TOWARDS ADVANCED FDIR COMPONENTS

Niklas Holsti, Matti Paakko

Space Systems Finland Ltd.
Kappelitie 6, FIN-02200 Espoo, Finland
Email: Niklas.Holsti@ssf.fi, Matti.Paakko@ssf.fi

ABSTRACT

Complex, autonomous spacecraft need powerful on-board Failure Detection, Isolation and Recovery (FDIR). We are developing a set of reusable FDIR software components called AFDIR. In addition to conventional failure detection methods (limit monitoring, trending, transient filtering, etc.), AFDIR includes Kalman filtering, Weighted Sum-Squared Residual test, Generalized Likelihood Test, Random Sample Consensus method, and various spacecraft simulations for computing "expected" values. AFDIR has two "integrative" diagnosis methods: *probabilistic reasoning* using Bayesian Networks, and *model-based diagnosis* using Causal Networks. The AFDIR *configuration space* data-structure compactly represents many configurations and allows selection of all spacecraft configurations for graceful failure-recovery. AFDIR is divided into an *algorithm layer* of purely callable subprograms, and a *structure layer* with a more convenient *declarative* interface. The chief data-structure is the *signal-processing network*.

1. INTRODUCTION

As spacecraft become more complex and more autonomous, on-board Failure Detection, Isolation and Recovery (FDIR) grows in importance. The FDIR subsystem should be able to detect and isolate more types of failure, perhaps including multiple failures or partial failures such as sensor performance degradation. Moreover, numerous spacecraft configurations are now available for recovery. When a failure occurs, recovery should choose the "best" spacecraft configuration allowed by the remaining healthy components.

FDIR methods and techniques are continuously improving, in the space domain and in other domains. Mathematical and statistical methods are intensively studied, and new *analytical methods* for failure-detection and sensor-fusion appear regularly. Recently, *probabilistic reasoning*, usually implemented by *Bayesian Networks*, has become a prominent diagnosis tool in ground applications, as have various forms of *model-based diagnosis*.

This paper describes a project that is developing reusable FDIR software components for some of these novel methods, as well as for traditional methods such as monitoring measured values against limits. The project is known as "Advanced FDIR" (AFDIR) and is Call-Off-Order 5 of the ESA frame project "Software For an Autonomous Satellite" (SFAS). *Space Systems Finland Ltd.* (SSF) is tasked with the major part of the AFDIR work, with *Astrium SAS* as the Prime Contractor (managed by Mr. Philippe David) and *ESTEC* as the ultimate customer (managed by Mr. Eric Bornschlegl). Expert support on probabilistic reasoning is provided by the *Complex Systems Computation Group* (CoSCo) of the University of Helsinki, under the leadership of Prof. Henry Tirri.

The AFDIR components are being implemented in Ada with some use of object-oriented programming (tagged types). The components will be available to the European space industry under ESA General Clauses and Conditions. The paper is organised as follows. Section 2 discusses the goals and methods of the project. Section 3 is an introduction to Bayesian Networks for probabilistic reasoning, and Section 4 introduces Causal Networks for model-based diagnosis. Section 5 presents our approach to defining spacecraft configurations. Section 6 explains how signal-processing networks connect the sensors to the FDIR methods. Section 7 concludes by reporting the current state of the project.

2. GOALS AND METHODS

2.1 Goals

The goals of the AFDIR project are:

1. To improve the on-board failure detection and diagnosis capability by new, powerful methods.
2. To improve mission survival and utility by gracefully recovering from failures and making full use of the available spacecraft configurations (including those with degraded operation).
3. To simplify FDIR implementation by making these new methods, as well as the traditional methods, available as reusable software components.

These goals are quite ambitious and pose significant technical challenges. One of the fundamental challenges is to understand which of the new FDIR methods are both useful and suitable for on-board use, considering their abilities and requirements. Practical challenges include implementing the methods under on-board constraints, and making them reusable by suitable encapsulation and sufficient documentation in the form of user manuals and examples.

