EUCASS2009-219

Space missions FDIR[1] ENGINEERING AND VERIFICATION

This paper has been prepared from experience of ASTRIUM Space Transportation and results from ASSERT[2] study [1]. Written by Michel Turin (GTI6), Christel Seguin (ONERA), Pierre Bieber (ONERA), and David Lesens (ASTRIUM Space Transportation).

Abstract

Space missions FDIR engineering is a difficult compromise: either you increase the coverage of your failure detectors but you will increase simultaneously the false alarm rate and thus decrease your availability rate. FDIR engineering is part of vehicle engineering and is defined all along the development taking into account failure modes of equipments and subsystems and their effects often known late. This fact explains why FDIR is often a matter of late modifications on architecture and Application Software.

FDIR verification is one of the most difficult and time consuming activity due to the fact that Hardware in the Loop tests with real equipments cannot simulate all failure modes.

Models can be used to depict the various system failure modes, their propagation inside the system and their potential interaction with the FDIR. Such models can be used to assess the system architecture dependability.

Introduction

Space missions: spacecraft, launchers, re-entry vehicle have a communal design goal: insure the mission in presence of failures. System level engineering defines several classes of mission according to RAMS requirements:

- Highly reliable mission: spacecrafts with up to 15 years mission duration
- Highly available mission: space mission with time critical events as boost manoeuvres, launcher/re-entry vehicles.
- High safety mission: manned vehicles or mission with some safety critical phases (as ATV[3] [2] with its “CAM” Collision Avoidance Manoeuvre)

To fulfil the RAMS requirements, avionic design has to define an automatic FDIR strategy based on Failure Detection, Failure localisation & Isolation and Failure recovery.

FDIR engineering

FDIR engineering is a difficult compromise: either you increase the coverage of your failure detectors but you will increase simultaneously the false alarm rate and thus decrease your availability rate. You can specialise the failure detection to each type of failure making FDIR more complex. I will give some example

[1] Failure Detection Isolation and Recovery
[2] FP6 IP project coordinated by ESA http: //www.assert-project.net
[3] The Automated Transfer Vehicle mission has been launched during 2008 with a successful automatic Rendezvous with the International Space Station.

of failure (or unexpected behaviour) during flight and the response of FDIR was not always the expected one.

Power system failure is a feared event due to the rapid failure propagation and their common mode effects (as thermal effects), and the FDIR reaction time shall be adapted.

Gyrometer failures are a common cause of spacecraft end of life; but it is difficult to detect the failure and start a recovery with back-up gyrometers early enough before the spacecraft will start spinning.

Thruster leakage has several failure effects: disturbance torques and pressurization decay, and the failure detection is rather difficult to discriminate between a nominal behaviour and a failure case. Thruster redundancy management is not easy with 6 DOF control.

Computer and memory upsets due to space radiation have an instantaneous effect which can be catastrophic, the FDIR strategy shall be adapted to recover the mission either nominally or in a degraded mode using independent context memory, back-up computers, or triple modular redundancy with vote.

These failures are only examples; they are many other types, the knowledge of the failure mode, its probability and its effect shall be known early during the vehicle design phase.

So, FDIR engineering part of vehicle engineering, is defined all along the development taking into account failure modes of equipments and subsystems and their effects on the system, often known late, when the detailed design is known and subsystems FMECA provided to the prime contractor . This fact explains why FDIR is often a matter of late modifications on architecture and Application Software.

FDIR verification

During system level validation, all requirements shall be validated including RAMS requirements. To do it, we build a representative “Hardware in the loop” flight simulation with all avionic equipments, redundancies and FDIR mechanisms.

FDIR verification is one of the most difficult and time consuming activity due to the fact that Hardware in the Loop tests with real equipments cannot simulate all failure modes. Our experience has shown that about 80% of validation test time is spent for testing FDIR with addition of several artefacts as noise injection, EMC perturbation, short-circuits, and mainly failure simulation. A classical approach is to validate FDIR at software level according to software requirements and not fully at system level. This is a weak point in the system verification: the real behaviour of equipments may differ from the expected one and FDIR function efficiency not really proven.

