Modelization, Failures Identification and High-level Recovery in Fast Varying Non-linear Dynamical Systems for Space Autonomy

Michelle Lavagna, Guido Sangiovanni, Andrea Da Costa
Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Italy

Abstract

This paper shows the realization and validation of a detection, identification and recovery module for hardware failures on ESA GOCE satellite, merging two independently developed modules. The work is currently focused on its attitude control subsystem, following a previous work on the power subsystem. This application is more challenging than the former because of two reasons. Control system, sensors and actuators are highly interdependent; hence to correctly understand which is the faulty device is definitely a hard task. Secondly, the dynamical behaviour of some components is related to very high frequencies, asking for a finer attention to be paid in the neuro-fuzzy model identification phase.

Introduction

The increased complexity of space missions implied, in recent years, a growing demand for on-board autonomy enhancement. In this context, autonomy means to carry out most part or the whole mission with a minimal assistance by Earth. To this aim, three characteristics are judged to be essential: the ability to decide which activities to perform and when to do them taking into account e.g. on-board resources constraints, the capability to really execute them, and the ability to cope with unexpected events as faulty devices or contingencies. The most relevant example of such a system is NASA Remote Agent, flown in 1997 on the DS-1 mission [6]. In general, those agents are formed by three modules dedicated to the aforementioned characteristics, i.e. a Planner, an Executer and a Failure Detection and Identification module (FDI). In previous works we developed a planner system [5], based on an innovative decision making approach and, independently a FDI module [3]. The present paper will show the first steps towards the merging of the two systems, with the implementation of a Reconfiguration module compatible with the FDI, using the same knowledge base of the planner.

The basic methods which underlie the Planner module are an iterative repair policy for constraints violations resolution and hierarchical decomposition of tasks. In particular, the decisional nodes involved in the solving process are handled with classical decisional methods coupled with rules expressed by means of Fuzzy Logic. The FDI module uses two different methodologies to solve the detection and the identification problem in an integrate perspective. Neuro-Fuzzy Inductive Reasoning (NeuroFIR) methodology allows creating a very reliable mixed qualitative and quantitative model of a general non-linear dynamical system using significant measured data. Whenever any system change occurs, e. g. because of damage, the so-called “envelope” approach evaluates the entity of the deviation
from the reference model and triggers an alarm. After deviation detection, i.e. when a failure presence is stated, Zadeh’s Possibility Theory provides a qualitative treatment of measures and uncertain symptoms in order to make a diagnosis of its causes. Once the identification is done, some recovery strategy is defined, involving the high level encoding used by the planner that rules the spacecraft.

The whole architecture is shown in Figure 1, where it is possible to notice the three modules and more in detail the connections among the elements of the failure manager. For lack of space, all technical details are not reported but they could be found in the references.

![Figure 1: Overall architecture for the FDI, Reconfiguration and Planner modules](image)

**Failures detection and identification**

The application of our FDI methodology [1], [2], [3] to a fast varying dynamical system like GOCE AOCS subsystem is highly challenging because of the recursive dependence and the deep connection of each elements. The nominal model is created by sampling at a particular frequency the I/O data, thus generating pseudo-static qualitative rules: this frequency depends upon the dynamic of the system. If the system has slow dynamics, e.g. the EPS s/s, a large time step could be used, while in presence of fast varying signals this step must be so small to allow their descriptions. A problem occurs: the higher the frequency is, the bigger the number of the rules and the deeper the mask must be. In general, in diagnosis problem only plant slow dynamics reproduction is sufficient, so a low-pass filter has been applied to the AOCS I/O data. The choices of the cut frequency of the filter and the time step for the sampling phase have been performed following two criteria called completeness and minimality. A set of samples is complete if it overlaps all the trajectories of the system in the phase space and it is minimal if cardinality is equal to the minimum number of samples sufficient for describing such trajectories [1].

Once the qualitative/quantitative nominal model is created, the Neuro-FIR simulation engine is able to forecast system outputs using the current input set. An envelope is also generated around the real trajectory representing a kind of interval
of acceptability of the prediction: if the actual value is outside the envelope, an
instantaneous error is calculated. Those errors are collected during a time window
whose size is defined by the user and an alarm is activated if such cumulative error
exceeds a threshold value, meaning that a deviation from the nominal behaviour,
i.e. a failure, is present. The envelope approach is thus based on dynamic bounds
around the measures instead of static ones like in the FMECA analysis.