We have tried to answer these challenges by a careful review of the state of the art, using the expertise of Astrium and CoSCo and the results of an earlier ESTEC and Astrium project [1]. Furthermore, we have defined a high-level software architecture that allows declarative definition of the spacecraft parts, spacecraft configurations, and FDIR functions. The first phase of the project was dedicated to requirements analysis and also produced a survey report on FDIR practice and prospects [6]. This led to a selection of the FDIR methods to be included, as follows.

2.2 Failure-Detection Methods

For failure detection, in addition to the simple and traditional methods such as limit monitoring, trending, transient filtering, etc., the AFDIR will include at least the following:

- Kalman filtering, to compute least-squares estimates of the real state from noisy or drifting sensors [8].
- Weighted Sum-Squared Residual (or Chi-Squared) test, which accumulates Kalman-filter residuals over a time-window and is more sensitive to small errors than a single-sample test.
- Parity-vector methods such as the Generalized Likelihood Test (GLT) [2], in which data from multiple, redundant sources (sensors) are cross-checked for consistency, and the aberrant data can be identified.
- Voting techniques based on the Random Sample Consensus method [4]. As GLT, this cross-checks redundant values, but is designed for errors with non-normal distribution (frequent occurrence of very large errors).
- Various spacecraft simulations for computing "expected" values. This includes attitude dynamics, reaction-wheel dynamics, magnetotorquer dynamics, orbit propagation, and a model of the Earth's magnetic field.

These methods are already well known in the literature, but not yet as common in space FDIR as they should be.

2.3 Diagnosis Methods

The above methods detect discrepancies between actual values (e.g. from sensors) and expected values (e.g. from models or redundant sensors). They can sometimes also identify the value or part that is in error. However, some failures are not directly visible in specific sensor values, but have indirect effects on other devices or functions. The AFDIR will include two "integrative" methods that can deduce underlying failures from multiple, superficial indications or symptoms: Probabilistic reasoning using Bayesian Networks and model-based diagnosis using Causal Networks. These methods are becoming common in ground-based systems, but not yet in space. It is hoped that the AFDIR project will make them more accessible for on-board use. Sections 3 and 4 describe these methods in more detail.

2.4 Isolation and Recovery Methods

When a failure has been detected or diagnosed, the actions for isolation and recovery must be chosen. Here we use the term "isolation" to mean isolating the failed part so that it cannot harm the spacecraft or its operation. This is usually done by changing the *configuration* of the spacecraft, for example by switching to a redundant, spare part, or by switching to an operational mode that does not use the failed part.

The recovery configuration has often been chosen to be a "safe" one, which ensures spacecraft survival but is not functional. Thus, mission performance has been interrupted until the ground controllers analyse the problem and command the spacecraft to a functional configuration. One of the goals of the AFDIR is to reduce the use of fail-safe

modes in favour of fail-operational modes. Failure-recovery should place the spacecraft in a configuration that isolates the failed part, but is still as functional as possible.

Such *graceful degradation* is only possible if the on-board recovery process can choose from all possible spacecraft configurations, rather than a small set of predefined configurations. The AFDIR supports this by providing a data structure, the *configuration space*, that compactly represents a large number of configurations. Section 5 explains this in more detail.

3. BAYESIAN NETWORKS

Failures often affect many different observable values and behaviours. The task of diagnosis is to consider all of this "evidence" jointly, and infer the most likely common cause: the part that has failed. A *Bayesian (belief) Network* (BN) is a special representation of the *joint probability distribution* of all the relevant variables, both observable variables (e.g. sensor values) and hidden variables (e.g. whether the sensor works).

The variables are usually assumed to be discrete, often Boolean. Each node in the network represents a variable. The directed arcs between nodes represent the probabilistic dependencies between variables. Each node contains the conditional probability distribution of the corresponding variable, conditioned according to the values of the parent variables. To use a BN, the observed facts (or beliefs) are entered by setting the states of the corresponding nodes, and then Bayes' rule is used to compute the posterior probability distribution of the other (unobserved) variables.

