. Through a case-based reasoning engine, a maintainer can enter a series of indicators to a problem, e.g., black exhaust smoke and lack of power, and be provided a number of solutions ranked based on their previous success at correcting the problem. This demonstration system was built as a module that could be integrated into LEEDS. GWT allowed the demonstration to be a highly user-interactive desktop application.

Also under the contract, CIMS evaluated two openly available Configuration Management systems. Since the current version of LEEDS software is designed around a custom CM system, and in the future we will be adding more functionality that will utilize and need to integrate with other CM systems, it was decided to look at a more fully-featured CM system. This analysis focused on the EIA-836¹ CM Data Exchange and Interoperability standard and MIMOSA. After comparing the capabilities of each system with the needs of LEEDS and evaluating the ease of implementation, MIMOSA was chosen to be the next CM system for LEEDS.

5.3. Platform Life-Cycle Decision Support Projects

The following projects were conducted during this contract period under the LEEDS[®] program:

Projects:

- Analysis of Users and Applications for LEEDS
- Analysis of Technologies for Next Generation LEEDS
- Implementation of LEEDS in a Portal-based environment

¹ <https://acc.dau.mil/CommunityBrowser.aspx?id=32229>

**Platform Lifecycle
Decision Support**

PROJECTS

- Development of a Demonstration Maintenance Decision Support Module for LEEDS Analysis of available data models, e.g., MIMOSA & EIA-836A.

Platform Life-Cycle Decision Support Projects

Analysis of Users and Applications for LEEDS

Problem

LEEDS is a web application that provides the following core functionality:

- A mechanism to construct an asset hierarchy that displays the asset, its subsystems and component parts.
- Resources can be associated with each level in the hierarchy. Resources include technical manuals, drawings, images and more.

Figure 1 shows the LEEDS asset hierarchy (System Tree) displaying an image of the selected component, the engine.



Figure 1: LEEDS Asset Hierarchy.

Initially, the next evolution of LEEDS was planning to focus on asset and fleet decision support. For example, LEEDS could provide support for economic decisions, such as repairing vs. replacing parts, or when to retire existing fleet vehicles and purchase new ones. However, in

order to be thorough, an analysis of the planned users and their required functionality must be undertaken. This effort will be the guiding requirements for implementation of future functionality for LEEDS.

Goals

The end result of the analysis is to enumerate the expected users of LEEDS, together with a list of the applications they would use. Utilizing the list, applications may be prioritized, and use-case scenarios may be developed for each class of user, which in turn, will drive our requirements for the LEEDS system. For the purposes of this exercise, the candidate users will be drawn from duties typical of a vehicle Program Manager's office, such as fleet or operations management, engineering and research; or from the unit level, such as operators or maintainers.

Approach

In order to derive a list of LEEDS users and applications, we held brainstorming sessions and collaboratively updated internal documentation. Members included those with knowledge of fleet and operations management, asset operation, asset maintenance, research, engineering, software and database development. Raw material we had to work with included:

- Existing LEEDS requirements.
- Existing LEEDS demonstration applications.
- Existing in-house applications with relevance to LEEDS.

The brainstorming sessions generated "user stories," which are scenarios describing tasks that a particular user would perform. From the user stories we were able to assign roles to various tasks, as well as identify application features that users would make use of. This led to:

- A list of user categories.
- A list of potential applications and their features.
- A list of enhancements to existing in-house applications.

Results

The major functions that an augmented LEEDS should support are:

- Equipment and technical data configuration management.
- Fleet operations status; e.g., deployment and location information.
- Asset service life decision support.
- Fleet life cycle decision support: investment planning.
- Logistics planning; e.g., parts sourcing and supply.
- Stakeholder communication and information exchange.
- Engineering change order support.

This functionality would be utilized by the following list of primary users:

- Operator
- Asset Maintainer

- Asset Subject Matter Expert (SME)
- Procurement SME
- Researcher
- Operations Manager
- Maintenance Manager
- Fleet Manager

A list of secondary users includes:

- System Administrator: administers the LEEDS system
- Applications Engineers and Software Developers: develops extensions to LEEDS
- Data Warehouse Administrator
- ETL tools, data generators, report generators (third-party tools that interact with LEEDS)

The following sections define the major applications that LEEDS would provide, together with the users of each application.

Maintainer User Interface

The Maintainer User Interface associates maintenance and tracking information with assets. This allows users to view a variety of asset information from a single location. A sample use case would be:

An operator brings a vehicle in for maintenance. The Maintainer accesses the asset using the Maintainer User Interface, views the maintenance history, finds that the faulty part has been fixed before, decides to replace it, looks up parts availability, and schedules the asset for repair.

Features:

- Retrieve and view technical publications.
- View real-time and historical vehicle data, analyze and diagnose asset issues.
- Compare asset performance to fleet performance.
- Initiate, document, and close maintenance service requests and record results.
- Provide information on parts availability.
- View asset maintenance history.
- View prognostic indicators.
- Update vehicle configuration information.
- Maintenance scheduling.

Users:

- Maintainers
- Asset SMEs
- Researchers

Fleet Maintenance User Interface

The Fleet Maintenance User Interface provides a Fleet Maintenance Manager with a view of the fleet. Statistics are available at the fleet level, for groups of vehicles, and for individual vehicles. A sample use case would be:

A Fleet Maintenance Manager views the maintenance history for an asset that has been recently serviced. A part was replaced, and it showed unusual wear. The manager identifies all assets that have the same part installed, and schedules them for inspection.

The Maintainer User Interface would be accessible from the Fleet Maintenance User Interface in order to provide maintenance information to the fleet maintenance manager.

Features:

- Fleet benchmarking, trending and problem reporting.
- Management of technical publications.
- Analysis of fleet-wide maintenance issues.
- View vehicle data for analysis and diagnosis.
- Manage the archiving of vehicle data.
- Provide information on parts availability.
- Initiate maintenance service requests.
- Develop condition-based maintenance (CBM) plans.

Users:

- Asset SMEs
- Procurement SMEs
- Maintenance Managers
- Fleet Managers

Researcher's User Interface

The Researcher's User Interface presents information used in asset research. For example, for vehicle assets the user interface would allow the researcher(s) to view and manipulate sensor data. A sample use case would be:

A Researcher is interested in reducing vehicle fuel usage. They choose a vehicle using the Fleet User Interface and download sensor data from that vehicle. A graphing tool allows them to view, for example, the sensor data, the RPM or shifting data. The sensor data can be used to design algorithms that identify shift patterns that may reduce fuel usage.

Features:

- Fleet benchmarking, trending and problem reporting.
- Maintenance of technical publications.
- Analysis of fleet-wide maintenance issues.

- View vehicle data for analysis and diagnosis.
- Download asset data to local database for cleansing, manipulation, and simulation.
- Receive notifications when new asset data has been uploaded to the database.
- Provide information for algorithm development.

Users:

- Researchers

Node Maintenance Tool

The Node Maintenance Tool is used to manage the software and resources of an on-board system.

Features:

- Update software and firmware for the on-board health monitoring system.
- Configuration and control of the health monitoring system.
- Collect health monitoring system log files.

Users:

- Maintainers

Developer's Analysis Tool

The Developer's Analysis Tool supports software and system developers to design and debug the health monitoring software and on-board system.

Features:

- View health monitoring system log files.
- View standard usage reports.
- View standard error reports.

Users:

- Applications Engineers and Software Developers

Fleet Operations User Interface

The Fleet Operations User Interface provides many of the same views of the fleet as the Fleet Maintenance User interface and adds operation planning and tracking information.

Features:

- Interactive map displaying asset locations, speed, heading, routes (planned and actual).
- Display asset readiness (status, cargo, health, fuel, performance, task assignments, and adherence).
- Access to positional, logistic, and fault data.
- Asset task and route planning.
- Exception reporting for assets operating outside a planned task/route.

Users:

- Operators
- Fleet Managers
- Operations Managers
- Maintenance Managers

Remote (On/Off-Board) Operator's Display

The Remote Operator's Display provides asset and mission planning and tracking information.

Features:

- Manage inspection checklist.
- Automatically populate current health data.
- Provide current health and logistics data.
- Auditory input and feedback.
- Input logistics data through cargo scanning.

Users:

- Operators

Reliability Centered Maintenance (RCM) Tool

The Reliability Centered Maintenance Tool supports the use of condition metrics in order to maintain assets before they encounter a failure.

Features:

- Support monitoring of assets.
- Scheduling of maintenance.
- Provide engineering support.
- Perform failure mode analysis.

Users:

- SMEs

Analysis of Technologies for Next Generation LEEDS

Problem

LEEDS was first developed in the year 2000 as a standalone MS Access database. In 2003 a conversion to a web-based application established the current architecture. LEEDS has had many incremental functional additions and improvements over time, but the original 'look' and underlying architecture design is still in place. The key problem with the existing implementation of LEEDS was it was outdated in terms of architecture and appearance. Specifically:

- LEEDS was designed for Internet Explorer 6, was not compatible with more recent versions of Internet Explorer, and was not compatible with any other browsers. This places restrictions on LEEDS users.

- LEEDS used an older technique for database interactions. Newer frameworks, such as LINQ, support a wider range of database sources and provide more flexible programming.
- The LEEDS user interface was HTML Frame-based that has fixed areas for features; contemporary design uses a more fluid layout viewable on a wider range of devices.



Figure 2. Existing LEEDS showing the use of fixed frames.

Goals

Evaluate current and emerging technologies as candidates for the implementation of a new LEEDS. These technologies target:

- Web and desktop application development frameworks.
- Web server alternatives.
- Database technologies.
- Communication protocols.

Approach

At the time of this exploration, the “Semantic Web” and personalization were beginning to reshape the world-wide web. At the same time, numerous frameworks were being developed to support development of new web applications. The development of LEEDS for the Semantic Web would lead us to evaluate a number of technologies, with the following considerations:

- The technologies could address our current LEEDS requirements and be flexible enough to address new LEEDS requirements.
- The technologies must be robust and scalable.
- The technologies should be relatively familiar for our typical developer base.

Figure 3 provides an illustration of the variety of technologies available for client applications.

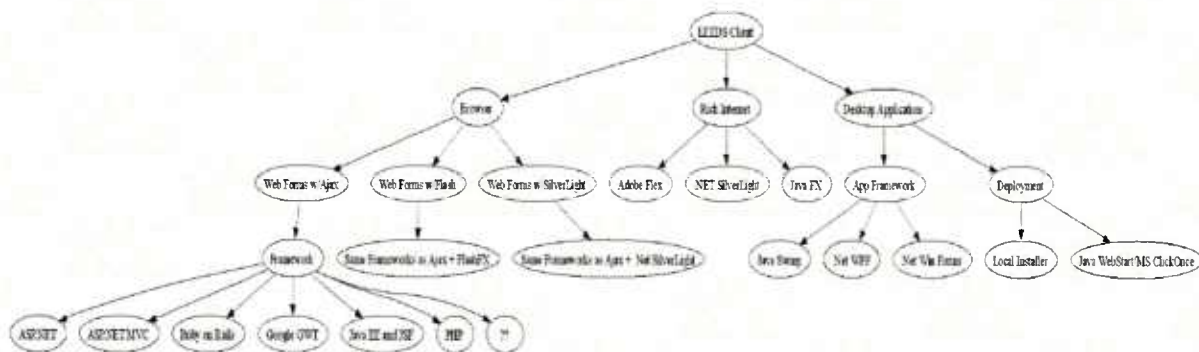


Figure 3. Client technology alternatives decision tree.

Desktop Applications

We investigated the following desktop technologies:

- Java Swing – Java-based user interface technology.
- .Net Windows Presentation Foundation (WPF) – latest Microsoft user interface framework.
- .Net Win Forms – traditional Microsoft user interface framework.

There are issues involving the deployment of desktop applications:

- Installation is required on a client machine, often with platform-specific considerations.
- Users decide when to acquire and install updates so several versions of the application can be found in the field, requiring developers to maintain multiple versions at the same time.

Two technologies were investigated in order to alleviate the limitations with deployment of desktop applications. These technologies support launching applications from the internet rather than downloading and installing the applications to keep all versions up to date. These technologies were:

- Java Web Start
- Microsoft ClickOnce

Rich Internet Applications (RIAs)

RIAs are web applications that share many characteristics with desktop applications. They run faster and are more interactive than traditional HTML-based web pages. Some of the RIA frameworks we investigated were:

- Adobe Flash and Adobe Flex
- .Net Silverlight
- JavaFX

However, RIAs also share a limitation with desktop applications – deployment. In the case of RIAs, it is not the application that needs to be deployed and updated but rather a browser plug-in that runs the application. The user must install the browser plug-ins prior to using the application, and often to every browser they use.

Web Application Frameworks

The majority of the evaluation was spent on client technologies, specifically web-based frameworks. Web applications have the following advantages over desktop applications and RIAs:

- No installation necessary; clients simply navigate the application in a browser.
- Web applications are updated on the server, so clients are all using the same and the latest version.
- User interface controls have caught up to the level offered by RIAs.
- By running in a browser they are “sandboxed,” which prevents them from negatively affecting the client machine.

One requirement we placed on the web application frameworks was that they must support Asynchronous JavaScript and XML (AJAX or Ajax). Ajax is a technique for communicating with a server without interfering with the current display. This makes for highly interactive web pages that do not require full page refreshes, making them appear more like desktop applications and RIAs.

We investigated the following web application frameworks:

- ASP.NET, and ASP.NET MVC (Model-View-Controller version of ASP.NET)
- Ruby on Rails
- Google Web Toolkit (GWT) – Java framework for creating Ajax applications
- Java Enterprise Edition and JavaServer Faces
- PHP – Scripting language embedded in HTML

Database and Communication Protocols

The analysis of database technologies and communication protocols was extensive enough to be moved to its own area named “GLEAM,” which is covered in another section of this report.

Results

We favored Java-based web applications because of our extensive Java experience, and because web applications solve the deployment problems mentioned above. We also used client frameworks with extensive libraries of user interface components.

We developed two proof-of-concept applications: one used Portal technology; the other used GWT.

A Portal is a browser application that hosts small, self-contained applications called “portlets.” The portlets can be arranged by the user and are able to communicate with one another.

The Portal provided the following features:

- Built in Ajax support.
- User personalization support and theme support.
- Extensive component library provided by JavaServer Faces.
- Ability to incorporate third-party portal applications.

An advantage of developing a portal-based application is that the individual portlets can be developed separately and “pulled” into the portal seamlessly. Unlike a web application, it is not necessary to modify the overall application to incorporate new features.

The GWT implementation also made use of built-in Ajax support and provided an extensive library of user interface components, including drag-and-drop. An additional benefit of GWT is that the application is written in Java and then translated into JavaScript for the browser.

We found that GWT is very useful in developing highly interactive, desktop-like applications for the browser.

Implementation of LEEDS in Portal Environment

Problem

The implementation of LEEDS in a portal environment grew out of our investigation of technologies for improving LEEDS covered in a previous section of this document entitled, *Analysis of Technologies for Next Generation LEEDS*.

Life Cycle Engineering and Economic Decision Support System (LEEDS) web application is/was designed to assist in making economic decisions directed at maintaining assets. Some of the answers it seeks to provide are:

- Is it more economical to repair or replace a component?
- Is it more economical to remanufacture an existing asset, or to purchase a new asset?

The LEEDS application draws information from a database of asset characteristics and analysis data. During the decision-making process, new information is added and existing information is updated.

The existing LEEDS application has several shortcomings:

- Only compatible with an older version of Microsoft Internet Explorer.
- Pages do not support user configuration.
- Adding new features means modifying the entire application.

- All of the content (drawings, manuals, images, video) are kept in the database.
- Appearance is dated.

Goals

Develop a LEEDS application with consideration given to the following software characteristics:

- Interoperability – Functional in a variety of browsers, on a variety of operating systems.
- Scalability – Functions independently of the size of the database.
- Usability – Ease of understanding, use of common presentation controls.
- Extensibility – Ability to add features without modifying the system.
- Modularity – Software broken down into reusable functional units.
- Security – User authentication and user groups.
- Performance – Provides timely responses.

Approach

As explained above, a Portal is a web application that hosts “mini-applications” called “portlets.” Each portlet serves a particular function; however, portlets are able to communicate with each other as well. The combination of individual portlets leads to a cohesive portal application. A portal application addresses our goals in the following ways:

- Compatible with all major browsers (interoperable).
- Users are free to configure their pages to suit their needs (usability, extensibility).
- Portal application is easily updated by simply dragging new portlets into the portal. This includes the ability to use third-party portlets such as email and calendar portlets (extensibility).
- Integrated Content Management System (CMS) provides uniform access to content, together with support for collaboration and versioning (scalability).
- Theme support allows the user to customize their environment and brand an implementation of LEEDS for a particular industry or purpose (usability, extensibility).
- Management of user roles and groups, providing controlled access to portlets based on roles (security).
- Developers can work on separate portlets without interfering with each other, and integration into the portal application is managed by the portal, so there is no need for additional code to tie a portlet into the overall application (modularity).
- The portlets access their data source and update their views without the need for full-page refreshes (performance).

Results

After evaluating various portal applications and component libraries, we decided on the LifeRay Portal (www.liferay.com) and the Icefaces JavaServer Faces implementation (www.icesoft.com). The reasons for choosing these were:

- Offered free community editions.
- Feature rich.
- Well documented compared to other options.

Approximately 50% of the existing LEEDS technology was fully implemented in the portal application at the time we concluded the proof-of-concept. A screen shot can be seen in Figure 4.

In addition to replicating a number of LEEDS functions, we also made the following enhancements:

- Implemented several user configurable themes and layouts.
- Offered a variety of user interface components that the user is free to choose from; e.g., navigation through an expandable tree or using a more compact menu component.
- Defined user groups that controlled access to portlets based on role. For example, a Maintainer would have a different view of the application than a Fleet Manager.

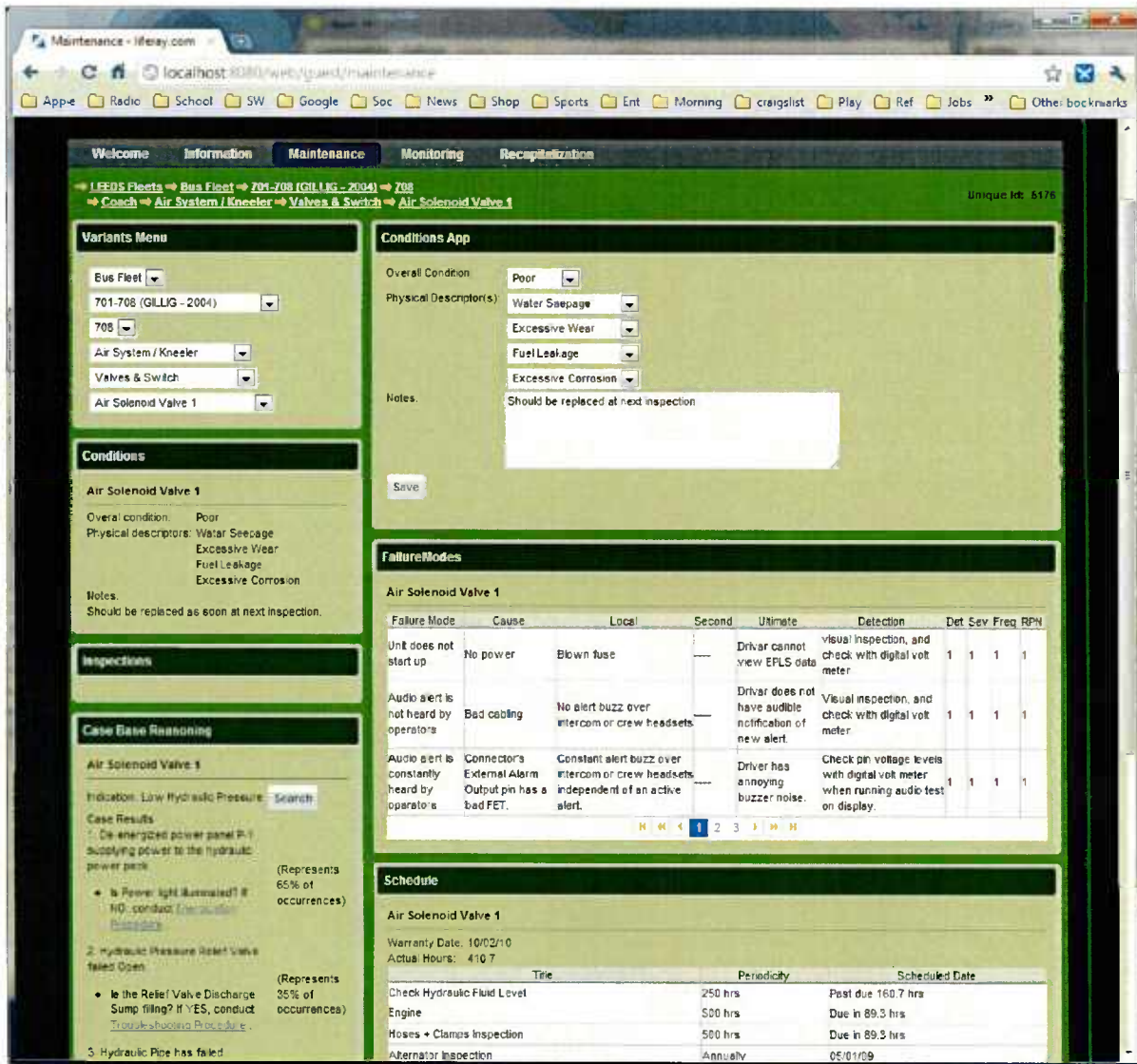


Figure 4. Screen shot of a portal page showing several portlets.

Development of a Demonstration Maintenance Decision Support Module for LEEDS

Problem

A common problem in maintenance facilities is the retention of experiential knowledge by specific employees. It is not uncommon for an employee to be known as the only person who can fix a certain type of problem. Often it is not the case that the person is a better maintainer; just that he has had more experience with that type of problem and the necessary maintenance procedures to alleviate the problem. The challenge is how to capture this knowledge and make it

available to all of the maintenance personnel so that certain problems do not have to wait for “the expert.”

Goals

The goal of the Maintenance Decision Support Module for LEEDS project is to develop a demonstration of a maintenance-decisions-support module that captures the knowledge gained through experience and makes that knowledge available to other maintenance personnel. The system provides a Case Based Reasoning (CBR) system to provide suggestions for maintenance based on previous solutions to problems. Thus, when a user asks for a suggestion on how to solve a problem with a set of indicators the system will output a set of potential solutions together with a rating as to how well that solution fits the existing problem. The user may then elect to perform one of the suggested procedures to address the problem. Upon completion of a successful solution to a problem, the system will be provided with the necessary information to update the CBR system. Figure 5 provides a high-level representation of such a system.

Approach

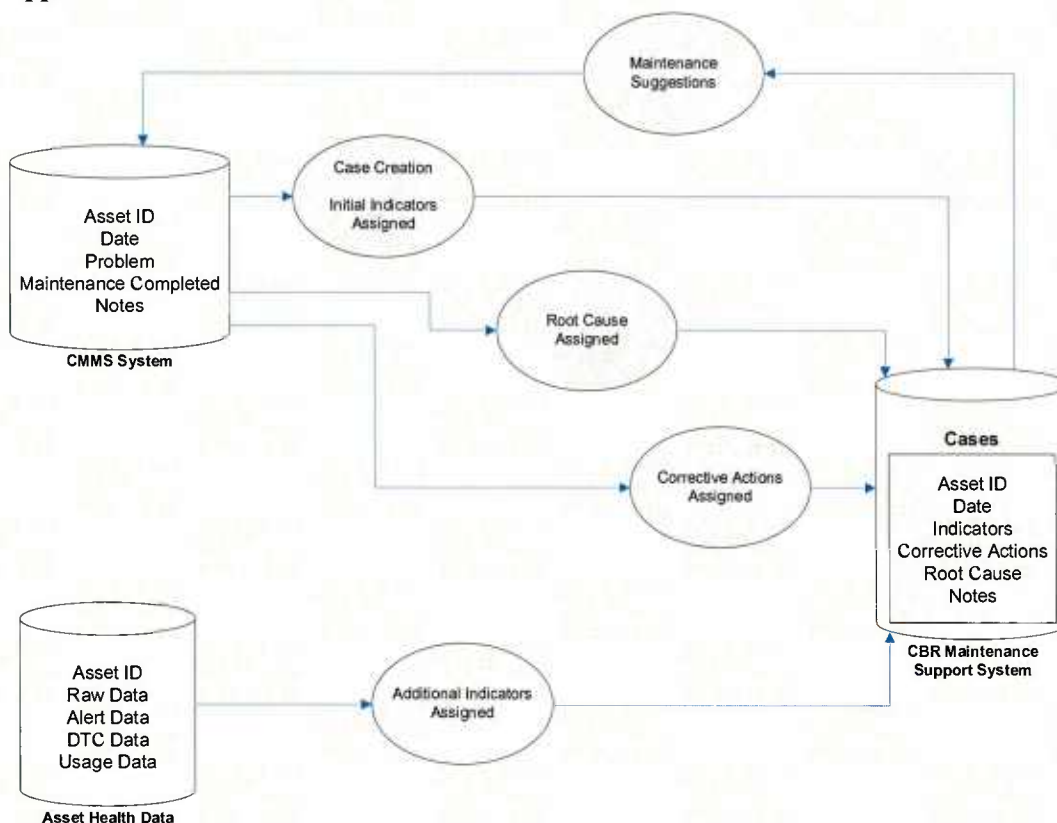


Figure 5. Maintenance Decision Support.

