7

A Novel Configuration-Driven Data Mining Framework for Health and Usage Monitoring Systems

David He[1], Eric Bechhoefer[2], Mohammed Al-Kateb[2], Jinghua Ma[1], Pradnya Joshi[1] and Mahindra Imadabathuni[1]
[1]*The University of Illinois at Chicago*
[2]*Goodrich Sensors and Integrated Systems*
USA

1. Introduction

Health and Usage Monitoring Systems (HUMS) and Condition Based Maintenance (CBM) systems are closely related systems since the data collected by HUMS can be effectively used by CBM systems to generate condition and health indicators of air-crafts in order to find out the components on which maintenance is required. Efficient and comprehensive automation of such CBM systems is, therefore, of utmost importance to take an advantage of the massive amount of data continuously collected by HUMS.

In this chapter, we present a novel framework for automating the CBM systems, based on supervised learning practices established in UH-60 Condition Based Maintenance (CBM) manual [3]. The proposed framework is based upon a concept of building a configuration-driven data mining tool in which the maintenance decisions of aircrafts' components are driven by configuration metadata. We seek to address a key design goal of building a generic framework that can be easily instantiated for various CBM applications. This design goal, in turn, raises the challenges of building a system that features modularity (i.e., the software structure is based on a composition of separate modules, which jointly incorporate with each other.), extendibility (i.e., the software should be easy to extend.), and maintainability (i.e., the software should be easy to maintain and update.).

To meet this design goal, we adopted a layered architecture for the proposed framework. The framework consists of three layers; 1) Storage layer, which concerns the storage of source data and configuration metadata used by the system; 2) Extraction layer, in which the source data and configuration metadata are extracted and passed to the upper (processing) layer; and 3) Processing layer, which is responsible for executing mining algorithms and reporting necessary maintenance actions.

This layered architecture uses two prominent techniques, namely XML-based metadata storage, and dynamic code generation and execution. The XML-based metadata storage provides a generic platform for storing configuration meta-data, whereas dynamic code generation and execution allows applications to be extended with non-compiled source code, which is stored within configuration metadata. Both the generic storage platform and the on-the-fly code generation features help significantly reduce the cost of the software

development life cycle (including software validation, verification, and maintenance), by allowing system engineers to supplement new functions, modify existing algorithms, consider new data sets, and conduct advanced analysis operations, without having to issue a new software release.

We have developed a functional software prototype of the proposed framework and examined the utility of the prototype to retrieve configuration metadata and, consequently, generate various maintenance reports. We will present examples of the reports generated by the software prototype, and an estimate of the time saved over the manual system. We will also demonstrate the exceedances report that shall list all simple exceedances, the associated count, and duration of those exceedances, as well as fault BIT report that shall list all BIT failures and their total instances. The software prototype provides the user with advanced and simplified modes for generating reports. The advanced mode gives the user a control on the report generation to specify, for instance, the type of aircrafts and analysis, the source of data and configuration files, the information to be displayed and hidden in the report, etc. The simplified, on the other hand, generates, in one single execution, all reports defined in the configuration files and using the default system settings.

We present two examples of the reports generated by the software prototype for the S-92A; the first example is for the fault BIT report that lists all BIT failures and their total instances, and the second example is for the exceedances [4, 5] report that lists all simple exceedances and their associated count. (See [6] for comprehensive list and details of the set of reports generated by the system).

In the research work presented in this chapter, we make the following contributions: 1) We address the motivation and demand for an automation of the HUMS CBM systems. 2) We present a novel framework for building Automated Condition Based Maintenance Checking Systems (ACBMCS). 3) We demonstrate the software prototype and explore the steps of generating various maintenance reports, in light of fault BIT and exceedances reports.

The remainder of this chapter is organized as follows. First, we outline the system and software requirement specifications. Second, we present the framework architecture. Third, we discuss in details the design of XML-based configuration. Fourth, we demonstrate an example of using the software prototype to generate maintainable reports. Finally, we conclude this chapter.

2. Requirement specifications

In this section we outline the system and software requirement specifications, which identify the end-users needs and expectations from the software application.

2.1 System requirements

The proposed framework has been designed to accommodate the following system requirements:

- The system should be configurable across various types of aircrafts.
- The system should perform different types of analysis to determine the health conditions.
- The system should process all the parameters required by analysis type and aircraft type.
- The system should configure the application algorithms based on the type of aircraft and the type of analysis being performed.

- The system should process all types of data used to monitor the various aircraft state parameters.
- The system should display the report to the user based on the algorithm executed.

2.2 Software requirements

The primary requirement for the software is to be generic. This generality has been address by building the software with the following features: 1) Modularity: The software structure is based on a composition of separate modules, which incorporate with each other through interfaces. 2) Integrity: Data, information, and knowledge stored and manipulated by the software must be correct and the relationship between them is consistent. 3) Extendibility and maintainability: The software should be easy to extend and maintain.

3. Framework architecture

The framework architecture is illustrated in Figure 1, which demonstrates that the software is composed of three connected layers; 1) Storage layer, 2) Extraction layer, and 3) Processing layer.

The storage layer concerns the storage of configuration metadata that drives the processing of the systems, as well as source data on which analysis is being done.

The extraction layer is the layer in which the source data and configuration metadata are extracted and passed to the upper (processing) layer.

The processing layer is responsible for executing the mining algorithms, based on the configuration metadata, against the corresponding data retrieved from source data files.

3.1 Storage layer

The storage layer is the lower lever layer which stores configuration metadata that drives the processing of the systems, as well as source data on which analysis is being done.

The configuration metadata (i.e., information representing aircrafts, analysis, parameters, algorithms, and actions report) is stored as a collection of XML documents. These XML documents are designed in a normalized way in order to reduce, or possibly remove, any information redundancy. To assure the integrity of XML documents content, these XML document are built according to an XML schema that expresses constraints on the structure and content of the XML documents. In addition, the XML schema employs the concept of archetypes which provides means of defining user-defined data types in the XML schema in a way that reduces nested definitions, as well as the concept of restrictions which defines acceptable values for XML elements or attributes. The design details of XML-based configuration are further discussed in the following section.

The source data, on the other hand, comes in the form of Raw Data Files (RDF) and Activity Data Files (ADF). An RDF is a collection of aircraft health data for a single aircraft operation, whereas an ADF is a set of indexes, to an RDF file, that provides a performance-optimized approach of retrieving source data of a single aircraft operation.

3.2 Extraction layer

This layer carries out the extraction of data from the source files. This data extraction is derived by the content of configuration metadata.

