Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 302716, 9 pages
http://dx.doi.org/10.1155/2014/302716

Research Article

FDIR for the IMU Component of AOCS Systems

Maurício N. Pontuschka[1] and Ijar M. da Fonseca[2]

[1] *Department of Computer Science, PUC-SP, 01303-050 São Paulo, SP, Brazil*
[2] *Aeronautical Mechanics Division, Department of Mechatronics ITA and DMC/INPE, 12244-456 São José dos Campos, SP, Brazil*

Correspondence should be addressed to Maurício N. Pontuschka; mauricio@realide.com

Received 16 May 2014; Accepted 11 July 2014; Published 1 September 2014

Academic Editor: Antonio F. Bertachini A. Prado

The main objective of this paper is the study of a FDIR for an IMU aiming at space applications with focus on the gyro signal analysis and the tests of the filtering algorithms. The algorithms have been tested by using lab data provided by the DMC LABSIM (Physical's Simulation Laboratory of the Space Mechanics and Control Division of INPE). The results have demonstrated good agreement with the concepts applied in this study. Automatic detection procedures are very important in the characterization of occurrence, definition of criteria, and device types in the scenario of AOCS FDIR. An IMU comprised of four gyros in a tetrahedral configuration is one of the assumed components for the AOCS (attitude and orbit control subsystem) considered in this work. The types of failures considered in this paper are the step abrupt change, ramp/drift/slow, stuck, cyclic, erratic, spike, and finally the stuck for variance alteration noise. An appropriate algorithm for the automatic detection of each type of fault is developed. The approach includes the mapping capability of fault event indicators to the IMU. This mapping is very important in the characterization of the occurrence, definition of criteria, and device types as well as associated fault identification for an AOCS.

1. Introduction

The FDIR acronym refers to three main functions related to software (SW) and hardware (HW): fault detection, referring to the ability to discover faults or the process of determining that a fault has occurred; fault isolation being the function of fault localization within the system by providing information pinpointing it; fault recovery referring to the process of limiting the fault propagation and enabling the service to be restored to an acceptable state.

FDIR belongs to a wide area called fault-tolerant control systems (FTCS). In other words, FTCS refers to the control systems with capability to accommodate component failures automatically. Such systems are capable of maintaining overall system stability and performance in the event of failures. The FDIR enters in this process by detecting, isolating, and recovering the system being monitored in the event of fault occurrence [1, 2]. In the FTCS scenario, a closed-loop control system, which can tolerate component malfunctions while maintaining desirable performance and stability properties, is said to be a fault-tolerant control system. Despite the many individual research in this area, systematic concepts, design methods, and even terminology are still not completely standardized for the specific area where FDIR and related literature appear [3]. As the subject is multidisciplinary considering the complete life cycle of FDIR systems, some standard terminologies come from software and hardware specification and development as well as from quality assurance.

The fundamental requirement for planning current space missions is autonomy. The autonomy aims the satellite's architecture improvement and special attention to the FDIR to be implemented in space. Automatic actions have been included in space missions since the beginning of the space conquer. Nowadays automatism plus autonomy represent the state of the art of space missions requiring spacecraft with the capability to operate as intelligent robots. In such scenario the FDIR appear as one of the most important

subsystems for the safety assurance of space missions. The importance of such subsystems is enhanced with the following statement regarding FDIR "It is a cornerstone for the satellite's autonomy enhancement" [4]. This means that the FDIR subsystem design, test, and implementation are seen as imperative to guarantee a dependable and autonomous system with a minimal risk of harmful failures. FDIR systems aim to provide three important functions: (i) monitoring, where the responses of the system are checked with respect to tolerances and generates alarms to the operator; (ii) automatic protection, where the FDIR automatically initiates an appropriate counteraction when a dangerous process state is detected; (iii) supervision, where the fault diagnosis is performed and contingency actions are taken [5]. As a safety-critical subsystem FDIR development must follow stringent standards from conception to implementation. The objective is risk reduction, in addition to robustness, performance, and reactive detection, isolation, and recovery from failures. According to the ECSS standard [6], dependability is terminology used to describe the availability performance and its influencing factors: reliability performance, maintainability performance, and maintenance support performance. When working with safety-critical subsystems as it is the case of FDIR it is important to take the clear meaning of the terminology, as defined in the standards. The definition and integration of FDIR technologies in the on-board software lifecycle is considered mandatory by different satellites manufacturers.

Regarding the on-board software (OBSW) FDIR some conventional FDIR techniques (for example, SW watchdogs, memory scrubbing, parameter monitoring, among others) are typically present in most satellites. More sophisticated software FDIR techniques have recently been developed. One example is the model-based FDIR with artificial intelligence (AI) support also called smart-FDIR, in addition, advanced FDIR AFDIR) techniques have been made available as reusable components. The FDIR includes the Kalman filtering, weighted sum-squared residual testing, and generalized likelihood testing.

Considering the currently available software FDIR techniques and the actually implemented FDIR schemes in different satellites, there exist a gap between the usage of these technologies and their early incorporation into the satellite's design phase. Designers' trend is toward incorporation of a generic FDIR architecture into the whole design. This architecture must be easy to maintain, extend, and adapt. The above mentioned gap is characterized by the lack of an early incorporable, reusable high-level, flexible, and maintainable FDIR generic architecture to be used customized according to the designers' need. The major satellite manufacturers are supporting the development of more generic software architectures aiming cost-cutting and further market competencies. FDIR is generally taken into consideration, only when all the equipment designs and specifications are available. It is believed that many recurring on-board problems and failures can be avoided if a solution is found for the problem of late FDIR definition and incorporation. Missions that faced problems, such as the Mars climate orbiter, Ariane 5, Titan IV-B, Mars polar lander, and deep space 2, motivate deep studies on how to avoid failures which may have shared the same cause in different missions. The relative expensive cost could be reduced if generic, maintainable, and extendable FDIR designs are available for reuse by different space projects designers.