Existing Computerized Maintenance Management Systems (CMMS) capture data related to the platform, the problem being reported, as well as the parts and labor that went into the repair. However, the data is typically not as simple as *problem* → *solution*. For example, a repair order may consist of a transmission problem but the work performed may include an oil change

because it happened to be due. There is no correlation between the problem and the oil change although they appear together on the same work order.

For the decisions-support system to be effective, we needed to capture problem indicators, maintenance actions applicable to the indicators, the failure mode, and if the problem was resolved. Utilizing this data, a case-based reasoning engine was created that examines the problem indicators and matches them to previous cases that resolved the problems.

To design the system, the users were identified as the maintainer and the subject matter expert (SME). Utilizing these roles, simplified use cases were developed to show the steps that would need to be taken between the users and the software. The maintainer is responsible for asking for assistance from the system. The SME retrieves the problem reports (observed conditions from the repair order or alerts and Diagnostic Trouble Codes [DTCs] from a platform-monitoring system), identifies the appropriate corrective actions from the repair order, creates the cases, assigns the failure modes, and closes the cases as having been resolved. As the collection of closed cases grows, the system will be able to better identify the corrective actions required through a matching algorithm utilizing the problem indicators.

Results

Initial demonstration software was developed utilizing Google Web Toolkit. Without a CMMS system to mate the software to, a page was developed for creating and populating Work Orders. Additionally, Alerts and DTCs may be created for demonstration purposes. The SME will work from completed Work Orders as a way of populating the system with the knowledge that already exists.

Additionally, when a maintainer needs assistance the SME will create a case with the known problem indicators and ask for suggestions. The system will then compare the list of indicators with the indicators that have cases associated with them in the system. Through a matching algorithm, the system will return a list of suggested actions to take as well as a rating of how effective those actions have been in the past for this type of problem. For example, a problem may have been solved 100 times through repair sequence 1, and it was solved 5 times using repair sequence 2. Repair sequence 1 will be recommended first, together with a confidence value that takes into account the successful repairs as well as how well the indicators match.

In some instances, a problem may not exist in the system; therefore, the system will return no suggestions. At this point, the SME and maintainer are still required to troubleshoot the problem. The resulting repair will then be entered in the system so that it may be suggested for the next occurrence of the problem.

In the next iteration of the software, the following enhancements would be beneficial: 1) linking the system through an API to a CMMS system; 2) defining a way of associating similar platforms (e.g., a Chevy Silverado and a Chevy Tahoe may share the same engine or

transmission so certain repairs should be applicable across both platforms); 3) incorporating failed procedures into the matching algorithm to reduce false positive matches.

Analysis of Available Data Models

Problem

Organizations with large-scale systems with many variants have to deal with the issue of configuration management. Configuration management (CM) is an engineering and business process approach to maintaining equipment over its lifetime, tracking what components are installed in the system, and providing information on how to recreate, test and maintain those components. Within configuration management there are 3 main thrusts when CM is integrated into the business systems: during design of the products; during manufacturing (as made); and during maintenance. As part of this project we investigated two data models that are used to track a product's configuration and maintenance.

Goals

LEEDS is RIT's tool for maintaining engineering and assessment data on a set of platforms. We investigated two international standards for organizing CM data: EIA-836² CM Data Exchange and Interoperability and MIMOSA³, to determine which standard to adapt for basing our next-generation LEEDS product upon.

Approach

From prior work investigating CM systems, we knew which particular standards we wanted to analyze in-depth. The initial review of standards consisted of reviewing material available on the internet for both models, and going through the XML which is used to describe the data models. We converted the XML to graphical models, enabling us to more easily visualize data relations and requirements. For MIMOSA, we joined the MIMOSA trade alliance organization, which gave us access to design documents and application notes.

After understanding the data models, we evaluated applicability to the LEEDS system by looking at which types of data are required by the data model and making comparisons to existing data collected by the current LEEDS system. Another factor in the analysis is which markets are implementing the standard. In the end, the MIMOSA system was selected for additional development of a test system and for building a database and generating code for evaluation.

Results

MIMOSA was selected as the better candidate for future LEEDS data designs. It has a broader industry acceptance and we have prior experience with it: our AHM system already uses a subset

² <https://acc.dau.mil/CommunityBrowser.aspx?id=32229>

³ <http://www.mimosa.org/>

of the standard. Also, as RIT expands its capabilities into diagnostics tools the MIMOSA design is better able to link AHM health data (sensor readings and alerts) to physical systems.

Remote Monitoring & Advanced Support Concepts



Mission Selection for LAV-583

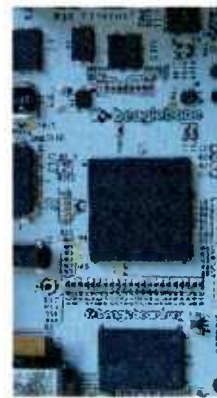
Find missions

☒ Where vehicle moved

☒ Where alert

☒ Longer than

Index	Start Time	End Time
55	8/19/04 11:00	8/19/04 18:33
58	8/20/04 19:14	8/20/04 21:44
99	8/23/04 17:29	8/23/04 18:40
101	8/23/04 19:21	8/23/04 20:50
111	8/24/04 10:15	8/24/04 15:18
126	8/26/04 14:25	8/26/04 18:29
143	10/18/04 15:03	10/18/04 17:01
144	10/20/04 13:43	10/20/04 15:05
145	11/12/04 15:32	11/12/04 16:43
160	12/7/04 10:47	12/7/04 11:15
168	12/8/04 14:47	12/8/04 16:17



6. Remote Monitoring & Advanced Support Concepts

6.1. Description of Remote Monitoring & Advanced Support Concepts

Remote Monitoring and Advanced Support Concepts is the research of enhanced health assessment technologies and health management approaches for high-value equipment and fleets of equipment. It is the decision-support technologies for use by maintainers, fleet managers and operations managers to allow them to manage and sustain their assets effectively. The core technology is health monitoring algorithms installed on equipment outfitted with appropriate sensors and asset health-management systems, communications networks, and appropriate software tools to aid stakeholders in their particular role in the life-cycle support effort.

The recent operations in Iraq and Afghanistan have confirmed the need for DOD transformation to more mobile and flexible military forces and to new logistics and sustainment concepts to maintain these units. A number of initiatives are underway within the DOD such as Condition Based Maintenance + to support this transformation. These system concepts rely on mobile communications networks, increased status reporting from weapon system platforms (battlefield OnStar) as well as planning tools within command and support services that allow them to rapidly analyze and respond to battlefield needs. However, significant technology gaps in remote platform health monitoring and condition-based sensing, particularly for legacy vehicles, are slowing this transformation. Further pressure is placed on these older systems as new logistics and support concepts being developed by the DOD also require improved visibility of the state of military equipment, including position, health, fuel, ammunition, and supply levels.

To support advancement of ONR initiatives in autonomic logistics, RCM, and CBM+, CIMS has performed projects in the following areas: Applied Remote Monitoring & Support Technologies, Asset Health Management hardware & software improvements, Advanced Anomaly Detection & Diagnostic Framework Technologies, and Improved Maintainer's Tools.

Applied Remote Monitoring & Support Technologies focuses on implementing AHM systems on multiple platforms for the purposes of collecting data for analysis and demonstrating new AHM algorithms & technologies. Asset Health Management hardware and software improvements consists of projects to develop new data acquisition and system health node hardware, as well as new software protocols and implementations that broaden the capabilities of the AHM system. Advanced anomaly detection & diagnostic framework technologies consists of data mining and algorithm development activities that advance the diagnostic and prognostic capabilities within AHM. Improvements to the maintainer's tools will provide the maintainers and system developers the ability to analyze data during missions through an improved user interface.

6.2. Remote Monitoring & Advanced Support Concepts Research Objectives, Progress, and Results

Applied Remote Monitoring and Support Technologies

Reliability Centered Maintenance is a tool for analyzing systems and failures on a platform. The results of the analysis are a series of recommendations surrounding preventative maintenance, condition monitoring, and system/sub-system redesign. Under this contract, CIMS facilitated an RCM analysis of the new LAV AT A2 Turret. The resulting analysis provided the LAV community with an understanding of the potential functional failures of the system and a preventative maintenance program design that would minimize failures while in the field.

Asset health management technology research has been applied to varying degrees, including implementation as part of the Embedded Platform Logistics System (EPLS). To further this research, CIMS performed implementation of AHM on multiple platforms to further collect data and implement new diagnostic and prognostics algorithms. AHM was implemented on a commercially available engine, representative of an Unmanned Aerial Vehicle (UAV) engine, which had previously been converted to run on JP-8. The implementation allowed for development of an algorithm for prediction of glow plug failures. AHM was additionally installed on two engines on an engine dynamometer, allowing for development of algorithms that can detect abnormal fuel use or high exhaust gas temperatures based on current operating conditions. An implementation of AHM was also performed on the Grizzly AVGP to enable a practical implementation of improved algorithms on a real-world mobile platform. To further advance AHM on the LAV, a study was performed on the vehicle data from the EPLS program, specifically related to known failures and the correlating data in an effort to improve diagnostic algorithms. This analysis resulted in a correction of some algorithms and a better understanding of the data required for improving diagnostic and prognostic algorithms.

AHM 3rd Generation Hardware Improvement

To advance the implementation and capabilities of the AHM hardware, CIMS performed multiple hardware development and feasibility studies. Low-cost vibration monitoring equipment is one of areas CIMS researched. CIMS designed and evaluated hardware that would collect and process vibration data from a 3-axis accelerometer utilizing algorithms developed in MATLAB. It was determined that the hardware selected was not capable of processing vibration data more than 20 kHz and that utilizing MATLAB for the algorithms still required extensive knowledge of the C code on the hardware. In order to be able to transfer data from the new hardware, a protocol was developed that would transfer the waveforms across a J1939 data bus. The speed of the data bus limits the transfer to a 6 kHz signal continuously. However, to mitigate data transfer issues, the protocol implements triggers which will send bursts of data across rather than continuous data, thus reducing traffic on the data bus.

Two additional hardware feasibility studies were completed for a compact USB interface to J1939/J1708 data buses and for running AHM on a low-cost ARM based processor running Linux. The USB interface was designed and determined to be capable of reading from both a J1939 and a J1708 data bus. The limitation on this interface is the susceptibility of USB to noise, which is of particular concern in vehicles. In an effort to run AHM on low-cost ARM-based hardware, CIMS chose an off-the-shelf processor board and built other daughter boards that were capable of collecting data from the vehicle data bus, an accelerometer, GPS, Wi-Fi, cellular and Bluetooth communications. The ARM processor was easily capable of running AHM and collecting data. This proof of concept also means that AHM could be run on a commercial piece of hardware like a cell phone if a vehicle is equipped with Bluetooth data bus adapters.

For its final area of hardware development, CIMS chose to add OBD-II capability to AHM. OBD-II is the standard data bus and communication protocol for light vehicles, i.e., typical cars and trucks. Because the military has a large number of support vehicles that are light duty, adding this capability to those vehicles would enable the military to monitor the majority of its vehicle fleet. The hardware was developed using a commercially available chip that interfaces with OBD-II and utilizes the power on the OBD-II data bus to cause the hardware to start and shut down. The major difference with the OBD-II protocol is that data must be requested from the data bus, as opposed to J1939 and J1708 which typically transfer data at a certain rate. As part of this program, CIMS developed software that would request and parse the data from OBD-II into AHM.

AHM 3rd Generation Software Improvements

Multiple software development projects were undertaken to improve the AHM software. Previously, AHM utilized MySQL for the onboard database, which is limited by licensing, reliability, size, and space management. So in an effort to remove MySQL, CIMS developed a flat file based data storage mechanism that is more compact, faster, reliable, and robust to system power losses. In addition, a flat file receiver was written that would automatically receive the files, parse them, and input the data to a Master database on the backend. The user interface for the receiver was a minimalistic design that informs the user if problems have occurred, i.e., a file was not received by the server. Combined, these efforts replaced MySQL and the MySQL replication process used for transferring data to the backend server.

After implementing AHM on many vehicles, including those selected for the EPLS program, it was determined that AHM had a slight flaw in handling the J1939 data bus signals. Originally AHM was designed to be utilized with a J1939 data bus designed and implemented on vehicles that previously did not have a data bus. Therefore, by design, no two ECUs would report the same signal. However, it was determined that standard vehicle data buses do not adhere to this rule. Therefore, we could potentially have two data buses reporting vehicle speed utilizing two different sensors. This becomes problematic when one sensor goes bad because the signal might jump between a good value (60 km/hr for example) and a bad value (0 km/hr if the sensor is

broken). This jumping back and forth would trigger the data to be marked as bad, even though one of the signals is good. CIMS implemented a signal selection technique that would allow the AHM programmer to prioritize the ECUs that it would like to utilize. The system would then evaluate the signals in priority order until it determined that one of them was good and then it would populate the data port with data only from that known good ECU and sensor. In addition, for tracking purposes, the system was designed to keep track of which ECU the stored data was received from.

With hardware having been developed to send waveforms off of a platform, a tool was needed that would allow the data to be visualized for evaluation. The tool allowed the various waveforms to be loaded and graphed, as well as the ability to save the data into CSV files for use in MATLAB. The tool was generally used for debugging purposes.

One of the future goals of AHM is to be able to have an engineer be able to set up all aspects of AHM without requiring programming skills. In a first step toward this goal, CIMS developed a tool for programming the data acquisition nodes so that an engineer does not need to write code in C. This tool utilized a database of J1939 signal definitions to allow the user to select which signals he wanted to output. Through a graphical user interface, the user can select the hardware channel that he would like to use, define any sensor conversion that needs to be done, apply sensor fault limits, define sampling and filtering rates, perform any calculations that the user would like, and select the J1939 signal definition to report the data through. After defining all of these factors for each signal on a DAC, the user can select for the system to create the programming file for the DAC node. The DAC programming tool was successfully demonstrated on the Grizzly vehicle when an AHM system was applied to it.

As previously stated, AHM was originally designed to be utilized on vehicles that did not have a data bus. As the system was designed by an engineer and implemented, all processing for vehicle alerts could be done at the System Health Node. However, as AHM was moved forward and added to vehicles containing existing data buses, AHM did not have the ability to process alerts, also known as Diagnostic Trouble Codes, from existing vehicle ECUs. Under this contract, a data model was developed and data parsers were added that could translate the existing DTCs into a format that could then be saved in the database. DTCs are now capable of being collected from both the J1939 and J1708 data bus.

In order to build an AHM application for a platform, an engineer must have an understanding of what signals exist on a platform. Previously an AHM system needed to be installed and data captured to a file for parsing later to identify the signals. In an effort to speed up this process, a tool was designed in MATLAB to parse the files. The tool provided an Excel workbook with multiple spreadsheets in it. The first spreadsheet was a list of signals being reported by the platform. The sheets that followed provide a graph of each signal that was captured allowing the engineer to determine if the reported signal is actually a valid signal. This tool allows for quick identification of signals for developing new applications.

Maintainer Tools

Once signal data has been collected on a platform, a user must be able to access and utilize the data. The Maintainer's Graphical User Interface (MGUI) developed by CIMS has been the typical tool used to access this data. During this program, a few improvements were made to the interface to make data analysis easier. First, a new charting library was utilized to provide more features and avoid licensing costs. Dynamic downsampling of datasets was employed to improve performance when visualizing large datasets. Users can add annotations and titles to graphs, which can subsequently be saved as images or printed. Selection of missions was modified to allow selection of multiple missions, filtering of missions by the data in the mission, and the ability to move forward and backward through the missions while viewing existing graphs. When viewing scatter (X-Y) graphs, the data may be filtered by time.

Previously, MGUI allowed the user to select data to be viewed based on what data exists in the database at the time the mission selection screen is populated. Thus, if a mission is ongoing, only data up to that point is available for graphing and as data is added to the database, it will not be available to MGUI for graphing. The interface for data access was modified to allow MGUI to continually add data to graphs on the screen as data is added to a mission. Thus a maintainer can pull up a graph of critical signals and view them in real-time as the vehicle is being driven. New data is simply added to the graph and the display is updated. This is useful both as a maintenance aid and as an algorithm development aid.

Advanced Anomaly Detection and Diagnostic Framework Technologies

As datasets are becoming available from AHM type systems, the next logical step is to identify what can be learned from the data. With a health monitoring system, these efforts generally revolve around development of diagnostics and prognostics for improved platform performance and reduced downtime. CIMS performed multiple projects aimed at better utilization of available data. In one instance, data was evaluated around an oil pressure DTC reported by one of the vehicle ECUs. Utilizing a regression based model to predict oil pressure, a threshold was applied to the differential between the predicted pressure and actual pressure. Through testing it was determined that the regression model could predict the oil pressure anomaly prior to the DTC occurring. This early detection could prevent costly breakdowns while the vehicle is in service.

Another critical issue with large signal sets across multiple platform types is the ability to identify the functional relationships between the signals. Utilizing a subset of signals and data to reduce overall complexity, CIMS evaluated the ability of the gridded residual regressor model to identify relationships between the signals. Utilizing this technique, CIMS was able to confirm that oil temperature and engine speed are the best signals for predicting oil pressure. It was determined that this method could be utilized for identifying relationships between signals when the correlated signals are limited in number, i.e., 2 or 3, but a boosted regression tree might be preferable if more dimensionality is needed.

In order to perform data analysis, the data must be able to be trusted as being valid. In efforts to ensure “clean”, valid data, CIMS implemented an expanded set of sensor fault detection methods. Specifically, multi-sensor fault detection (MSFD) was necessary to identify signals that might appear to be valid on their own, but when compared to other signals appear problematic. CIMS implemented two MSFD methods, cross-correlation and Dempster-Shafer. The cross-correlation method was utilized on data from the LAV to identify temperature sensor faults in the planetary wheel hubs that may not have been identified by a simple limit-based algorithm. The Dempster-Shafer method was utilized to identify a noisy battery shunt current measurement on the LAV.

Although real-time diagnostics and prognostics of mechanical systems have either been done or have proven feasible, applying prognostics to electrical systems has not been well established. CIMS chose to perform a feasibility study on a switched-mode power supply on a HALT test fixture to determine if signals could be correlated and predict failure. The study was successful and the real-time correlation predicted failure approximately 30 minutes prior to actual failure. As this was on an accelerated test, detection of the failure under normal circumstances may have provided a significant advanced warning.

AHM utilizes a relational data table structure with asynchronous data storage. This is done to reduce the quantity of data being written into the data store. However, for data analysis purposes on the backend, the data typically needs to be resampled to a synchronous data set. CIMS tested the performance of a typical data warehousing system. The first step was to translate the data into the data warehouse, followed by a series of queries to verify performance. Once the data has been translated, there is a significant performance improvement in accessing data because the synchronization step has already been done. This improvement can be significant when analyzing large data sets.

One of the recent areas of study for the AHM system surrounds improving driver behavior to reduce fuel consumption. Early models for improved behavior simply took into account engine speed and vehicle speed. However, CIMS was interested in modeling the mechanics of the system to be able to identify whether outside factors are affecting the fuel efficiency, i.e., hills or wind. CIMS struggled to model the system due to the unpredictability of system variables, such as vehicle mass, etc. However, it was determined that modeling the fuel consumption allowed identification of engine manufacturers and designs which could allow for further enhancements of the algorithms.

Another application of AHM technology at CIMS was for monitoring of a Solid Oxide Fuel Cell. An analysis was performed to improve the prognostics surrounding reactant ignition in the fuel cell. The baseline was performed using sequential Monte Carlo simulation and vector quantization techniques. Varying techniques were then applied to attempt to improve the prediction of failure without increasing false positives. It was determined that Boosted

Regression Trees, Gaussian Process Regression, Minimax Classifiers, and Additive Groves could all be utilized with near-perfect prediction results.

6.3. Remote Monitoring & Advanced Support Concepts Projects

The following projects conducted during this contract period under the RM&ASC include:

Projects

- Applied Remote Monitoring and Support Technologies
 - Implementation of AHM for a commercially available engine, representative of a UAV engine
 - RCM Analysis of the LAV AT Turret
 - LAV Engine Performance Data Collection
 - EPLS Alert Analysis
 - Grizzly Vehicle AHM Development
- AHM 3rd Generation Hardware Improvements
 - Vibration Data Acquisition Controller
 - Waveform Data Bus Protocol
 - Feasibility and Prototype of a USB-based Interface for Vehicle Communications (Compact Bus Interface)
 - Feasibility of Running AHM on Low-cost ARM Based Hardware Running Linux
 - OBD-II implementation
- AHM 3rd Generation Software Improvements
 - Flat File Database System for Real-time Health Monitoring Applications
 - Flat File Database Receiver and Platform Data Synchronizer
 - Multi-Source Signal Support in Asset Health Management (AHM)
 - DAC Graph Tool
 - DAC Node Programming Tool
 - DTC Data Model J1939 and J1587 Listener Improvements
 - Bus Data Analysis and Visualization
- Maintainer Tools
 - Improved User Interface for the Maintainer's Graphical User Interface (MGUI) Tool
 - Live Mission Data Viewing of Asset Health Management (AHM) Data in the Maintainer's Graphical User Interface (MGUI) Tool
- Advanced Anomaly Detection and Diagnostic Framework Technologies
 - Data-Driven, Supervised Anomaly Detection
 - Functional Relationship among Signals for Anomaly Detection and Virtual Sensing
 - Multi-sensor Fault Detection

- Switched-Mode Power Supply Data Analysis
- Vehicle Data Warehousing and Analytics
- Fuel Usage Analysis
- Solid Oxide Fuel Cell Diagnostics

**Remote Monitoring &
Advanced Support Concepts**

PROJECTS

Remote Monitoring & Advanced Support Concepts

Implementation of AHM for a Commercially Available Engine, Representative of a UAV Engine

Problem

The Unmanned Aerial Vehicle (UAV) is one of many platform assets used by the U.S. Armed Forces. Reliability of these assets is a top priority, leading to the application of Condition-Based Maintenance (CBM) methodologies to ensure the asset will perform as required to complete a mission. Remotely operated systems such as the UAV require the ability to monitor system functionality to detect potential failures in a timely manner.

Goals

The objective of this effort is to develop a next-generation Asset Health Management (AHM) system, which includes enhancements to hardware and on-board and off-board software, using a commercially available engine that is representative of a UAV engine as a demonstration platform. Figure 1 shows the representative UAV engine test stand setup.



Figure 1. UAV Engine Test Stand.

The AHM application will be developed from the work done in several key areas:

1. Reliability Centered Maintenance (RCM) evaluation.
 - a. Identify UAV functions and functional failures to determine which signals to monitor in order to implement failure detection and CBM.
2. Identify Sensors and signals.

- a. Select sensors and hardware for monitoring the UAV to predict failure modes for specified functional areas as a result of the RCM.
3. Develop CBM methodologies.
 - a. Identify a set of actions a reasonably trained maintainer can accomplish on a system to eliminate a functional failure or improve the system performance.
4. Data Acquisition (DAQ) development.
 - a. Develop deployable data acquisition hardware capable of monitoring identified signals, including vibration. This DAQ hardware will be designed for use in demanding military environments.

Approach

An RCM analysis was performed to identify the functions and functional failures of a UAV system. The end result would be to categorize failures by severity and frequency of occurrence, and identify maintenance tasks.

One example for a specific symptom of low cylinder pressure can be caused by several different faults:

- Fault examples: Wear or cracks in cylinder or rings, carburetion faults, valve malfunctions, weak spark, mistimed spark.
- Fault frequency: Low.
- Fault severity: Moderate to high.
- Prognostics horizon: can be long.
- Sensors availability: Sensor in UAV test stand; unknown expense for production engines; sensor itself could lead to seal leakage faults.
- Likely features: Limits, cylinder-cylinder differential, double peaks (detonation), rise and fall times indicating intake or exhaust obstruction.
- Fault injection: Carburetion, electrical.

The next step was to analyze the maintenance procedures from a functional breakdown perspective. This examines each subsystem and lists the failure effects and performance issues that a maintainer can detect with minimal added equipment. Figure 2 illustrates the functional systems of the UAV and sensor signals identified. The functional analysis was used to determine which additional sensors and equipment, if any, are needed to allow the maintainer to segregate between a system failure (in this case the engine) and external element failures.

A diagnostic engine will be incorporated into AHM as a result of the functional analysis. The idea is that, given vehicle condition information primarily in the form of alerts, operating context, and sensor data, the diagnostic engine will process a series of "rules" to determine a probable cause of an indicated failure.