Models based engineering

Models can be used to depict the various system failure modes, their propagation inside the system and their potential interaction with the FDIR. Such models can be used to assess the system architecture dependability early during the development phase. During the ASSERT study we have demonstrated successfully on ATV study case the fault tolerance of Avionic architecture which includes a triplex system

with several redundant sensors and thrusters and a back-up system in charge of the collision avoidance manoeuvre. For this verification we have used ALTARICA[4] [3] modelling tool, see the following §.

Nevertheless, FDIR engineering and RAMS engineering being tightly coupled activities, it is interesting to have a common tool able to provide a model of the system taking into account functional and dysfunctional behaviour on which we can trust. This model is the basis of proof based system and software engineering as it is defined by ASSERT objectives. A common model used by Engineering team and RAMS team is an important factor of communication and should reduce the misunderstanding and speed up the maturation of FDIR functions as I will try to explain:

Today, for complex systems it is agreed to have a functional model which is used for validation of requirements and preparing flight operations; integrating the failure model with the functional model is possible as it is recommended now : among the possible modelling techniques, AADL[5] [4] is a system level language permitting to describe the architecture of your system at organic level, the AADL error annex based on the same principles is an interesting extension; the system level modes and transition can be described taking into account errors propagation and common mode failures provided by RAMS experts and subsystems FMECA. From this AADL model, it is possible to start the verification with ALTARICA tools used to verify the dependability properties of the system.

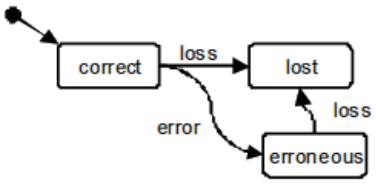
Model based engineering has a major advantage: early during the development we can analyse together the failure model and the FDIR model and tune this last one according to the identified failures mode. We can measure the probability of system loss, and prove the safety by introducing multiple failures. The analysis can be static (fault tree method) or dynamic (sequences of failure events).
As these two models are completely linked together exactly as a strategic game: you find some failure for which you are not enough resistant and you improve your FDIR strategy and so on... This strategic game explains the difficulty of our space projects. Having a common description tool should facilitate the FDIR maturation

Some basic FDIR mechanisms

Failure detectors are in charge of detecting equipment failures. We have the following definition: the test coverage of the equipment is the ratio between detected failures over the total failures (detected and non detected): $C = \Sigma Ci * \lambda i \ / \Sigma \lambda i$.

Health monitoring detector:
Comparison of the health signal with a threshold or detection of a frozen signal (alive detector), the failure status is memory less. Variant of threshold detectors exist with failure status memorized until a reset is sent. 2 types of failure event are identified, error and loss according to the following state diagram. The lost state is detected by a failure detector; however the erroneous state cannot be detected easily if we have not a reference for comparison.

[4] OCAS-ALTARICA is a commercial tool developed by DASSAULT from academic studies (LABRI).
[5] Architecture Analysis and Design Language developed by SEI and is now an international SAE standard //la.sei.cmu.edu/aadl/currentsite/index.html



An equipment design with health monitoring will convert an erroneous behaviour of the equipment into a well known behaviour: the service provided by the equipment (or the function) is considered as lost and the failure is not propagated. If the failure detector is itself erroneous we will have false alarm or undetected error.

Failure masking: the junction will merge the output of 2 functions and provide a consolidated output : if at least one input is correct and the other one is not erroneous the output is correct, if both inputs are lost the output is lost, otherwise the output is erroneous.

Safety barrier: If the alarm is not raised the output is identical to input, else the output is set in a fail safe state easily detectable.

Default value: If the alarm is not raised, the output is identical to input, else the output is set with a default value which protects the system against failure propagation, and the default value may be for example the previous input value.

Median value: this selector will choose one input among 3, with the algorithm of median value, any single erroneous input is eliminated , if 2 inputs are erroneous the output is erroneous, but the lost state cannot be detected.

Voter between 3 functions*:* If at least two of its inputs agree on a value and this value is not lost, then the Voter selects this value for its output. In the case of two inputs lost then the Voter selects the third input value for its input, otherwise the Voter output is erroneous. In order to cope with voter failures, reliable broadcast can be implemented: distributed vote with interactive consistency between 3 voters with multiple exchanges is possible and has been implemented on ATV computers™ (ASTRIUM).