In general, any complete FDIR arrangement must be distributed on both HW and software SW levels in any on-board computer architecture. The FDIR concept at the HW level can be seen for instance in a processor watchdog timer or in an individual watchdog that receives beacon signals from different units. The memory error detection and correction (EDAC) function is also present on the HW level as a classical FDIR component. The HW level is referred to as HW FDIR and at SW level as SW FDIR. Considering FDIR analysis and design the following approach can be used [7].

(i) Define failure scenarios, detection strategies and levels of autonomy.

(ii) Partition FDIR functions into on-board hardware and software or TM/TC actions.

(iii) Define FDIR architecture as centralized (single node), distributed FDIR (multiple nodes with majority voting) or a combination of both.

(iv) Create a FDIR analysis tool in UML to derive the FDIR actions.

(v) Create a FDIR model of the system incorporating the FDIR algorithm using UML.

(vi) Evaluate and optimize the system architecture performance when exposed to different failure scenarios.

(vii) Demonstrate the FDIR algorithm running appropriate computer simulations. The MARC (modular architecture for robust computation) demonstrator handling failures is an option for demonstration.

(viii) Assess the performance and reliability by using appropriate communication protocol (MIL-STD-1553B for example) to handle critical commands/telemetry.

FDIR must be part of many spacecraft subsystems for safety reasons. AOCS is one of those space vehicles safety-critical subsystems once it failures may lead to mission loss. This paper represents an initial study in the scenario of the Brazilian Amazonia-1 satellite AOCS. The AOCS diagram is shown in Figure 1. The Amazonia-1 AOCS software comprises the FDIR module that allows performing detection, isolation, and recovery consistent with predefined failures. The FDIR is a dynamic table in which predetermined set of failures are programmed. The Amazonia-1 AOCS includes a tetrahedral IMU (inertial measurement unity) comprising four gyros as shown in Figure 1. That IMU is the object of the paper for the sake algorithm development aiming the detection and isolation regarding failures. In this sense this study may apply to any AOCS subsystems comprising IMUs.

The data for simulations and validation of the algorithm developed in this work was obtained from the DMC's LABSIM (Space Mechanics and Control Division's SIMulation LABoratory–Laboratório de Simulação da Divisão de Mecânica Espacial e Controle), at INPE (National Institute for Space Research).