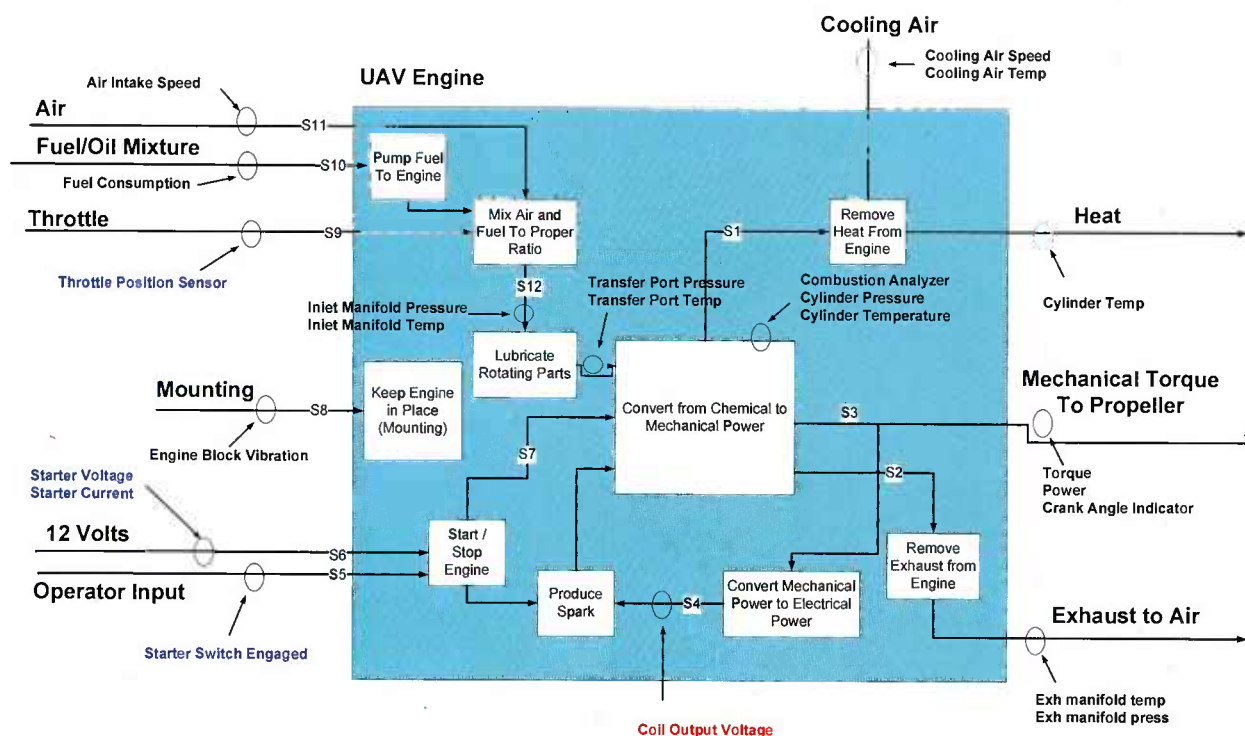


Figure 2. UAV Functional Diagram.

Results

A more robust AHM system was developed and applied to the representative UAV engine, which monitored functional systems as identified through the RCM analysis. The analysis identified 11 high-priority and 5 low-priority signals to be included in the AHM system. One failure, in particular, was identified and investigated further. The glow plugs used to pre-heat the combustion chamber of the engine were seen to fail on occasion. To better understand this failure and develop a predictive model, testing was performed to accelerate glow plugs to failure. The data from this testing was used to develop preliminary rules to be applied in AHM for prediction of glow plug failure. The methodologies used to develop these preliminary rules can be applied other systems in a similar manner.

RCM Analysis of the LAV AT Turret

Problem

The U.S. Marine Corps and the Program Manager-Light Armored Vehicle (PMLAV) were evaluating upgrading the LAV Anti-Tank (AT) legacy turret with a new design. Prior to finalizing a design, PMLAV wanted to complete a full RCM analysis of the platform in order to reduce maintenance downtime and cost, and to increase readiness and availability.

Goals

The objective of this project was to improve reliability of the LAV AT A2 turret platform while reducing the overall lifecycle support costs without jeopardizing mission capability, crew safety

or environmental quality. The desired outcome is to provide appropriate and cost-effective maintenance policies that maintain system functionality while reducing safety hazards and environmental effects of failures. A key goal is to develop maintenance technology requirements where Condition Based Maintenance (CBM) technology could be implemented in conjunction with USMC Autonomic Logistics (AL) initiatives.

Approach

The Reliability Centered Maintenance (RCM) process, following the SAE JA1011 "*Evaluation Criteria for RCM Process*" and SAE JA1012 "*A Guide to the RCM Standard*" was used in the analysis of a new U.S. Marine Corps' LAV AT A2 turret design. The RCM process provided an in-depth assessment of the functions, functional failures, and failure modes of the new LAV AT A2 turret design.

The assessment of the turret occurred during a 12-day training and analysis session in San Diego, CA. The team consisted of facilitators from RIT, a depot field service representative, PMLAV quality assurance, Marine Corps operators, maintainers, and logisticians for the LAV AT A2.

The RCM analysis began with an assessment of the system's functions and functional failures, followed by the identification of failure modes and consequences, and finally, when possible, the creation of detailed maintenance tasks to address the identified failures. This particular analysis session gave the Marines and PMLAV Turret Design team a chance to communicate their thoughts and ideas through a structured approach, where the details of the analysis were captured in a RCM database developed at CIMS, shown in Figure 3.

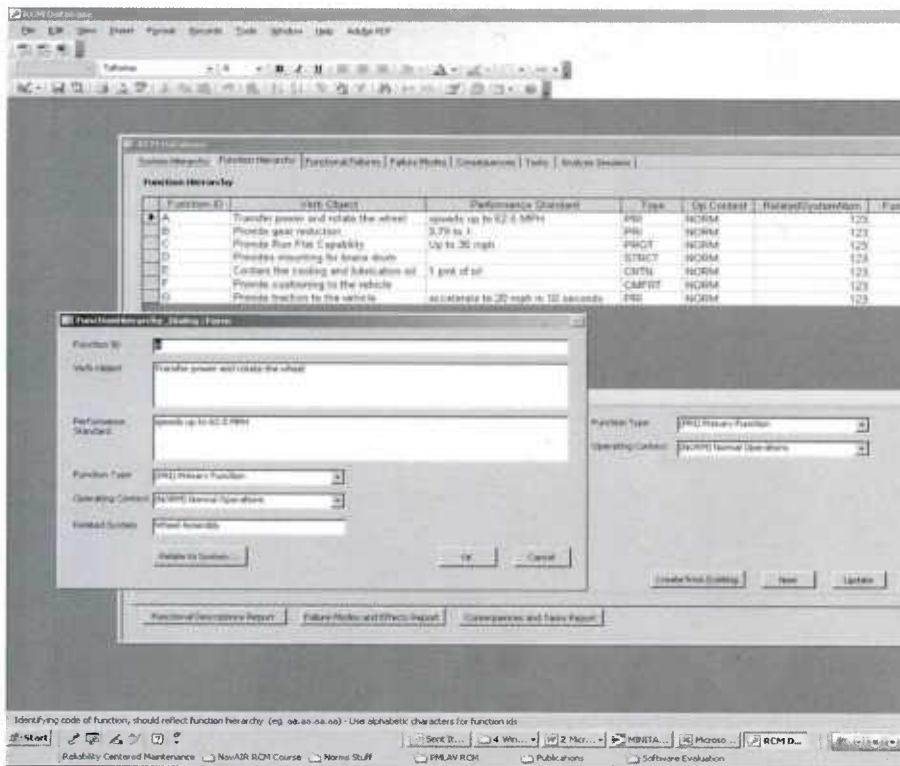


Figure 3. RCM Database

As a result of the dynamic nature of the design, there was limited insight into the proposed system; for this reason, the Stryker vehicle TOW turret platform was used as a reference design. Also, because current maintenance procedures or manuals for the proposed equipment do not exist, this RCM analysis should be considered a baseline for this system.

Results

Following the analysis, RCM coordinators used the collected data to create a maintenance policy that attempts to eliminate or reduce the consequences of functional failures on the operation of the asset. This policy may then be incorporated into procedures used to maintain the asset or system. Based on the evaluation the following maintenance policies were identified for the LAV AT A2 Turret:

- Scheduled Tasks
 - Zonal inspections and walk-around checks
 - Operator maintenance and preventative maintenance
 - Operational maintenance
- Default Tasks
 - Recommended design changes
 - Failure finding

Following a review of operator maintenance procedures and instructions for the current LAV AT Turret, the team prepared a synopsis of critical new maintenance tasks that would be needed for the LAV AT A2 Turret. The RCM analysis of the LAV AT A2 turret resulted in the identification of the following functions and failures that the new maintenance tasks would address:

- Functions
 - 2 Primary
 - 9 Secondary
- 238 Failure modes
 - 2 Severe safety consequence
 - 2 Hidden

Tasks were created based on the knowledge of the PMLAV team of what components may be in the new LAV AT A2 Turret, together with the maintainers' and operators' knowledge of the current system. This research is presented in Table 1.

The normal outcome from a typical RCM analysis is a reduction in the number of maintenance tasks, based on the analysis, from the currently implemented maintenance program. Current maintenance requirements for the LAV AT A2 Turret are not yet clearly defined because of the dynamic nature of the new turret design.

Table 1. Recommended Maintainer Tasks to be Included in TM

W – Weekly M – Monthly S – Semiannual A – Annual B – Biennial

Item No.	Interval						Item to be Inspected Procedures	RCM Task #
	W	M	S	A	B	Hours		
New			◎				RCM: Traverse Drive and Mast Drive Timing Belt: Further study recommended to determine wear of belt vs. time.	215, 289, 323, 385, 462, 523, 596
New			◎				RCM: Check electrical ground connections for corrosion. Clean and add dielectric grease if necessary.	564

Item No.	Interval						Item to be Inspected Procedures	RCM Task #
	W	M	S	A	B	Hours		
New				⊙			RCM: Inspect resilient mounts and replace if needed.	627, 635
New			⊙				RCM: Inspect slip ring for corrosion and water damage.	212, 267, 282, 361, 364, 469, 520
New			⊙				RCM: Check and adjust bore sights.	207
New			⊙				RCM: Inspect turret ring bearing for corrosion; Lube. (current design not known)	213, 271, 280, 362, 390, 470
New			⊙				RCM: Grease and lube turret drive gear components if necessary.	591
New				⊙			RCM: Inspect elevation drive and motor.	222, 247, 284, 328, 393, 425, 455

LAV Engine Performance Data Collection

Problem

A large source of data is necessary to support prognostic and diagnostic development. Although the Embedded Platform Logistics System (EPLS) program was planning to install an Asset Health Monitoring (AHM) System on approximately 700 Light Armored Vehicles (LAV), there would be significant benefit to collecting data in a controlled environment. This data could also be utilized to develop algorithms for accurately calculating unavailable parameters from available signals, as well as determining the effects of AHM data processing on the ability to utilize the data for data mining.

Goals

Under another program, RIT was contracted to collect LAV engine data on an engine dynamometer utilizing a National Instruments (NI) data acquisition system. The goal of this program was to develop an AHM system that would process the collected data into the AHM format. Additionally, RIT was performing a data collection and analysis program for evaluating

biofuels for the Department of Transportation. This program provided another source of data for a Cummins diesel engine. The data collected could then be used for development of new algorithms for diagnostics and prognostics, revision of existing algorithms, and to gain a better understanding of the effects of the AHM approach on long-term data usability. In addition, since the engine is running in a controlled environment, algorithms could be tested for repeatability and accuracy.

Approach

The AHM data collection system that was implemented on the engines on the dynamometer was based on the most recent ONR-developed AHM code, with many of the features from the EPLS system included. The system utilized the same data variances for downsampling and alert limits utilized in the EPLS system as a starting point. For the LAV engine, data was collected with the NI system and then run through AHM as a post-process.

The EPLS system was the starting point for the LAV engine sensor set for the other program; however, different sensors were used and additional signals were collected. This would allow for an expanded data set and a comparison of sensor and data accuracies vs. the data collected in the field. For the Cummins engine, data was collected through the existing J1939 data bus for the engine.

Results

The data collection system was utilized to collect approximately 14 hours of LAV engine data and approximately 570 hours of Cummins engine data from the dynamometer. This data was collected and processed into the AHM data format in a database on the server. The data is available for data-mining purposes. Additionally, the dynamometer was used significantly for testing new algorithms. For example, utilizing data on the engine dynamometer allowed for development of an algorithm that could predict the fuel usage from engine torque, engine speed, intake manifold pressure and engine load. The predicted value could then be compared to the measured fuel usage, allowing AHM to identify when the vehicle was using significantly more fuel than necessary.

Similarly, an algorithm was developed that could predict the exhaust gas temperature from engine speed and torque under normal circumstances. Comparing the predicted temperature and the actual exhaust gas temperature would identify potential engine issues. Development of these algorithms takes a significant amount of time and data. Many variations of the algorithms were developed and run under varying engine speed and load profiles to determine the best algorithms for final implementation. The resulting algorithms are applicable to the MTRV because they actually collect engine speed, load, and torque, as well as intake manifold pressure from the vehicle data bus.

EPLS Alert Analysis

Problem

The Embedded Platform Logistics (EPLS) system had been deployed on a small number of Light Armored Vehicles (LAVs) for pilot testing of system functionality. The reliability of the EPLS system and the ability to differentiate between an actual system failure and false alarm is dependent on the accuracy of the sensors and EPLS software. Data collected on 16 LAVs over several months required an in-depth analysis to validate EPLS system performance and its effectiveness in identifying failures.

Goals

One objective of the analysis was to classify alerts as: actual system faults, sensor faults, EPLS system faults, or other anomalies in order to assess the overall system effectiveness and look for opportunities to improve algorithms. Another objective was to identify the anatomy of a specific LAV failure and create a knowledge base for future learning and application to improve EPLS functionality and support condition-based maintenance policies. The main goal of this project was to develop better diagnostics capabilities to enhance the EPLS system and relate a set of symptoms to a diagnosis of a root cause for a failure.

Approach

The preliminary analysis of the EPLS system on the LAV evaluated active alerts based on the criteria listed below. The results were used to determine the importance and whether further investigation into the alert was necessary.

- Frequency and duration of occurrence.
- Number of vehicles affected.
- Severity of alert and potential for system/vehicle failure.
- Alerts for components and systems that are known to have had failures.

The alerts analyzed in more detail were selected based on frequency of occurrence, number of vehicles affected or the severity of the alert. Once an alert was chosen, the first pass was to select relevant missions with the alert and look for mission-to-mission similarities for a single vehicle and commonalities across platforms. Good missions with no active alert present, with similar operational context, were also selected as a baseline “good vehicle” for comparison. Operational context can be defined as operating and environmental conditions present.

The general rule for selecting relevant mission data to review is as follows:

1. Identify an alert of interest and filter to select missions where the alert is active.
2. Determine active modes during the alert, such as *engine on* or *drive*, to determine the operational context for comparison and filter missions especially for alerts that had multiple modes associated.

3. Randomly select five missions with alerts, from beginning, middle, and end of the data range for comparison.
4. Randomly select vehicle missions with no active alert while in same modes determined when the alert was active, as baseline for good vehicle for comparison.
5. Create a set of rules and definitions, describing conditions leading to the alert, which can be used to define alert context, which can be applied to future instances of the alert.

The defined conditions leading to an alert can be used in an algorithm that can be applied across multiple vehicles and missions, reducing the amount of time spent sorting through mission data manually.

Results

Eight vehicles and ten alerts were analyzed in depth for EPLS data available shown in Table 2. In 3 of the 10 alerts, it was found the multiple vehicles with the same alert resulted in the root cause being business-rule related. The available data supported the fact that changes were made to the business rules to correct the alert occurrences. The other alerts did not have enough supporting data to draw a strong conclusion as to the root cause. It is clear that from the data analyzed that the EPLS system was still being refined to more accurately report actual alerts. Better refinement of the system will allow for more efficient data analysis and the ability to discern between alert and sensor fault.

Results of this comprehensive analysis can be used to optimize future data analysis efforts on EPLS or other vehicle data. Results discussed here can be used to improve EPLS system functionality on the LAV and to maximize its performance in determining actual alerts on the vehicle or sensor/system faults. Several recommendations can be made as a result of this effort:

1. Assessment of LAV system functionality to support more accurate alert limit and mode definitions in the business rules.
2. Improved organization and communication between field and back office to report and track changes made in the field.
 - a. Maintenance data would be useful and allow for data comparisons before and after a maintenance event.
3. Larger data sets with more vehicles to support analysis conclusions.

Tools were developed to filter mission data more efficiently. These tools can serve as a building block for future analysis tool development.

For each alert analyzed, a set of conditions to identify specific features were created to look for similar instances across different missions and vehicles. It is important to note that in the absence of an active alert, the ability to look for an alert based on defined operating conditions is necessary. This can be useful when these instances are periodic and not active long enough at any given point to activate an alert.

Table 2. Alert Analysis Results.

Alert ID	Description	Root Cause	Associated Vehicles
1762	Hydraulic Pressure	Alert Limits definition	110, 138, 157, 164
167, + Multiple BIT Alerts (electrical)	Electrical system anomaly; Multiple alerts and sensor faults active as a result of this anomaly	EPLS System Ground Fault	160
14008	Secondary FF pressure drop	Alert/mode definition	110, 137, 155, 157
1127	Turbo Pressure	Bad STE-ICE sensor	110
14942	Primary FF pressure drop	Alert Limits /mode definition	157
14967	Primary Fuel Filter Input Pressure	Filter Restriction	137, 157, 159, 164
14006	Alternator Field Voltage	Unknown	110, 138, 157, 159,
190	Engine Over Speed	Governor, Operator error	110, 135, 157, 164
190	Engine Under Speed	EPLS System	138, 157, 164
14019	Engine Coolant Level	EPLS System	110, 137, 138, 157, 159, 164
100	Engine Oil Pressure	Unknown	110, 137, 138

Grizzly Vehicle AHM Development

Problem

As algorithms are developed for the Asset Health Monitoring (AHM) system, most testing is performed through data playback of previously recorded data. There are limitations to this as the data being run through the algorithms in AHM during playback have already been subjected to downsampling. However, when AHM is run on a vehicle, the algorithms are run on the raw data being received by AHM (i.e., without downsampling). This can have a large impact on the performance of algorithms, especially those designed to run based on a number of samples instead of those running on a specific schedule. A demonstration platform with an AHM system would be beneficial to analysis of algorithms against raw data, especially allowing the platform to be driven under specified conditions to obtain a specific result.

Goals

The goal of this program is develop an AHM-equipped platform that can serve as a real-world platform for testing, validating and demonstrating new algorithms in AHM.

Approach



The Grizzly Armored Personnel Carrier is part of the Canadian Armored Vehicle General Purpose (AVGP) family of vehicles; see Figure 4. The Grizzly is a six-wheeled vehicle utilizing many of the same components as the U.S. Marine Corps' Light Armored Vehicle (LAV). RIT was in possession of two Grizzly vehicles previously provided by the Marine Corps on another grant project. An AHM system was developed for the Grizzly that could collect data from a slightly expanded sensor set than was used on the LAV during the Embedded Platform Logistics System (EPLS) program. This platform will allow further development of algorithms for improvement of a monitoring system for the LAV.

Results

Figure 4. Grizzly in CIMS Research Bay.

The Grizzly was equipped with a System Health Node, Power supply, Data Acquisition Nodes, a wireless planetary temperature measurement system, and various sensors. The System Health Node main board was upgraded to an Intel Atom processor-based system, which provided additional computing power, more memory, and expanded storage abilities. The AHM software utilized the most up-to-date algorithms for sensor fault detection, battery prognostics, oil life, and multi-modal anomaly detection. This also was the first mobile test platform to utilize the Flat File storage mechanism instead of MySQL.

Vibration Data Acquisition Controller

Problem

AHM systems need to acquire vibration data as part of asset monitoring. Cost-effective commercial systems that support rapid development and deployment for research purposes are not available. CIMS initiated a project to create a low-cost data-acquisition platform capable of monitoring vibration in various types of equipment, and to support algorithm development using MATLAB or Simulink.

Goals

The goal of this project was to develop a deployable data-acquisition node (DAC) capable of monitoring vibration of various types of equipment. The node would be designed for use in demanding military environments. The node would have the ability to use advanced tool sets for algorithm development, which could be downloaded via the communications bus. The data gathered by this node would be communicated over the same bus. From the data collected, condition-based monitoring could be developed. Vibration data could be used to detect many failure modes including bearing fatigue, cracked or broken gears, loose armature laminations, etc. If the potential for failure is detected soon enough, proper maintenance could be performed before serious failures occur. This node can be used to detect gear defects such as propagating cracks in real-time applications.

A second goal was to simplify the process of creating algorithms that are run by the hardware. CIMS researched the feasibility of using code generated by MATLAB for embedded vibration

analysis, instead of custom-written C code. One of the advantages of this concept is the ability to develop a signal-processing algorithm in MATLAB using standard MATLAB algorithms. Only after thorough testing and debugging in MATLAB would the algorithm get deployed to the DAC node.

An additional goal was to use the fixed-point and floating-point simulators available in Texas Instruments Code Composer Studio to evaluate the performance of MATLAB-generated algorithms. These two emulators were chosen because they are representative of the fixed-point and floating-point processors currently available from Texas Instruments.

Approach

A prototype vibration DAC was developed meeting the following general requirements.

- Monitor vibration from 3 accelerometers simultaneously on analog channels 1 – 3.
- Support a generic input on analog channel 4.
- Support a digital tachometer input.
- Support transmission of waveforms and other data over a J1939/CAN bus or over Ethernet.
- Support software-configurable sample rate for analog channels.
- Provide a hardware filter for each channel with software-configurable gain and cutoff frequency.
- Support configurable calibration values (gain and offset) for each analog channel.
- Support algorithm development using MATLAB.

The key hardware component is the embedded processor. For this application we wanted a processor capable of running algorithms generated by MATLAB. Our goal was to reduce the complexity of algorithm development to the point that algorithms can be developed, targeted, and tested by a subject matter expert, rather than a specialist in embedded development.

A key decision was choosing either a floating-point or fixed-point digital signal processor. Floating-point processors have much greater dynamic range, but at the expense of lower overall processing throughput, higher component cost, and greater power consumption. In many cases fixed-point processors can do everything a floating-point processor does, but they require that all signals be scaled appropriately to make maximum use of the more limited dynamic range of fixed-point calculations.

The processor we selected was the Texas Instruments TMS320DM6437. This is a fixed-point digital signal processor that was chosen in part because of the built-in CAN interface, but also because it was claimed to be supported by the MATLAB code generation tools.

An additional technology decision was the analog signal processing chain up to and including the analog-to-digital converter. In previous projects we used high-resolution, multi-channel A/D converters. These converters use a multiplexer to select one of up to 16 channels, routed to a

single A/D converter. There are two negative side-effects to this design. First, the multiplexer arrangement means that samples of multiple channels cannot be taken simultaneously. Second, the multiplexing arrangement means that the effective sampling rate of the A/D converter is the speed of the converter divided by the number of channels. For example, a 10 k sample per second A/D converter with 8 channels has an effective sampling rate of 1.25 kHz. For these reasons, we chose to use individual A/D converters for each channel. While this raises the cost, it means that all channels can be triggered to sample simultaneously, retaining the full sampling rate of the analog to digital converter.

Because the sampling rate is software configurable, we also needed to have a user-programmable anti-aliasing low-pass filter on each channel. This was accomplished by using a switched-capacitor analog filter with digital control. I/O pins from the processor were used to set the filter cutoff frequency as appropriate for the user-configured sampling rate.

See Figure 5 for an overview of the Vibration DAC architecture.

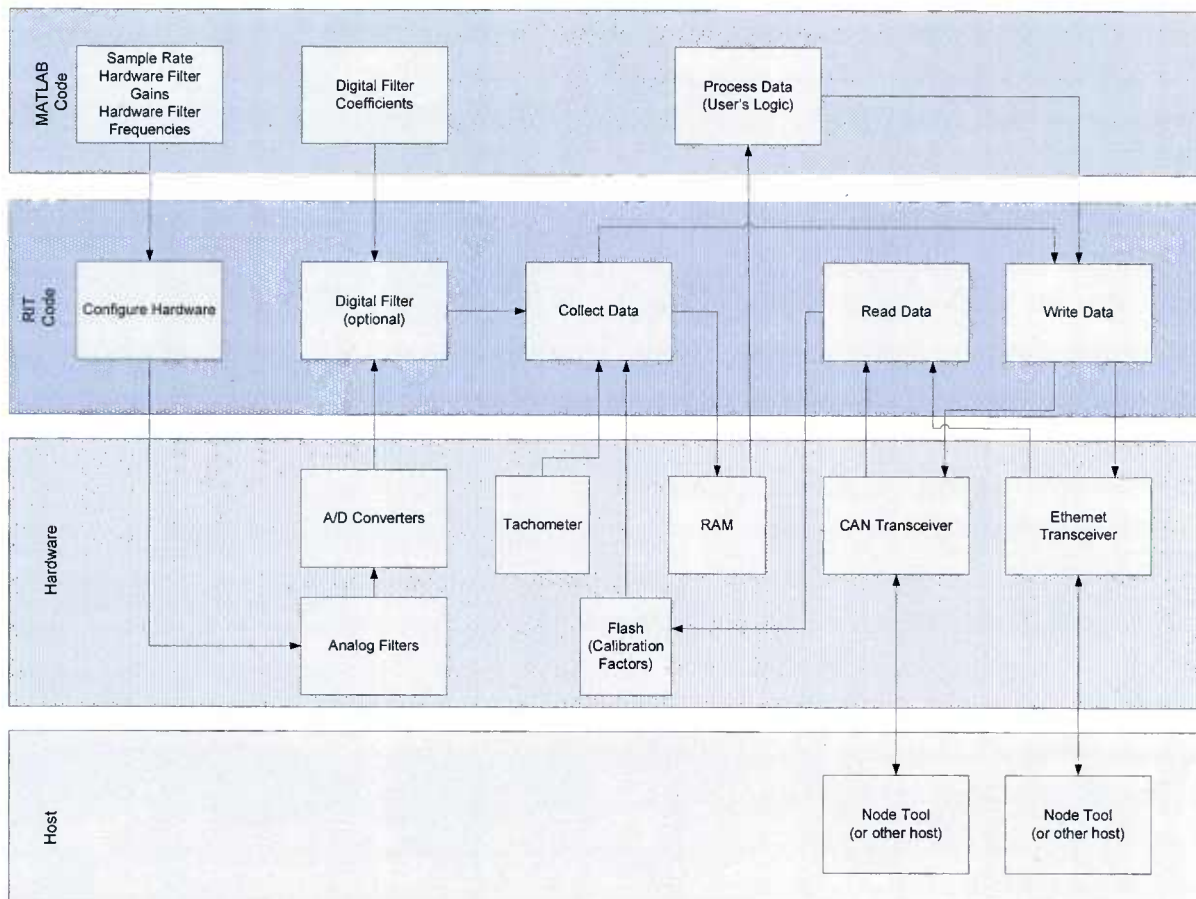


Figure 5. Vibration DAC Architecture Overview

Given the requirements for the DAC, it was decided to partition the software into several tasks executing in parallel. See Figure 2 for an overview of these tasks.