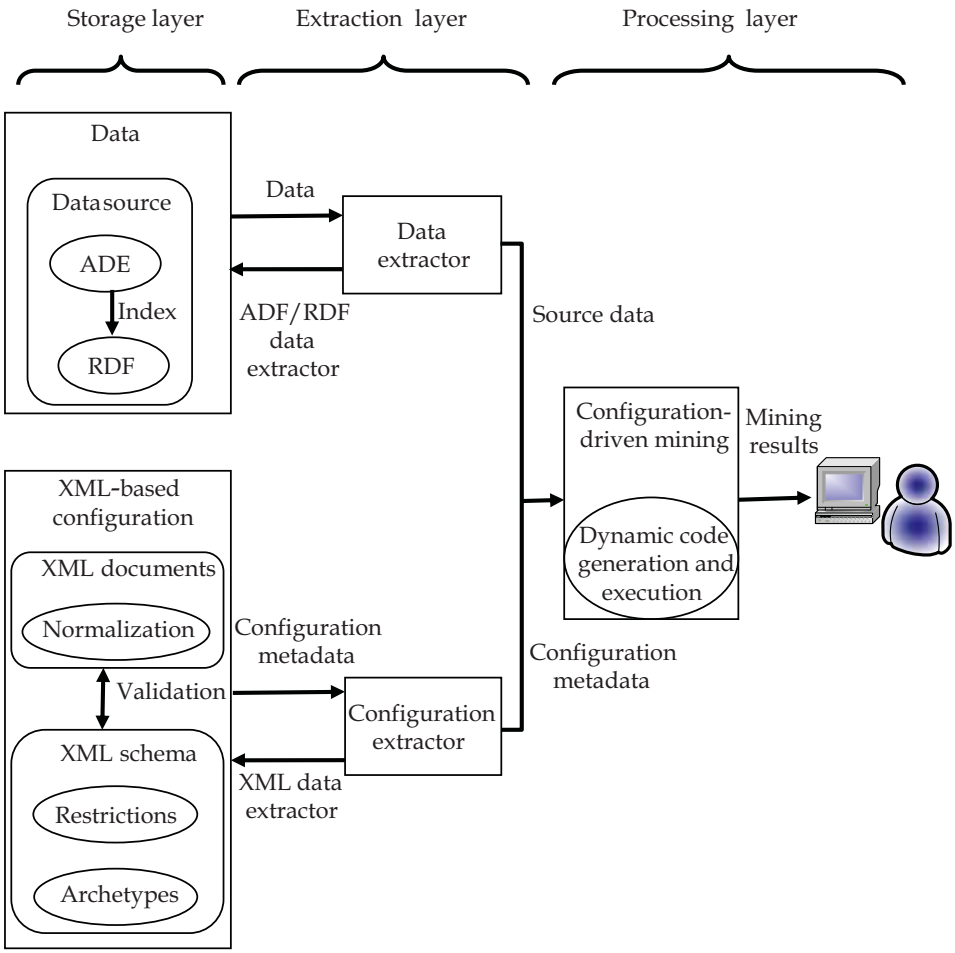


Fig. 1. Framework architecture

3.3 Processing layer

The processing layer is responsible for executing the mining algorithms, based on the configuration metadata, against the corresponding data retrieved from source data files. This layer uses dynamic code generation and execution technique to load algorithms, which should be executed, on the run-time from configuration metadata. The dynamic code generation and execution technique is a key building block of the proposed framework that allows the software to dynamically; 1) Wrap the code into fully-functional assembly source code (with all the required dependencies); 2) Compile the source code into an assembly; and 3) Use the assembly reference to create an instance of the application object.

4. XML-based configuration

The XML elements in the XML-based configuration and their interaction are shown in Figure 2.

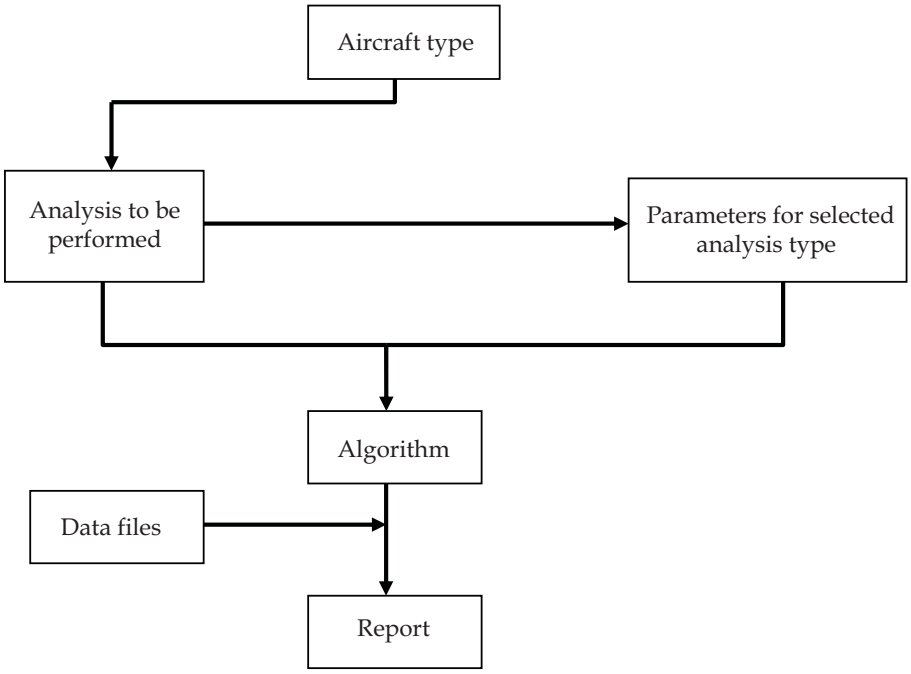


Fig. 2. XML elements in XML schema

The XML-based configuration design involves design of the XML schema and the XML documents holding the configuration information. The XML schema includes information regarding the organization of the elements, sub-elements and attributes which is in accordance with the structure of the system. The different restrictions are assigned on each of the elements, sub-elements and attributes to ensure data integrity. The XML documents, which contain the actual configuration information, are then validated against the schema to check for discrepancies and insure data integrity.

4.1 XML schema elements

XML schema of configuration files models five key elements illustrated in Figure 2:

- AIRCRAFT: The aircraft element consists of the main element Aircrafts in which all possible aircraft types can be defined. The sub-element Aircraft, of the main element Aircrafts, gives the details of one type of aircraft. Each aircraft is given a specific ID, defined as an attribute called airID, to distinguish it from other aircrafts and help in referencing. The sub-element airname of Aircraft contains the information of the name of the aircraft. Figure 3 shows the structure of this element.