Two axes gyrometers FDIR™ (from ASTRIUM).
Four gyrometers (2 axes) are mounted on adequate axis. The design permits to detect the failed gyrometers and to provide the angular rate projected on the 3 axes of the vehicle.
With 3 gyrometers we have 6 axis data which can be processed by a coherency algorithm in order to detect the failed axis data. So with 3 gyrometers we are tolerant to any single gyrometer failure. With 4 gyrometers, we are tolerant to any double failures with the condition that the 2 failures do not occur simultaneously; after the first failure we reduce the number of valid gyrometer to 3 and we are then single failure tolerant. This design is used for ATV vehicle.

Thrusters Management System (TMS) in presence of failure (from GTI6) [5].
For a 6 DOF control system, the problem is to select the adequate thrusters (at least 6) among the possible ones declared correct by the failure detectors with minimum fuel consumption. In the frame of the ESA study “Integrated Multi-range Rendezvous Control System” the Thrusters Management system algorithms trade-off has been analyzed. Different Force/Torque requests are sent to the assessed TMS algorithms, no particular problem has been identified with the selected design.

Transient error detection® (from CNES)

Space electronics are sensitive to radiation, and Rad Tolerant technology is costly and has low performances. As transient errors (SEU) are predominant, they shall be processed in priority over other types of failures. The SEU probability is typically 1E-3 to 1E-4/hour for a Rad tolerant computer but may have much higher probability with COTS components.

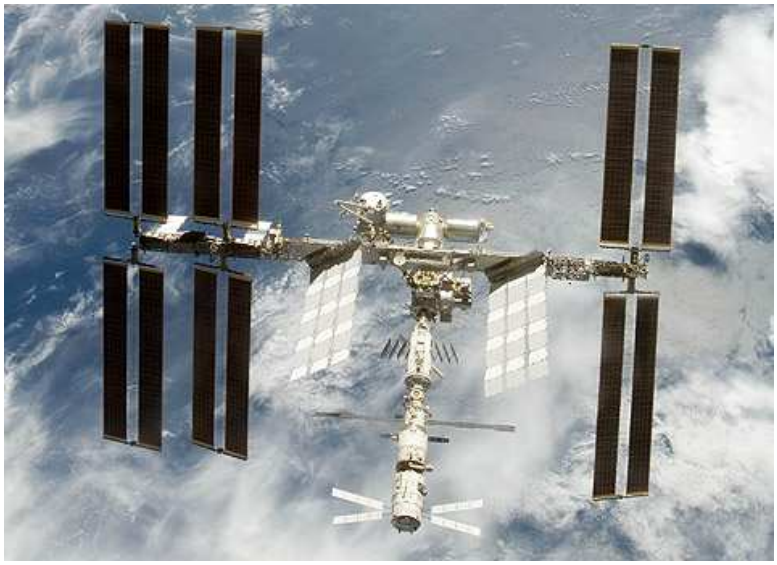
The principle "Duplex Multiplex in Time (DMT) patented by CNES"[6][6] is to duplicate the application process in the same processor (time redundancy) and compare the results, the recovery depends of application: either restart from a context memory (backward recovery) or continuation with one cycle lost (forward recovery). The comparison will be done on the acquired data, the processed data and the output commands. Comparison may be strict (bit to bit) if the input data coherency can be guaranteed.