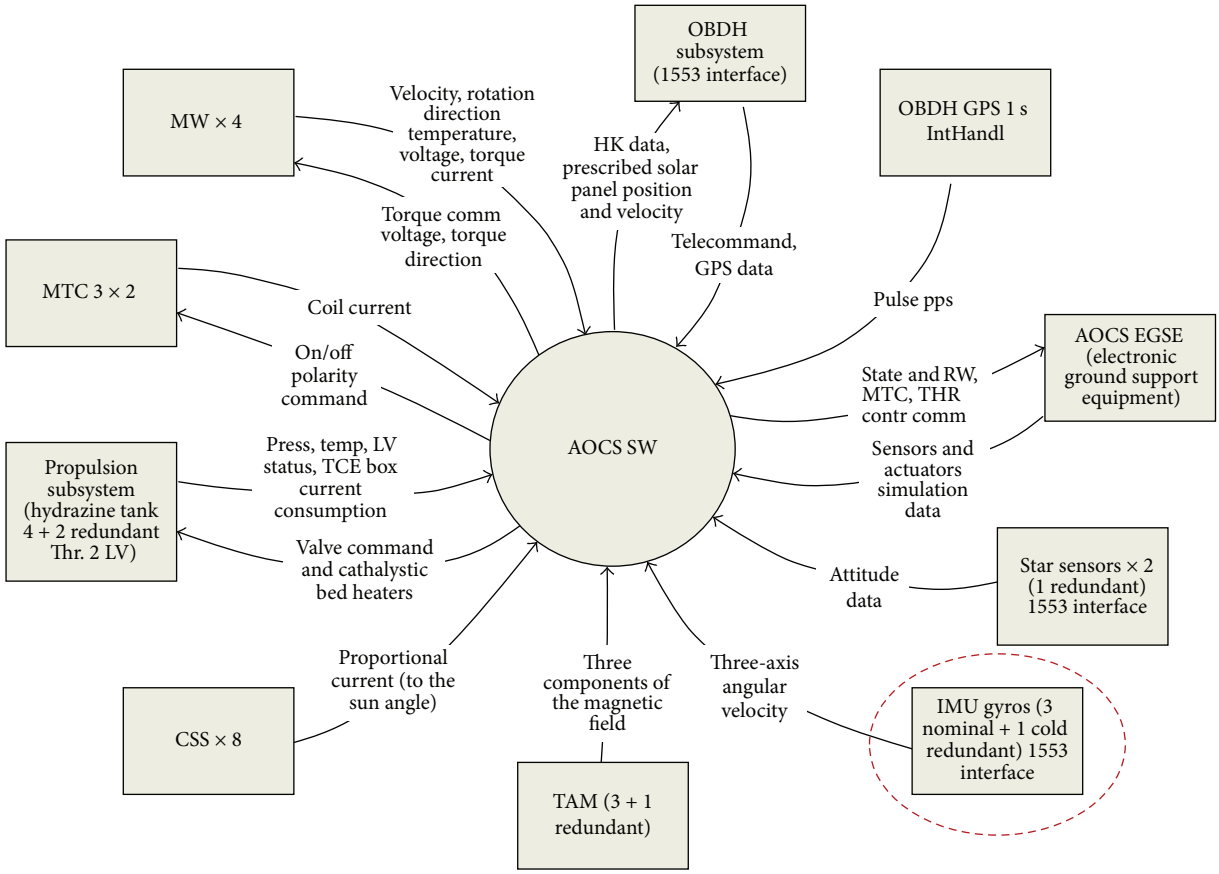

Figure 1: Amazonia-1 AOCS enhancing the IMU inside a dashed cycle.

The types of failures considered in this paper are the step abrupt change, ramp/drift/slow, stuck, cyclic, erratic, spike, and finally the stuck for variance alteration noise. An appropriate algorithm for the automatic detection of each type of fault being analyzed is developed. As a result of this approach a mapping of events fault indicators to the IMU is obtained.

2. Discussion on FDIR

This study focuses on the FDIR occurring in inertial units. Those inertial units were originally applied in control of critical system operations in military combat aircrafts and spacecrafts [8, 9]. Project of these categories include the ability to operate in dynamic environment demanding huge computational effort to maintain acceptable levels of stability and operability. The stability during the flight requires the use of inertial sensors to provide output acceleration, rate, and angular position. The purpose of those control devices is to establish reliability, performance, and integrity to provide the lowest error rates aiming at the mission success. Inertial units may be constructed as arrays of strap down sensors with different degrees of freedom. Each sensor provides electric signals that may contain errors from different natures. There are algorithms to compute various types of these known errors for fault identification [10]. As scope of this work we selected the following error types: step abrupt change, ramp/drift/slow, stuck, cyclic, erratic, spike, and stuck for variance alteration noise.

In most cases the algorithms that were created to identify the errors are based in signal analysis acting as filters that raster the signal searching specific patterns [11] that indicates the presence of known errors. In these cases the computational construction presents a loop in its main structure to read each byte of an input sequence. The problem is how to make several analyses simultaneously by a single processor unit. These algorithms have to be combined in a way that the system could achieve the best level of performance.

The strategy is this paper is to use a single loop structure and a set of buffers to provide all the analyses in a sequence inside the loop. The sensor provides the raw signal to the DIF that initiates the analysis. The first buffer reads the first byte of the raw signal to make it available for the next level of analysis. In this level the first filter will uses the information contained in the first buffer. Each filter requires a different quantity of information necessary to begin the analysis. If the information is not sufficient to initiate the analysis, the flow of execution is sent to the next loop where a new value is read from the sensor. On the other side, if the information is sufficient the first filter process provides a new signal to the next buffer. The next buffer is processed by the next filter in the same way of the first one. So we can provide in the same

loop all the filter combination necessary to achieve better performance in the analysis of the complete set of filters.

When identifying if one of the known errors has occurred we have to ensure that all of filters have applied the signal modifications of each buffer in order to process the algorithm, guaranteeing the simultaneous analysis. In the case of identification of any used patterns in the analysis, an exception signal has to be emitted in order to alert that the referred pattern has been identified. It is difficult in this time to say that this identified pattern is really about a real error, but the finding is fundamental information about on how the system is working and if system behavior is as was expected. Pattern identified, the next step is to send the information to a higher layer of analysis which uses artificial intelligence, state diagrams or even a set of rules focusing on the interpretation of the context that the systems is inserted and if the finding really indicates an error condition.

This means that the lower layer is responsible for providing good quality information to the interpretation layer (Figure 6). This time the analysis does not use a signal processing base anymore (as used in the previous layer). A data structure is built from the signal processing analysis. At this point the structure includes the necessary information to identify actions to be taken.

The most critical system environment usually restricts the available computational resources. In attention to this it is recommended to design adaptive systems that can use efficiently the available resources and still guarantee de process efficiency. When we use a set of layers to implement the analysis, we are using all possible pattern matches that can be found in first place, but when a pattern match happens, the effort is to check if it is in fact an error or not. In this case some of the checking can be disabled to make possible the new verification. Some level of checking has to be maintained to guarantee that the main systems are still working, but the most important task in this situations is to decide if the analysis will mark the pattern matching as an error or not. This choice can decide the system's mission success.

In practice it is almost impossible to predict all kind of errors, but it is certainly possible to find a set of errors. It is recommended to incorporate all those known errors in the FDIR system. In this paper the complete set of those errors and how to identify, isolate, and recover from each one are used to increase a knowledge base. This base is updated each time new information is discovered in the computing error process. It is not a purpose of this work to discuss about how that information is updated on orbit operating satellites. However, new satellites could carry error information of past missions in their FDIR and include a safer procedure to achieve the mission success.