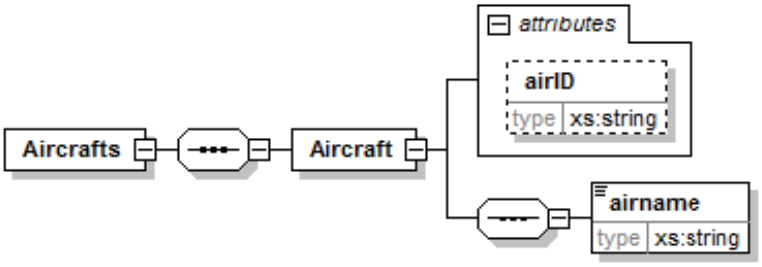


Fig. 3. Structure of the AIRCRAFT element

- ANALYSIS: The analysis element, as illustrated in Figure 4, consists of the main element AnalysisSets, which defines the set of all possible types of analysis that can be performed in CBM. The element AnalysisSets contains a sub-element Analysis, which carries the information regarding the analysis type. The sub-element name of Analysis gives the name of the analysis, and the attribute anaID assigns a specific ID to each analysis type.

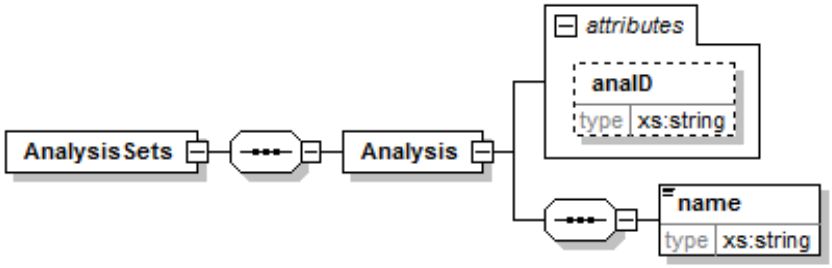


Fig. 4. Structure of the ANALYSIS element

- PARAMETER: The parameter element consists of the main element Parameters. As shown in Figure 5, it has a sub-element Parameter which holds the information of every parameter type. The attribute paramID assigns a unique ID to each parameter type which is used to call it by reference later. The sub-element paramname gives the name of the parameter and usedAt assigns the parameter to analyses and aircraft types at which the parameter is used. The sub-element assignment of usedAt holds this information in the attributes analysisID and aircraftID.
- ALGORITHM: Figure 6 demonstrates that the algorithm element consists of the main element algorithms, which has a sub-element algorithm in which every type of possible algorithm is defined. The attribute algoID assigns a unique ID to each algorithm type that is defined. The algorithm type is distinguished by the analysis and aircraft that uses it as well as the way it is executed. The sub-element used assigns the algorithm to analyses and aircraft types that use the algorithm. The sub-element assignement used holds this information in the attributes IDanalysis and IDaircraft. The sub-element algoinfo holds the information regarding the types of conditions to be executed in

condition and the types of possible actions on executing the algorithm condition in action. The element algoinfo is modeled as IF condition THEN action. The types of conditions and actions are defined as archetypes called conditiontype and actiontype, respectively.

- REPORT: The report element, illustrated in Firgure 7, consists of the main element reports". It has a sub-element report in which every type of report to be generated is defined. The attribute repID assigns a unique ID to each type of report that is defined. The type of report is distinguished by the algorithm that uses it and by the way it is executed. The sub-element for assigns the report to a type of algorithm. The sub-element result of for holds this information in the attribute algoID. The sub-element reportinfo holds the information regarding the types of conditions to be executed in condition and the types of possible actions on executing the algorithm condition in action. The element reportinfo is modeled as IF condition THEN action. Similar to the algoinfo element, the types of conditions and actions are defined as archetypes called conditiontype and actiontype, respectively.