- MATLAB Task – Runs the user’s MATLAB-generated algorithm at a rate specified by the user. The system was designed to support up to four MATLAB tasks.
- Ethernet ISR Task – Dedicated to reading and writing data from/to hardware registers. Communicates data packets with Ethernet task through transmit and receive queues.
- Ethernet Task – Processes data generated by the MATLAB tasks. Communicates data packets to/from Ethernet ISR task through transmit and receive queues.
- CAN ISR Task – Dedicated to reading and writing data from/to CAN hardware registers. Communicates data packets with J1939 task through transmit and receive queues.
- J1939 Task – Processes data generated by the MATLAB tasks. Communicates data packets to/from CAN ISR task through transmit and receive queues. This task also accepts analog channel calibration values over the J1939/CAN bus and writes them to flash memory.
- ADC Task – Dedicated to triggering the A/D converters at a rate specified by the user. Reads the converted data and places in channel buffers, where it can be accessed by MATLAB tasks.
- Debug Task – Sends raw analog and digital input data over the J1939/CAN bus to an external RIT tool that calculates channel calibration factors. This task will run only when enabled by the tool.
- Idle Task – This task runs only when all other tasks are idle. It will initially do nothing, but is available for future use.

The Texas Instruments SYS/BIOS real-time kernel was used to create and manage all of these tasks.

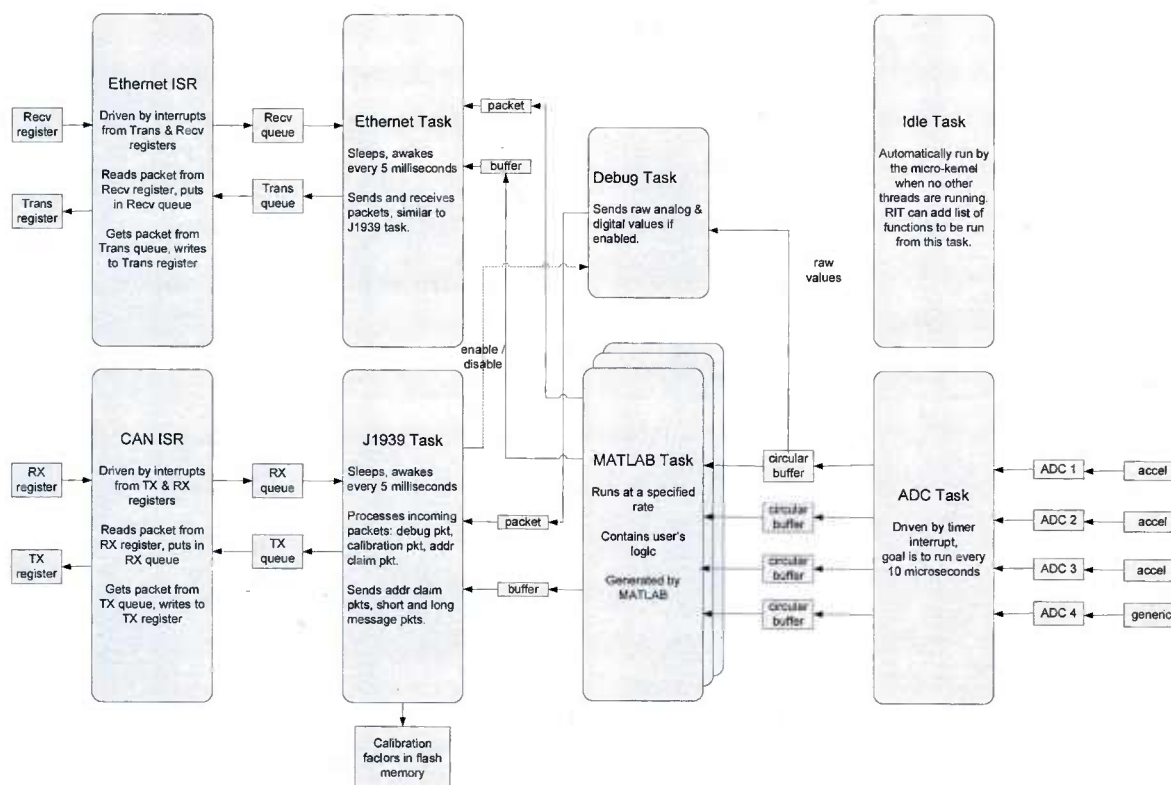


Figure 6. Vibration DAC Software Architecture

Writing algorithms in MATLAB and generating embeddable code was researched and a detailed workflow concept was developed. This workflow concept was used to generate all MATLAB algorithms for this project. All algorithms were developed and tested with MATLAB R2011a.

The following is an overview of the generation of MATLAB code.

1. Write and test the desired algorithm in MATLAB.
2. Create a MATLAB function containing your algorithm. This function will call DAC library functions to acquire channel data and send calculated results.
3. Write a single initialization function in MATLAB. This function will configure the sample rate, gain, and cutoff frequency for each analog channel.
4. In MATLAB, generate C code for each of your functions.
5. Compile and link the generated code in Code Composer Studio.
6. Download your compiled code to a vibration DAC and test.

Tests were conducted on a sample of MATLAB generated algorithms, comparing the processor cycles used by a fixed-point processor and a floating-point processor. The processors chosen for these tests were a Texas Instruments TMS320DM6437 fixed-point processor and a Texas Instruments TMS320C6748 floating-point processor. These processors were chosen because they are typically used in embedded projects and they are currently in production.

Identical algorithms were used in both sets of tests. All algorithms were generated using MATLAB R2011a and were compiled with Texas Instruments Code Composer Studio 4.2.

Results

The prototype vibration DAC was evaluated in our lab using various algorithms generated using MATLAB R2011a. The generated C code was compiled with Texas Instruments Code Composer Studio 4.2.

- Timing tests showed that the fixed-point processor chosen for the vibration DAC could not run the MATLAB-generated floating-point code efficiently.
- A single processor had to do all of the work (running MATLAB code, data acquisition, CAN communication to/from the host, and micro-kernel overhead).
- We were unable to achieve the desired 100 kHz sample rate. The best we could achieve was 20 kHz. Sampling any faster caused the CAN and task threads to “starve.” This was true even after increasing the external memory interface to its maximum speed and turning on processor data caching.
- We were unable to truly achieve simultaneous sampling on all channels. In practice, there are several microseconds between each trigger of the A/D converters.
- The vibration DAC often could not “keep up.” For example, when calculating a 512-point FFT once every 2 seconds and sending the results (257 half-precision data points) over the data bus, only about half of the data packets were actually sent.



Figure 7. Vibration DAC Printed Circuit Board

An alternative processor that might be a better match to the project goals is a Texas Instruments OMAP-L138 dual-core processor, with the following features:

- Floating-point digital signal processor (DSP)
- ARM9 processor
- Xilinx Spartan-6 FPGA

Software modifications which would be recommended include:

- Use the DSP to run only the signal processing (MATLAB) code.
- Create four SPI ports with the FPGA that run in parallel and perform the entire data acquisition loop in hardware, presenting the DSP with all the results at once and perfectly synchronized.
- Use the ARM9 for all CAN and Ethernet communications.

This modified design approach would subdivide the tasks currently executed by a single fixed-point processor and allow these tasks to be executed in parallel.

Developing “embeddable” algorithms in MATLAB is not straightforward and requires more than a little knowledge of C programming and the ability to integrate the code generated by MATLAB with the vibration DAC source code. Developing using MATLAB is possible, but still requires the engineer to have a thorough understanding of the embedded target, its architecture, and its limitations. In particular:

- The class (data type), size, and complexity (real or complex) of all data in MATLAB must be specified so that MATLAB can generate proper C code. MATLAB supports dynamic data typing and sizing, but C does not.
- In our experience generating MATLAB examples for this project, getting the MATLAB function “right” so that MATLAB can properly generate code can take a considerable amount of time. MATLAB generates many errors that must be debugged and corrected.
- MATLAB tends to allocate a lot of data on the stack (17,408 bytes in our FFT example). This can lead to runtime errors that are very difficult to debug. We cannot know in advance what algorithms a MATLAB engineer might write, so we cannot predict stack sizes in advance.
- MATLAB generates many C files (typically about 20) that must be copied by hand to the correct project subdirectory in Code Composer Studio, where they are compiled and debugged.

Fixed-Point Processor Tests

The following tests were run for the TMS320DM6437 fixed-point processor. This is the processor used on the vibration DAC.

For these tests, the C64x+ CPU Cycle Accurate Simulator (little endian) was used in the Texas Instruments Code Composer Studio 4.2 IDE.

Table 3. Fixed-Point Tests.

Test	Operation	CPU Cycles	Notes
1	1000 single-precision floating-point multiplications.	70,025	Generated from MATLAB.
2	1000 16-bit integer multiplications.	24,020	Generated from MATLAB.
3	1000 32-bit integer multiplications.	22,020	Generated from MATLAB.
4	Compute 512-point single precision floating-point FFT.	3,251,911	FFT generated from MATLAB.
5	Compute 512-point single precision floating-point FFT.	3,180,879	FFT generated from MATLAB. Same as test 4 above but without the

			magnitude calculation.
6	Compute 512-point fixed-point FFT.	301,819	FFT generated from Simulink.
7	Compute RMS of 512 single precision floating-point samples.	132,986	Generated from MATLAB.
8	Compute standard deviation of 512 single precision floating-point samples.	160,080	Generated from MATLAB.

Floating-Point Processor Tests

The following tests were run for the TMS320C6748 floating-point processor.

For these tests, the C674x CPU Cycle Accurate Simulator (little endian) was used in the Texas Instruments Code Composer Studio 4.2 IDE.

Table 4. Floating-Point Tests.

Test	Operation	CPU Cycles	Notes
1	1000 single-precision floating-point multiplications.	21,021	Generated from MATLAB.
2	1000 16-bit integer multiplications.	24,020	Generated from MATLAB.
3	1000 32-bit integer multiplications.	22,020	Generated from MATLAB.
4	Compute 512-point single precision floating-point FFT.	831,345	FFT generated from MATLAB.
5	Compute 512-point single precision floating-point FFT.	785,286	FFT generated from MATLAB. Same as test 4 above but without the magnitude calculation.
6	Compute 512-point fixed-point FFT.	301,819	FFT generated from Simulink.
7	Compute RMS of 512 single precision floating-point samples.	58079	Generated from MATLAB.
8	Compute standard deviation of 512 single precision floating-point samples.	66338	Generated from MATLAB.

The tests show that for floating-point operations, the floating-point processor uses approximately 60% to 70% fewer cycles than the fixed-point processor; fixed-point processors must emulate floating-point operations whereas floating-point processors can perform these operations in hardware. For fixed-point operations, both processors use the same number of cycles.

A floating-point processor is a better choice for the Vibration DAC. MATLAB and Simulink generate floating-point code. The choice of a floating-point processor would allow the MATLAB engineer to simply generate code without avoiding the use of floating-point and alleviate the engineer from using the difficult-to-use MATLAB Fixed-point Toolkit.

Waveform Data Bus Protocol

Problem

The development of a low-cost data acquisition device capable of monitoring vibration created the need to transfer these waveforms over a data bus to clients for further processing. For example, clients can use these waveforms for condition-based monitoring, anomaly detection, or failure mode analysis.

Goals

An objective of a related CIMS project was to develop a method for real-time vibration monitoring and data collection for unmanned aerial vehicle (UAV) engines. An Asset Health Monitoring system was implemented to calculate frequency data from the engine and record it in a database.

Given this objective, the goal of this project was to develop a protocol for transferring large waveforms over a data bus. This protocol would be used initially on the UAV engine monitoring project, but could be used in the future on any project where it was necessary to transmit waveforms over a data bus.

The protocol was designed for use with the J1939/CAN data bus. This data bus was chosen because it is supported by RIT's data acquisition nodes and it is commonly used in both commercial and military applications.

Approach

The metadata needed to describe the waveform data was considered, and it was determined that the metadata should include:

- Sampling rate in hertz.
- Number of points per waveform segment.
- Data type (e.g., 16-bit integer, 32-bit floating-point, 64-bit floating-point) – to be included only if the data type is variable.
- Scaling factor – to be included only if data is sent as an integer, to allow the consumer to scale the samples back to floating-point.

J1939 standard messages are limited to 8 data bytes; this is too limited for the transmission of waveforms, so J1939 long messages were used. Because J1939 long messages are limited to 1785 data bytes, the size of the data type chosen was important. IEEE 754 double-precision

floating-point (64 bits), single-precision floating-point (32 bits), and half-precision floating-point (16 bits) were considered.

Half-precision floating-point was chosen because it provides the best balance between data resolution and data size. There are 10 bits in the fraction and 5 bits in the exponent, with one sign bit. This provides, on average, approximately 3.3 significant decimal digits per sample while still retaining sufficient dynamic range for the vibration data being monitored in this experiment.

The decision to fix the data type at half-precision floating-point also eliminated the need to include the data type and scaling factor in the metadata.

The protocol was designed to support both continuous waveforms (streams of data) and non-continuous waveforms (periodic snapshots of data). Each of these collections starts with a header that identifies it as continuous or non-continuous. The header also contains a protocol version number, stream ID, sampling rate in hertz, a fragment number, the total number of fragments in this collection, and the first set of data samples.

For non-continuous collections, the header is the only fragment sent and contains the entire data snapshot. For continuous collections, the header is the first of many fragments. Subsequent fragments do not repeat all of the metadata included in the header; they contain only the stream ID, a fragment number (that increments with each fragment), and a set of data samples.

Results

The waveform protocol was implemented in RIT's second-generation four-channel DAC (a producer of waveform data) and in our Asset Health Monitoring software (a consumer of waveform data).

The DAC node used in the UAV engine monitoring project captured data on three analog channels from a 3-axis accelerometer at a sampling rate of 3.2 kHz per channel. A programmable timer and a digital encoder signal from a sensor mounted on the crankshaft were employed to trigger a data capture; when the timer fired, the capture occurred on the next rising edge of the encoder signal. The waveform protocol worked very well in transmitting the non-continuous waveforms (snapshots) captured in this experiment.

While the protocol was designed to support continuous waveforms, in some circumstances it may not be possible to send such waveforms over the bus. The limiting factors include:

- DAC node processor speed and workload (how fast can it capture and send data).
- DAC node memory (how much waiting-to-be-transmitted data can it store).
- Data transfer rate of the CAN bus (how fast can it transmit data).

The J1939/CAN bus supports a 250 kilobit per second data transfer rate. Discounting packet overhead (64 bits per frame), the maximum theoretical frame rate of a fully loaded bus with full data fragments (7 payload bytes) is approximately 1700 packets per second, or an aggregate

throughput of approximately 95 kHz. The maximum sampling rate possible depends on the size of each sample. Using half-precision floating-point, this corresponds to an aggregate sampling rate of approximately 6 kHz (for a single channel). This also assumes a 100% loaded bus; other messages (particularly if they are of higher priority) must be reserved, further reducing the sampling rate. This is part of the reason why triggered, non-continuous sampling strategies are so important.

Users of the waveform protocol must keep these limiting factors in mind. When unable to send continuous waveforms, we recommend that data acquisition nodes employ intelligent triggering to acquire and send data snapshots. We recommend the use of one or more of the following techniques:

- Triggering based on a timer.
- Triggering based on a digital input event.
- Triggering based on the receipt of a specific trigger message on the data bus.

Feasibility and Prototype of a USB based Interface for Vehicle Communications (Compact Bus Interface)

Problem

Vehicle monitoring systems for modern commercial vehicles such as the MTVR have multiple data bus interfaces, including J1708 and J1939. Current monitoring solutions (including those used for EPLS) include internal bus interface adapters. This has a negative impact on bus integrity because the System Health Node CPU cannot always be located close to the vehicle's diagnostic bus. It also has the disadvantage of being tied to a specific computer bus architecture (in this case, the legacy bus PC/104).

Goals

The main goal was to determine the performance and reliability of the proposed architecture, which involved development of a prototype dual CAN + J1708 + Power Interface board that communicates with a System Health Node CPU, followed by analysis of the feasibility of this architecture as well as the suitability of commercially-based (COTS) hardware for this application. This included designing a circuit board and building several samples as prototypes, together with developing software to allow it to read the vehicle's data buses and forward live vehicle data to the System Health Node for processing.

Approach

We would use a commercially available ARM-based microcontroller, develop a USB bus interface, test it using protocol simulators as well as actual vehicles, and verify that its performance meets expectations.

Results

A circuit board was built using an ARM7TMDI microcontroller. This microcontroller has two built-in CAN channels as well as several serial ports, one that we adapted to communicate over the J1708 bus. The J1708 protocol, as well as USB profiles, was implemented in C targeted for this ARM processor.

Performance was measured, and it was determined that this is a feasible approach for next-generation EPLS-like bus interfaces. The system was able to keep up with CAN and J1708 traffic comparable to that seen on the MTRV. Further performance improvements could be made by implementing some rate limiting and filtering on the interface board (e.g., dropping repeat messages); however, during our testing this did not prove to be necessary.

In terms of reliability, the USB bus was found to be very susceptible to typical noise and interference in a vehicle environment. This noise often caused the USB host (on the main system CPU) to detect errors and disconnect the bus device. It was not always possible to reset the device in software; occasionally a full system reboot or power cycle was required. We believe that this does not disqualify the idea of a USB interface to the vehicle data bus, but careful attention must be paid to noise immunity (cable shielding and ground practices, as well as close attention to impedances and EMI filtering).

Feasibility of Running AHM on Low-Cost ARM Based Hardware Running Linux

Problem

Asset Health Monitoring (AHM) deployments to date have run on embedded PC architecture, including PC/104 and PC/104 Plus, as well as single-board x86 computers (e.g., Intel Atom chipset). These architectures have advantages during development, but at the expense of higher cost, larger size, and greater power consumption/dissipation than a more fully embedded solution.

Goals

The purpose of this project is two-fold:

- Determine the feasibility of an alternate approach: rather than running asset health monitoring software on a small PC, run it on a more capable fully embedded platform similar to a modern cell phone or tablet processor.
- Determine the benefits and pitfalls of such an architecture by building a demonstration system and testing it in a realistic mobile environment.

Approach

Requirements Development

A set of requirements was developed. Rough, high-level requirements include:

- Low projected volume cost (<\$200 system).
- Wireless communications: Wi-Fi and Cellular communications; optionally support Bluetooth if practical.
- Vehicle communications: OBD-II (legacy and CAN), J1708, and J1939 (CAN).
- GPS Receiver.
- Power: 12V and 24V DC in. Total power consumption < 1 watt.
- Ability to run AHM software.

CPU Selection

We selected the BeagleBone¹ development board, which is based on the Texas Instruments AM3359 ARM Cortex A8 superscalar architecture microprocessor. It was chosen, in part, to shorten the development schedule and allow us to focus our hardware efforts on the vehicle and communication interfaces required by our application. However, because the BeagleBone is “open” hardware (meaning that full schematics and design data are made available to the end user) the capabilities of the board can be integrated into our own design on our own custom hardware. This would lead to a smaller, more cost-effective design path for a future, production-volume asset-monitoring solution.

Vehicle Interface Board

We designed and prototyped a vehicle interface circuit board, including power supply, CAN, J1708, and OBD-II interfaces. This board included a six-axis accelerometer, GPS receiver, and circuits implementing the low-level OBD-II physical protocols for all manufacturers of U.S. sold vehicles.

Communications Interface Board

We also created a wireless circuit board that included a Wi-Fi/Bluetooth module and a Mini PCI Express cellular modem slot.

Results

The hardware architecture was shown to provide a robust software development environment. Base operating system configuration was similar to our previous work on x86 architectures, so the transition to ARM was relatively straight-forward. Most of the key open source software components of the AHM system were available for the ARM as installation packages, allowing us to integrate familiar components like gpsd (GPS interface) and ntp (time synchronization) with little effort.

AHM software was installed and run with simulated and actual vehicle data, proving that the ARM hardware can run the monitoring software as effectively as larger, much more expensive x86 solutions (such as PC/104). Typical AHM operation used less than 30% of the CPU capacity of the ARM even at high load. The SDIO based storage solution proved to have high enough

¹ <http://beagleboard.org/bone>

write performance to effectively be equivalent to or better than previous PC/104 implementations of AHM.



Figure 8. BeagleBone with RIT's Vehicle Interface Cape showing

Communication to the cellular radio and GPS both proved to be seamless and reliable. Our vehicle data bus interface (OBD-II) design included support for modern (2008 and newer) and legacy (prior to 2008) vehicle bus standards. Each OBD-II bus type (J1850 PWM, J1850 VPW, ISO 9141-2, and CAN) was tested against sample vehicles for each bus, and communications were shown to be reliable, proving the feasibility of the integrated ARM/OBD-II design.

Challenges included integration of the Wi-Fi module. This device was chosen Wi-Fi module was selected because of its supposedly “seamless” integration with the processor. This device provided challenging to configure, in part because the “stacking” nature of the BeagleBone does not provide sufficiently controlled impedances for reliable communication. We identify this as a potential area of continued development, as we believe the problem would be solved by integrating the CPU and Wi-Fi on a single, impedance-controlled circuit board.

OBD-II Implementation

Problem

Previous Asset Health Management (AHM) efforts focused on monitoring expensive and high-value platforms utilizing heavy-duty engines. These platforms either have existing data buses such as J1939 and J1587 typical on heavy duty vehicles both within and outside the military, or they are high enough value to justify the benefits of adding a data bus, such as Light Armored Vehicle (LAVs).

Computing technology improvements have drastically reduced the cost, size, and power of on-board computers while increasing their capabilities. Systems that cost thousands of dollars 10

years ago are available today for only a few hundred dollars, and have made deployment of the same heavy-duty AHM technology on light-duty non-tactical military vehicles a feasible option; however, many of these vehicles use the Onboard Diagnostics bus (OBD-II), specified by SAE J1979 and required by law on light-vehicles.

Goals

- Produce an OBD-II data bus interface capable of running in AHM on the low-cost AHM hardware platform. Ideally the implementation can also run on COTS low-cost platforms such as ARM processors and Android mobile devices.
- Ability to request current powertrain data, such as RPM, speed, throttle position, etc., at the fastest possible speed from the vehicle.
- Ability to request for current active Diagnostic Trouble Codes (DTCs), as well as pending and “permanent” DTCs, and associate freeze-frame data.
- Support the three pre-2008 OBD-II protocols as well as the 2008 and later CAN-based OBD protocol. In particular, support the “Multi-PID” request format on CAN-based vehicles to increase data rates by 6x.

The ultimate goal is to support not only existing AHM diagnostic capabilities, but collect the data required to add the additional functionality of monitoring fuel economy, a focus for light-duty non-tactical vehicles.

Approach

Implementation based on the SAE J1979 specification, and verification of real-world behavior on multiple light vehicles, such as a model year 2004 Honda CRV, 2005 Honda Element, 2008 Honda Fit, 2004 Volkswagen Passat, 2000 Dodge Grand Caravan, 2003 Jeep Liberty, and 2010 Volkswagen Golf. Simulators based on data collected from these vehicles provided a quick way to test code in development against pre-CAN format OBD data and CAN format OBD data.

Most data was collected using COTS hardware; however, some testing was done with the low-cost hardware prototype board which had the same “STN” chip found in the COTS variant. This chip is compatible with the protocol implemented by the “ELM” OBD-II interface chip, which initially became popular in the hobbyist community. The STN is a higher-quality, but compatible, chip aimed at commercial and embedded markets. These chips implement every possible physical layer for OBD-II interfaces, thus we need only be concerned with the communication with the vehicle ECU.

To allow running the code in multiple environments and not be tied to AHM, the code was implemented as a lightweight library separate from the AHM framework. Incoming data is parsed and published as events to any software modules listening for the events. When integrated with AHM, an AHM function listens for these data events and translates the data into AHM data streams which can be processed normally by AHM functions.

Additionally, the requesting of data was separated from the processing of data, which simplified the design and made testing easier.

Results

An OBD interface meeting the goals was produced. The interface was tested via simulation as well as real-world prototype usage in two applications:

1. A prototype low-cost, ARM-based system running Linux and the full AHM technology with the OBD interface, and an AHM application to record the data on a Honda CRV. The hardware is discussed in the section “Feasibility of running AHM on low-cost ARM based hardware running Linux.”
2. An Android smartphone using a COTS Bluetooth OBD adapter on a variety of vehicles. A small Android-specific application was made that ties the generic OBD interface code directly to the AHM Flat File Database system to record the data without pre-processing.

Future work opportunities include:

- Development of new algorithms specific to fuel economy and driver behavior, not only for gasoline vehicles, but focusing on novel areas in alternative fuels (electric, hybrid, biodiesel).
- Configuration of existing AHM algorithms such as (sensor fault detection) for light vehicles.
- Development of production-ready system and testing on a large fleet of vehicles.
- Support for reading OBD test results.