3. Approach to Develop the FDIR Software

The FDIR software has to be built after the components of the satellite are defined. This is necessary because the FDIR analysis has to explore all possible known errors that may be associated with the component arrangements. Accordingly the architecture and components used on FDIR software have

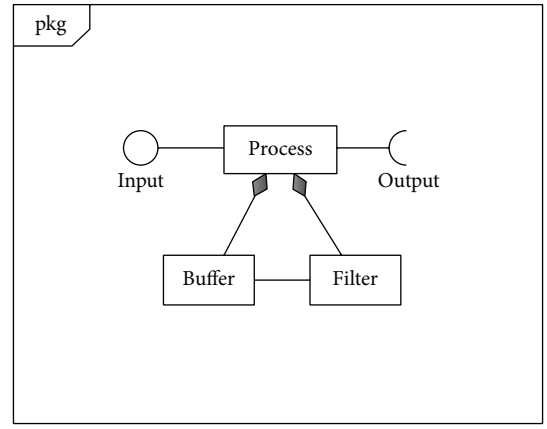

Figure 2: UML schematic model to the signal process.

taken into account the hardware system arrangement. This study proposes an approach to build a unique software architecture that could encompass different FDIR applications. Such architecture can lead to reuse of code of already tested software components. Further, the code reuse may result in better reliability, performance, accuracy, cost reduction, and quality improvement in future software developments, instead of building all the software from scratch in each new project. A software component that is already used in several other missions with good performance and accuracy certainly provides the same quality levels and benefits in further projects.

The main software component of the signal processing phase (Figure 5) combines the several signals analyzes in a single loop that can be configured according to the project needs. It was developed a framework for FDIR that accommodates different filter analyzes and an appropriate algorithm for a future automatic detection of each type of fault being analyzed.

Most signal analysis is implemented in a loop that reads a sequence of bytes organized in an array [12]. In this paper such array is the digital signal that is read from an IMU component [10]. The main architecture is responsible for receiving that signal and maintaining a main loop to process all filter algorithms. Each algorithm has to have a specific buffer to receive the signal from the previous algorithm so as it obtains the information necessary to make its analysis. The result of the analysis will be stored at the buffer of the next algorithm. At this point we can think of each algorithm as a process linked to two buffers, the one comprising the used data and the other storing the results. This approach is presented using UML 2.2 notation with class diagrams for the conceptual model [13]. The UML means unified modeling language and appears in the avionics literature as base for other modeling approaches, such as the category DSML (domain-specific modeling language) to model simulation testing systems of avionic software [14].

Figure 2 shows the scheme of a general model process for the signal. Figure 3 shows that each process has its own buffer and filter. Figure 3 also shows that the output of a filter is connected to the next filter input. Each buffer is independent

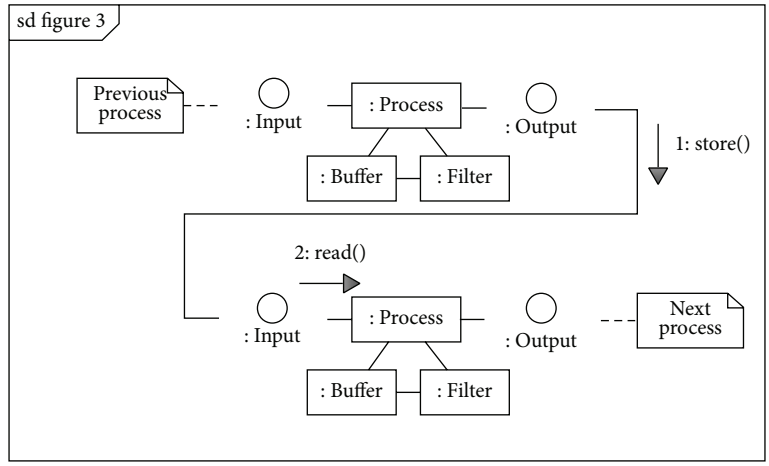

Figure 3: UML schematic model of the processes integration.

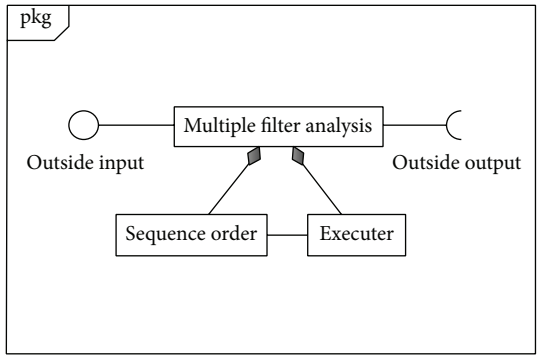

Figure 4: Multiple filter analysis main architecture.

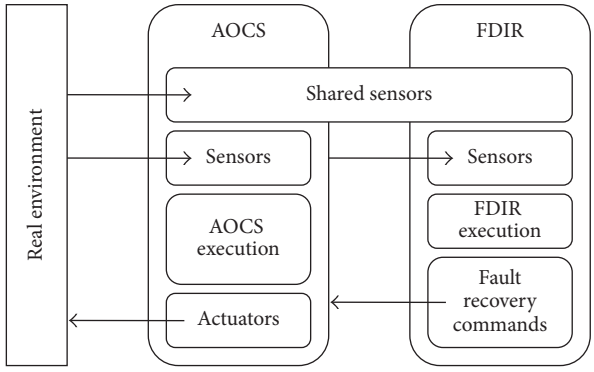

Figure 5: Integration between AOCS and FDIR modules.

for each process; the data stored in a buffer has to be used only by the associated process. The software has to be constructed in a structured paradigm even when using the same oriented object like diagram for modeling purpose; Figure 3 presents in fact the integration of various unit processes.

Figure 4 shows that the main loop has to generate an outside signal input, a main loop, a connection to the outside output, and also a form to establish the correct order of the process executions. This model executes a single step of analysis in each process filter. The reading loop is not inside of the filter algorithm but outside as the main loop that is shared by all processes.