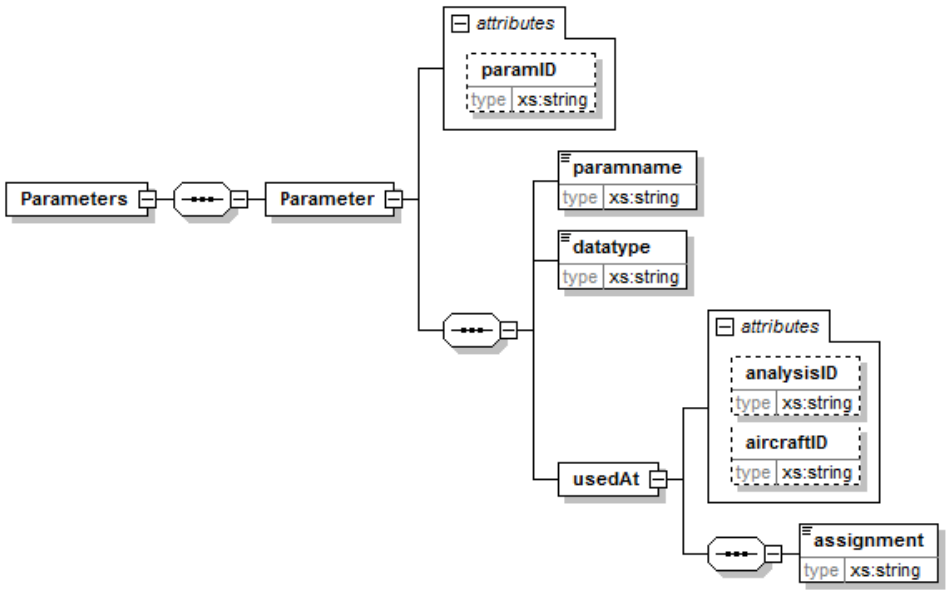


Fig. 5. Structure of the PARAMETER element

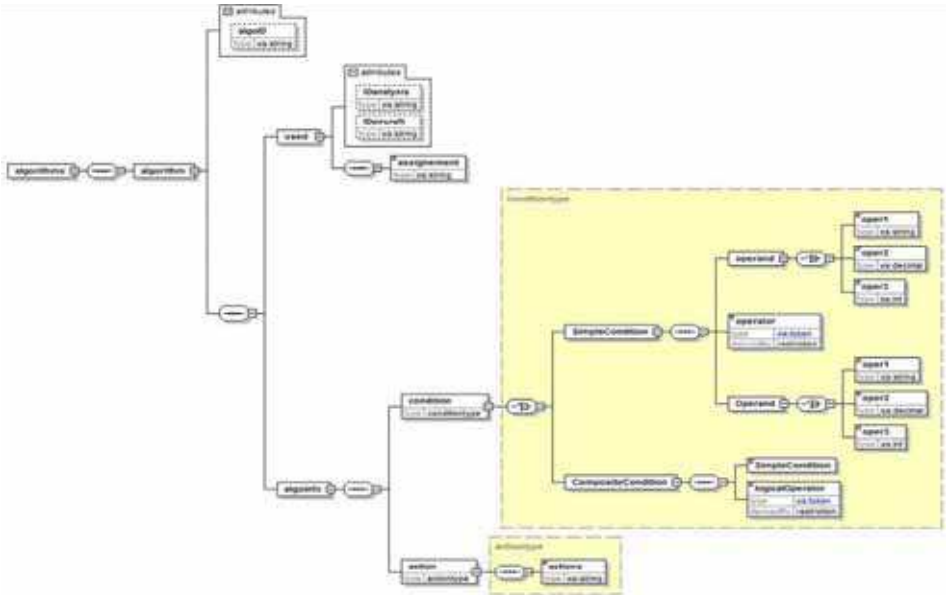


Fig. 6. Structure of the ALGORITHM element

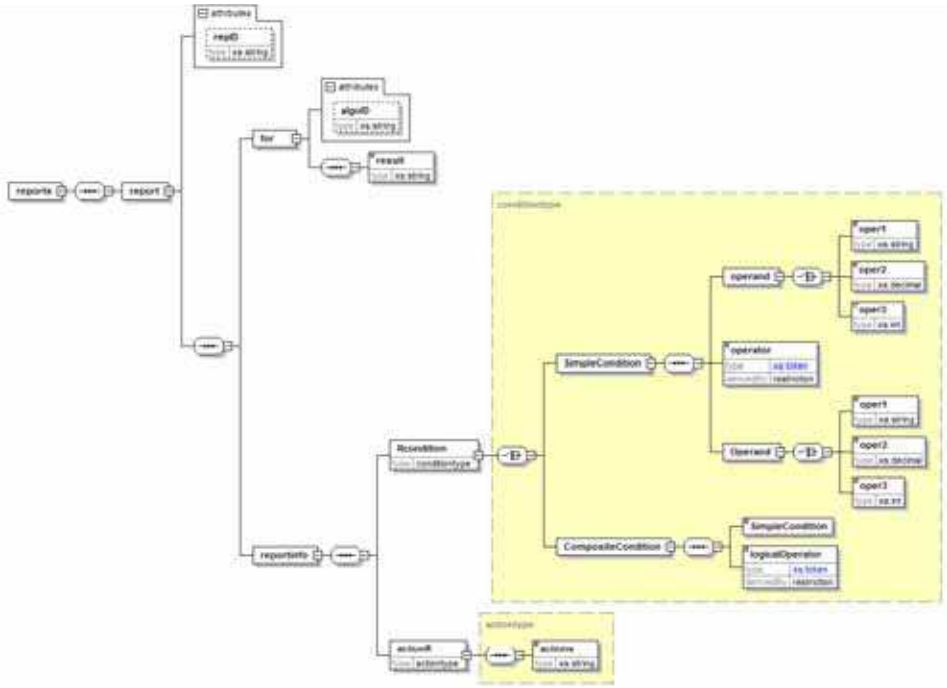


Fig. 7. Structure of the REPORT element

4.2 XML schema restrictions

In the design of XML schema, we employ the concept of archetypes which provides means of defining userdefined data types in the XML schema in a way that reduces nested definitions. The following archetypes are defined in the schema:

- CONDITIONTYPE: As shown in Figure 8, the conditiontype element is modeled as a choice element where the choices are simpleCondition and compositeCondition. The sub-element simpleCondition is a sequence element of operand, operator and operand. compositeCondition is a sequence element of simpleCondition, logicalOperator and compositeCondition.

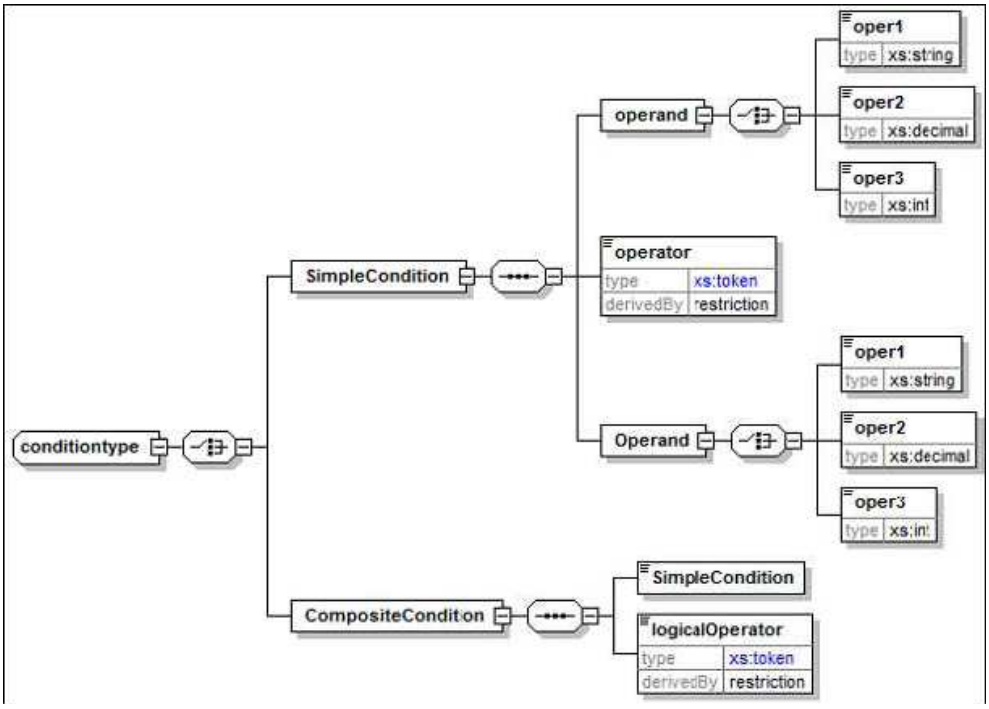


Fig. 8. Structure of the CONDITIONTYPE archetype

- ACTIONTYPE: The element actiontype, illustrated in Figure 9, consists of the sub-element actions where the corresponding action to a condition is listed.

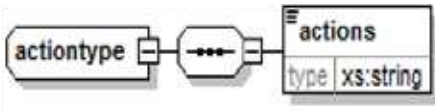


Fig. 9. Structure of the ACTIONTYPE archetype

5. Software prototype

To examine the utility of the proposed framework, we have developed a functional software prototype of the framework. The prototype has been developed under .Net Framework 3.5, in C# using Microsoft Visual Studio 2008 Professional. In this section, we present a step-by-step example of using the software prototype to generation fault BIT and exceedances reports.

5.1 Software installation

The software release comes in the form of a single MSI (Windows Installer) file. Running this installation file guides the user in the installation process of the software application. Figure 10, Figure 11, and Figure 12, respectively show the initial installation screen, the installation options, and the installation confirmation screen which indicates that all software components have been actually and successfully installed.

