Spacecraft Health Monitoring and Management Systems

Massimo Tipaldi
Compagnia Generale per lo Spazio
Via Tiengo s.n.c., 82100 Benevento (Italy)
Email: mtipaldi@cgspace.it
massimo.tipaldi@ohb-system.de
mtipaldi@unisannio.it

Bernhard Bruenjes
OHB System AG
Universittsallee 27-29, 28359
Bremen (Germany)
Email: bernhard.bruenjes@ohb-system.de

***Abstract*—Spacecraft health monitoring and management systems (also referred to as FDIR (Fault Detection, Isolation and Recovery) systems)) are addressed since the very beginning of any space mission design and play a relevant role in the definition of their reliability, availability and safety objectives. Their primary purposes are the safety of spacecraft/mission life and the improvement of its service availability. In this paper current technical and programmatic FDIR strategies are presented along with their strong connection with the wider concept of on-board autonomy, which is becoming the key-point in the design of new-generation spacecrafts. Recent projects developed at OHB System AG have brought to light some issues in the current FDIR system design approaches. These findings pave the way for innovative solutions, which can support and not rule out conventional industrial practices.**

Keywords*—FDIR; Spacecraft Autonomy; Spacecraft Health Monitoring; Packet Utilization Standard.

I. Introduction

Spacecraft Health Monitoring and Management systems (also referred to as FDIR (Fault Detection, Isolation and Recovery) systems) act as supervisory controllers and ensure that the mission objectives and constraints are met and that the spacecraft is protected from failures leading to a service deterioration or even worst to the spacecraft loss. Satellite evolution and the need of designing more and more autonomous and dependable missions are the main factors driving FDIR system enhancement. Space missions can benefit from more autonomous FDIR systems in different ways, such as reduction of ground intervention, reduction of the occurrence of service interruptions, improvements of the system reactivity etc. FDIR systems are basically featured by the following capabilities:

- fault detection, i.e. the determination of the presence of faults in a system and their times of occurrence
- fault isolation (and classification), i.e. the determination of the type, severity and location of faults
- fault recovery, i.e. basing on the fault severity and possible effects, the process of choosing the best action to recover from them.

FDIR requirements are allocated between the space flight and the ground segments according to some factors such as mission type and objectives, spacecraft orbit and ground visibility profile, operations concepts, communication bandwidth and latency. Moreover important constraints (e.g. robustness, reactive detection, quick isolation/identification and limited on-board resources (CPU and memory)) have to be taken into account for the selection of FDIR system approaches. Today's rapid progress in computing power supports the transfer of FDIR functions from the ground to the flight segment and the enhancement of the spacecraft on-board autonomy. FDIR conception and implementation in a space system can be a rather complex field, since the analysis of S/C failures can extend to very sophisticated considerations. Within a spacecraft, the effects of faults can propagate through the boundaries of many subsystems and can result in diverse fault symptoms. FDIR information processing and physical components are tightly integrated and intertwined, therefore the structure and properties of physical spacecraft components determine the functions included in an FDIR system. Its design strongly also depends on the spacecraft operational modes and mission phases. FDIR systems have to be carefully designed in order to ponder the needed HW redundancy versus the diagnosis capability and the recovery actions embedded in the On-Board Software (OBSW). Basically, FDIR system concept can be shaped between two extremes, i.e. having a straightforward on-board FDIR (where only vital elements are checked on-board and the related recovery is completely performed by ground segment) or making use of a more sophisticated on-board FDIR concepts (where failures are identified at the lowest level possible and solved autonomously by the spacecraft and the ground intervention is reduced to the minimum). Hardware redundancy is also a key-element especially for those functions whose failure can lead to mission loss. FDIR system testing is also a challenging process due to the inability to reproduce all the possible faults and their combinations during the test campaigns. This paper draws on real satellite projects carried out at OHB System AG and is organized as follows. In this section an overview on spacecraft on-board autonomy and health management concepts is given. Sections II and III summarize the current industrial approaches and practises used to design, develop, validate and operate FDIR systems, whereas section IV describes the challenges of future space missions which demand more and more high-rich on-board autonomy and health management capabilities. Finally, section V concludes the paper.