In Figure 4 the executer reads the next process identification from sequence order, sends the next word of information to the process buffer, and executes the respective filter step. If the filter succeeds in processing the information the executer takes the next process from the sequence order. If the filter fails in processing the information, one more word of information is read from the outside input and sent to the same process until it returns a success signal. Some internal loop analysis has to consider a specific size of information to be able to process the signal, so that each process acts so as to provide sufficient information to the next level of analysis. Each process that is executed will store the results in the next buffer process and check it for success or failure. The workflow returns to the main loop to provide more information to feed the system if a failure is detected.

4. FDIR Analysis Approach

Up to this point we have presented the procedure to obtain a clear signal in the sense that all the set of signal processing to be used is applied. We have to be sure that a correct set of filters and associated correct order are implemented in each specific context the used system. The procedure mentioned above provides different groups and sequence of filters to allow more flexible analysis for different subjects. The procedure that has been presented up to now can handle several filters in a configurable sequence of execution with a minimal delay. The next step of this strategy is to start the pattern matching for the identification of the signal meaning. In this phase the system should start the analysis by asking a pattern matching from a set of historical data of known errors from each applied context to the executed system. The main purpose of this phase is to build a model that could represent what is being monitored and then make and compare the data with those that were expected for that point of the mission. Each kind of mission could have some different approach modeling the behavior and structure.

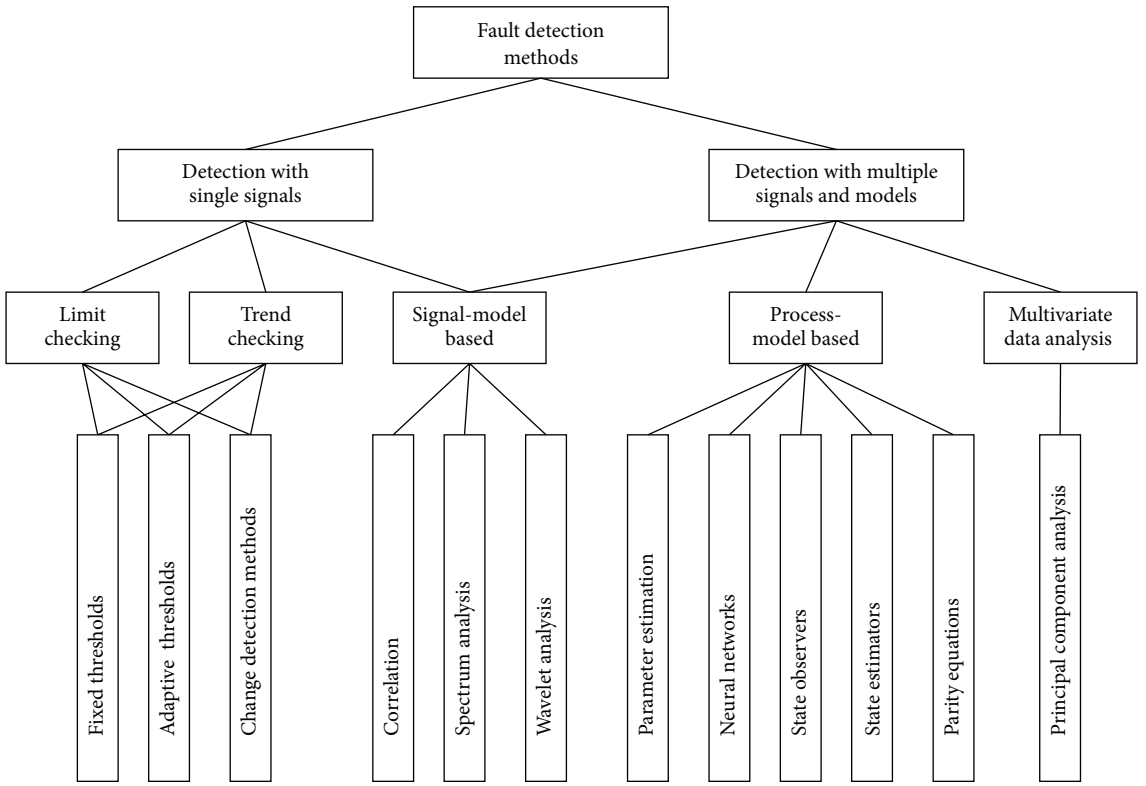

Figure 6: Overview of different fault detection methods [16].

To accomplish a valid comparison it was built and assembled some integration component to generate a consistent comparable system compatible with the historical data. It is necessary a good model to represent the historical data and also an approach to represent the behavior and the structure of the system under study in the paper. The first step is to provide an integration model to hide all specific elements from the main architecture. In that direction, the system has been partitioned in two parts, one that is independent and another that is highly dependent with respect to the context. The dependent part is built by observing the system's specific hardware (sensors and actuators). The independent part takes into account all of the system's common features and, consistently, cover all the system's particularities. For the analysis presented in this paper, some known pattern designs were studied to see how they could help to make that independent part of a FDIR. The adopted patterns for this work are classified into the categories [11] as shown in Table 1.

Table 1 enhances the different categories of design patterns. The FDIR solution uses this sort of patterns to organize the information about the observed system and to build the data structure and the desired comparison.