It is also possible without excessive work to port the full AHM functionality to the Android operating system; in particular, the ability to run AHM on Android COTS systems is a compelling future option for several reasons:

- Their low cost and increasing popularity means that staff may be already carrying them for other purposes; thus AHM can be added to vehicles with only an incremental hardware cost to interface with the OBD-II bus simply by running AHM as an “app” on the device.
- Ability to use the additional data sources typically found on such devices:
 - Accelerometer/gyroscope to monitor driver behavior for both safety and fuel economy.
 - Barometer to measure air pressure (improves context of fuel efficiency).
 - GPS for location, from which the user can acquire temperature and wind via weather lookup to provide context that includes fuel efficiency (headwind/tailwind, etc.).

Flat File Database System for Real-time Health Monitoring Applications

Problem

Embedded Asset Health Management (AHM) systems record a large amount of data constantly while they are powered, and they typically operate in an environment which is susceptible to sudden power loss. This situation leaves little time for database recording and for maintenance operations to be performed on the database.

The existing system utilizes MySQL, a relational database management system (RDBMS). This is a well-known and understood commercial off-the-shelf (COTS) product for general purpose data storage, and most typically is designed to run on the server. We use the MyISAM data storage engine in MySQL, which does not support transactions but gives the best write performance and best ability to control database size. However, the lack of transaction support means serious reliability problems result when sudden power loss occurs.

The main benefit of using MySQL, and RDBMS in general, is that is they utilize the standard SQL protocol for accessing and modifying the data. The same technology in the embedded system can also be used to communicate with enterprise servers, where SQL-based RDBMS makes sense and is commonly deployed in that setting.

Although we have done much research on space management with MyISAM specifically, and reliability in MySQL by keeping the database open on disk for the minimal amount of time to reduce the likelihood of corruption on power loss, we have hit the limits of capability that MySQL can provide to us.

We aim to drastically improve on a number of issues we have with the current solution:

1. **Reliability** – When MySQL crashes, whether via power-loss or other means, MyISAM formatted database tables containing data collected across all missions can become corrupted.
2. **Space Management** – Deleting rows does not free disk space or necessarily prevent the database from growing, and in fact does not in many “opaque” database formats.
3. **Overhead** – MySQL runs as an external daemon (server) process, which increases startup time, memory usage, and CPU overhead by using the TCP networking protocol even though client and server are on the same machine.
4. **Data Density** – Data density (records stored per byte) in MyISAM formatted MySQL databases is already quite competitive against other RDBMS, but as a general purpose solution the majority of space used still resides in data indexing on primary keys.
5. **Replication** – MySQL has a built-in replication mechanism, but it is designed for keeping two highly available servers synchronized and is not well suited to the embedded/disconnected monitoring problem.
6. **Licensing** – MySQL in certain cases requires per-unit licensing fees.

Goals

- **Simple** - The file format should be easily described, read, and manipulated.
- **Compact** - Improve bytes per record compared to existing solution.
- **Reliable** - Errors must be detectable and ignorable. An error such as missing/added/modified bits in a file must not prevent the rest of the file from being interpreted.
- **Robust** - An error in the data storage in one mission should not affect data from other missions or prevent recording of new missions. A clock reset while powered off should not prevent recording of further missions nor eliminate the knowledge of the order of missions. The solution must also be tolerant of system power loss.
- **Fast** - Minimize disk writes and CPU utilization.
- **Portable** - Implementation should as portable as AHM itself.
- **Standalone** - Reduce complexity by eliminating dependency on an external daemon.
- **Licensing** - Prefer open-source products with no per-unit licensing costs if possible.

Approach

After consideration of other solutions and existing problems we chose a binary flat file solution consisting of data chunks wrapped in frames to allow detection of corruption and recovery of unaffected blocks from corrupted files. While we selected a binary file format, we wanted to re-use open formats as much as possible, so the binary format is defined using the [Google Protocol Buffers Format](#) (Protobuf).

The designed solution also allows the database to be split up into “slices,” which allows for varying data synchronization techniques. This is discussed further in the section “Flat File Database Data Receiver and Platform Data Synchronizer.”

Results

The technology selection addresses the following goals:

- **Simple** – Code generators exist for many languages to read protobuf format files into an object format.
- **Portable** – Implementations to read/write exist for many languages, with primary support for Java, C++, and Python, which at least one of those options is available on virtually every platform today.
- **Standalone** – protobuf is a code library that can be used as part of an existing program; thus a separate software installation and running server is not needed.
- **Licensing** – Google’s implementation is BSD-licensed, and third-party implementations for other languages are typically likewise liberally licensed.

The goals of **compact**, **reliable**, and **fast** will be addressed in the benchmark evaluation.

Benchmark Evaluation

Operating on the embedded platform, we find, compared to MySQL, that:

- The system is able to recover from a crash or power-loss in only a few seconds or less.
- The storage required on disk drops by 64% if we consider MySQL binlogs only, or by 83% if we include both database and binlogs.
- The data required to be transmitted (after compression) drops by 22%.
- CPU effort (time * usage) to create a new database is reduced by 85%, time by 93%.
- CPU effort to load data into the database is reduced by 96%, time by 99%.

The following graphs detail the results of a benchmark of the AHM drivers for MySQL vs. the protobuf-based flat file implementation for writing 10k or 100k records. The minimum value is taken over several runs.

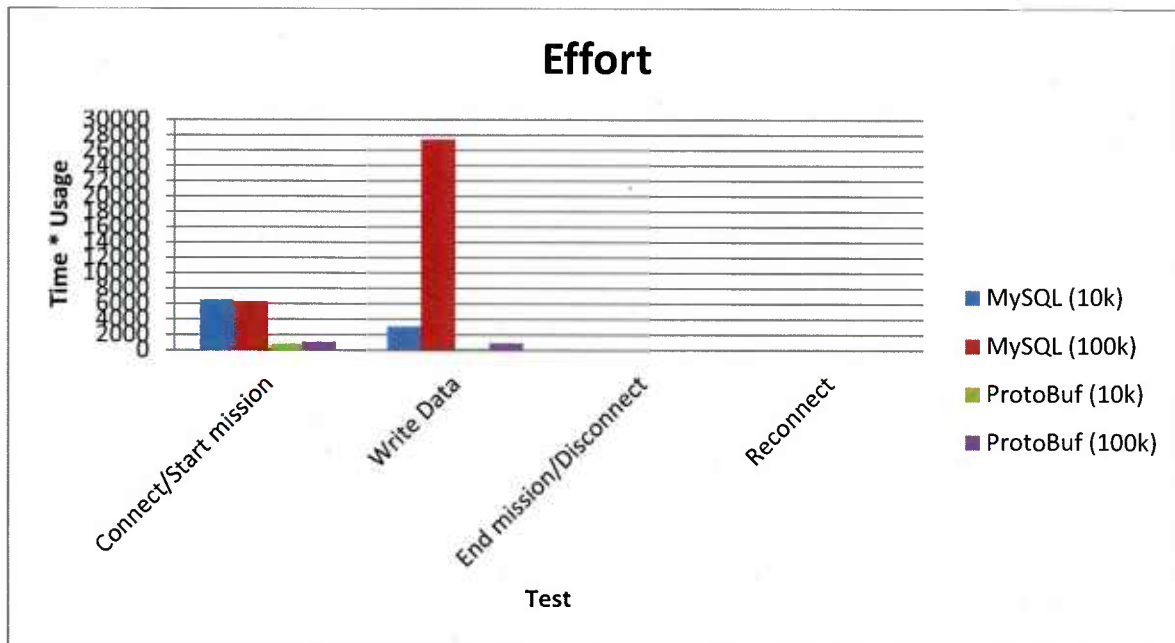


Figure 9. Effort Benchmark.

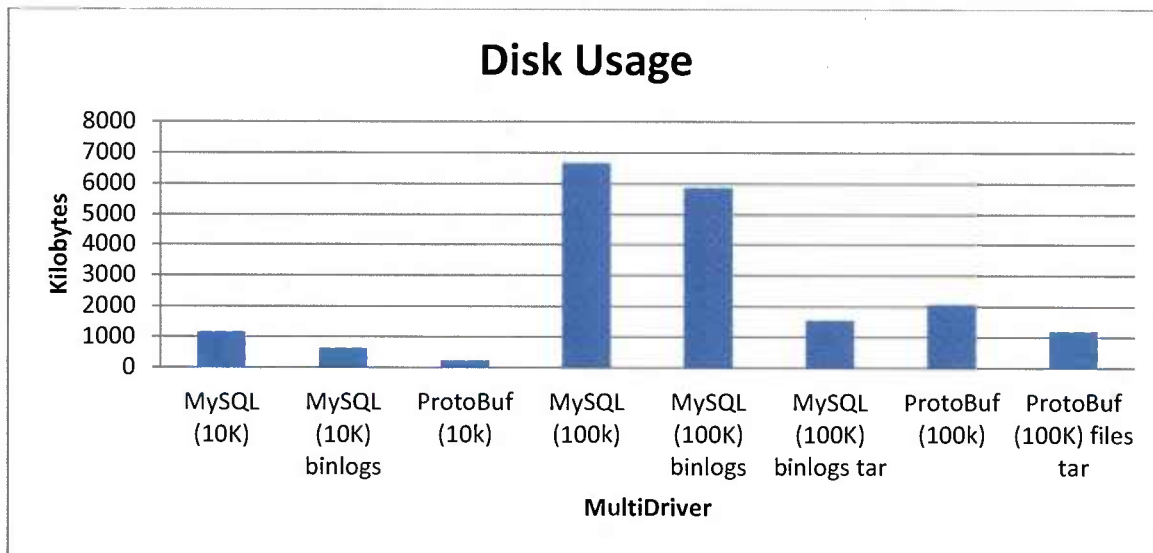


Figure 10. Disk Usage Benchmark.

Flat File Database Data Receiver and Platform Data Synchronizer

Problem

The updated Asset Health Management (AHM) system can use a binary flat file to store data as an alternative to SQL-based relational database management systems (primarily MySQL). With the data stored, a data transfer and synchronization process is required to replace the existing MySQL replication and/or MDS (Maintenance Data Synchronizer) method.

Goals

The primary goal is to replicate the MDS functionality, but reading from the flat file repository instead of the single-vehicle SQL database. We also wanted an improved user interface that allows for managing large fleets of vehicles and better reporting and guidance for errors, something MDS was not able to do. The original SQL writer backend from MDS was reused.

- Automatic upload of data from monitored platforms.
- Automatic synchronization of uploaded data into a multi-platform master database.
- Design for scalability to allow management of large fleets.
- Minimize administrative time by focusing interface on platforms with synchronization problems.

Approach

An overview diagram of the data management architecture:

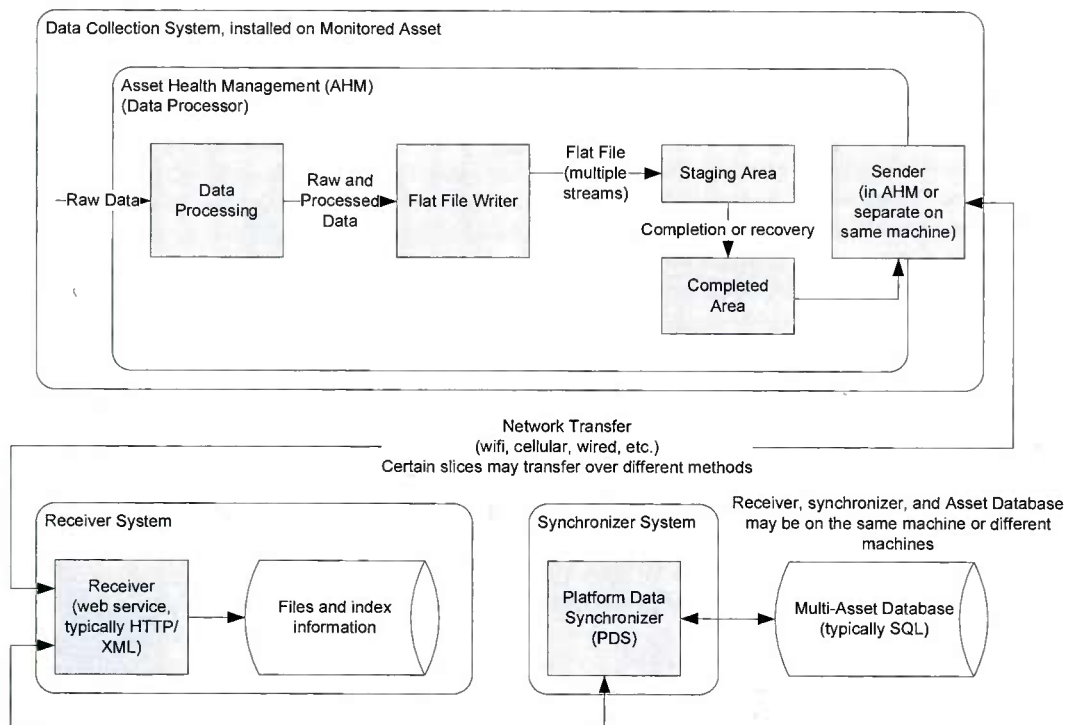


Figure 11. End-to-End Flat File Architecture.

Asset Health Management System

The AHM system (previously developed) collects raw data on a monitored asset and outputs raw sensor data as well as processed diagnostic, health, and summary data.

AHM Flat File Writer

The driver writes collected operational data in the form of “missions” and slices and segments within those missions.

AHM Segment Sender

The segment sender connects to the PDS (Platform Data Synchronizer) receiver web service when a network connection is available.

The segment sender is not the only method to transfer data. Each data segment file stands alone, so it can be transferred to the receiver via alternative networks or manually via physical media transfer such as USB storage devices.

Platform Data Synchronizer Receiver

The PDS receiver implements a RESTful web service over HTTP protocol, exposing a repository of AHM data segments, allowing for embedded platform upload (storage) and download (retrieval) for synchronization or analysis.

Platform Data Synchronizer

The PDS tool observes the data segment repository managed by the PDS receiver and synchronizes (writes) the data into the already existing enterprise fleet storage format (SQL Database revolving around MIMOSA-CBM standards). It is also the only directly visible component to the AHM system for managing health data collection and synchronization. Additionally, PDS provides functionality to detect and guide administrators through problems and their solutions, such as:

- Invalid credentials (username/password) in the segment sender on the embedded platform.
- Mismatch between embedded node and platform assignment. The PDS Receiver ensures that a particular node only transmits data for the platform it is known to be installed on, as a security precaution and also as a data integrity to prevent multiple nodes from declaring that their data is for the same platform.
- Errors in the data recorded (such as sensors not known to exist to the fleet storage system).
- Network connectivity problems with PDS receiver or fleet storage system.
- Vehicles that have not reported data in an abnormally long period of time.
- Missing data segments.
- Platforms deployed to the field but not recorded in the fleet storage system.

The screenshot shows the PDS Dashboard interface. At the top, there are tabs for 'Vehicles', 'Segments', 'Repository Log', 'PDS Log', and 'Problems'. Below the tabs is a 'Filter' input field and a 'Show by tag:' dropdown menu set to 'All Vehicles'. The main area contains a table with the following columns: Title, ID, Status, AutoSync, Last Mission ..., Last Mission ..., Last Sync, Last Segment, Receive Age ..., and Tags. The table lists several vehicles, all with a status of 'Unknown'. Below the table is a 'Modify' section with four buttons: 'Turn On Auto-Sync', 'Turn Off Auto-Sync', 'Set Receive Age Limit', and 'Clear Receive Age Limit'.

Title	ID	Status	AutoSync	Last Mission ...	Last Mission ...	Last Sync	Last Segment	Receive Age ...	Tags
Liban Volvo	3090	Unknown	☐						(Maint_Dev)
SAIA_6666	6666	Unknown	☐						(Maint_Dev)
SAIA_6667	6667	Unknown	☐						(Maint_Dev)
SAIA_7902	7902	Unknown	☐						(Maint_Dev)
SAIA_8503	8503	Unknown	☐						(Maint_Dev)
SAIA_8531	8531	Unknown	☐						(Maint_Dev)
SAIA_8689	8689	Unknown	☐						(Maint_Dev)
8699	8699	Unknown	☐						(Maint_Dev)
J2007	12007	Unknown	☐						(Maint_Dev)
Cougar(old)	1622341	Unknown	☐						(Maint_Dev)
PDS_Test-2	2	Unknown	☐	04/22/11 16:21	04/22/11 16:21	05/11/11 11:22	05/11/11 11:22		(PDS_Test)

Figure 12. PDS Dashboard.

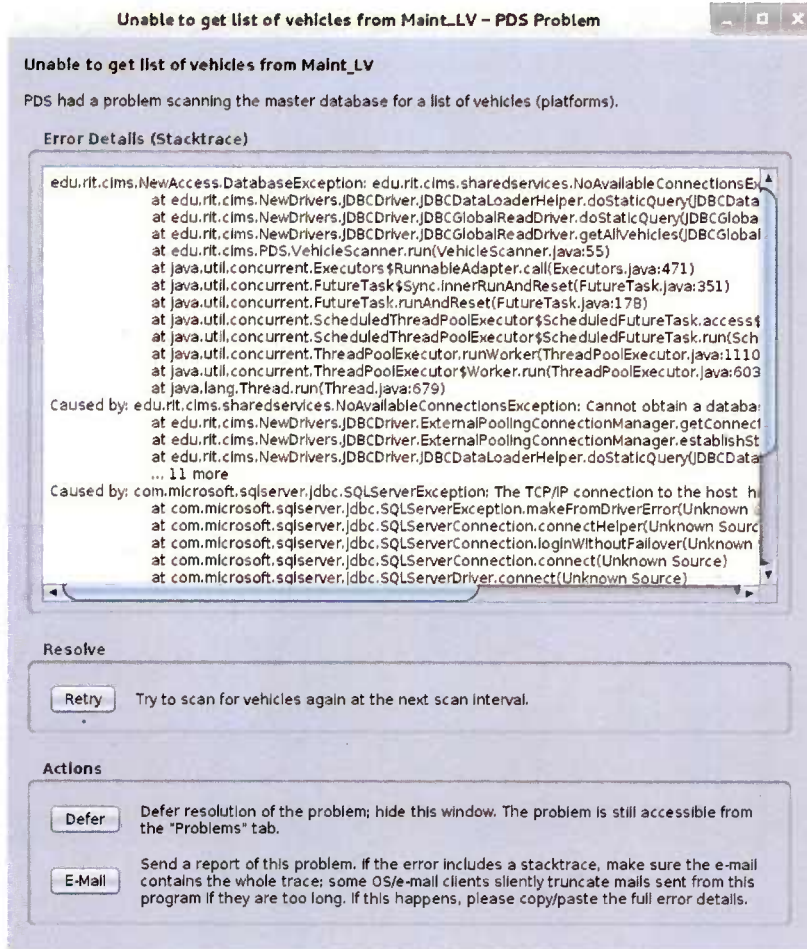


Figure 13. PDS Problem Resolution Screen.

Results

The resulting system has been used successfully for autonomous transfer of data from platforms into the multi-platform master database. The PDS dashboard provides a good view into problems occurring in any part of the system (from the monitored platform, to the receiver, the synchronizer itself, or the database), all from a single location. The most common problems of connection issues and database version mismatch (between platform and master database) are detected immediately and handled in a clear manner.

In terms of performance, the SQL writing backend is the bottleneck in a typical setup, and it is the same as from the previous MDS tool. We investigated ways to speed up this synchronization by investigating data-loading approaches that are proprietary to Microsoft SQL Server, rather than the non-proprietary approach we used (Java JDBC PreparedStatements). We tried the following methods:

- **Batched PS:** Create a SQL PreparedStatement for inserting, set parameters and call addBatch, then executeBatch.

- **Multi-Insert:** Use new syntax in SQL Server 2008: INSERT INTO X (C1, C2) VALUES (V1, V2), (Y1, Y2), (W1, W2)....
- **Temp Table:** DECLARE @MyTemp TABLE in same format as destination table, use a multi-insert to load it, then INSERT INTO DMCalcNumData SELECT * FROM @MyTemp.
- **Union All:** Multi-insert using the more compatible "UNION ALL" syntax: INSERT INTO X SELECT V1, V2 UNION ALL SELECT Y1, Y2....
- **Text Bulk:** Write batch to a CSV file then use BULK INSERT. File may be placed on local disk (SSD), ramdisk (RAM), or uploaded to remote server via UNC network path.
- **Binary Bulk:** Same as "Text Bulk" but write to binary file in SQL server's format and use proprietary ".fmt" file to describe it, then BULK INSERT.

The bulk insert methods only work well when PDS/MDS run on the same machine as the SQL Server and when utilizing a "RAM disk" for temporary import storage. In this best case it is possible to double-write performance. However, due to the large effort to convert the code base to write binary bulk files, the fact that PDS/MDS would no longer be able to operate remotely, and the technique being specific to Microsoft SQL Server, we decided not to deploy this solution and remain with the existing batched PreparedStatement solution.

Multi-Source Signal Support in Asset Health Management (AHM)

Problem

Prior vehicle applications for the AHM system typically involved a vehicle with only a single data bus or with no pre-existing data bus at all (for which one was added for the project). However, newer vehicles in both military and commercial vehicles are commonly equipped with multiple, existing data buses as well as multiple ECUs reporting the same data element even on the same data bus. For example, the MTRV (Medium Tactical Vehicle Replacement) has both a J1939 and J1587 data bus, and there is overlap in the data elements reported by these buses, as well as multiple ECUs reporting the same data (e.g., vehicle speed).

The previous solution to the problem was not adequate. In J1587, AHM had the ability to map a MID (identifying an ECU) and PID (data element ID) to a port (data streams); thus it was possible to map each ECU to a unique port, but only if it was known in advance what ECUs exist and which PIDs they report. For J1939, there was not an appropriate solution; the previous solution would combine all reports from ECUs into a single data stream, as if it came from the same sensor.

For the data streams that were separated (a single J1939 result and multiple J1587), it was possible to build a combination function that selected the "best" source that would become the primary element. For example, vehicle speed reports from three ECUs would be combined into the final determination of "vehicle speed" reported by AHM.

To summarize, there were three primary issues with the multi-source support in the previous software:

1. The number of ECUs and data elements reported by each must be known in advance of AHM installation (specific to vehicle series).
2. Multiple ECUs on J1939 were not supported properly at all. In fact, on J1939, source addresses of ECUs are not well-known and defined as in J1587.
3. The number of “redundant” ports is fixed and specified manually in the AHM application.

Goals

Our goal for this project was to resolve the current problems by making AHM more flexible in this situation. The following additional requirements allowed us to solve the problems:

1. The AHM application configuration does not need to know how many ECUs (from any data bus) will report a particular data element, thus making the application applicable to a wider range of vehicles by not needing to rely on specific knowledge about ECUs.
2. Each ECU’s data will be mapped to a unique port.
3. AHM is able to dynamically create ports while running based on a template port definition so that each ECU can have a unique port.
4. AHM will keep track of the source of data for the port, so that functions can ask “what data bus did this port come from and what is the source ECU address?” and use this information for selecting the “best” port.
5. AHM streams can be placed into groups, so that a function may ask for the data for all streams belonging to a group.
6. AHM places all streams dynamically created for ECUs into a group for the data element (e.g., all ports containing vehicle speed reports from ECUs placed into one group).
7. The intention is to combine all of the ECUs reporting a particular data element into a single data stream, formed by selecting the “best” source at each particular moment.

The requirements all revolve around a fundamental shift in AHM data handling: instead of the set of ports being fixed in the design, the set of ports is dynamic, created based on the number of ECUs reporting particular data elements while AHM is running. This also means that the number of inputs into a function changes while running when a function asks for a group of ports.

Approach

Evaluation of data collected on previous projects with the multi-source problem was analyzed to ensure that the implementation would provide the features necessary to allow intelligent combination of multiple sensors into the best guess at the true signal. Bus data captures from these vehicles were used to test the implementation under simulation.

Results

We successfully implemented a system that solved the original problems and addressed all of the goals for the updated system. The following components addressed this issue:

Stream Group Definitions

The AHM application configuration contains a definition for stream groups. Stream groups have a name (such as “speed” or “rpm”) and initially start out with no ports within. A stream group has a template that is used to create new ports on demand for as many sources of information arise during monitoring. For example, there may be 2x J1939 ECUs, 3x J1587 ECUs, and 1x GPS unit reporting vehicle speed. Functionality exists in AHM to query for the streams in a group and ask about the source of the information (such as bus type and ECU source address).

Updated Unified Function Architecture

The Unified Function architecture in AHM is a universal system of writing functions taking a series of configuration parameters, inputs, and outputs. This architecture was enhanced with inputs and outputs that can read from or write to stream groups. This structure is novel for the AHM system as they are the first input and output types that change the number of data streams (ports) while the system is running. In addition to the raw data, the inputs also return source identifiers containing arbitrary information (at this time, bus type and source address). The outputs are also given source identifiers and dynamically create new streams as new sources are written. The source identifiers provide functions with a basis for discriminating their inputs when faced with multiple choices for the same information (e.g., vehicle speed).

Updated J1939 and J1587 Listeners

The listeners are able to direct a certain data element (such as vehicle speed) at a “stream group” instead of a singular port. When a new ECU reports, a source identifier is created with bus type and address. As discussed earlier in the updated Unified Function Architecture, as new source identifiers are created, new ports are created in the stream group.

Updated Sensor Fault Detection Functions

The multi-source signals provide a stronger case for SFD, because redundant sensor information can be compared to find faulty sensors and come to a stronger conclusion for the true physical state. The SFD functions were updated to read and write stream groups. For the most part, these changes were easy as the SFD changes largely were simply adjusted to have the algorithms applied in a loop for however many sources were presented to it.