978-1-4799-2069-3/14/$31.00

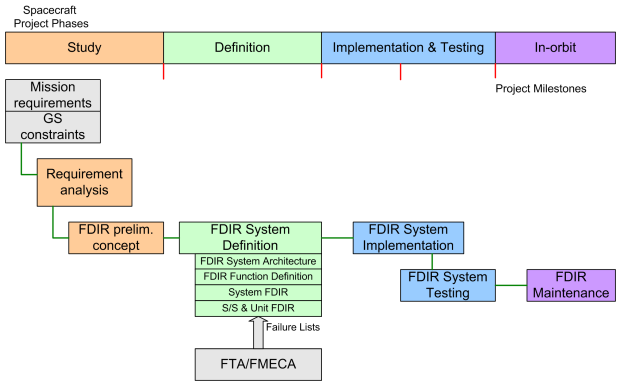

Fig. 1. FDIR Development vs Project Phases

II. Current FDIR development process and industrial practises

This section summarizes the current industrial practises used to design, develop, validate and operate FDIR systems. FDIR system development has to be regarded as a broad system-level activity and requires in-depth knowledge of space system engineering. FDIR development practises are derived from the development processes defined by the European Co-operation for Space Standardization ([1] and [2]). Fig. 1 shows the different phases featuring the FDIR system development and their relationship with the spacecraft project phases.

A. FDIR early life cycle phase

FDIR system requirements and functions stem from the mission/high-level system needs & constraints and the outcome of the Fault Tree Analysis (FTA) and the Failure Modes, Effects and Criticality Analysis (FMECA). FTA is a deductive analysis where top-level failures (e.g. loss of power, loss of fuel) are addressed and all the basic faults that lead to that state are identified. Conversely, FMECA is an inductive analysis and it computes the casual effects on the system of a given fault combination. After specifying all the FDIR functions, the implementation into the OBSW and/or spacecraft subsystems/units is performed, followed by a verification of all the implemented FDIR requirements. Current space projects suffer from the fact that FMECA/FTA data become consolidated too late in the engineering process, usually when spacecraft development has already passed its initial phases, therefore FDIR functions and designs are band-aids to already-existing design. Changes in the subsystems design can further jeopardize the process of achieving a stable FDIR design. In order to counteract all these issues, new approaches in the spacecraft development process underpinned by formal modeling techniques are necessary (see IV).

B. FDIR system architecure

During the definition and early development phases, FDIR concepts are further detailed and translated into a FDIR architecture and FDIR function identification and allocation. Failures are usually deployed on a set of five hierarchical levels (see Fig. 2) which are characterized by clearly defined interfaces between them, their severity, the function involved in the detection and the recovery sequence ([3]). The highest FDIR level is in charge of the execution of vital functions to ensure the S/C integrity, while lower levels operate on subsystem or unit level and are usually software driven. A higher level is triggered by the adjacent lower level, only when the latter is not able to solve/isolate a fault. In this case, FDIR functions allocated to the higher level perform command and control functions of the next lower level by using its housekeeping data and alarm information. The overall FDIR system is usually implemented by means of hardware and software resources with a high level of interaction between them and, as a result, each S/C subsystem is part of the overall FDIR architecture. The adoption of a hierarchical FDIR in the real project world is justified by both technical and programmatic reasons (e.g. the establishment of a control structure with clear defined interfaces and the allocation of FDIR functions and levels within the project organization).