It is developed an approach to interpret the signal and convert it into high-level information representing an information model. The model is consistent with the above patterns. The model involves a set of attributes that represents the system status and also a more complex structure data as can be seen in the structure's patterns. The model is changed according the signal readings. Some events are established to start the comparison routines. The period could be different in different type of analyses. When referring to the historical data, the reference is a set of routines that have to be started as the system finds a match. The set of routines represents the knowledge base and must contain the routines for all known errors that were related to the context in which the system runs. If a new error is found in one type of the equipment, even if the mission of the equipment fails, it can be used to build a new routine and be communicated to all other FDIR equivalent systems to prevent that the error occurs again. The objective of these routines is to seek for a possible occurrence of a problem. The objective of those routines is to seek for a possible occurrence of a problem, each routine for each type of problem. Each routine has to be built as an independent component and having its own structure and behavior. When the system is set with these considerations it is possible to plug some contingency routines with the leaves of the decision tree.

In Figure 5, the shared sensors provide signal read from the real environment and the FDIR sensors provide the signal read from the AOCS actuators and sensors. The focus is on the FDIR execution module which contains some important modules to support the FDIR tasks. The FDIR must first detect and diagnose a fault before reacting to it [15].

There are several methods for approaching the FDIR system for spacecrafts. The choice of each method depends of the space mission requirements.

Figure 6 illustrates several fault detection methods in different levels in the following categories: limit checking,

Table 1: Patterns category and description.

Category	
Behavior pattern	Description
Command pattern	Command objects encapsulate an action and its parameters.
Weak reference pattern	Decouples an observer from an observable class.
Protocol stack	Communications are handled by multiple layers, which form an encapsulation hierarchy.
State pattern	A clean way for an object to partially change its type at runtime.
Strategy pattern	Algorithms can be selected on flying.
Creational pattern	Description
Adapter pattern	"Adapts" one interface for a class into one that a client expects.
Bridge pattern	Decouples an abstraction from its implementation so that the two can vary independently.
Facade pattern	Creates a simplified interface of an existing interface to ease usage for common tasks.
Pipes and filters	A chain of processes where the output of each process is the input of the next.

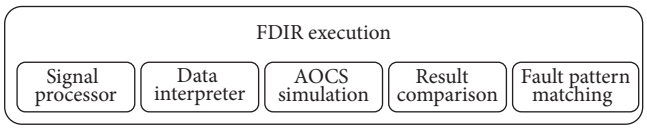

Figure 7: FDIR execution modules.

trend checking, signal-model based, process-model based, and multivariate data analysis. All these methods are characterized by a configuration which starts from the most physical level to the more conceptual level. Each method is developed for specific purpose in detecting failures in a particular scenario associated with the system being monitored.

Figure 7 illustrates the organization of an FDIR in terms of its module. The first one is the signal processing unit that is responsible to provide a clean signal for analysis. The second is the interpretation of the signal in order to give a good input data for the simulator unit. The third is the simulation unit that implements the computer simulations. Next is the result comparison unit which compares the result of the simulation with the input data and checks if the data received from the sensors monitoring the actuators demonstrate good performance with the expected AOCS behavior. If the results are consistent with the expected ones, they are used as input for an error pattern matching in order to try to find the recovery procedure to be started on the AOCS module. On the contrary, the ground base is alerted to assume the situation and provide the solution for the problem. Assuming that FDIR execution process is organized in a set of levels, it is possible to identify the level where each module is executed. The levels are (i) equipment-level where it is located the signal processor module; (ii) functional-level where it is located the data interpreter module; (iii) operation-level where it is located the AOCS simulation module; (iv) decision-level where it is located the result comparison and the fault pattern matching modules [4].

The AOCS simulation module is modeled by the behavior and creational design patterns. The AOCS simulation will indicate the expected response of the AOCS execution and compare the results. This action is divided in two main parts: (i) Verifying if the AOCS functioning is acceptable, and (ii) verifying if there are some software errors that could cause a failure situation. For the first part, the AOCS main software is encapsulated and compares the results from both modules. The results are expected to be the same. If a difference is detected then the result could indicate a possible failure situation. The point is to find if the error has occurred in the AOCS module or in the FDIR module. So, this can be used only to detect the occurrence of the error instead of identify it. The other kind of error is the identification of a software problem. The detection is based on nondesirable result from the actuator actions, for the detected inconsistency in the AOCS simulation. This verification is inside the AOCS simulator that includes the expectation of each actuator action and then implements the verification about the success of its action. If the result is not consistent with the expected one, there is a probability of a software error occurrence. Another source of this kind of problem could be associated with hardware problems. That is why it is only possible to infer that an error has occurred, not the associated source of error. The result has to be submitted to the error pattern matching module to determine the error source.

The error pattern matching module includes a set of rules that provides a clue about the current status of each subsystem. Also it includes the ability to start a subsystem unit check to verify the hardware good functioning. When the results are analyzed and a conclusion about the problem is achieved the data are communicated to the AOCS system which start some different kind of actions such as configuration adjustments; sensors isolation; actuators isolation; change the strategy of some algorithm that will have to perform the commands with a different set of hardware devices, and call the ground base for a procedure solution.