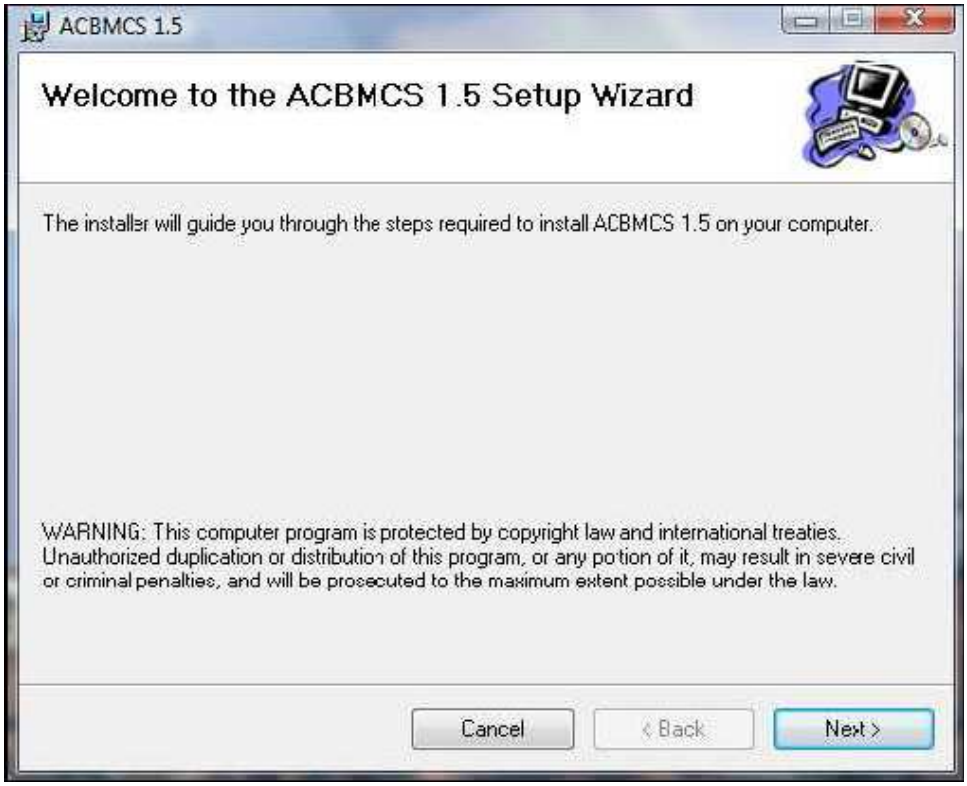


Fig. 10. Initial screen of installation wizard

Fig. 11. User can select installation folder and security level

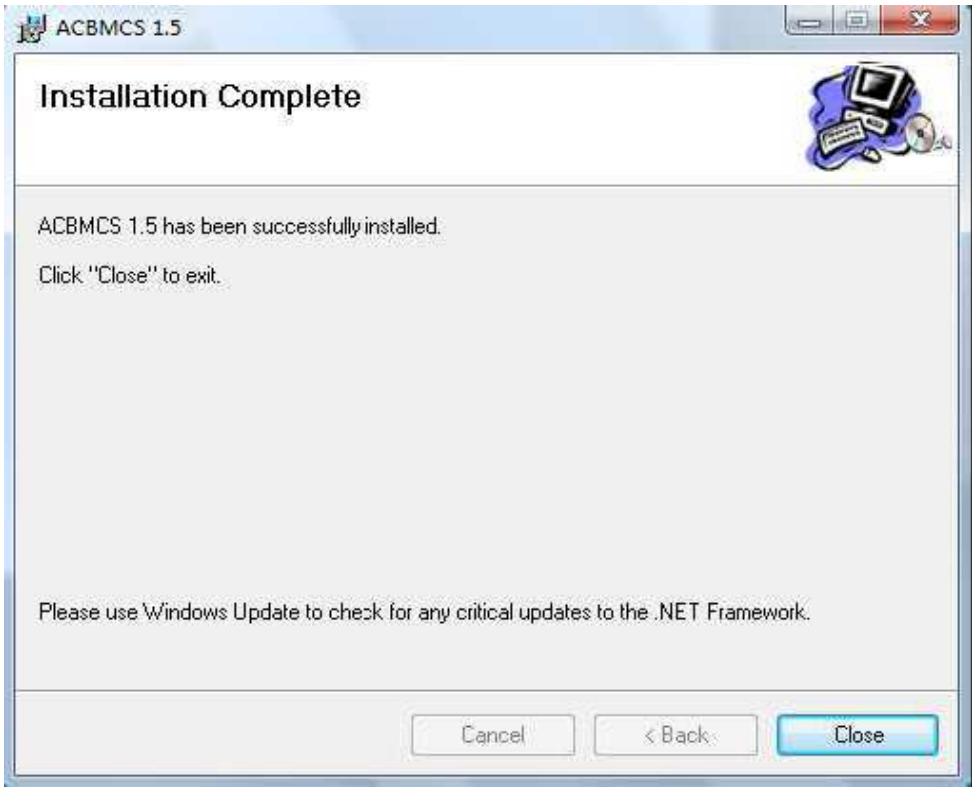


Fig. 12. Software installation confirmation

5.2 Application modes

As illustrated in Figure 13, the software features two modes available for the user; an advanced mode and a simplified mode.

Fig. 13. Advanced and simplified modes

5.3 Advanced mode

The advanced mode gives the user a control on the report generation to specify:

- The source of configuration files: The application initially loads configuration files located in the default path (Figure 14). However, the software provides the user with the option of changing the path configuration files (Figure 15 and Figure 16).