ATV Jules Verne

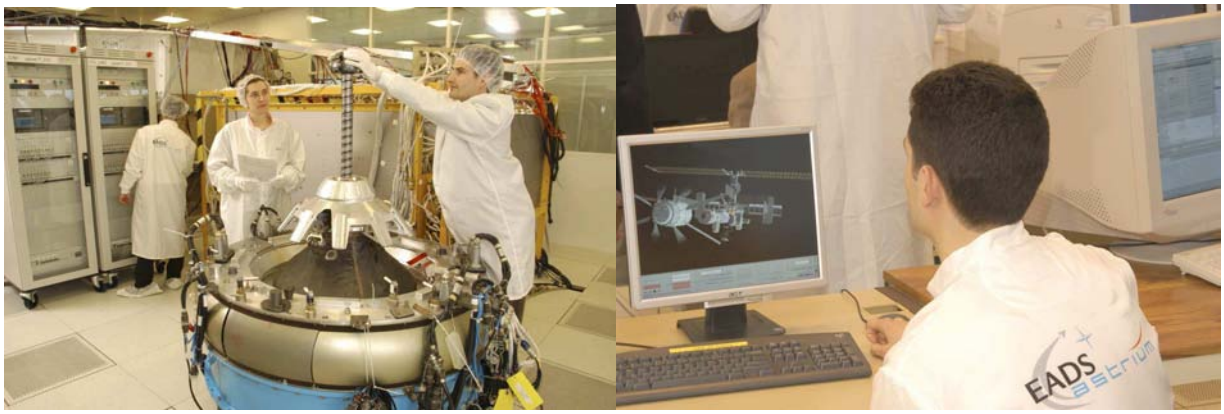
The mission Jules Verne has been fully successful during the year 2008:

9 March	Launch by Ariane 5
14 March	CAM demonstration
3 April	Automatic Docking on the ISS
4 September	De-docking
29 September	De-orbitation

The vehicle ATV developed by ESA with ASTRIUM-ST as prime contractor, has been docked automatically to the International Space Station.



[6] How to cope with SEU M.Pignol CNES

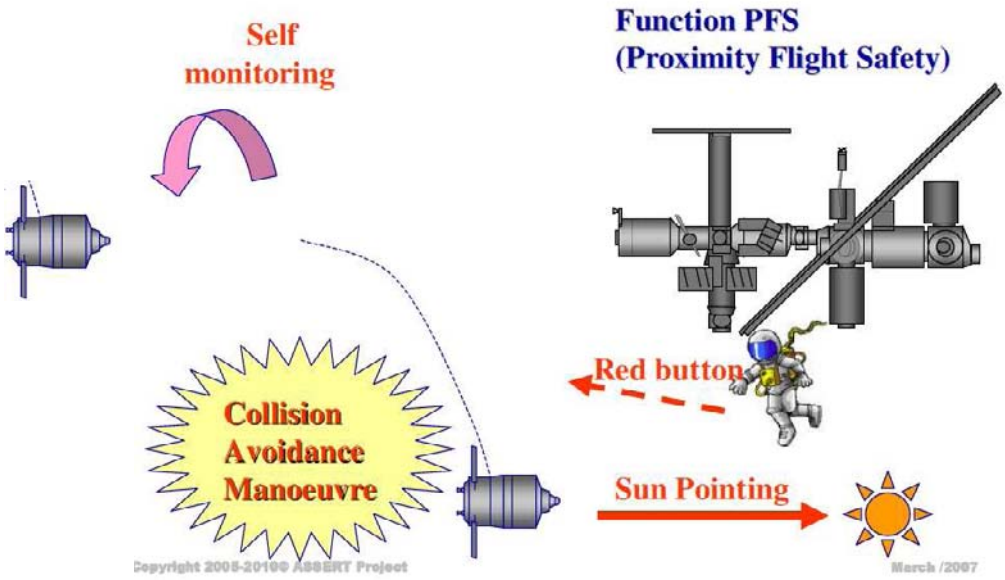


ATV mission

ATV is a complex vehicle interesting for our purpose: the FDIR design and verification.
We can recall here some customer requirements:

- Safety and reliability critical functions whose failures are time critical shall have automatically activated redundancy.
- no single failure or operator error shall have major or critical consequences, loss of mission is in this category.
- no combination of double failures or operator errors shall have catastrophic consequences, one possible catastrophic consequence is the collision with the ISS.

.