The identification phase aims to judge which type of failure has occurred:
Possibilistic Logic has been chosen to manage the uncertainty related to this
process. One knowledge island (KI) for each failure is defined: given the full
certainty of a malfunction, some symptoms must be present or absent with a certain
degree of necessity. After the definition of the knowledge base, the identification
phase can begin by defining the fuzzy set of possible causes of the symptoms
observed: it contains all the disorders that are (more or less) consistent with the
symptoms. In the other two fuzzy sets there are disorders whose some of its certain
consequences are present or some of its impossible consequences are absent, and
the disorders whose effects cover the observations. This approach allows to
contemplate multiple and unknown failures, under the condition that the KIs
consider all the details of all the possible interactions [2].

Planning system: domain database main features

Lower level: state variables

The whole planning scenario can be described at two levels of abstraction: ‘state
variable’ and ‘activity’. The state variables are parameters useful to describe the
current state of the spacecraft: typically there are two kinds of variables: either
internal, like a payload device, or involving the whole system, like attitude or
position. The actions performed either by the system itself or by some other
external events can change over time the value of those variables.

<table>
<thead>
<tr>
<th>State Variable: DFACS (attitude control)</th>
<th>State Variable: EGG (gradiometer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicates:</td>
<td>Predicates:</td>
</tr>
<tr>
<td>Ultimate Safe Mode (USM)</td>
<td>Science</td>
</tr>
<tr>
<td>Coarse Pointing Mode (CPM)</td>
<td>Acquisition</td>
</tr>
<tr>
<td>Fine Pointing Mode (FPM)</td>
<td>Anomaly</td>
</tr>
<tr>
<td>Calibration (CAL)</td>
<td>Stand_by</td>
</tr>
<tr>
<td>Drag Free Mode (DFM)</td>
<td>Getter</td>
</tr>
<tr>
<td></td>
<td>Survival</td>
</tr>
</tbody>
</table>

Example of transitions matrix:

<table>
<thead>
<tr>
<th></th>
<th>Sc</th>
<th>Acq</th>
<th>An</th>
<th>St_b</th>
<th>Get</th>
<th>Surv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sc</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Acq</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>An</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>St_b</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Get</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Surv</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![Figure 2: State transitions example](image)

Each variable has a finite set of values, described by logical predicates. We must
define legal transitions between couple of predicates, in order to define the
permitted evolution of the system over time. Figure 2 reports an example of
variables and enabled transitions. Moreover, it has to be noted that normally
variables have a particular preferable predicate (e.g. payload turned off); according
to that, the transitions applied can be seen as ‘perturbations’ of that predicate, here
called ‘default value’ of the state variable.
**Higher level: activity**

The activity is the main character in the scheduler perspective. A generic activity typically has: a finite duration, a set of state variables involved in its execution and some amount of resources demand (power, memory...) to be accomplished.

We settled two activity types: the goal and the subgoal. This distinction is made in a logical framework: a goal is a ‘stand-alone’ activity, that is some task either representing the aim the physical system has been built for (i.e. a scientific observation) or useful for the proper course of the mission (i.e. memory dumping).

Every other task needed to acquire the system right configuration to execute a goal activity, is here defined as a ‘subgoal’. In this perspective, we can imagine goals as requests explicitly invoked by final users or ground controllers, or even by the spacecraft itself, whereas the subgoals as activities introduced by the planner to guarantee, as stated, the proper execution of goals. We note that which subgoal is needed every time is strictly related to the current spacecraft status, but, thanks to the default values, we can foresee the states transition requested once (from the default to goal state) and so which subgoal. In this way causal nets of activities can be automatically generated before the real planning process occurs, similarly to the hierarchical task network techniques; this kind of aggregated activities is here called a ‘procedure’, and it will be the real subject of the scheduler.