The inputs to the BN will usually not be raw sensor readings, but discrepancy-measures from lower-level failure-detection methods. This answers questions such as: "Given a 70% probability of a discrepancy between sensors 1 and 2, and a built-in-test warning from sensor 2, but no built-in-test results (positive or negative) from sensor 1, how likely is it that sensor 1 (or 2) has failed?"

For FDIR purposes, two notable advantages of BNs are that they can deal with *missing* data and with *probability* data. The first property makes it easy to use the same BN in different configurations, for example with different sensors active. The second property means that the basic failure-detection methods, such as the GLT, do not have to discretize their error-measure against a fixed threshold. Instead, the probability of the error-measure (e.g. chi-squared) can be fed as such into the BN.

Fig. 1 shows a BN for spacecraft attitude sensors. The BN contains input nodes for failure-detection tests (two bottom rows), part health (top row) and on/off-status (second row). The middle row output nodes indicate probabilities for part failures. For example, the autonomous star camera 1 (ASC1) failure depends only on the ASC1 health. On the other hand, ASC1 failure and on/off status affect the ASC1 built-in test (BIT), several correlation tests (CT) and a GLT test.

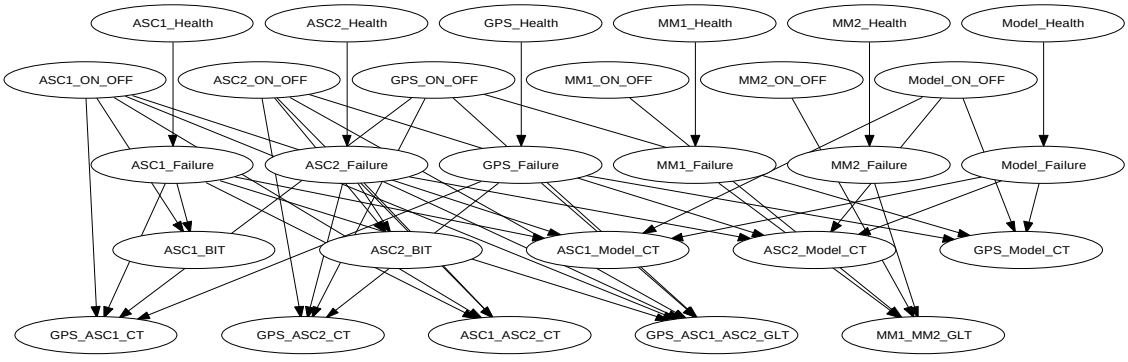

Fig. 1. A Bayesian Network

There is some similarity between a BN and an FMECA or FTA structure, which should make BNs easier to understand for engineers. The AFDIR will contain software tools to prepare and pre-process BNs on the ground, following [5], and on-board software components to evaluate BNs as part of FDIR execution. See [7] for additional information on BNs.

4. CAUSAL NETWORKS

Model-based diagnosis is a set of methods for reasoning about system behaviour by means of a *physical* or *causal* model of the system. For the AFDIR, we have chosen a type of model called *Causal Networks* (CN) [3]. In this model, a system is composed of parts, where each part has zero or more Boolean inputs and computes one Boolean output as a function of the inputs. The parts are connected in a directed acyclic graph by connecting some outputs to inputs. The idea is to model error propagation by Boolean equations of the form "if the input to component C is correct, and C works, then the output from C is correct." The identifiers in the Boolean equations are divided into two groups: *variables* and *assumables*. A variable usually represents correct status or data at some point in the system, such as "the input to component C is correct". Some variables are observable. An assumable usually represents the correct operation of a system component, such as "component C works", and is not directly observable.