ATV Avionic design

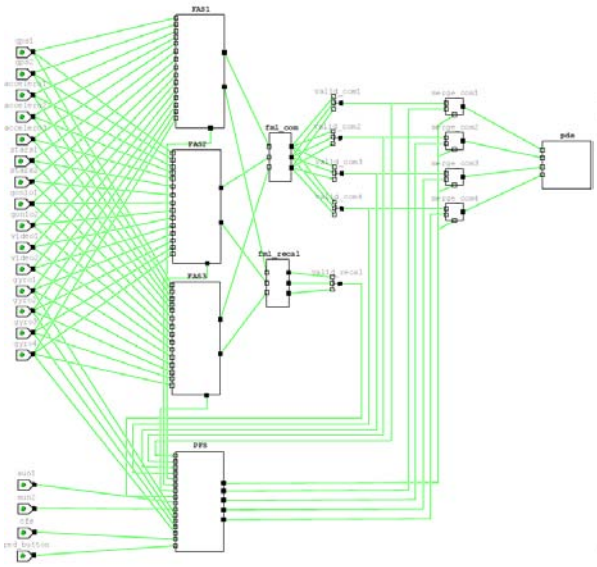
ATV avionic is designed with the following redundancies; we have analysed only the GNC function during this study.

- 3 computers organised as a triplex with votes dedicated to the nominal mission, it includes the Flight Application Software (FAS)
- 2 safety processors dedicated to the CAM manoeuvre, it includes the Proximity Flight Safety software (PFS).
- 4 two axis gyrometers supporting two failures
- 2 GPS receivers
- 2 video-meter and 2 tele-goniometers dedicated to rendezvous
- 2 star trackers
- 4 accelerometers
- 2 sun sensors
- 4 propulsion drive electronic driving 28 thrusters

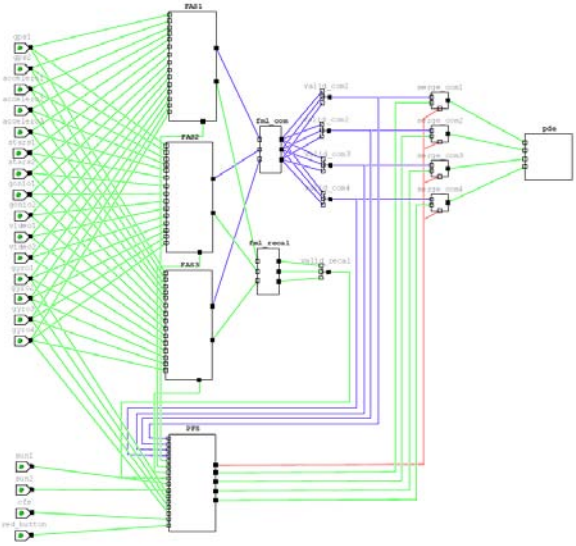
ATV study case

The ASSERT study has developed the ATV functional and organic models of avionic with the communication between the different units and the failure modes of each unit. The failure propagation effects including common mode failures are identified and described in the models. Then the probability law of each failure event is given, we have selected an exponential law as we are doing currently for reliability analysis. According to the methodology, failure propagation can be of 2 different types: loss of the function or erroneous function. This distinction is important for the FDIR mechanisms: the mechanism will detect with a 100% probability the function loss but erroneous functions can be detected only if we develop a failure detector. Failure detectors have coverage less than 100% and have a false alarm rate. As example, failure detectors type can be health monitoring type, a comparator or voter if we have 2 or 3 active instances of the same function or a running model of the function used as the reference or a more complex algorithm with Kalman filtering. A library of failure detectors is necessary for the efficiency of the FDIR design. ONERA has provided a first library during the study.

Once the avionic models and the failure models are established, we can add the failure detectors and the recovery mechanisms. Hierarchical graphic diagrams of the models are provided; some example of failures are indicated by the change of colours of the part of failed function (green is OK, blue is erroneous and red is lost).



Nominal case



Erroneous FAS detected by monitoring, CAM is started

It remains to analyse the efficiency of the design and compare the results with the requirements.
OCAS-ALTARICA provides several verification tools, we will analyse some of them and show how to use it to validate our system and provide quantified results. These tools are analysis tools and not simulation tools; the main interest of analysis is the exhaustiveness, because the combination of failure events may be very high.

Fault tree analysis: this method analyses the consequences of failure events whatever the sequence of events. This analysis is not convenient when the model includes feedback, but it is the simpler and more rapid tool.

Sequence generator by inference: this tool search the combination of failure events (cuts) which will set our system toward a feared event (the target): for example an incorrect recovery, a loss of system, an erroneous output… several options are possible for the failure events: simple combination, combination with permutation, combination with repetition; the depth of the analysis (the number of successive events to be analysed) is an important parameter. Increasing this number will increase drastically the length of the analysis, but some interesting sequences may be long. The search is stopped when the feared event is reached and the output file is limited to the minimal cut.