Although procedures are stated before the planning process, some parameters can be defined only during the resolution of plan, because of their temporal or logical dependence on other procedures. A typical example is the duration of a slew manoeuvre, depending from the current attitude and attitude to be acquired, which, in turn, depends from e.g. the position of the ground target to reach and the position along the orbit. For this parameter and as well as many others we define come custom functions: ad hoc routines called by the scheduler and linked with specific activities, having as inputs the state variable values, temporal allocation of activity and other parameters settled by the user (e.g. slew rate) and modifiable only either by the user herself or by other software modules outside the planner.

![Diagram](image)

**Figure 3:** Procedure structure example (goal: take a picture)

**Resources management**

We have identified three kind of resources, on the basis of different behaviour: we have aggregate (e.g. battery charge), renewable (e.g. memory) and punctual resources (e.g. power). We note that also for resources propagation could be useful to build special custom functions: thinking about the battery resource, we need to compute the on-board generated power, depending on spacecraft attitude and solar arrays configuration, for memory usage we need to know the way used by the payload to acquire data and how the communication system can dump data to Earth.
Reconfiguration: basic ideas

Speaking about actions undertaken whenever something wrong has been detected, we start defining two different levels on which the system can work: the first, dealing in a very small time horizon and quick reaction, is called ‘recovery’, an example could be the MIR module of the Remote Agent Experiment [6]. It deals with the Executor level of detail and real time constraints. The second level, more abstract, is the high level reconfiguration, or, simpler, the reconfiguration. It deals with the Planner perspective, that is, activities, resources, temporal relations and a wide time horizon: its aim is to take into account for futures planning processes the degraded conditions of the spacecraft, keeping exploiting the system at the best. Recovery and reconfiguration are in some way complementary: a quick reaction to unexpected events (namely, failures), to allow anyway the execution of planned activities and to avoid catastrophic effects due to failed devices, and a meditated imposition of further constraints for futures activities, e.g. a less availability of resources, implying a plan re-computation under the new conditions. In this paper we focused on the second aspect, the so-called high-level recovery.

We assume that the Reconfiguration module can be informed about the occurred failure in two ways: either by the Failure Detection and Identification Module (FDI), able to understand what happened, or by a simple comparison among state variable values currently recorded and forecasted in the previous planning session, hence blind about the real reasons of the observed behaviour. The assumption to be able to map the telemetry data recorded by on-board sensors into the planner’s variables, coded into logical predicates, and to monitor the status of on-board resources, is here posed.

Generally speaking, the Reconfiguration module is a rule-based framework, where antecedents can be occurred failure codes and spacecraft status, the subsequent some variation to be done into the planner database. If some rules are activated, a re-planning session is invoked and the schedule recomputed.

More in detail, we can divide the subsequent in two categories, depending on which part of the database they affect: changes for the static database and changes on the policy of planner. The first ones are variations on the boundaries of resources, (e.g. due to loss of a solar panel string), variations of transitions matrix or default values for state variables, parameters of custom functions, duration of activities and so forth. We remark that for each goal activity it is possible to define several paths to accomplish it, namely with different set of requested variables; therefore the system will check whether the new conditions (e.g. the default value for ‘main_antenna’ now is ‘broken’) are compatible with the current way to execute an activity (e.g. the ‘dump_memory’) and then, whenever possible, will switch on the new legal way for the activity (e.g. using the low gain antenna: less data rate, less power, increased duration for the same amount of data). Furthermore, some goals might be described like operative modes of the spacecraft: for GOCE the acquisition of data is continuous for several orbits, and no matter precisely whenever the mode is started, making this kind of activity very different from e.g. taking picture; for this reason we allow to temporarily suspend this kind of goals, after the reconfiguration will be
possible to re-activate the goal, or, if the conditions make the activity infeasible, to delete it. If the goal can be re-activated, the reconfiguration module provides to put the spacecraft into the right state requested by the goal itself.