As a future study the suggestion is to build the FDIR system which verifies the results in order to check the strategies that were chosen, build a good set of known problems to form a knowledge base to be used in future missions, as well as studying good strategies aiming at adapitative systems. The aim of such work would be to provide a dynamic system

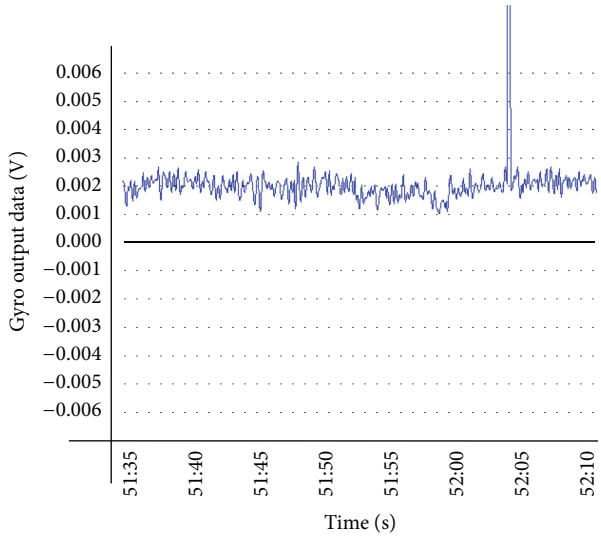

Figure 8: Graphical image of the gyro raw signal.

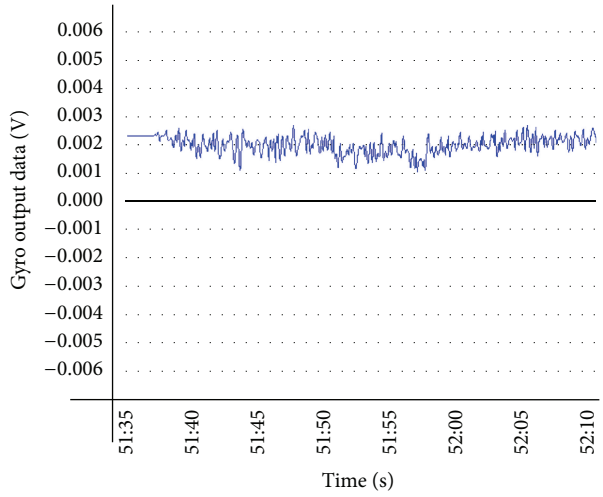

Figure 9: Graphical image of the gyro signal with the impulsive filter activation.

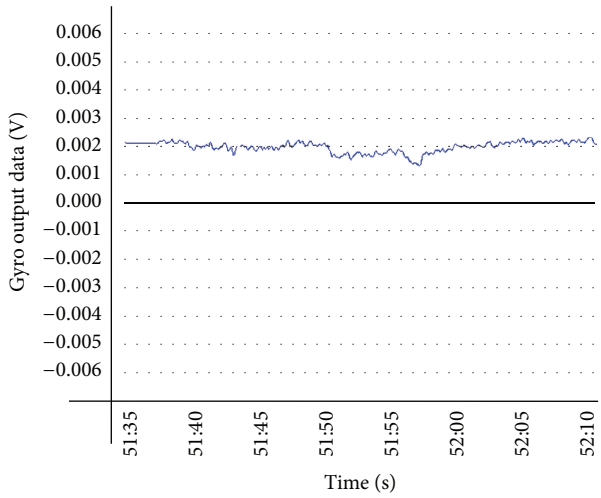

Figure 10: Graphical image of the gyro signals with the impulsive and Kalman filters activation.

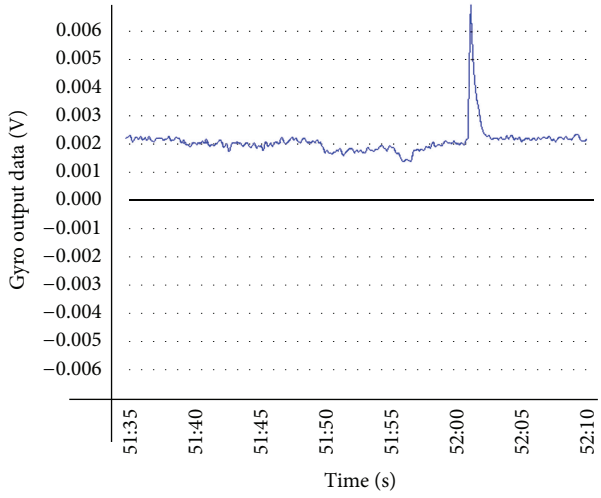

Figure 11: Graphical image of the gyro signals with the Kalman filter activated and the impulsive filter deactivated.

that can grow with new patterns of new studied errors so as to contribute for FDIR systems more proactive instead of reactive systems.

5. Simulations and Results

For the sake of the approach assessment a set of data obtained for real case at INPE-LABSIM was used in the computer simulations. As a first stage of the development we have implemented the filter module for a FDIR using the approach of the single loop. This module uses a file that was recorded in INPE- LABSIM containing up to 100 minutes of a gyro data. The software uses those to simulate the reading of the real sensor and presents a graphical view of the signal and show how it changes when a set of filters and associated parameters are changed and combined. The results are shown in Figures 8 to 12.

Figure 8 shows us the raw gyro signal that was recorded at INPE-LABSIM. Figure 9 shows the same file but with the impulsive filter activated. The filter can eliminate the undesirable signal peaks that could lead to an erroneous understanding of the satellite behavior.

Figure 10 shows the gyro signals by taking into account two combined filters, the impulsive and the Kalman filters. Depending on the required gyro signal analyses, it is necessary to make an adequate combination of filters.

In Figure 11 it is presented the signal with the Kalman filter activated and the impulsive filter deactivated.