Generic sequence generator: this tool is closed to the previous one but provides some advantages; the output file has several options minimal cuts, minimal sequences and non minimal sequences.

During the verification done, we have chosen the generic sequence generator with repetition option a depth analysis of 5, and an output file with minimal sequence which is convenient for the nature of our analysis (output file is readable). The duration of analysis is between few seconds to 2 minutes.

Aralia tool: Not experimented

Study Results

We have given in the table 1 the results of the analysis which gives for each feared events the number of failures (between 1 to 5) leading to the feared event, the probability of occurrence of the feared event and the time needed for the analysis. The model being simplified and the reliability data being not representative of the real ATV flight hardware, the values cannot be considered as significant; they are only provided to show the interest of the method.

The first column gives:

- Functional static, limited to a static view of ATV functional architecture
- Functional dynamic , includes a dynamic view of ATV functional architecture
- Global dynamic, includes the organic (hardware) ATV architecture

Three feared events were considered:

- loss of the output of the FAS functions
- erroneous output of the FAS functions
- global ATV loss (loss of the output of the FAS and PFS functions)

Results associated with the static functional model were obtained using the fault-tree generation approach and results associated with the dynamic model were obtained using the sequence generation approach. Results obtained with both approaches are similar but the fault-tree approach was able to compute scenarios that combine a great number of failure events (up to 4 failures) in a much smaller amount of time.

		FAS output	FAS output	global ATV
	cuts	**loss**	**erroneous**	**loss**
functional	*1*	8	0	0
static	*2*	13	33	28
	3	70	142	622
	4	0	235	2756
	5	0	0	X
	proba	8. 1e-5	8.7 1e-9	2.8 1e-9
	time	0.03 s	0.21 s	4.37 s
functional	*1*	8	0	0
dynamic	*2*	13	33	28
	3	70	142	622
	4	X	X	X
	5	X	X	X
	proba		8.7 1e-9	
	time	2 min 5 s	2 min 5 s	2 min 5 s
global	*1*	5	0	0
dynamic	*2*	2	24	22
	3	25	230	397
	4	X	X	X
	5	X	X	X
	proba			
	time			

Lesson learnt

The state machine of the process model is very dependant on the hypothesis on the failure model, so a very careful analysis of the failure types and common mode failures is only possible with RAMS experts; ALTARICA modelling could be a good tool to facilitate the dialog between RAMS and data processing experts. ALTARICA language issued from academic world is rigorous and it is not possible to define short-cut during the design phase.

It is very important to define first the failure models hypothesis and ALTARICA will be a good tool to formalise these failure models; the FDIR model could be an automatic translation from other design models defined at software engineering level as SCADE [6].

Failure model and FDIR model: these 2 models competing against together have not the same observability: the failure model receives the failure events, but the FDIR model observes only the output data. So, 100% coverage is impossible, and we will have to accept only a partial proof based on an estimated probability law. Modelling should reflect these different observability levels: separate the models in order to test the better FDIR model which answer to a system level requirement with an agreed hypothesis of failure model.

References

[1] FP6 IP project coordinated by ESA: http://www.assert-project.net

[2] ATV Jules Verne ESA http://www.esa.int/esaMI/ATV/index.html

[3] OCAS-ALTARICA is a commercial tool developed by DASSAULT
http://www.cert.fr/tis/theses/jthese_05/ressources/articles/humbert.pdf
http://www.springerlink.com/content/231tpt5272011585/
http://www.christophekehren.com/issc04.pdf

[4] Architecture Analysis and Design Language developed by SEI
http://www.aadl.info/aadl/currentsite/index.html

[5] "Optimization of Spacecraft Thruster Management Function", A.Vankov, Journal of Guidance, Control and Dynamics, Vol. 28, No 6, pp. 1283-1290, November-December 2005.

[6] How to cope with SEU/SET at system level M. Pignol CNES
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=1498181&isnumber=32176

[7] SCADE developed by ESTEREL technology http://en.wikipedia.org/wiki/SCADE