Periodicity of a goal and its priority could be changed too: e.g. we can increase the calibration occurrence rate of a payload or decrease the importance of the secondary payload, for example when the ability to send data to Earth is reduced. The second category involves a kind of ‘active role’ for the planner: according to which failures, which current status and on which changes occurred by the previous cited rules, the system can decide to insert new activities, either to warn ground controllers about the on-board situation, to make the next planning process easier (e.g. turn off some device before every eclipse if the battery is degraded) or to deeply understand the situation on-board, making a more accurate identification process. Moreover, since the planner module here considered rely on a decision-based system [4],[5] it could be easy to tune conflict resolution policies by changing some weights on different criteria, depending on the new priorities of the mission and maintaining the planning effort low, even though e.g. resource boundaries became a very constraining condition to meet: this can be done making the planner more incisive and decreasing the preference for conservative repair actions. We note also that the occurrence of multiple failures will be not a problem, each of them will be propagated for rule activation: a special check module will guarantee that double activity insertion request will not be sent to planner or no conflicts rise among rules (e.g. we want avoiding scheduling a mode no more attainable because of new transitions matrices).

Figure 4: Overall architecture of the Reconfiguration Module

Simulation Results

As reported in [2] the reference scenario selected for the validation of the aforementioned approach is the ESA mission GOCE, studied during the European Space Agency research project Smart-FDIR, coordinated by Alenia Spazio, Turin, Italy, with Politecnico di Milano acting as subcontractor. The objective of the
GOCE mission is to produce high-accuracy, high-resolution, global measurement of the Earth gravity field. It is a nadir-pointing satellite on a sun-synchronous dawn-dusk orbit, with 6 solar arrays (4 body mounted and 2 on wings) and a Li-ion battery made up of 8 cells. Sensors are one GPS receiver, a star tracker, a coarse sun sensor assembly, a coarse Earth sensor assembly, a 3-axes magnetometer, plus the gradiometer which calculate the accelerations. Actuators are 8 micro-thrusters, 2 ion thrusters and magnetic torquers. Alenia has developed one simulator for the power subsystem (EPS) and one for the AOCS subsystem (called E2E simulator), which produce I/O data both for nominal and faulty modes. Among all the possible combinations, two scenarios and four reconfiguration examples have been chosen to show the performances of the Smart-FDIR approach.

Reference models and detection parameters

Simulations last one orbital period, i.e. 5370 seconds, in EPS case and 1.5 periods in E2E case; reference models has been built sampling data every 50 seconds. This value has been chosen by deeply analyzing the dynamical behaviour of the spacecraft subsystems: it represents a good compromise satisfying the two criteria of completeness and minimality for both the slow subsystem and the fast-filtered one. A Butterworth digital filter with a stop frequency of 0.04 Hz has been used to remove fast dynamics from E2E signals whose frequencies are around 0.03-0.08 Hz. Size of the cumulative error window and threshold for the alarm activation have been fixed in 5 and 3 respectively, thus meaning that failure identification module has a minimum innate delay of 150 seconds. All details regarding the choice of the inputs/outputs variables used to create reference models are reported in [2].

Triple failure on the EPS subsystem

At time 1000 s, two failures are simultaneously injected in the EPS simulator; the first one is the solar array string loss (code F21) and the second one is the complete loss of a battery string (F25/F26). Moreover, at time 3500 s, another failure happens: the battery connection to the charge regulator switches off (F1 or CR). Failure F21 produces a reduction of 1/27 (less than 4%) of the power supplied by the solar array so it is defined “hardly to be detectable” with traditional methods; in a similar way, after the injection of F25/F26, battery tensions has a fall of 12.5%, but still remains in the acceptable range. These two failures are identified after three time steps, i.e. when the alarm threshold is reached, with a “complete coverage” result, thus showing the excellent efficacy of the proposed approach. The last failure (F1 or CR) is identified one time step after its injection but the F25/F26 is judged absent while it is still present. This misidentification depends on the fact that this failure causes the state of charge of the battery to be constant, thus inactivating one symptom of the F25/F26 knowledge island. Figure 5 shows also that it is possible that the minimum voltage is reached (F3) because of the loss of one battery string: the main advantage of the proposed approach is that the reaching of such a critical limit is recognized as an incipient failure thanks to fuzzy set theory.
failures on the AOCS subsystem