New Sensor Selection Function

The additional information from port source information and SFD provided by the multi-signal processing allows for an improved selection of a final source. For example, there may be 4 ECUs and a GPS unit reporting vehicle speed, but the function should select the “best” speed. The function considers the following when making the determination of the best source:

1. The type of data bus the data came from (such as J1939 or J1587). The J1939 bus is more modern and often has higher resolution data reported at higher speeds.
2. Whether or not sensor fault detection (SFD) has detected that a particular source is faulty or overly noisy.
3. The source address of the reporting ECU. An AHM application engineer may have done a study on a vehicle to determine that a particular ECU has a better reliability or better measurement of the data element. For example, perhaps engine speed (RPM) is reported by engine ECU, transmission ECU, and brake system ECUs, and the engineer can set a preference in that order.
4. The last source selected for the output.

Preferences are set by the application engineer. The function will try to select the same source and stick with it; however, it will select a new source if it stops reporting or is marked by SFD as having bad data. The newly selected source is selected by preference rules set by the application engineer. For example, the following logic could be applied at each moment of time to combine a group of speed sensors into a single speed-sensor output:

1. Select the stream that was chosen last time, if it is still reporting and valid (per SFD).
2. Otherwise, select the first reporting and valid of J1939 transmission or engine ECUs.
3. Otherwise, select the J1587 engine ECU, if it is reporting and valid.
4. Otherwise, select the first reporting and valid signal seen from J1939 data bus.
5. Otherwise, select the first reporting and valid signal seen from J1587 data bus.
6. Otherwise, select the first reporting and valid signal (such as GPS).
7. Otherwise, all signals are not reporting or invalid, so select the first such signal; the resulting output would be marked as not reporting or invalid as a result.

DAC Node Graph Tool

Problem

The development of a Data Acquisition Node (DAC) capable of transferring waveforms (e.g., vibration data) over a J1939/CAN data bus created the need for a tool capable of receiving and visualizing this data.

Goals

DAC Graph Tool was created to fill this need. Our goal was to use this tool in testing and debugging of the waveform protocol itself, as well as the testing and debugging of the waveform processing algorithms running in the data acquisition nodes.

Approach

DAC Graph Tool was developed to read waveform data transmitted on the bus using the RIT waveform protocol developed for this purpose. It was developed in Java so that we could reuse Java code from our Asset Health Monitoring software that receives and processes waveform

data. We also used the same Java graphics library used by our Maintainer's Graphical User Interface (MGUI) Tool to display the waveforms.

Results

All of our goals were implemented. Shown in Figure 14 is a screen shot of the tool displaying waveforms from a DAC node connected to a three-axis accelerometer.

DAC Graph Tool was developed to read waveform data transmitted on the bus using the RIT waveform protocol developed for this purpose. DAC Graph Tool can display up to four graphs of waveform data from different sources. The number of graphs to be displayed is configurable. The titles for each graph as well as the labels on the X and Y axis are also configurable. Each graph will update continuously as waveforms are received.

The tool also saves the waveforms it receives to CSV files. In our testing, these files were used to capture raw and calculated waveforms transmitted by the RIT DACs. These waveforms were imported into MATLAB to verify that IIR/FIR filtering algorithms and FFT algorithms executed by the DAC node were correct.

This tool was used in the testing and debugging of the waveform protocol itself as well as the testing and debugging of the RIT vibration DAC and the RIT second-generation four-channel DAC.

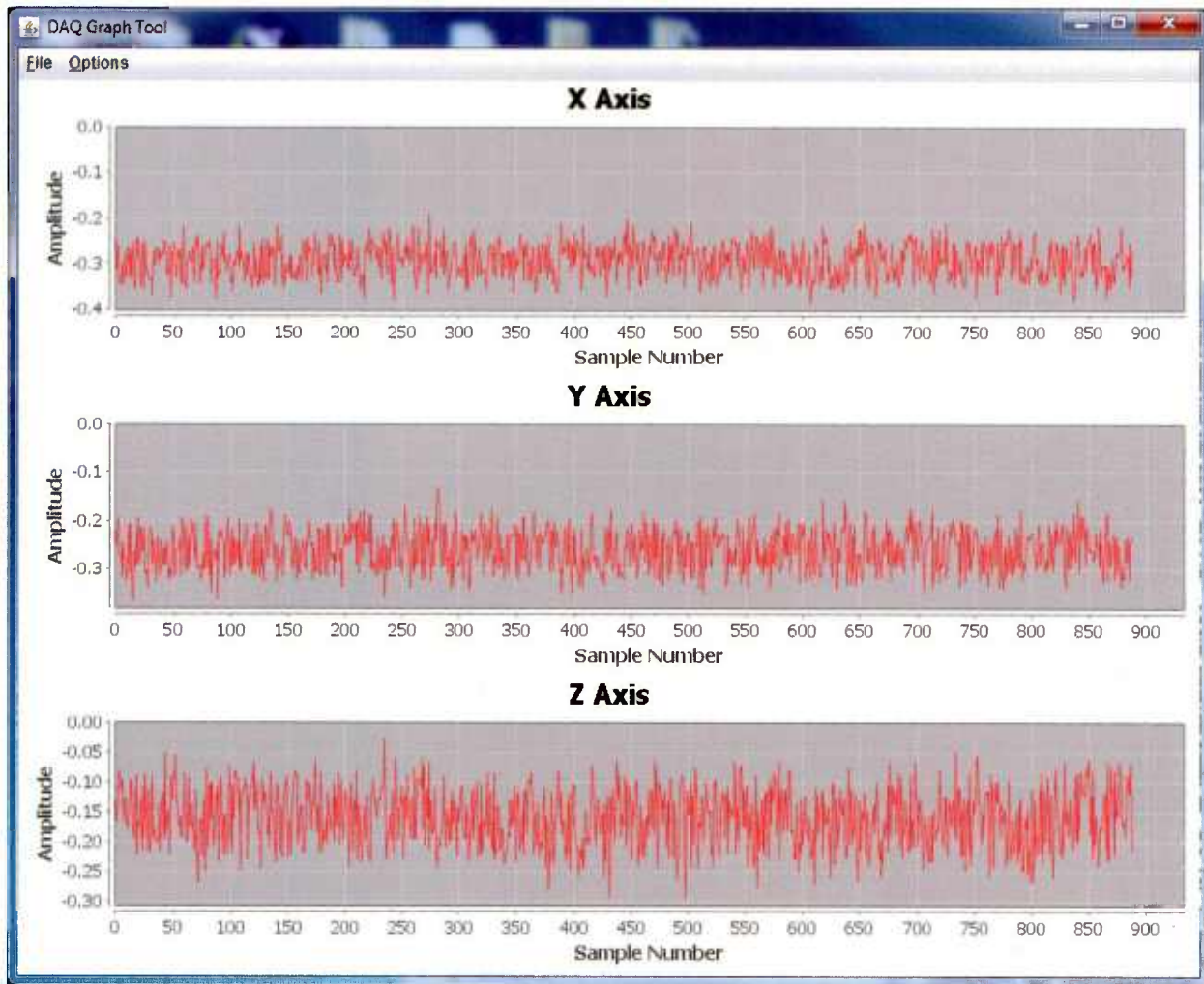


Figure 14. Graph Tool displaying waveforms from a three-axis accelerometer.

DAC Node Programming Tool

Problem

Asset Health Management systems typically employ several data acquisition controllers (DACs) to acquire and process data from vehicle sensors. The programming of these DACs requires an engineer experienced in data acquisition systems and embedded programming in C, the language most often used in the programming of embedded processors. Such programming is difficult and requires detailed knowledge of computer programming languages and embedded programming concepts.

Goals

The goal for this project was to develop a simplified DAC node programming tool (DAC App Tool) that allows a complete DAC application to be written as a collection of simple graphical “flow charts.” This will allow application development with less dependence on embedded

software developers. A subject matter expert can develop embedded software with a simple drag-and-drop graphical interface, and the tool will automatically generate appropriate embeddable firmware for the target device.

Approach

The typical processing of a signal in a DAC node can be described simply as:

- Read a raw digital or analog value from a sensor.
- Process this value in some way; e.g., convert it to a pressure in kilopascals.
- Put this processed value in a data packet and send the packet on the data bus.

DAC App Tool was developed to allow an application engineer to perform this processing by creating function block diagrams from a library of graphical function blocks. Function blocks are available to read sensor values, perform arbitrary calculations, make logical decisions (conditional branching), and send values on the data bus.

Overview

DAC App Tool was designed to support building and transmitting data packets on a J1939/CAN data bus, commonly used in military ground vehicles. This bus is also in common use on passenger and commercial vehicles.

App Tool supports a hierarchical project tree. A project typically contains several DAC programs. A program is a complete application that, when compiled, is flashed to a DAC node. A program contains one or more PGNs (Parameter Group Numbers), each representing a data packet to be sent on the bus. A PGN contains one or more SPNs (Suspect Parameter Numbers), each of which is a data value calculated from an analog or digital input.

Configuration

DAC App Tool is linked to a PGN/SPN database. This database contains the definitions of the PGNs and SPNs that the application engineer will be allowed to configure in a DAC program. This allows different databases, containing different PGN/SPN definitions, to be associated with different projects.

App Tool also supports a configurable FIR or IIR filter of arbitrary order on each analog input channel. If the application engineer chooses to use such a filter, App Tool provides an interface for entering the filter coefficients.

Flowcharts

A SPN represents a data value that is calculated from an analog or digital input. This calculation is represented graphically by creating a “flowchart” of function blocks.

A predefined palette of function blocks is provided. Function blocks come in four types: *Get*, *Calculation*, *Conditional*, and *Return*.

A flowchart starts with a *Get* block that gets the current value of an analog or digital input. There must be only one *Get* block per SPN.

One or more *Calculation* blocks are then used to perform arbitrary calculations on the value returned by the *Get* block.

Conditional blocks can be used to make a true/false decision. This introduces branches in the flowchart.

Finally, a *Return* block is used to indicate the value to be placed in the data packet and transmitted on the CAN bus. Each branch in the flowchart must be terminated by a *Return* block. There are three types. *ReturnValue* returns the calculated value. *ReturnError* returns a J1939 error indicator. *ReturnUnknown* returns a J1939 unknown indicator.

A flowchart represents a value “flowing” through the chart from left to right. The *Get* block obtains a value. *Calculation* and *Conditional* blocks accept a value from the block to their left, process it in some way, and make it available to the block to their right. The *Return* block places this value (or an error/unknown indicator) in the data packet for transmission on the bus.

The function blocks available to the user in App Tool are listed in an external file named *Functions.xml*. This design allows a different file to be used in each project, if desired, to control the available blocks. This also allows new function blocks to be added if and when they become available in the DAC application library.

Executable Program Generation

DAC App Tool stores each graphical program in a separate XML file, and stores a reference to each of these program files in a master XML file. Each XML program file must be translated to an executable program before it can be run on a DAC node. App Tool translates each XML file to a C source file, compiles it, and links it with an application library. The application library contains the code for all function blocks, as well as core operating system code for the DAC.

To compile a program from App Tool, the application engineer invokes a *Compile* command. App Tool also supports a *Compile All* command that compiles all programs in a project. App Tool was designed to call an external shell script to compile and link the C code; it is the shell script that actually invokes the compiler and linker. This design allows the compiler or linker to be changed by changing the script alone; there is no need to rebuild App Tool itself. App Tool captures the errors, warnings, and informational messages generated by the compiler and linker, and displays them within App Tool to aid in debugging. In this project, the Freescale Code Warrior compiler and linker were employed.

After a DAC program has been successfully compiled, the engineer invokes a *Flash* command to download the executable image to a DAC node. This image is copied to the DAC node over the J1939/CAN bus.

Results

DAC App Tool was successfully used to program the DACs on the Grizzly Light Armored Vehicle. It was also used to program the DACs used for vibration analysis on a UAV (unmanned aerial vehicle) engine and on an engine undergoing tests on an Engine Dynamometer. The DAC nodes used with App Tool on these projects were RIT's second-generation four-channel DAC nodes.

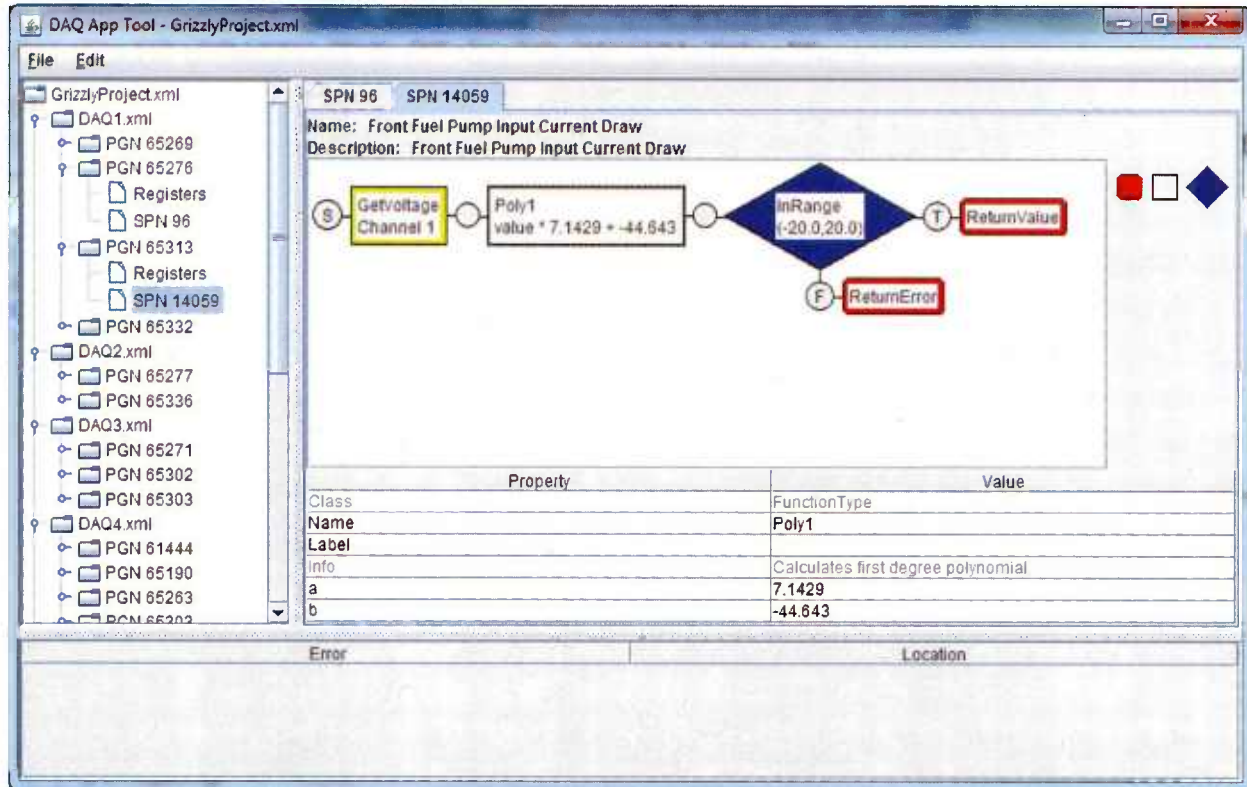


Figure 15. DAC App Tool Example

DTC Data Model J1939 and J1587 Listener Improvements

Problem

While our health monitoring system collected a variety of data from the J1939 and J1587 data buses, we did not collect Diagnostic Trouble Code (DTC) data. The DTCs, or “faults,” are reported by the engine control unit (ECU). The ECU gathers data from a variety of sensors. Occasionally sensors will trigger a fault report, usually indicating a performance or maintenance problem that requires attention.

Goals

Our goal was to collect DTC data from the J1939 and J1587 data buses. DTC collection enhances our health monitoring system in several ways; for example:

- We can provide fault information on display devices other than the vehicle dashboard.

- We can give context to sensor data analysis both before a fault becomes active and while the fault is active.

Approach

Achieving this goal required several modifications to our health monitoring system. Every effort was made to use common code, but since the J1939 and J1587 standards differ, some bus specific modifications needed to be made. The changes are outlined in the following sections.

Database Changes

A common database definition was developed for DTC data. The DTC fields are:

- Port ID: Identifies the source of the data, J1939 PGN or J1587 MID.
- Time: The time at which the DTC was reported.
- Primary Code: The primary code of the DTC. A PID for J1939 and a SID for J1587.
- FMI: The Failure Mode Indicator of the DTC.
- Secondary Code: Optional field, not used in J1939/J1587. Provided an additional discriminator for OEM codes.
- Occurrence Count: Optional occurrence count for a DTC.

DTC Data Model

A common data model was developed for both J1939 and J1587 DTCs. The data model contains all the information defined for the DTC database record.

DTC Bus Packet Parsers

Separate packet parsers were developed for J1939 and J1587. The parsers decode the bus packets and create DTC data objects for use by the health monitoring system.

Bus Listeners

Bus listeners are used to transform bus packets into data object that can be used by the health monitoring software. The health monitoring system already contained bus listeners for both J1939 and J1587. The bus listeners take packets off the bus, determine the packet type, and pass the packets to the appropriate packet parsers which create the data objects. The bus listeners were modified to recognize DTC packets and invoke the DTC parsers.

Results

The target platform for exercising the DTC processing was the Medium Tactical Vehicle Replacement (MTVR). This vehicle housed both a J1939 and J1587 bus.

We successfully achieved our goal to collect and record DTC data from both the J1939 and J1587 data buses. During our development, we use simulations that acquire bus packets from files rather than a vehicle bus. The packet files are populated by capturing them from vehicle buses, in this case, an MTVR. Using packet files known to contain DTCs, we were able to

confirm that the packets were parsed into DTC data objects, and those objects were successfully written to the DTC database records.

Bus Data Analysis and Visualization

Problem

A good, cost-effective vehicle data-monitoring system takes advantage of the existing vehicle infrastructure (sensors and data buses). The ability to quickly collect, interpret, and analyze the existing information content on a vehicle (or vehicle family) greatly reduces the development time of the overall monitoring system.

Goals

The objective of this project was to accelerate the assessment of the existing vehicle sensors.

Approach

To facilitate the assessment process, we identified the need for: 1) software for capturing data on a vehicle during operation in a non-intrusive manner; and 2) software for parsing of the captured data and transforming it into a compact summary in a user-friendly format. Information from SAE J1587 and SAE J1939-71 specifications was consulted for data acquisition and analysis. LabVIEW and MATLAB were selected for data capture and data analysis, respectively. Excel spreadsheet was selected for the user interface.

Results

The outcome is the toolbox containing two LabVIEW-based, data-capture applications (one each for J1708 and J1939), two MATLAB applications (for parsing 1708 and 1939 data packets), and the process for capturing the data and parsing the captured files into spreadsheet summaries. Figure 25 and Figure 17 illustrate J1708 and J1939 report summaries, respectively.

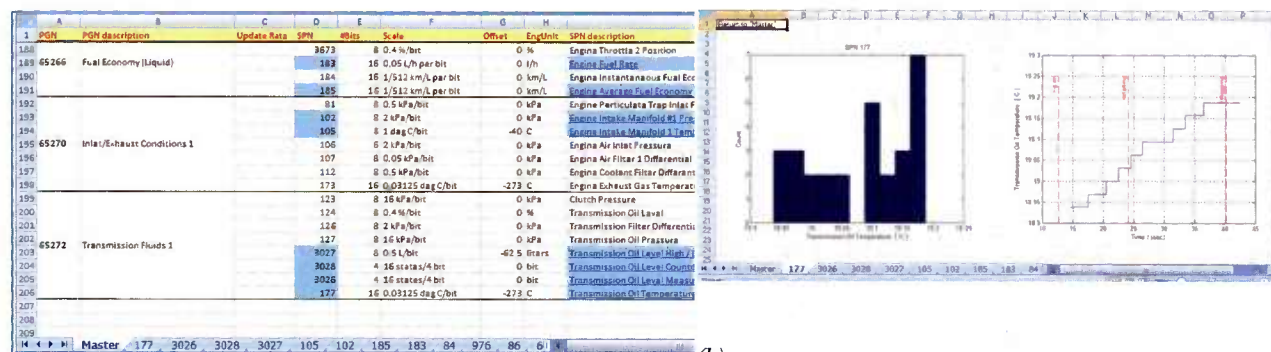


Figure 16. J1939 spreadsheet report: (a) masters sheet containing the PGN/SPN summary with linked sheets for data samples; (b) one of the linked data sheets for a particular SPN (SPN = 177 in this case) with the return hyperlink.

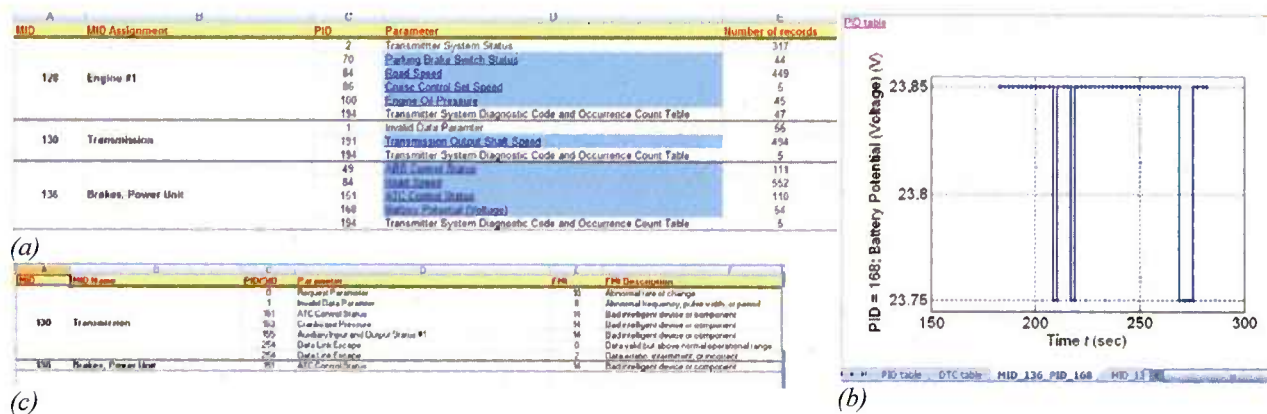


Figure 17. J708 spreadsheet report: (a) PID table with hyperlinks to individual MID_PID details; (b) example MID_PID details – battery voltage from MID = 136, with the return hyperlink; (c) diagnostic trouble codes (DTCs). Note: PID = Parameter IDentification character, MID = Message Identification.

Improved User Interface for the Maintainer's Graphical User Interface (MGUI) Tool

Problem

To support the data analysis tasks in this project, the existing version of MGUI requires improved data analysis features, visualization of new data types, and improved report generation.

Additionally, the existing commercial charting library we used as the basis for the original MGUI code was not receiving updates, had an outdated interface that made adding new features difficult, lacked the chart types we wanted in future work, and had been acquired by a company charging much higher prices.

Primary Goals

There were many goals for this work, many of them small convenience features or bug fixes (more than 100 tasks). The most notable new features are:

- Update to a charting library that supports more graph types that we want to use, preferably one that is free and open-source to eliminate licensing costs for development or distribution.
- Drastically improve performance on viewing huge datasets by dynamically down-sampling based on zoom level.
- Ability to add annotations and titles to charts that are preserved in image export and printing.
- Ability to filter lists of collected missions based on data within.
- Ability to view multiple missions in the same graph.
- Ability to switch to view previous and next missions quickly without closing the charting view.
- Ability to filter scatter graph data by time, giving the scatter plot a 3rd axis (x, y, and time).

- Improved user interface: allow panning graphs with mouse or keyboard, and reordering graphs.

Approach

Primarily, we solicited feedback from the data analysts that used MGUI on a daily basis to form the user interface requirements. Iterative development and frequent collaboration with the data analysts ensured that the most valuable features were implemented and implemented properly.

Results

All of our goals were implemented. Updated versions of this software were used both internally at RIT and externally in support of the US Marine Corps Embedded Platform Logistics System, and with the Solid Oxide Fuel Cell research efforts. Below are screen shots highlighting some of the major MGUI improvements:

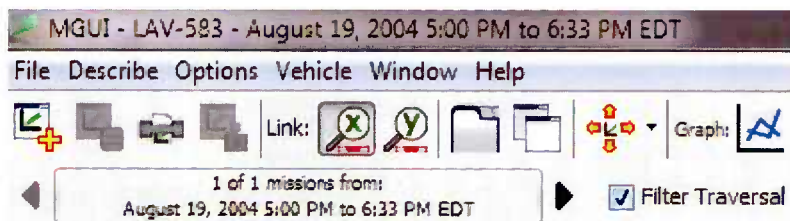


Figure 18. Mission navigation possible from graphing screen.