The FDIR system hierarchical structure consists of the following levels:

- **Level 0**: faults and failures local to a unit that can be recovered by local correction (e.g. EDAC (Error Detection and Correction), short currents, over-voltage, data bus failures); they have no impact on system performance.
- **Level 1**: faults and failures that can be detected outside the unit and are handled at subsystem level (e.g. failure of a unit or of a unit's communication interface, which can be recovered by having recourse to the redundant path); the OBSW has to monitor unit data, e.g. against specific thresholds for failure detection; such failures can degrade subsystem performance.
- **Level 2**: failures that are handled by the OBSW at system level (e.g. failure of a system function, caused by the propagation of undetected unit faults and failures); inter-cross data checks, functional monitoring or subsystem/system level complex verifications are the basis for the failure detection process at this level; these failures can cause loss of system/subsystem service and/or performance.
- **Level 3**: anomalies that are related to the Processor Module (i.e. the spacecraft On-Board Computer (OBC)) which executes the central OBSW and therefore level 1 & 2 FDIR functions; at this level, failure management consists of Reconfiguration Modules checks (which oversees the health of the OBC and flight software), hardware alarm monitoring and execution of OBC resets, reconfigurations and switches to Safe Mode as appropriate.
- **Level 4**: failures that can not be handled autonomously by the spacecraft which executes reconfiguration procedures (according to some patterns usually stored in a non-volatile memory) and then performs a transition into Safe Mode for ground intervention; level 4 handles all the critical failures that can lead to a loss of mission, therefore the driving requirement is usually the total mutual independency and hot redundancy between the nominal system functions and the ones devoted to the detection and recovery from such critical events (see Herschel/Planck [4] mission).

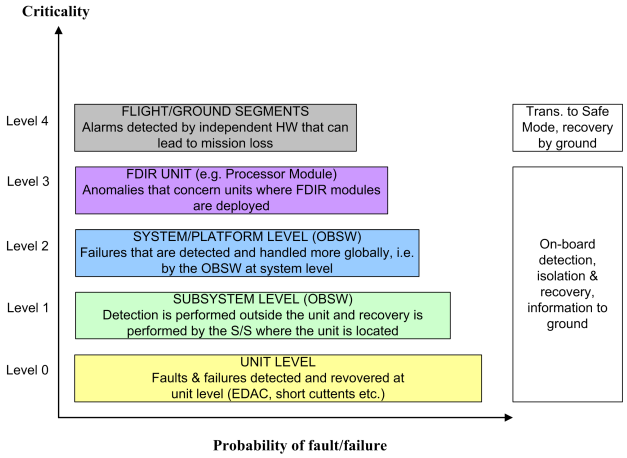

Fig. 2. FDIR Hierarchical Structure

The general trend in spacecraft manufacturing is to convert many hardware implemented functions into software modules in order to support a more cost-effective implementation. Such approach is also coming as common practise in the FDIR field ([5]), therefore the OBSW is playing a more and more relevant role in the FDIR function implementation.

The operational concept of a typical spacecraft includes the definition of one or more safe mode configurations that represent the ultimate reaction of the system level FDIR to spacecraft severe anomalies. Safe mode can be triggered both by on-board critical events or by ground. The spacecraft can remain in this mode without ground segment intervention for a specified period of time. In safe mode, the communication link to Ground, a specific power supply profile and thermal survival functions for relevant equipment are maintained, all non-essential on-board units/subsystems are powered off and the spacecraft is oriented to a particular attitude with respect to the Earth or the Sun. The recovery of the spacecraft from safe mode to nominal mode needs to be commanded by ground.

C. Verification & Validation approaches for FDIR systems

SW requirements derived from FDIR requirements and corresponding SW artifacts normally undergo the traditional Verification & Validation (V&V) process of the SW applications where they have been allocated. Such process includes requirement consistency & completeness analysis, SW unit tests, code inspection, SW metrics collection and analysis, SW/SW integration tests, fault injection to verify the fault tolerance mechanisms implemented in SW, SW system-level tests (see ECSS-E-ST-40C [6]). SW V&V activities are defined according to the associated SW criticality levels, ranging from Level A (software whose anomalous behavior would cause or contribute to a failure resulting in a catastrophic event) up to Level E (software whose anomalous behavior would cause or contribute to a failure resulting in a negligible event). Once all the HW and SW items concurring at the implementation of system-level FDIR requirements have been validated and integrated, FDIR requirements are globally verified at system/spacecraft level. To this respect, the Hardware/Software Interaction Analysis (HSIA) plays an important role in highlighting flaws from the very beginning of a project. All the V&V techniques used in the current projects do not scale to autonomy-rich systems, where high environment sensitivity tends to dramatically invalidate the "if it works at all, it will always work" motto. As a result, the confidence gained per test goes down, so more tests must be performed. But, the tests must vary the environment and system state trajectory to a large extent, which makes the current approaches unfeasible. This remark paves the way for different solutions (see IV).