The top part of Fig. 2 depicts a spacecraft telemetry system with redundant buses, encoders and transmitters, a radio-frequency switch, two nadir antennas and a zenith antenna. The variables and assumables for a CN are also shown. The bottom part of the figure shows the CN graph and Boolean equations for Encoder 1. The first equation says that if encoder 1 gets telemetry and power and is OK then it sends data. The rest of the equations state that the encoder does not send data if it does not get either telemetry or power.

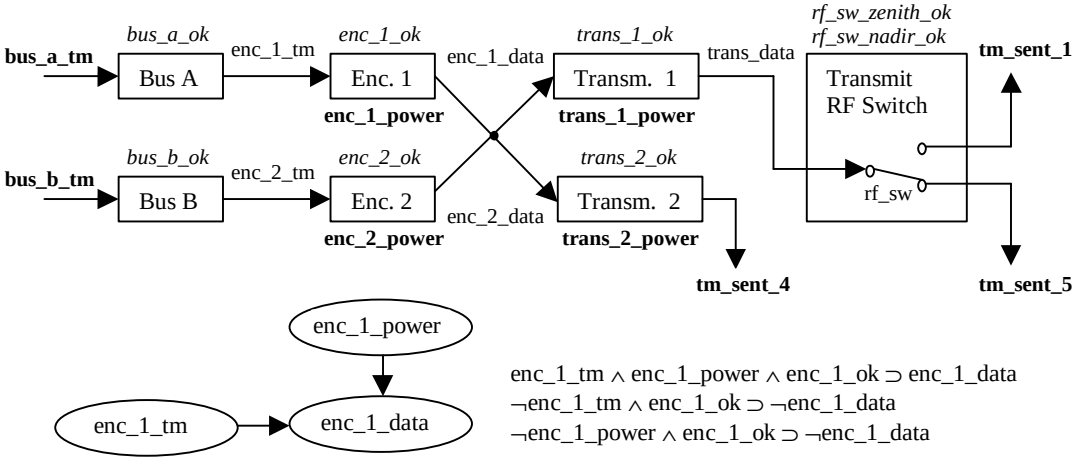

Fig. 2. A Causal Network

The goal of the diagnosis is to explain the observed variable values (e.g. the detected errors) by choosing suitable values for the assumables, which is the same as choosing components that could have failed. For example, if we observe that the input to component C is correct, but the output is wrong, then we must assume that "component C works" is false. It is possible to find the *minimum number of failures* that must have occurred to explain the observations.

A CN is very similar to a circuit diagram or a functional block diagram. This should make them even more usable by engineers than BNs. However, CNs are somewhat heavier to compute than BNs. The AFDIR will contain software tools to prepare and pre-process CNs on the ground, and on-board software components to apply CN as part of FDIR.

5. CONFIGURATION SPACES

The configuration-space data-structure lets the AFDIR evaluate a large set of possible spacecraft configurations to find the best recovery configuration. To explain this data-structure, we must first explain some terms related to parts, states and configurations.

The AFDIR considers the spacecraft to consist of a fixed set of *parts*. A part can be a physical part, such as a sensor or a power supply, or a software function such as the AOCS control-loop. Each part can be in some *state*, where different parts may have different sets of states. The state describes both the physical state of the part, such as *on* or *off*, and the role of the part, such as *on line* or *standby*. For software parts, the state will often reflect the "mode", so the states of an AOCS control-loop could include *rate-reduction*, *Sun-pointing*, and so on. A *configuration* defines the state in use for each part. Thus, it expresses both which parts are on and which are off, and the operational role of each part.

For a given spacecraft, only some configurations are allowed, and the rest are forbidden for reasons of spacecraft design or limited resources. There may be so many allowed configurations that listing them explicitly would be cumbersome and error-prone. An AFDIR *configuration space* structure represents the set of configurations *implicitly*, by constructing it from simple, partial configurations that are combined with set operations. Operations are supplied for *n*-tuple redundancy, *m*-of-*n* redundancy, free and constrained hierarchical composition, unions, intersections, etc. Fig. 3 shows a configuration space for spacecraft attitude sensors. A *Part_Range* for sensors in state *on* is *Join*ed with a *Quorum*, which shows how many sensors can be in different states. A *Union* of the *Join* with a *Listing* where all sensors are o*ff* completes this configuration space.