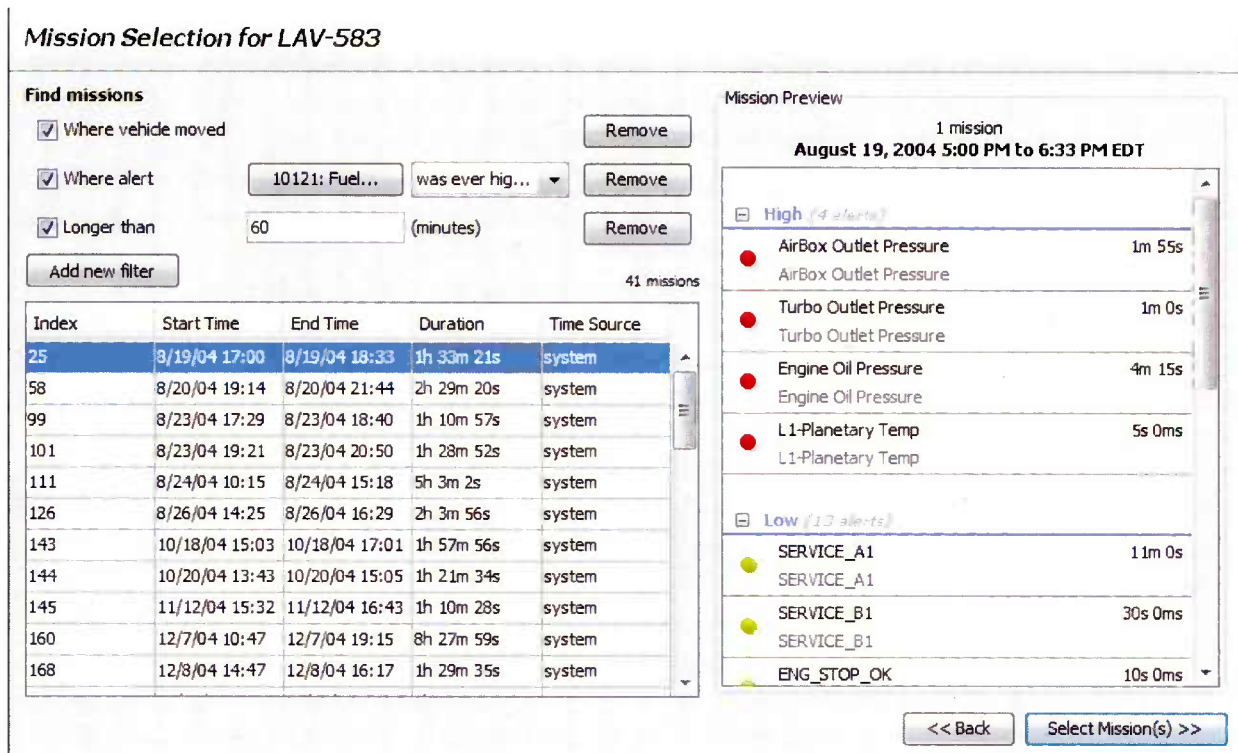


Figure 19. Mission selection window mission with filters.

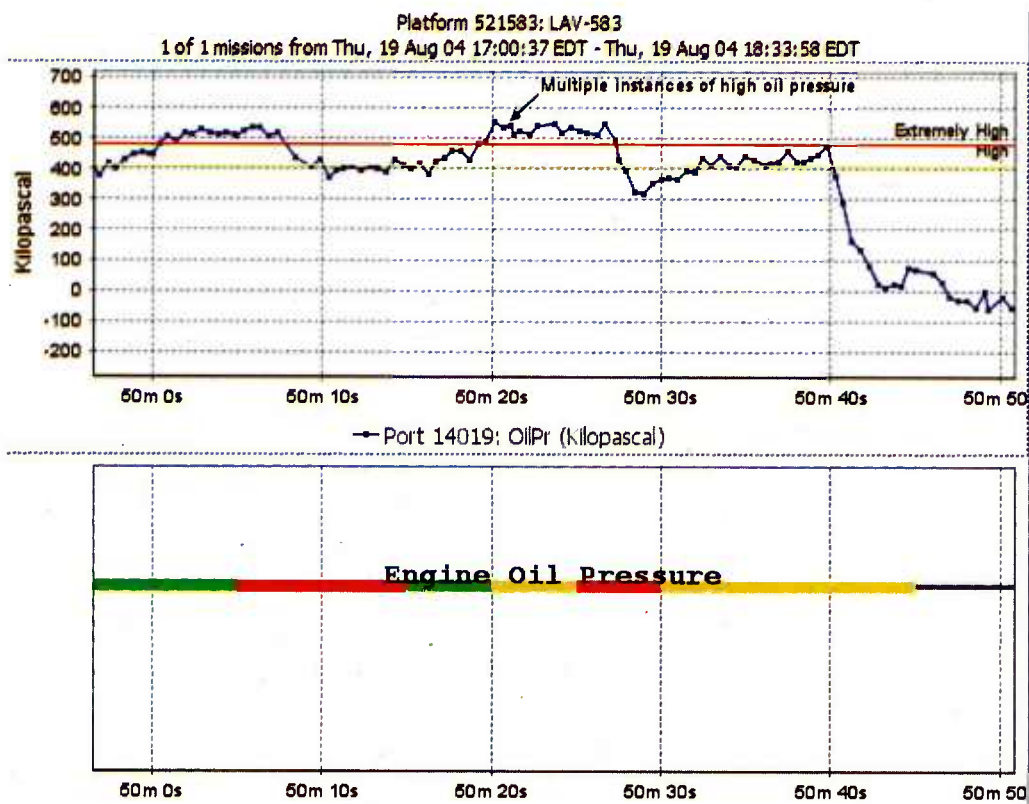


Figure 20. Display of new annotations and titles features.

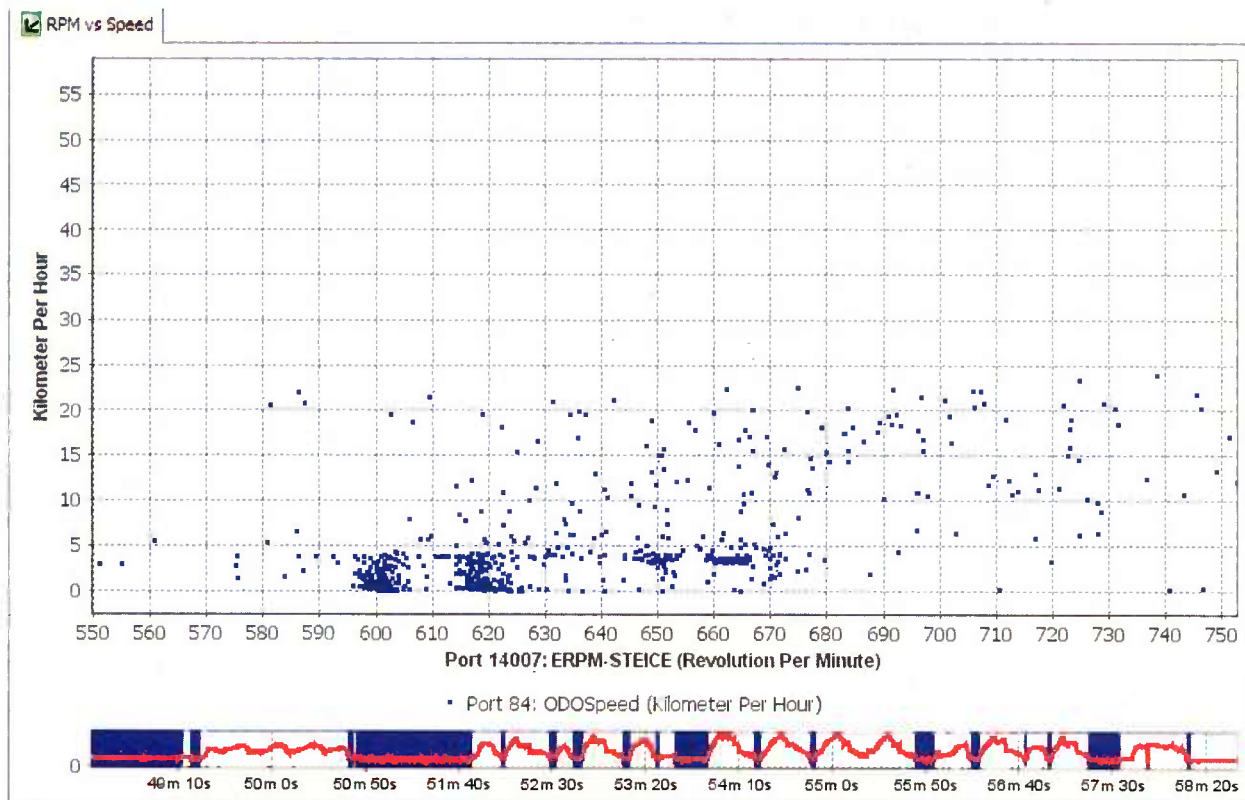


Figure 21. Updated scatter plot with time axis. Zooming on the time axis changes the points displayed. Zooming on X or Y axis updates highlighted regions in the bottom graph. This allows relating points to time in either direction.

Live Mission Data Viewing of Asset Health Management (AHM) Data in the Maintainer's Graphical User Interface (MGUI) Tool

Problem

The Maintainers Graphical User Interface (MGUI) tool was originally intended to be a utilized for post-mission viewing of data collected with the Asset Health Monitoring (AHM) system. However, it was determined that the same tool could be extremely useful for viewing data while the AHM system was collecting data in real-time. This is both useful for a maintainer validating a problem in real-time, as well as a developer validating that a new algorithm function is working properly during simulated missions.

Goals

The goal is to modify MGUI to view missions that are in progress, specifically the ability to update graphs as new data is coming in for the mission. Previously, MGUI could view an ongoing mission, but it would only see data up to the point at which the mission was loaded by MGUI. In order to view newer data, the user would need to exit back to the mission selection screen, select the same mission again, and reload the appropriate graphs for analysis.

Approach

The MGUI code was modified to allow the tool to load new data as it was loaded into the database. Previously, all mission data was loaded when the mission was selected. Thus for an ongoing mission, the data would only load up until the mission was selected and MGUI would never view any of the new data as the database was populated. To do so, the system would query for the new data at a given interval, e.g., once per second. New data was determined based on the last time data was loaded in order to prevent reloading the same data over and over. After new data arrives, the graph is repainted to show the new data. This approach minimizes data loading while providing the user with the most recent data available for evaluation and decision making.

Results

MGUI is now capable of displaying data as it is received into the AHM system. This obviously requires MGUI to be connected to the system running AHM. This feature has been most helpful for validating features and debugging problems with new algorithms under development for the AHM system. This was especially useful as some missions required either running the simulated data to completion or ending the simulation before validating the results. The simulation can now be ended at the point where a problem occurs or the desired result is validated, thus allowing for shortened development times.

Data-Driven, Supervised Anomaly Detection

Problem

Given large amounts of vehicle, machinery or process data, we identify and use storage, access, and analysis techniques for supervised anomaly detection. Effective anomaly detection is the first step in implementing diagnostic and prognostic systems via data processing, data mining, and machine learning. The supervised, data-driven approach shown in Figure 22 is based on our experience with analysis of vehicle failures, in which empirical investigation is impossible due to the expense of commercial or military vehicles and their limited availability. We identify failures and study the data and anomaly detection methods, with the ultimate goal being the creation of failure libraries from which diagnostics and prognostics can be created. ONR can use these results to enhance development of production-quality diagnostic and prognostics. These will simultaneously reduce the costs of maintenance and unexpected failures of vehicles and other machinery.

Goals

1. Investigate use of maintenance data, vehicle Diagnostic Trouble Codes, and Asset Health Monitoring Alerts to obtain ground truths for machinery.
2. Investigate anomaly detection techniques for identified faults.

Approach

Our approach comprises the following steps:

1. Identify candidate failure events;
2. Use visualizations to understand the associated signals;
3. Select and train models that yield an anomaly metric;
4. Compare the performance of the models.

For this work we used vehicle data such as vehicle speeds, pressures, temperatures and position, collected from 9/2009-7/2010 for approximately 100 commercial vehicles (see *Vehicle Data Warehousing and Analytics* for more details). In addition, we had maintenance data for 24 vehicles for the period 1/2010-11/2010. We examined the maintenance records by reading the details to find failure events relevant to the engine, transmission, and braking systems. We found a single failure with sufficient data to analyze with this method. We also used vehicle diagnostic trouble code information to find events.

Through this process we confirmed the event identified by reading the maintenance records and found a series of similar events for a vehicle for which we had no maintenance records. The signal visualizations provided the information we needed to prepare the data for building the models and to understand the vehicle processes. We selected models from numerical analysis and machine learning fields. We selected classification models, which are purely data-driven and require no engineering insight, as well as regression models for which some engineering insight is required. With this approach, we trained several models, compared the results, and were able to identify a difference in performance for the two model types.

Results

Two general classes of oil pressure anomaly detection methods are investigated: regression-based methods, where the oil pressure is predicted from other recorded vehicle data, and the residual (prediction error) is compared to thresholds; and classification-based, in which the vehicle signals are classified as anomalous or not for each time period. The regression-based methods we investigated are: Gridded Residual, Boosted Regression Tree, and Feed-Forward Neural Networks. The classification-based methods we investigated are: Gaussian Mixture Models (GMM) and Replicator Neural Networks. We found that the regression-based detection methods performed better than the classification-based detection methods.

Because there is a single fault, we could not use the normal receiver operating characteristic method to measure quality of our anomaly detection methods. We originated a detection-quality estimation method to overcome this, which sets a detection threshold high enough to guarantee zero false alarms, then estimates detection quality using the detection horizon interval. Figure 23 shows that the oil pressure anomaly was detected hours before the vehicle diagnostic trouble code became active.

While using the GMM classification algorithms, we found some unexplained inconsistencies between the model training quality and the anomaly detection performance. After investigating

the discrepancy, we originated a useful regularization criterion to prevent use of such discrepant GMMs.

These results are documented in a paper, *Case study: Models for detecting low oil pressure anomalies on commercial vehicles*, which is being submitted for publication.

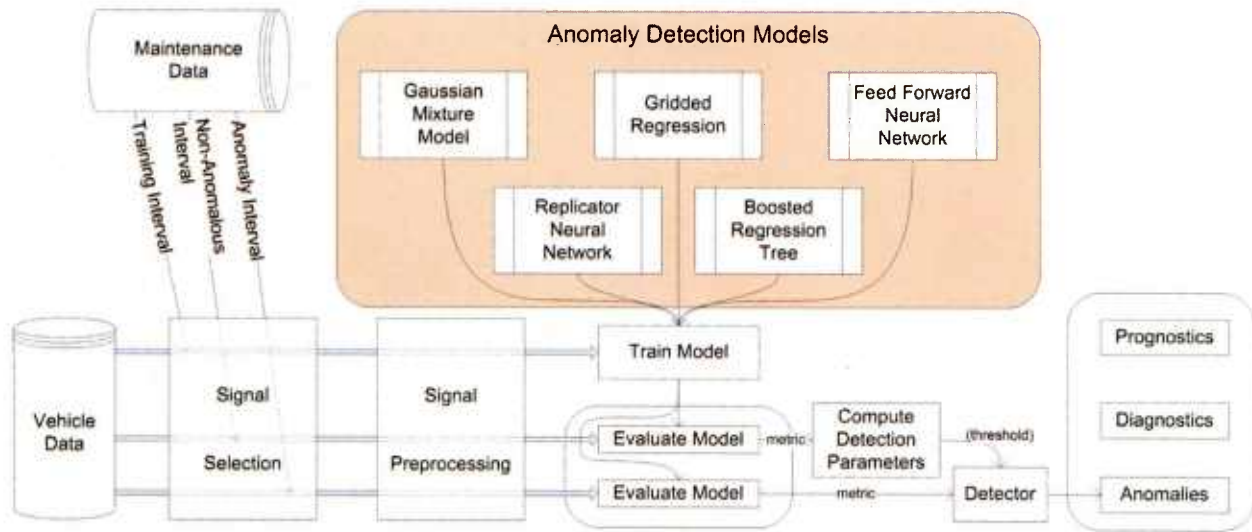


Figure 22. Data-driven anomaly detection block diagram.

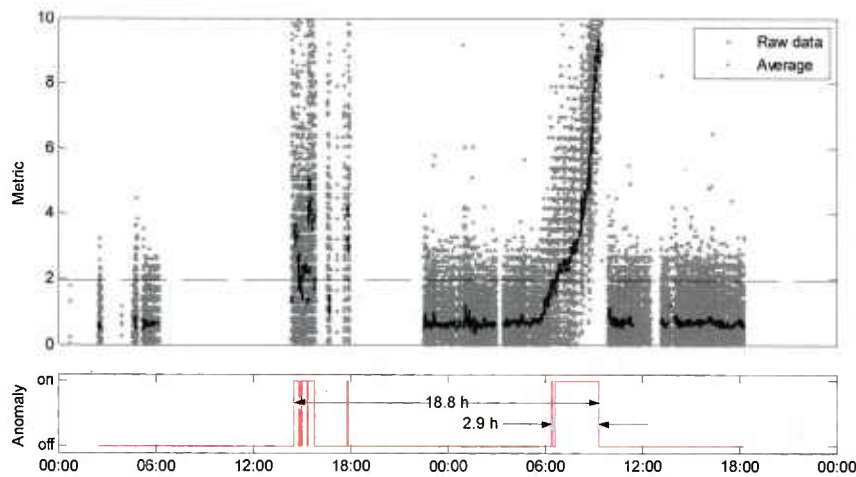


Figure 23. Anomaly detection performance for the Gridded Residual method. The top graph shows instantaneous and average absolute value of the residual (in units of standard deviations), as well as the detection threshold. The lower graph shows the detected anomalies and the detection horizon.

Functional Relationship among Signals for Anomaly Detection and Virtual Sensing

Problem

Data-driven anomaly detection, diagnostics, and prognostics algorithms are expensive to design and implement. When they are added to an existing system, without the benefit of the original engineering specification, the cost to determine the relevant signals and their relationships through engineering analysis can be high. However, understanding the relationships is critical to reduce the complexity of the problem, identifying the most important signals for each algorithm, and simplifying the system by ignoring unimportant signals. Signals' functional relationships also provide a framework for sensor fault detection so that anomaly detection, diagnostic, and prognostic algorithms can ignore unreliable signals rather than malfunction. The problem addressed by this effort is to automatically identify the functional relationships among a large group of signals, allowing an anomaly detection algorithm to be created with reduced engineering effort.

Goals

Create an automated method (or methods) to identify small groups of related signals, given a large group of related and unrelated signals. In this problem, n signals are related when $n-1$ of them can be used to predict the other signal. For example, the engine oil temperature and engine coolant temperature signals exhibit a [mutual] functional dependence – each is a reasonable predictor of the other because of the physics of heat conduction in the engine subsystems. On the other hand, atmospheric pressure, air temperature, and RPM would not be expected to have a functional dependence because weather, season, and climate factors drive pressure and temperature, and RPM depends on vehicle speed, terrain, gear selection, pedal position, engine power, etc.

Approach

Our initial survey of the data, as well as our anomaly detection results, indicated that an approach utilizing exhaustive search would be fast enough for an initial trial. Based on this, our approach is:

1. Identify the target set of signals S .
2. Determine the maximum size n of the subsets $\mathcal{T}_i$ of S to test;
3. Step through member of $\mathcal{T}$, training a regressor to predict one of the signals from the other $n-1$ signals;
4. Evaluate the performance of each regressor from step 3; and,
5. Select the best regressors, based on the evaluations of step 4.

Results

Using $n = 3$, a set of 13 signals (positions, rates, torques, and temperatures), data from one vehicle, two weeks of training data, and the gridded residual regressor, the model of normal

distributions over predefined subdomains, we found all 858 regressions and their performance. We used the gridded residual regressor because of its good performance in the anomaly detection work. The results are illustrated in Figure 24. The residual (histogram in the upper-right graph) is the difference between the actual value and the predicted value. For this set of signals on this vehicle's data, about 30% of the derived models have a standard deviation that is at most 10% of typical values of the predicted signal value. The data also confirms that the signals chosen for the oil pressure anomaly detection work (oil temperature and engine speed) are the best choices for creating an oil pressure anomaly detector based on two other signals. This particular regressor can be trained at a rate of about 20 per minute. To scan all useful sets of 3 signals out of 100 requires building almost 500,000 models; this is within the reach of even small organizations.

For higher dimensionality anomaly detection problems, other quickly-trained regressors such as the boosted regression tree, might be preferable. The gridded residual is not as well suited for functions of four or five variables because the training data would be sparsely distributed within the bins.

The observed modeling and training performance show that this is a feasible method for automatically screening signals to determine functional relationships that could be further investigated for use in anomaly detection, diagnostics, or prognostics.

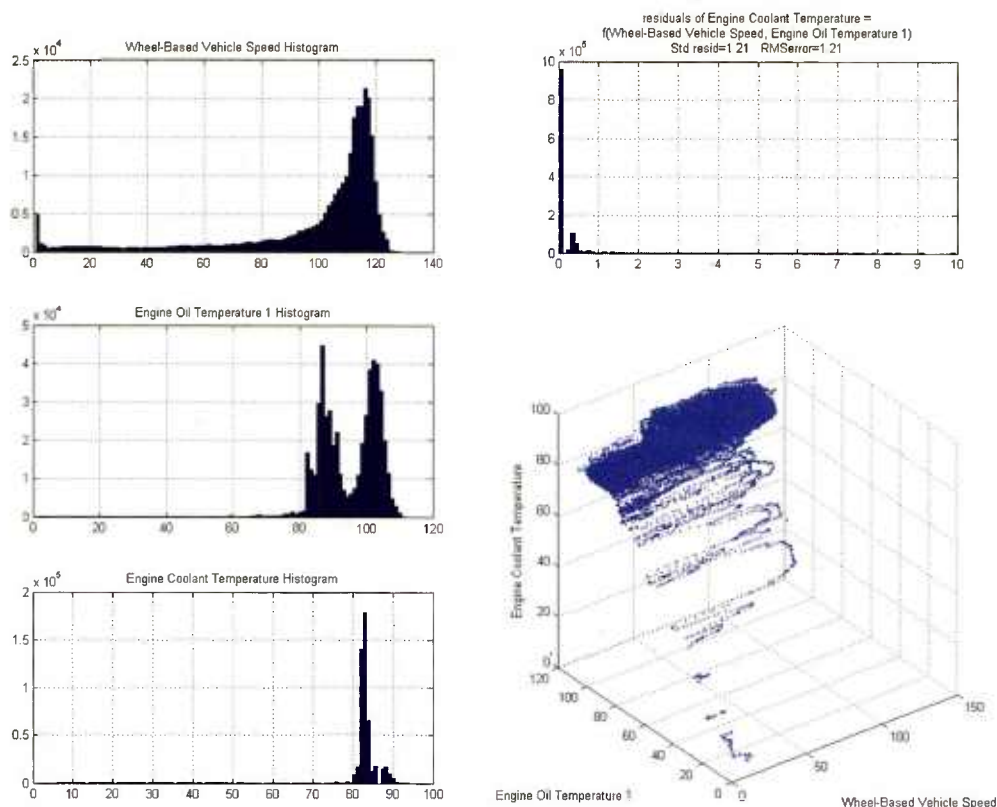


Figure 24. Illustration of prediction of engine coolant temperature, based on vehicle speed and oil temperature. The charts on the left show histograms of the speed and each temperature. The histogram on the upper-right shows the prediction error histogram (in units of standard deviations). At the lower-right is a graphical visualization of the relationship between the signals.

Multi-sensor Fault Detection

Problem

Measurement reliability greatly affects the *health monitoring system* (HMS) of the critical components of a vehicle's electrical system (battery, alternator, and starter). *Sensor fault detection* (SFD) is a subsystem of HMS responsible for detecting unreliable signals. In addition to three typical categorical health states – *green* (OK), *yellow* (needs attention), and *red* (deadlined) – an SFD-equipped HMS includes a *black* (unobservable) state. There are two types of SFD:

1. Single-sensor fault detection (SSFD) considers signals individually and assesses the signal reliability based on the observed values (in-range vs. out-of-range), rates of change, autocorrelations, and other metrics.
2. Multi-sensor fault detection (MSFD) considers a group of redundant or physically related signals and assesses the reliability of individual signals based on their consistency with the other signals within the group.

Goals

To achieve a comprehensive and powerful SFD, we considered and identified general approaches for MSFD of wide applicability and then customized for selected applications in ground vehicles.

Approach

Two approaches are examined. When the measurements are statistically correlated, or more generally, a functional relationship among the measured signals exists, a correlation-based algorithm is a good choice for MSFD. We applied this approach for ensuring signal integrity of battery voltage V_{Batt} , alternator voltage V_{Alt} , and starter-motor voltage. High cross-correlations among these signals allow reliable MSFD, and multiple comparisons of individual cross-correlations are used to determine the compromised signal. The correlations were computed sequentially as the data became available. This approach is not limited to the electrical signals; it can be used whenever the measurements are sufficiently correlated.

When the measurements are not sufficiently correlated but a number of probabilistic relationships exist, the Dempster-Shafer algorithm was found to be effective. An MSFD based on Dempster-Shafer requires rule definition, setting up an inference engine, and verification.

Goals

The objective of this study is to produce a case study for electronics prognostics. More specifically, the goal is to induce a realistic failure to a commonly used circuit, which was chosen to be a Switched-Mode Power Supply (SWMPS), analyze the data, and assess the opportunities to detect the failure before the device stops performing its intended function, i.e., providing DC power.

Approach

Because electronics components can operate in normal operating conditions for many years, inducing failures is based on accelerated aging using the Highly Accelerated Life Test (HALT). Figure 27 depicts the basic principle behind the HALT: the components are characterized by their *strength* with respect to different stressors, or *loads* (for electronics components these loads are temperature, vibration, and over-voltage). In normal operation the components are designed so that the load distribution is well-separated from the strength distribution, so that overlap, which corresponds to failures, is low (see Figure 27a). As the components age, their strength distribution widens and shifts closer to the load distribution, increasing the overlap between the two distributions (see Figure 27b). HALT, on the other hand, shifts the load distribution towards the strength distribution to accelerate failures of new components.