III. FDIR ELEMENTS IN SPACECRAFT OPERATIONS

Traditional spacecraft operations are mainly performed with Flight Control Procedures (FCPs), Mission Time Line (MTL) and On-Board Control Procedures (OBCPs). FCPs are executed step-by-step by a ground operator, which involves sending telecommands (TC) to the spacecraft and checking Telemetry (TM) downlinked to ground. Missions with limited ground station coverage can also use the MTL, which is a sequence of time-tagged TCs loaded from ground and executed by the OBSW when their time tag expires. OBCPs are script-like files, written in a high-level programming language and compiled on ground into some efficient representation (for instance, Java bytecode), suitable to be processed and executed by an on-board virtual machine. MTL concept is limited since it is based on the assumption of successful commands. Several ESA missions adopt the OBCP approach for FDIR concept such as Rosetta, Venus Express and the dual mission Herschel/Planck. System availability requirements drive the choice of the FDIR strategies, including the adoption of MTL and/or OBCPs. For missions requiring a high level of availability (like the telecommunication spacecraft or complex interplanetary space missions), FDIR is often very sophisticated and aims at minimizing the effect of a fault with regard to the whole satellite, leading to a reduction in mission outage. On the other hand, when the required availability level is medium (e.g. scientific Earth observation missions), satellite saving is preferred to mission follow-on. This is reflected in having two kinds of FDIR operational modes, i.e. the Autonomous Fail Operation mode (where a redundant unit can be reconfigured and called into operation) and Autonomous Fail Safe mode (where the satellite is transferred into Safe Mode for ground to intervene). The extent of higher on-board autonomy with the respect to FDIR system also depends on the criticality of the considered mission phase.

The Packet Utilization Standard (PUS, see [7]) defines the operational model of a spacecraft, which consists of a set of extensible services that can be invoked through a service request (telecommand source packet), such invocation resulting in generation of zero or more service reports (telemetry source packets). Amongst the other things, PUS addresses the FDIR operational concepts and provides a set of services which allow the standardization of the fault management between ESA missions. In particular, the On-board Monitoring PUS Service 12 plays the role of fault detector, whereas the On-board Event Reporting PUS Service 5 allows the fault isolation. The recovery is performed by the On-Board Event-Action PUS Service 19 that triggers some actions which can be included in the On-Board Command Schedule (MTL, PUS Service 11) or the On-board Control Procedures (OBCPs, PUS Service 18). Such recovery actions can also entail the transition to the Safe Mode, and therefore the ground detailed analysis and recovery. PUS gives a higher level of flexibility both during the ground testing and the spacecraft operational life, e.g. monitored item

thresholds can be modified, it is possible to update the recovery action according to changed recovery needs.

IV. FDIR SYSTEM EVOLUTION IN THE NEXT YEARS

Recent projects developed at OHB System AG have brought to light some issues in the current FDIR system design approaches. These findings pave the way for innovative solutions, which can support and not rule out conventional industrial practices. The purpose of this paragraph is to outline such shortcomings and provide some possible solutions. As a rule, future space missions will be featured by ambitious goals resulting in demanding performance and availability requirements. Ground operation teams have to handle increasingly large volumes of telemetry data and interplanetary spacecraft missions pose an additional challenge because of the time delay between command transmission and reception by the spacecraft. Moreover, the lack of a well-established analytical methodology supporting FTA/FMECA activities, system-level FDIR conception down to its implementation causes serious discontinuities throughout all the project phases and hampers the process of a stable and consistent FDIR design. Space and scientific communities are strongly aware of all these issue and improvements in both FDIR programmatic aspects and development strategies are necessary. A central role is played by the FDIR toolset environment, which should provide a seamless support throughout the different FDIR development phases.