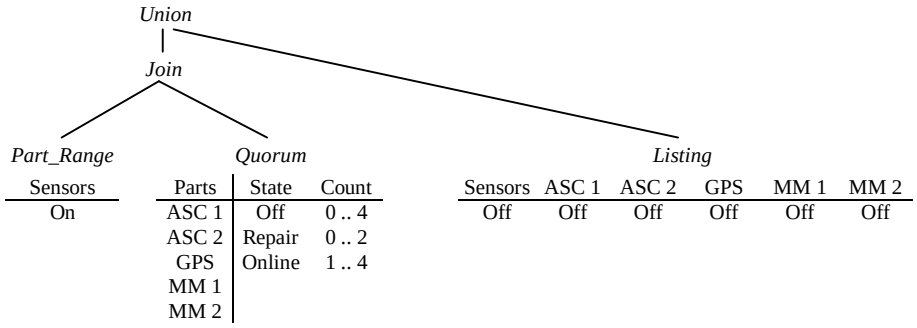

Fig. 3. A Configuration Space

A configuration space constructed in this way can be traversed (searched) by the AFDIR recovery process. Each configuration can be evaluated in terms of the *quality of service* (QoS) it provides for the several spacecraft functions, such as attitude control, TC/TM operations, etc., and in terms of its overall *utility* for mission success. When a failure is detected in a part, the failure is isolated by limiting that part to some subset of its states. For example, a totally broken part is limited to its *off* state, while a sensor with degraded precision can be limited to *off* and *coarse-control* states. The AFDIR recovery process finds the "best" configuration that obeys all such limitations.

6. AFDIR SOFTWARE ARCHITECTURE

As for all on-board software, the AFDIR has to satisfy constraints regarding processing resources, real-time performance, and quality. However, additional problems are caused by the goal of reusability. One problem is that some FDIR methods are very simple to program, for example the checking of sensor values against min/max limits. When such a method is encapsulated as a component, reuse is profitable only if it is very easy and causes very low overheads. On the other hand, some advanced FDIR methods have a complex interface to the inputs and outputs. The "glue code" needed to reuse an encapsulated component may be complex, again reducing the profit of reuse. An FDIR subsystem interacts with both low-level aspects such as sensor sampling, and high-level aspects such as spacecraft configuration and mode. The interface of the AFDIR components must adapt to the spacecraft's peculiar constraints on data availability, unequal sampling periods, skewed and jittered sample times, units of measurement, etc.

The architecture chosen for the AFDIR addresses these problems in two ways. Firstly, the AFDIR is divided into several levels, starting from the simpler mathematical algorithms such as Kalman filters, and ending with the advanced BN and CN levels. Each level depends only on the lower levels, and so the AFDIR user has some flexibility in which parts of the AFDIR to reuse. Secondly, each AFDIR level is divided into two layers: An *algorithm layer* that contains purely callable subprograms, and places no constraints on the software architecture of the caller, and a *structure layer* that implements a declarative interface to the AFDIR. This simplifies the programming, as long as the AFDIR architecture is followed. To avoid the architectural constraints, the user can access the AFDIR via the algorithm layers, but the interface is then procedural and more complex.

The chief data-structure of the structure layers is the *signal-processing network*. It is similar to the "block diagrams" used with MATLAB and other tools, and consists of signals and signal-operators that are connected into a data-flow network. A *signal* represents a stream of values that can be sampled or continuous. A *signal-operator* is a processing function that takes some signals as inputs and produces some signals as outputs. The AFDIR provides predefined types of "source" signals for sampled sensor data and software variables, and predefined signal-operators for scaling, resampling, interpolating, filtering, and so on. There are also generic signal-operators into which an AFDIR user can

program any type of signal-processing function. Fig. 4 shows a signal-processing network for a correlation test. The signals *S* and *R* are synchronised, subtracted and normalised, and finally it is checked if the norm is below a threshold. The correlation test result is an input for a BN.