Figure 12 shows the gyro signals output taking into account the both filters activated but in an inverse order as compared with Figure 9. This filters order inversion has eliminated some characteristics of the gyro output because some oscillations were interpreted as signal peaks. So it

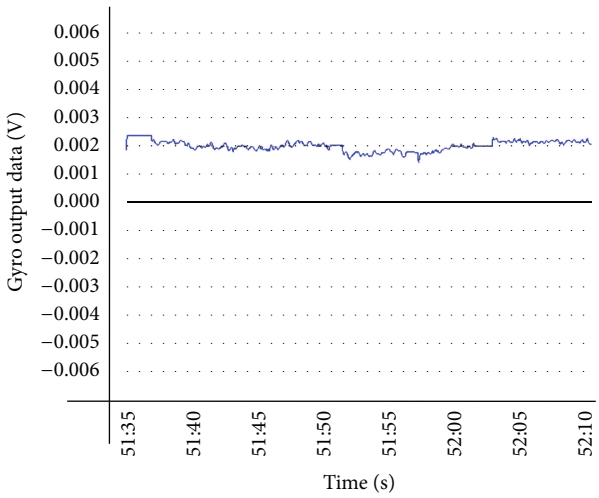

Figure 12: Graphical image of the signal with the Kalman and impulsive filters activated.

is recommended a careful analysis of the results when establishing the correct order of the filters.

6. Conclusion

Automatic detection procedures as developed in this paper are very important in space applications as characterization of occurrence, definition of criteria, and device types in the scenario of AOCS FDIR. This work implements a FDIR study for an IMU for space applications concerning failure detection, isolation, and recovery from faults affecting the hardware performance under of the FDIR software monitoring. The types of failure considered in this article are the step abrupt change, the ramp/drift/slow, the stuck, the cyclic, the erratic, the spike, and the stuck for variance. The validation approach was accomplished by using a set of a gyro data provided by the DMC's LABSIM at INPE. Analyses of the gyro signals were made by using different filters and also a combination of them. The results have demonstrated good agreement with the concepts applied in this study. In particular, referring to the filter applications, it should be pointed out that the inverted order could lead to wrong conclusions since some oscillations that could be interpreted as peaks may not be exactly peaks. So a careful analysis of the results is recommended when establishing the correct order of the filters. A framework for FDIR is developed that accommodates different filter analyses and an appropriate algorithm for a future study detection of each type of fault being analyzed. The FDIR software developed in this work includes the capability to implement fault event mappings for IMU.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the INCT for Space Studies, CAPES, Aeronautical Mechanics/ITA/SP, and Computer Science, PUC/SP, for the support to accomplish this work.

References

[1] M. Blanke and R. J. Patton, "Industrial actuator benchmark for fault detection and isolation," *Control Engineering Practice*, vol. 3, no. 12, pp. 1727–1730, 1995.

[2] M. Bartyś, R. Patton, M. Syfert, S. de las Heras, and J. Quevedo, "Introduction to the DAMADICS actuator FDI benchmark study," *Control Engineering Practice*, vol. 14, no. 6, pp. 577–596, 2006.

[3] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, vol. 32, no. 2, pp. 229–252, 2008.

[4] X. Olive, "FDI(R) for satellites: how to deal with high availability and robustness in the space domain?" *International Journal of Applied Mathematics and Computer Science*, vol. 22, no. 1, pp. 99–107, 2012.

[5] R. Isermann, "Model-based fault-detection and diagnosis—status and applications," *Annual Reviews in Control*, vol. 29, no. 1, pp. 71–85, 2005.

[6] ECSS-P-001B, Glossary of Terms, July 2004.

[7] Senior & Phil Ireland, "Modular architecture for robust computing (MARC)," in *Proceedings of the International SpaceWire Conference*, West Park Conference Centre, Dundee, UK, 2007.

[8] W. J. Kubat, "Application of strapdown inertial navigation to high performance aircraft," AGARD Lecture Series-Strap-Down Inertial Systems 95, 1978.

[9] R. J. Patton, F. J. Uppal, S. Simani, and B. Polle, "Robust FDI applied to thruster faults of a satellite system," *Control Engineering Practice*, vol. 18, no. 9, pp. 1093–1109, 2010.

[10] E. Oliveira, *Detecção e Isolação d e Falhas em Unidades de Medidas Inerciais com Redundância Mínima de Giros de Fibra Óptica [Ph.D. thesis]*, INPE, Brazil, 2011.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, Reading, Mass, USA, 1994.

[12] U. K. Krogmann, "Failure management in spatio-temporal redundant, integrated navigation and flight control reference-systems," in *Proceedings of the IEEE Position Location and Navigation Symposium (PLANS '90)*, pp. 330–337, Überlingen, Germany, March 1990.

[13] UML Object Management Group, *Group. UML 2.2 Super-Structure Specification (Formal/2009-02-02)*, UML Object Management Group, Needham, Mass, USA, 2009.

[14] L. A. Wang, B. Liu, and M. Lu, "A modeling language based on UML for modeling simulation testing system of avionic software," *Chinese Journal of Aeronautics*, vol. 24, no. 2, pp. 181–194, 2011.

[15] M. Muenchhof, M. Beck, and R. Isermann, "Fault-tolerant actuators and drives—structures, fault detection principles and applications," *Annual Reviews in Control*, vol. 33, no. 2, pp. 136–148, 2009.

[16] R. Isermann, *Fault-Diagnosis Systems: an Introduction from Fault Detection to Fault Tolerance*, Springer, Berlin, Germany, 1st edition, 2006.