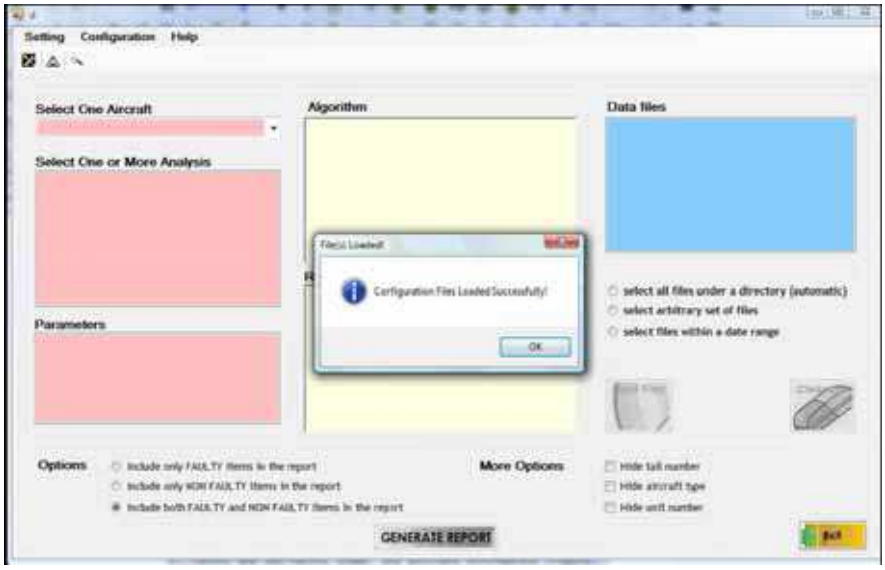


Fig. 14. Configuration files loaded from the default path

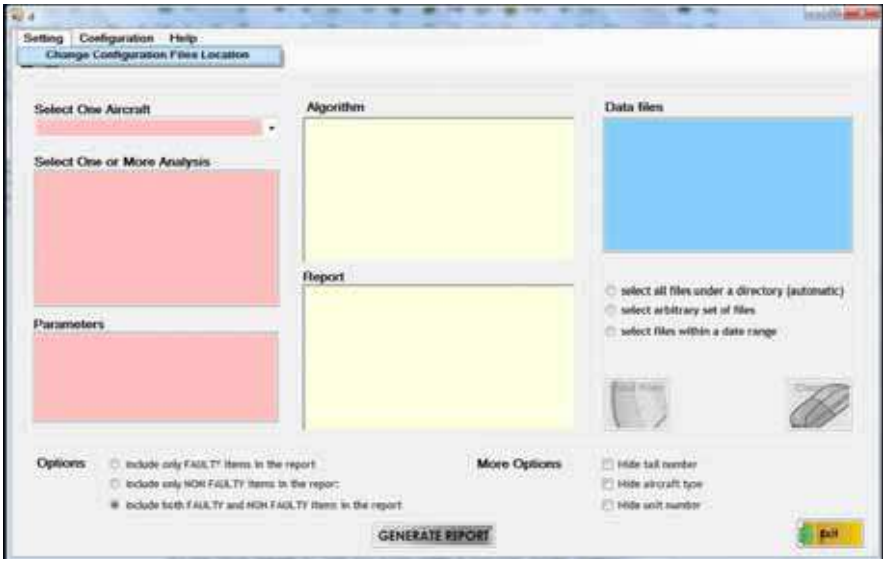


Fig. 15. Change the path of configuration files

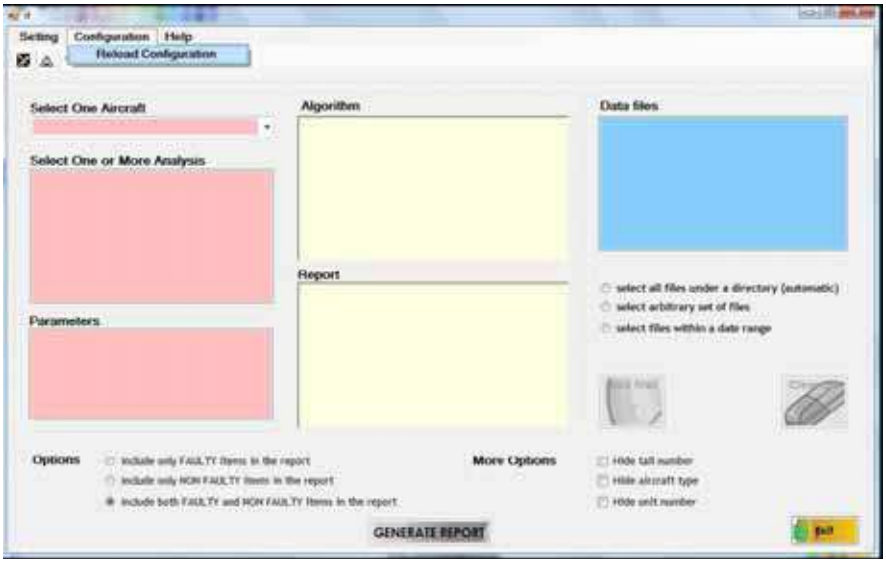


Fig. 16. Reload the configuration files

- The type of aircrafts and analysis (Figure 17): The user first selects the type of aircraft, and based on the selection, the related analysis are displayed. Upon the selection of the analysis (which can be multi-selection), a list of the related parameters and the corresponding algorithm(s) and report(s) are displayed.

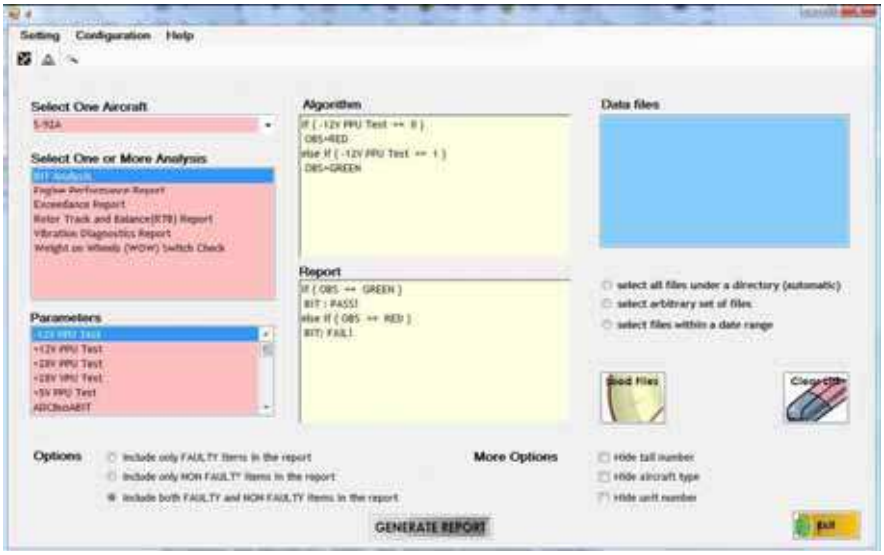


Fig. 17. Select aircraft and analysis

- The source of data files: The user has three options to load source data files. The first option is to load all files under a specific folder (Figure 18). The second option is to load a set of arbitrary files (Figure 19). The third option is to load files within a certain range of dates (Figure 20).