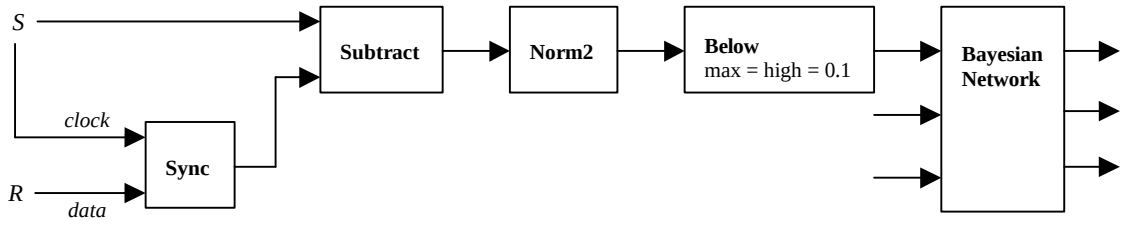

Fig. 4. A Signal-Processing Network

All the AFDIR failure-detection and diagnosis methods are provided as signal-operators. To construct a signal-processing network, the AFDIR user writes declarative Ada statements that create signals with specific parameters, and instantiate signal-operators with specific input signals and parameters. The network is thus created at program initialisation, and then evaluated periodically by the AFDIR.

7. PROJECT STATUS

At the time of writing, the AFDIR is nearing the end of its Architectural Design phase. Prototypes of the analytical failure-detection methods, BN and CN have been implemented and tested in a limited way. As a pilot application we have defined a hypothetical satellite called ASOS, for "Advanced Smart Observation Satellite". ASOS resembles the small Earth-orbiting satellites being built today, with an attitude-control system using autonomous star cameras, magnetometers, magnetotorquers and reaction wheels. AFDIR will be used to implement the FDIR subsystem of the ASOS on-board software, but the rest of the on-board software will be just a harness for testing the FDIR part.

The AFDIR is expected to be completed in the third quarter of 2001. Because the project is future-oriented and in several respects tries to extrapolate current FDIR practice, we appreciate the opportunity to present it at this stage to a wide audience, in the hope of constructive feed-back that could guide the final design and implementation phases.

8. REFERENCES

1. Borde, J., Implementation of AOCS Configurations. Technical Note 5: Fault Management System and Redundancy (WP 250). Matra Marconi Space, reference S260/JB/NT/17.96, Issue 1, 6/26/96.

2. Daly, K.C., Gai, E. and Harrison, J.V., Generalized Likelihood Test for FDI in Redundant Sensor Configuration. *Journal of Guidance and Control*, Vol 2, No 1, Jan.-Feb. 1979.

3. Darwiche, A., Model-Based Diagnosis under Real-World-Constraints. *AI Magazine*, Vol 21, No 2, pp. 57-73, Summer 2000. http://www.cs.ucla.edu/~darwiche/realize.ps.

4. Fischler, M.A. and Bolles, R.C., Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis. *Communications of the ACM*, Vol 24, No 6, pp. 381-395, June 1981.

5. Huang, C. and Darwiche, A., Inference in Belief Networks: A Procedural Guide. *International Journal of Approximate Reasoning* 11:1-158, 1994. http://www.cs.ucla.edu/~darwiche/cs262a/hd_inference_ijar.ps.

6. Huvinen, J., Advanced FDIR: Survey Report on FDIR and Robustness. Space Systems Finland, reference SSF-SFAS-RP-007, Issue 1.0, 10.04.2000.

7. Myllymäki, P., Material on Bayesian Networks and related topics. http://www.cs.helsinki.fi/research/cosco/Bnets/.

8. Welch, G. and Bishop, G., An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, TR 95-041, June 6, 2000. http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html.