After the end of the transitory, a failure in the microthruster n°3 (MTA3) is injected at time 3500 s and removed after 1000 s; then a failure on the GPS system is introduced at time 6500 s; Figure 6 shows the capability to associate to the observed symptoms either a sensors or an actuators failure. While GPS failure is identified three time steps (150 s) after its injection, MTA3 failure is identified with a bigger delay because of the presence of the filtering phase (8 time steps, i.e. 400 s). The unknown failure signal activates at 6800 s because the spacecraft drifts under erroneous measurements so the angular errors exceed the limits related to nominal case (“out of bounds” symptom). The presence of this fuzzy symptom in the second and third KIs is responsible of the high variation of the failure controller and unknown failure signals; though a degree of possibility of these failures is present, it does not lead to a misidentification because the alarms are not activated and the coverage is not complete, thus showing the advantages of the proposed approach.

Reconfiguration examples

Case 1: Loss of panel string and Degradation of battery

The system changes some parameters in the custom function devoted to compute the generated power. In this way we can propagate properly the battery charge profile.

We re-compute dynamically the new maximum capacity of the battery and, eventually, the efficiency on the charge lines. If one of two efficiencies falls below the pre-fixed threshold of 0.6 for charge or 0.89 for discharge, an insertion rule is activated, asking for a dedicated Earth contact.

Case 2: Charge regulator switch off

This is a severe failure that can compromise the overall mission. An emergency link is inserted with the maximum priority, meaning that the first available contact will be used. We note that for less critical failure a kind of trade off among ground stations, their estimate workload and temporal closeness of contact window is automatically performed inside the decision based planner.

Case 3: Earth sensors failed

We show the reconfiguration after an Earth sensor failure as it is a more interesting example than the GPS one which simply implies switching on redundant instrument. If the Earth sensor fails, it becomes impossible to acquire nadir pointing without star trackers or other sophisticated devices. The Coarse Pointing Mode can’t be maintained, the new mode for safety will be the Ultimate Safe Mode (very coarse pointing to the sun, more aerodynamic drag). If the spacecraft is in the science mode, it uses star trackers for attitude determination, so the effect of the failure is not immediately visible, but in future emergency situations the system will act according to new database.

Case 4: Corrupted Gradiometer data
In case the scientific data are unexpectedly corrupted, we decide to increase the calibration activity duration from one day to one and a half. In the same way we can increase the occurrence rate e.g. twice a month. Actually, in GOCE mission this activity is performed only during a specific phase of the whole mission horizon (the POP phase); we assume that calibration will be performed also in other portions of the mission.

Figure 5: FDI results for the EPS faulty scenario

Figure 6: Forecasted signal with envelope, Knowledge Island and FDI results
Case 1
- Failure: degradation_of_solar_arrays
- Failure: degradation_of_battery
- Resource: 'energy', the new boundaries are: min, 70, max 72
- Parameters of custom function 'energy_calculate' have been changed, cell_number
- Activity to be inserted → link. This rule is enabled for 10000 time units

Case 2
- Failure: battery_switch_definitely_stuck
- State Variable: DFACS, switch on another attribute has been requested
- State Variable: EGG, switch on another attribute has been requested
- Resource: 'power', the new value/default is: 300.000000
- Parameters of custom function 'energy_calculate' have been changed, charge_battery_efficiency
- Activity to be inserted → emergency_link. This rule is enabled for 10000 time units

Case 3
- Failure: earth_sensor_failed
- State Variable: DFACS, the transition matrix has been changed

Case 4
- Failure: EGG_corrupted
- Activity Params: EGG_calibration, the usual duration value has been changed, the new value is 129600

Figure 7: Reconfiguring actions

Conclusions

This paper has shown the successful improvement of the FDI module and the first steps toward the realization of a complete Remote Agent. The FDI module is now able to identify failures on fast-varying dynamical system as proved on the GOCE AOCS subsystem. The developed reconfiguration module represents the link between this item and the planner: it uses the same knowledge base and takes advantage from the decision-based approach, enabling in a simple way – with conditional rules – to modify the resolution policy. In the same manner, it becomes easy to introduce new useful activities, resulting as self-generated by the spacecraft and then coherently allocated by the planner. Generally speaking, this reconfiguration system could be thought as a complementary module to the classical recovery, acting on a more abstract level, thus able to adapt the higher level mission tasks to new degraded conditions.

References