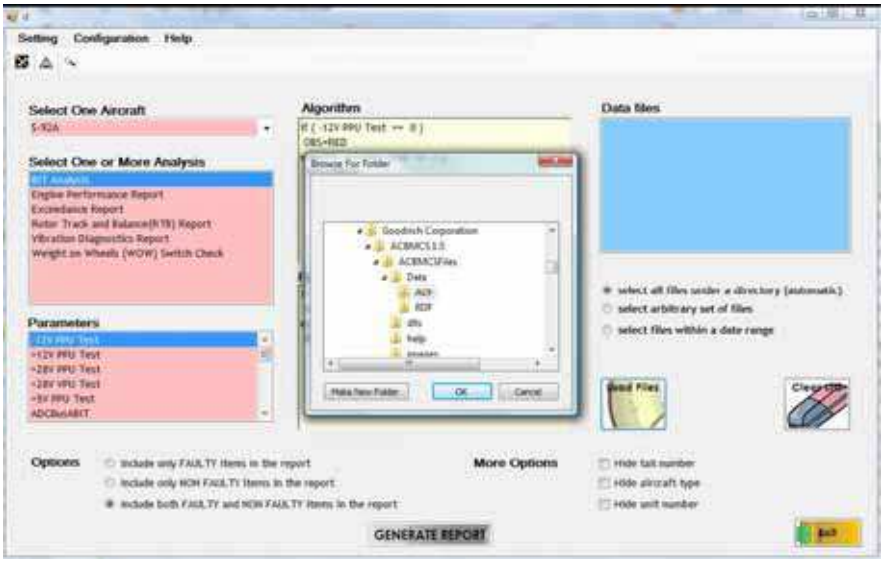


Fig. 18. Load all files under a specific folder

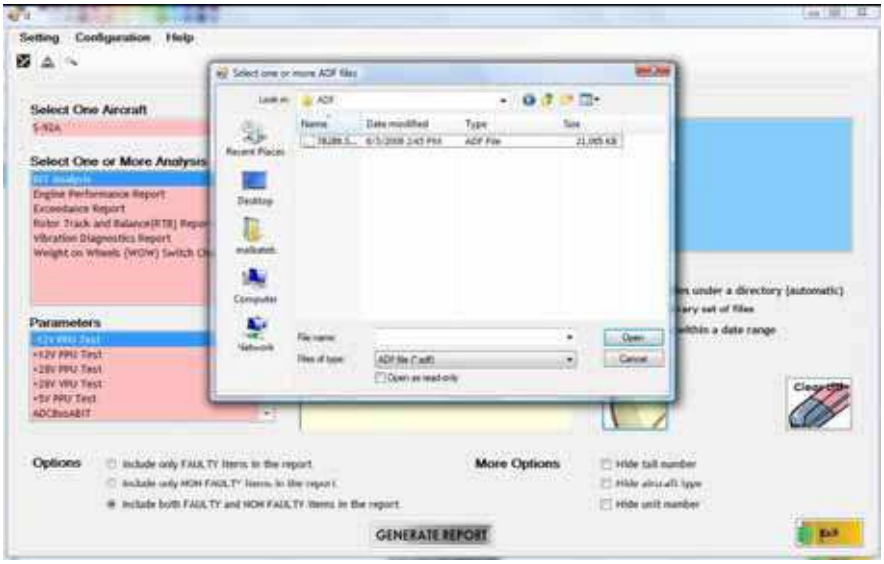


Fig. 19. Load a set of arbitrary files

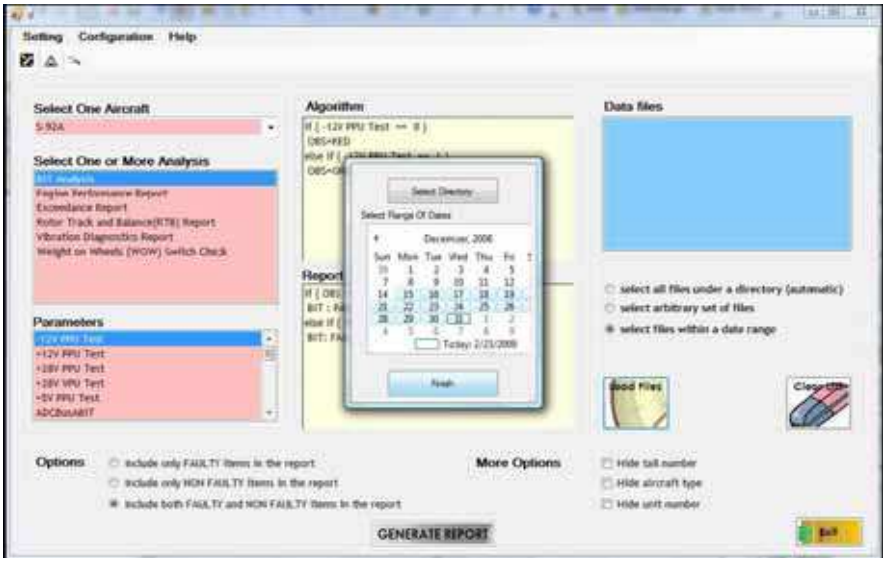


Fig. 20. Load files within a certain range of dates

- The information to be displayed and hidden in the report; faulty and non-faulty items (Figure 21), and aircraft information (Figure 22).

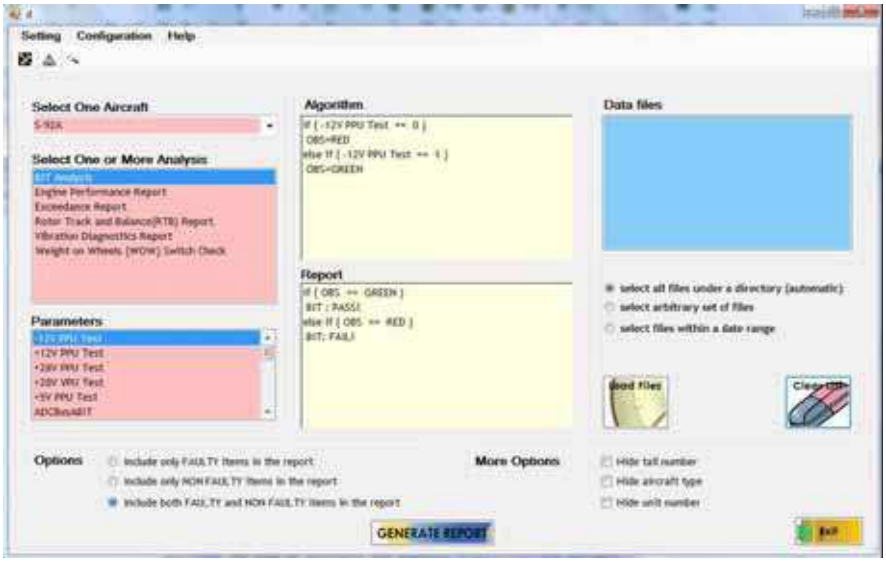


Fig. 21. Display faulty items, non-faulty items, or both

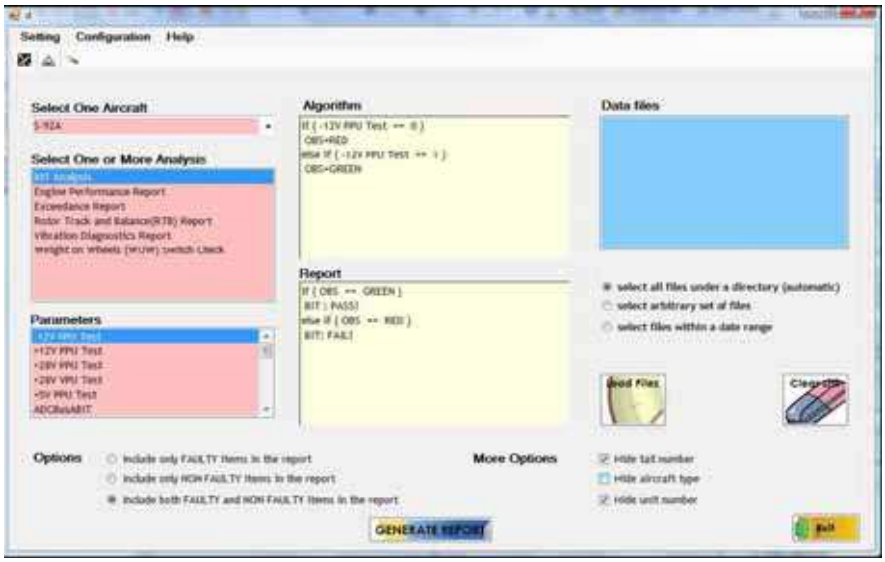


Fig. 22. Show or hide aircraft information

The last step is to generate the maintenance reports, which is achieved by clicking on the GENERATE REPORT button. Figures 23 and 24 and Figures 25 and 26 show the output maintenance reports for the fault BIT and exceedances analysis, respectively.