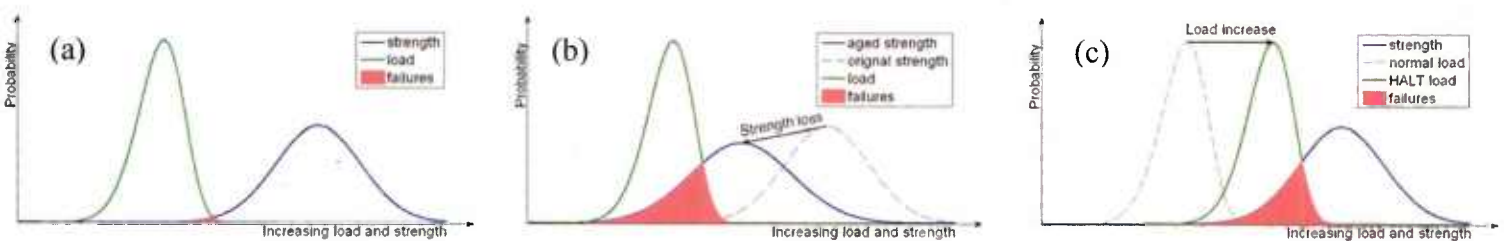


Figure 27. Distributions of strength and load. The overlap signifies failures. (a) Normal operation; (b) Aging; (c) HALT.

Figure 28 shows the circuit diagram of the SMPS, a photograph of the circuit, and a sketch of the loading in three-dimensional space: temperature, voltage and vibration.

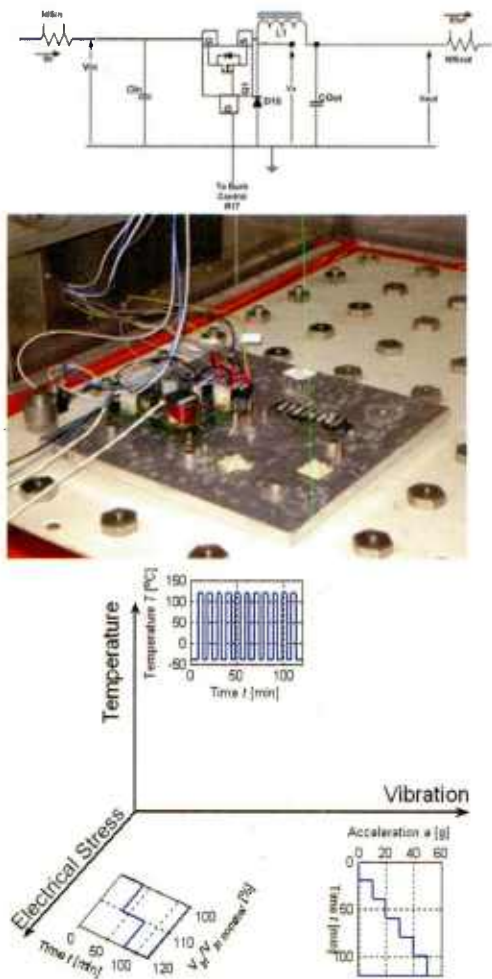


Figure 28. SMPS:
circuit diagram;
photograph of the
device mounted inside
the HALT chamber;
load profiles.

Automated controls for the HALT chamber and a separate, multi-resolution data collection system have been developed. The slow-changing variables, such as temperature, and average output voltage were saved continuously, while the waveforms of the high-frequency signals are saved in bursts. In addition, custom data visualization and analysis tools have been developed. A strong negative correlation between temperature and output voltage have been established (Pearson correlation coefficient ~ -1). To enable real-time monitoring, a window for the computation of the correlation coefficient was selected.

Results

The part was driven to failure while operating at an elevated temperature. The real-time correlation detected the failure in progress approximately 30 minutes before the part failed to produce desired output voltage, as shown in Figure 29.

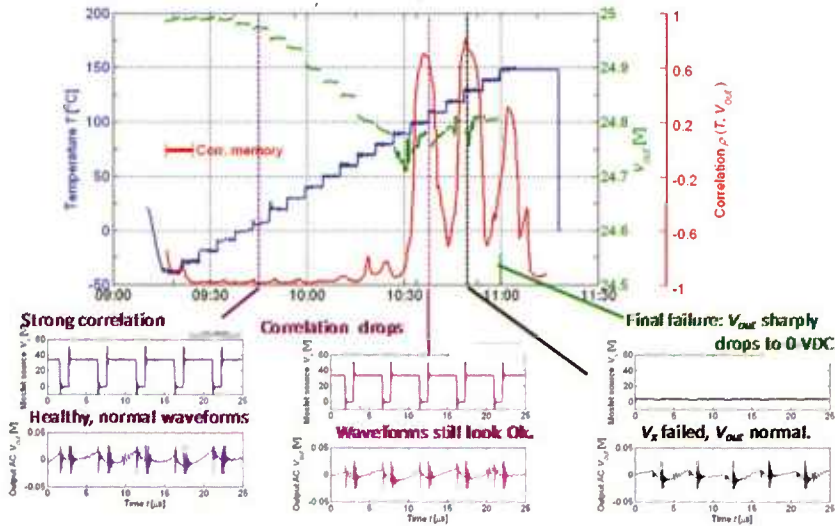


Figure 29. Temperature, output voltage, and their correlation coefficient are shown in the main plot (the color of traces corresponds to the color of the axes). The characteristic waveforms are shown in separate plots.

Vehicle Data Warehousing and Analytics

Problem

When vehicle signal processing involves several signals, such as engine RPM, oil temperature, and oil pressure, and the signals are stored in the MIMOSA schema, or the similarly designed EPLS database, the asynchrony and the data schema complicate the processing. Alternate data schema could make processing more convenient and faster.

Goal

Investigate the application of typical data warehouse schema to asynchronous, sparse, vehicle data, highlighting differences in space required, and assess software performance and complexity.

Approach

Our approach comprises the following four steps:

1. Select one of the best-practice data warehouse schema patterns. Data warehouse schema influence storage requirements; more normalized schema save space and provide improved access time, but required table joins increase the complexity of the access software and may require more resources in the database server.
2. Develop a schema instance for our vehicle data. The table designs specify the intra- and inter-table constraints and data storage details. The snowflake data warehouse schema was selected for storage because of its highly orthogonal structure and compatibility with existing business intelligence processes. The dimensions we defined include: data collection system, driver, trip, fleet operating company, distance driven, time driven, speed range, time, and vehicle. This permits the use of analytics software to view, for example, fuel consumption vs. speed range and miles-on-vehicle. The fact tables include engine and vehicle signals that are recorded at 1- and 10-second intervals.

3. Try to create the data warehouse efficiently. The performance of this step sets a limit on how many vehicles' data can be stored in the data warehouse. The data warehouse is populated from the relational data store through with an Extract-Transform-Load (ETL) process. For our ETL system, we had to compensate for these characteristics of the source table: (a) values are stored in the source table with one sample of one signal per row; (b) values for a particular signal are sparse – they are not stored unless they are sufficiently different from the latest stored value; (c) signal values' time stamps are not synchronized. This required that our Transform step, in ETL, must read each signal's [time, value] pair from the source database, resample the values to match the times in the facts tables, filter the resampled stream to avoid aliasing, collect the values for the given time, and insert the record into the fact table. Our 1- s fact table has 28 signals. The 10- s fact table has 11 signals. Some signals that are commonly used in analysis are calculated during ETL. For example, the engine power is pre-calculated from engine speed and torque.
4. Experiment with data access and analysis to determine usability. This involves comparing the software accessing the un-normalized, asynchronous, and sparse data with software using the data warehouse

Results

The snowflake schema pattern is suitable for vehicle data warehouses. Our ETL code is a hybrid of Microsoft SQL Server Integration Services (SSIS) and Matlab, because implementation of the synchronization, resampling, and filtering was not suited to SSIS. The database transfers and computations process about 600 seconds of vehicle data per second, after accounting for fixed start-up times (on a server with dual 3 GHz 64-bit processors, 12 GB of memory, and 1.8 TB RAID-5 storage).

Fact table access for data analysis is simplified because a single query returns the synchronized and filtered data. This replaces software to input, resample, and synchronize the raw signal data. Query performance was 19 microseconds per row from the relational data store, and 7.9 microseconds per row from the data warehouse, when each returned row comprised the time and three signal values. Figure 30 shows sample signals from the data warehouse. The raw data (not shown) is much noisier.

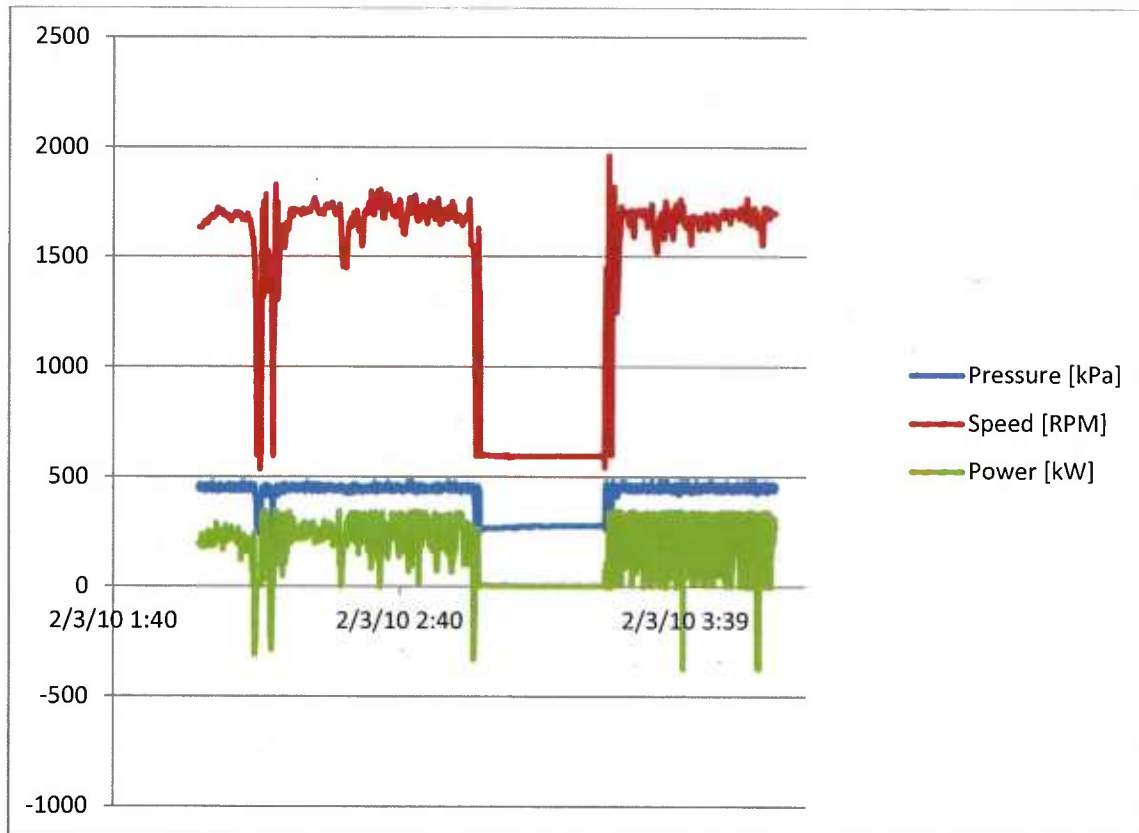


Figure 30. Engine oil, Pressure, Speed, and Power from fact table. Negative power is due to engine braking, coincident with reduced engine RPM.

Fuel Usage Analysis

Problem

Vehicle condition and driver behavior are the two controllable factors that affect vehicle fuel usage. On the other hand, vehicle and engine manufacturer and model, vehicle loading, wind direction and speed, and wheel rolling resistance also affect fuel usage. The effects of these factors must be included in analysis so that low or high fuel usage can be assigned to the correct cause. For example, a fuel use rate of 6 miles per gallon in a commercial tractor-trailer truck could be from a carefully driven vehicle in excellent shape which happens to be fully loaded and driving into a strong wind. The same usage on a lightly loaded vehicle could indicate poor condition or overly aggressive driving.

Goals

Our goals are to investigate models of mechanics of vehicle motion, to apply the models to stored vehicle data to determine model parameters, and to analyze fuel use in different contexts. Since vehicle forward motion is well modeled by Newtonian mechanics, our goal is to identify and select a model, and use it to analyze fuel use. The components of these models include speed, acceleration, aerodynamic resistance, rolling resistance, engine output, and braking

effects. Simple models may fail to account for critical signals, while overly complex models may require signals that are unavailable or are only approximately known. Specifically, the vehicle mass directly affects the energy required to overcome rolling resistance for initial acceleration and for gaining elevation.

Approach

Our approach comprises the following five steps:

1. Develop a model of vehicle energy production (based on engine power) and energy used to accelerate the vehicle, to overcome aerodynamic drag and rolling resistance, and to gain or lose elevation;
2. Develop a mechanism to segment trips (where there is no significant change to vehicle load) into short (e.g., 30-second) segments. The segments when wheel brakes are used are discarded, as vehicles provide insufficient braking data and segment boundaries are adjusted so the vehicle is in motion throughout each segment;
3. Obtain a best (least-squares) solution for the estimated vehicle mass, as well as aerodynamic drag and rolling resistance coefficients;
4. Analyze several trips for several vehicles to determine the performance of this method;
5. Follow up with a feasibility investigation of the use of a machine learning model to predict fuel consumption based on other engine sensor signals.

Results

The model was based on Newtonian mechanics and energy conservation, with least-square error minimization used to extract model parameters. We analyzed data for five vehicles, including wind direction and speed data for some trips to allow for more accurate estimation. Notably, we compared estimates over segments of a trip in which the vehicles were continuously moving, thus precluding any significant changes in the vehicle load. For that vehicle (covered in Figure 31), the mean mass estimate is 22.6 tons and the standard deviation is 3.3 tons – about 15% of the mean. In the second half of the trip, the mean is also 22.6 tons, while the standard deviation is 2.4 tons (about 11% of the mean). The vehicle cargo weight is available for two trips of the same vehicle, with the resulting estimated vehicle empty weight differing by 8400 kg between the two. However, the wind was westerly during one of the trips, and easterly during the other. This highlights the uncertainty of mass estimates without accurate wind speed and direction data. Because some critical data is unknown, the ultimate performance of this method cannot be completely assessed. Although it appears to estimate vehicle mass consistently over a trip, the differences between trips indicate that estimation accuracy is unacceptably low without further data. Results for the coefficients of aerodynamic drag and rolling resistance are similar to those for the vehicle mass, supporting our conclusion that precise modeling is possible only with sufficient data availability.

As an extension effort, we studied the computational predictability of fuel consumption given a number of on-board sensor readings. Figure 32 shows instantaneous engine fuel rate prediction

from engine speed and percent torque demanded. Boosted Regression Tree models were able to model consumption efficiently, producing residuals for 10 long-range freight trucks (30+ deliveries each) with near-zero mean and standard deviation of at most 8 liters (2.1 gallons) per hour. Furthermore, we were able to train accurate Support Vector Machines from this same data to differentiate between engine manufacturers and designs, allowing for increased effectiveness of fuel predictive models in application (via manufacturer-specific model selection).

By combining these estimates, it is possible to calculate higher level usage estimates, such as the amount of fuel used to move one ton one mile, or to visualize the relationship of fuel required to move one ton one mile and speed, to determine the most economical operating speed for a particular load.

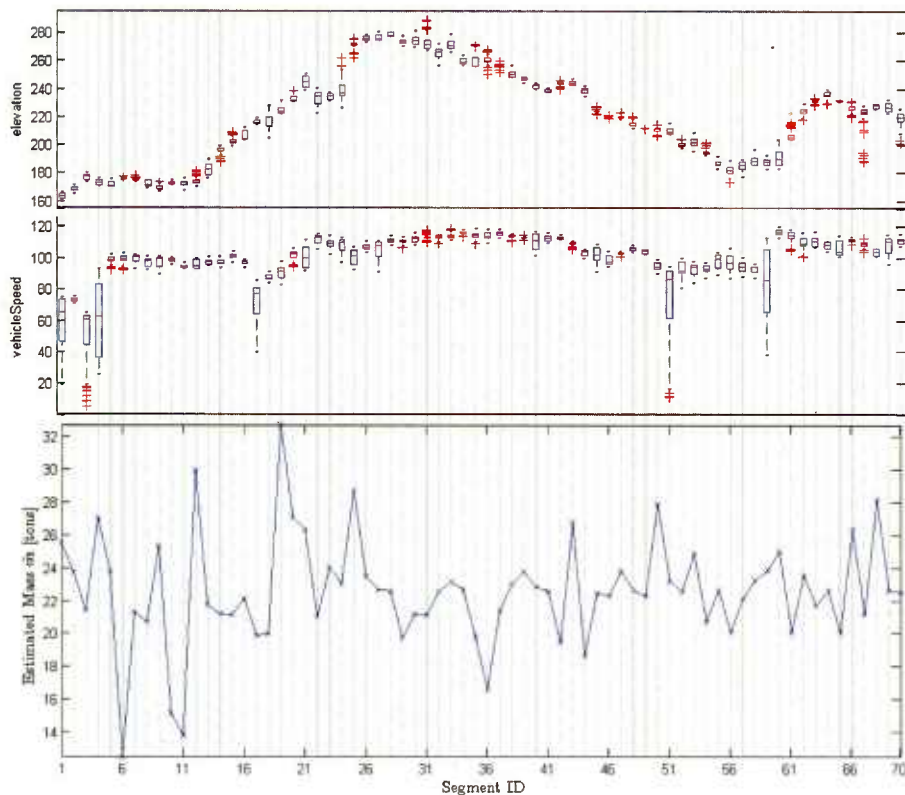


Figure 31. Vehicle Elevation, Speed, and Estimated Mass. These are computed for discrete motion segments in a single trip.

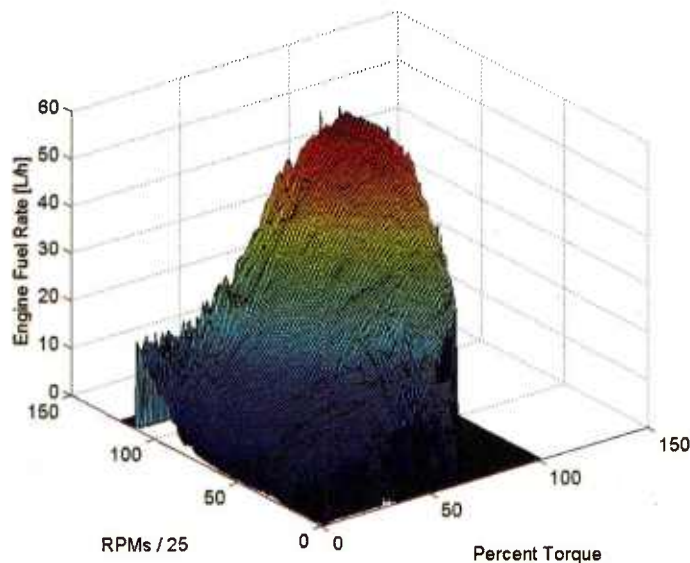


Figure 32. Predicting Fuel Consumption from On-Board Sensor Readings.

Solid Oxide Fuel Cell Diagnostics

Problem

Fuel cell auxiliary power systems are effective at assisting in efficient force deployment, but must be robust in operating conditions to guarantee reliability in the field. Under separate funding, we studied the forecasting of one particular failure mode (reactant ignition) in solid oxide fuel cell systems using sequential Monte Carlo simulation and vector quantization⁴. Despite our successes, detection rates failed to converge to 100%, especially when constrained to low (< 5%) false alarms accepted.

Goals

In order to rectify this situation, we worked to identify, implement, and validate more advanced classification methods that outperform particle filtering in forecasting solid oxide fuel cell reactant ignition. Preference was given to methods also applicable to other domains.

Approach

We studied methods including (but not limited to): Feed-Forward Neural Networks (FFNNs), Gaussian Mixture Models (GMMs), Infinite Gaussian Models, Gridded Regression Models, Kernel Support Vector Machines, Structural Support Vector Machines, Boosted Regression Trees, Gaussian Process Regression, Minimax Classifiers, and Additive Groves. Performance was evaluated in terms of detection and false alarm rates, with training and testing performed on data drawn from both fuel cell test systems.

⁴ Ardis, P.A., Nenadic, G.N., Walluk, M.R., Smith, D.F.; "Forecasting Reactant Ignition in Solid Oxide Fuel Cell Systems", ASME 10th International Fuel Cell Science, Engineering & Technology Conference; San Diego, CA, USA; 2012.

Results

The majority of studied methods outperformed the baseline but retained a significant number of missed detections (see Figure 33). However, the last four listed all performed at near-perfect levels (see Figure 34) and should be used as solutions for forecasting in the field. Furthermore, we studied the use of Artificial Prediction Markets to fuse the results of multiple classifiers, but do not recommend their use as they did not produce consistent improvements at all operating points and increased training time in comparison to other fusion methods.

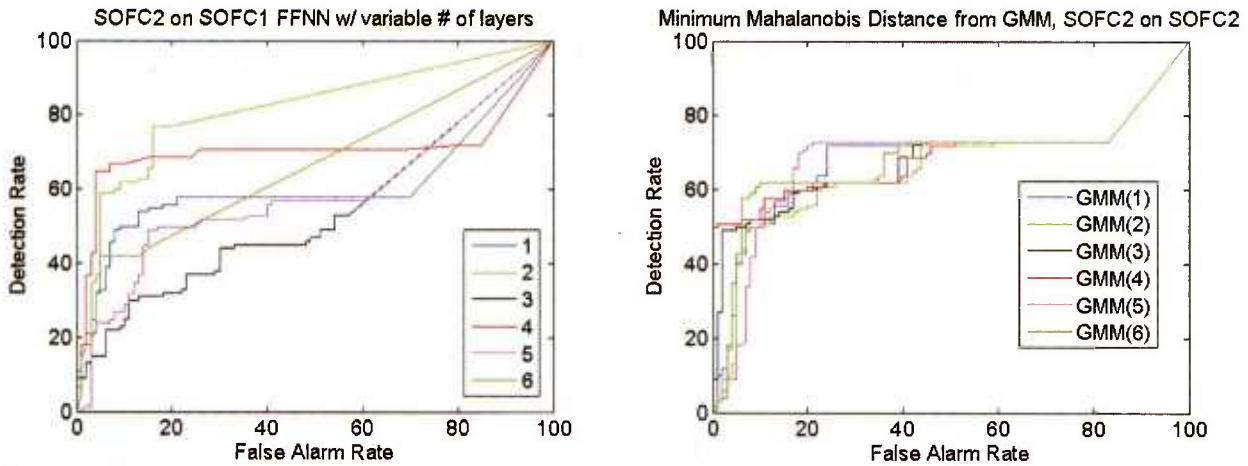


Figure 33. Typical Receiver Operating Characteristic (ROC) Curves.

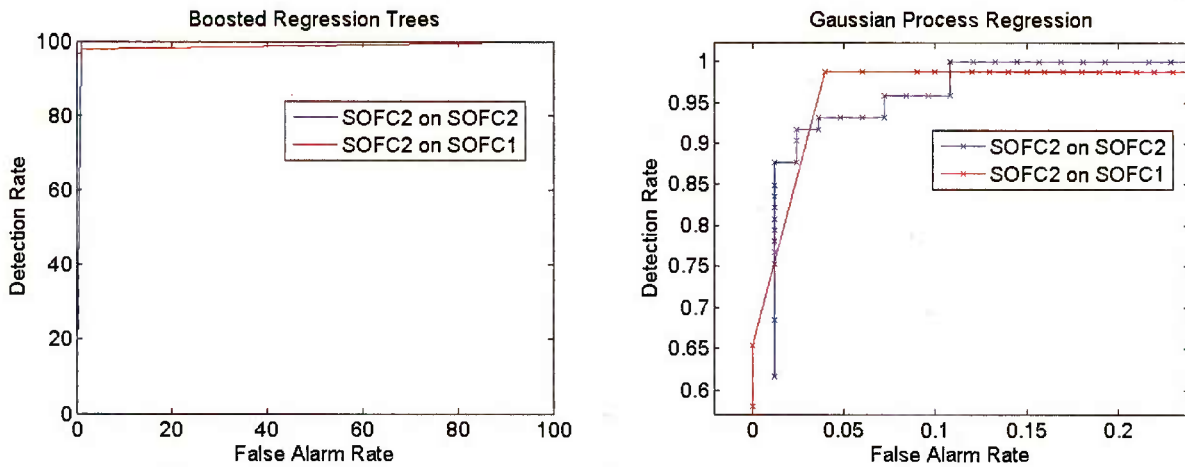


Figure 34. ROC Curves of Top Performing Methods (Right: Zoom-In).

System Resilience & Survivability



7. System Resilience and Survivability

7.1 Description of System Resilience and Survivability

System resilience is represented by system performance degradation in response to disturbance or hazard events, and the ability to effectively and quickly restore performance/function after the event. Hazard events may be failures within the system or may be an external attack or a series of attacks. More resilient systems can be achieved by reducing system vulnerabilities and minimizing disruption when hazards occur as well as through increasing the system adaptability and ability to recover to a stable state after disruption. Survivability is associated with the level of system vulnerability and is a subset of resilience.

Prior to undertaking any work in this specialized research area, CIMS determined it would require a reliable source of real-world data to utilize in the research, to benchmark potential results against, and to help ensure that our R&D efforts would be relevant to DoD needs. Unfortunately, and despite numerous inquiries, we were unable to locate a military partner both willing and able to provide the necessary real-world platform data. Accordingly, CIMS expended no budget in system resilience and survivability.