HUMS-ACBMCS

GOODRICH

MAINTENANCE REPORT

DATE:

2/23/2009 3:52:10 PM

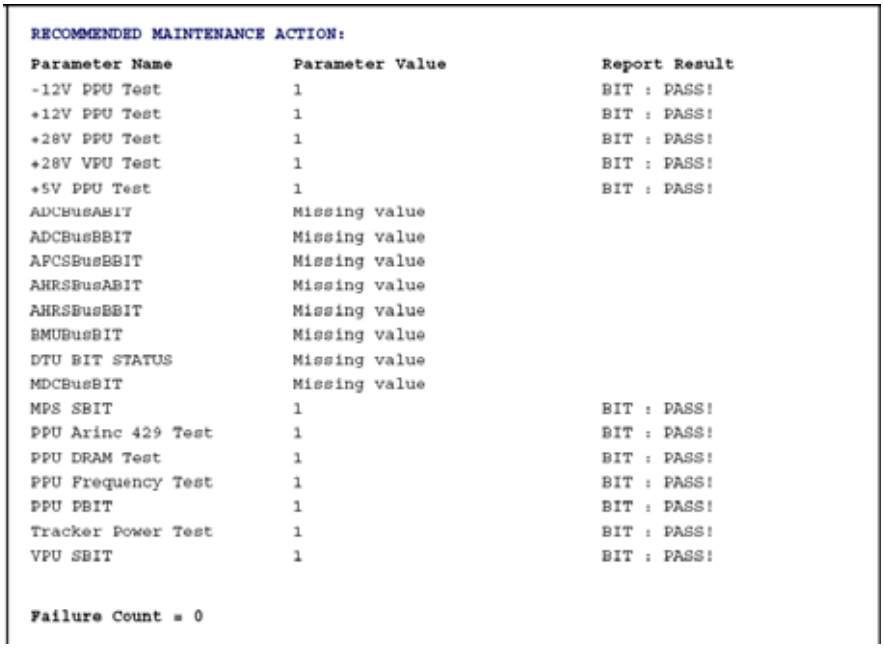
DESCRIPTION:

This report is generated by the Automated Condition Based Maintenance Checking System (ACBMCS) to demonstrate the recommended maintenance action.

ANALYSIS TYPE:

BIT Analysis

Fig. 23. Report header of the fault BIT analysis



RECOMMENDED MAINTENANCE ACTION:

Parameter Name	Parameter Value	Report Result
-12V PPU Test	1	BIT : PASS!
+12V PPU Test	1	BIT : PASS!
+28V PPU Test	1	BIT : PASS!
+28V VPU Test	1	BIT : PASS!
+5V PPU Test	1	BIT : PASS!
ADCBusABIT	Missing value	
ADCBusBBIT	Missing value	
APCSBusBBIT	Missing value	
AHRSBusABIT	Missing value	
AHRSBusBBIT	Missing value	
BMUBusBIT	Missing value	
DTU BIT STATUS	Missing value	
MDCBusBIT	Missing value	
MPS SBIT	1	BIT : PASS!
PPU Arinc 429 Test	1	BIT : PASS!
PPU DRAM Test	1	BIT : PASS!
PPU Frequency Test	1	BIT : PASS!
PPU PBIT	1	BIT : PASS!
Tracker Power Test	1	BIT : PASS!
VPU SBIT	1	BIT : PASS!

Failure Count = 0

Fig. 24. Report body of the fault BIT analysis

Fig. 25. Report header of the exceedances analysis

Fig. 26. Report body of the exceedances analysis

5.4 Simplified mode

The simplified mode generates, in one single execution, all reports defined in the configuration files by using the default system settings. Default settings reflect the default path of both the configuration files and source data files.

6. Conclusion

In this chapter, we present new framework for automating CBM systems. The utility of the proposed framework has been verified through a software prototype that demonstrates various functionality provided by the framework.

This work opens an avenue for several future works including, for instance, extending the data extraction module to manipulate the Health Indicators (HI) data from source data files, as well as conducting further complicated and advanced data mining and analysis operations using the proposed framework.

7. References

[1] Robert Hess, Alan Duke, and David Kogut (1999). Helicopter track and balance theory. Aircraft Maintenance Technology,

http:/ / www.amtonline.com

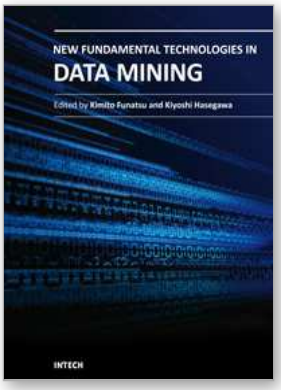
[2] Goodrich Corporation. *Design standard for condition based maintenance system for army aircraft*. ADS-79-SP CBM

[3] US Army Aviation and Missile Command. *Uh-60 condition based maintenance (cbm) manual*.

[4] Goodrich Corporation. *Configuration requirements specifications for the s-92 hums exceedance processing and monitoring*. DOC.NO.: 3019554-CRS-0104

[5] Goodrich Corporation. *S-92 imd hums system specification*. DOC.NO.: SES-920273

[6] Mahindra Imadabathuni, Pradnya Joshi, David He, Mohammed Al-Kateb, and Eric Bechhoefer (2009). Application of the automated condition based maintenance checking system for aircrafts. Processings of The International Conference on Industrial Engineering and Systems Management (IESM 2009). MONTREAL - CANADA



New Fundamental Technologies in Data Mining

Edited by Prof. Kimito Funatsu

ISBN 978-953-307-547-1
Hard cover, 584 pages
Publisher InTech
Published online 21, January, 2011
Published in print edition January, 2011

The progress of data mining technology and large public popularity establish a need for a comprehensive text on the subject. The series of books entitled by "Data Mining" address the need by presenting in-depth description of novel mining algorithms and many useful applications. In addition to understanding each section deeply, the two books present useful hints and strategies to solving problems in the following chapters. The contributing authors have highlighted many future research directions that will foster multi-disciplinary collaborations and hence will lead to significant development in the field of data mining.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821