From a technical stand-point, activities focus on innovative mechanisms combined with well-proven methods of conventional FDIR with the aim of reducing the number of safe mode events and increasing the spacecraft operational time. While defining a FDIR function, the current way forward is to map one or more failure modes to a telemetry point (discrepancy), resulting in an isolation/recovery action, if the FDIR threshold of the telemetry data is exceeded. However, there may be failure modes which do not only affect its assigned telemetry point but also other telemetry points which are assigned to the detection of different failure modes and different recovery actions. This may lead to a recovery which does not solve the problem in a first FDIR step. This is due to the fact that current FDIR approaches are based on stringent and rigid procedures. Simple diagnostic routines usually process symptoms in isolation, which can result in incorrect diagnoses or contradictory deductions. Additionally, they are not able to cope with the space environment, which is only partially observable by the FDIR monitoring function and is time-variant. Thus, a model-based FDIR is needed, having the capability of fusing information coming from different sources and reasoning about anomalous observations in presence of uncertainties, system dynamic evolution and partial observability in order to estimate the system health. The way-forward supported by some preliminary assessment at OHB System AG is to keep the FDIR hierarchical structure presented in II-B and to endow FDIR system with more effective and powerful detection and recovery means ([8] and [9]). As for the algorithm selection, at least three factors have to be taken into account, that is the reaction time needed for the system diagnosis, the kind of data to handle (discrete, continuous and hybrid) and the required computational resources.

Another important aspect of the new FDIR approaches is

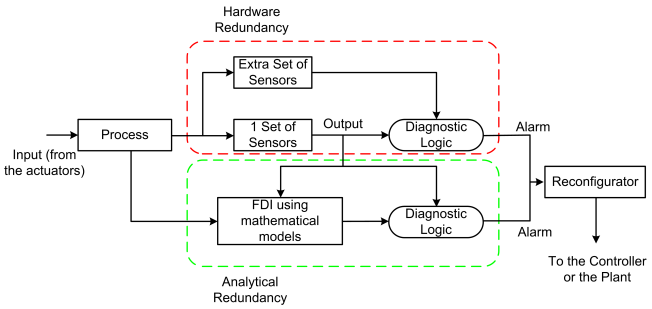

Fig. 3. Concepts of hardware redundancy and analytical redundancy for FDIR

the exploitation of the analytical redundancy over the hardware redundancy (see Fig. 3, where a control-system view of the FDIR is given with a clear separation between the fault detection and isolation (FDI) step and the controller reconfiguration step). The basic concept of hardware redundancy is to compare duplicative signals generated by various hardware, such as measurements of the same signal given by two or more sensors. On the other hand, analytical redundancy uses a mathematical model of the system together with some estimation techniques for the fault detection and isolation. As the analytical redundancy approach generally does not require additional hardware, it is usually a more cost effective approach compared to the hardware redundancy approach. However, the analytical redundancy approach is more challenging due to the need to ensure its robustness in the presence of model uncertainties, noise, and unknown disturbances. Generally, the analytical redundancy approach can be divided into quantitative model-based methods and qualitative model-based methods. The quantitative model-based methods, such as the observer-based methods, use explicit mathematical models and control theories to generate residuals for the FDI. On the other hand, the qualitative model-based methods use artificial intelligence techniques, such as pattern recognition, to capture discrepancies between observed behavior and that predicted by a model. In both cases, fault diagnosis is achieved by first, then residual generation and evaluation and last, application of an appropriate decision logic.

Autonomy is an important feature of currently developed and future missions. In the near future, satellites will be able to receive, process and achieve high-level goals even in an uncertain or dynamically varying context. They will be endowed with an ergonomic interface so that ground operators can easily define and transmit such high level requests. Autonomy accommodation and the integration of planning systems and dynamic reprogramming capabilities into the flight software can be achieved by organizing the OBSW architecture along three hierarchical levels, that is to say the decisional level, the operational level and the functional level (see Fig. 4). These levels are characterized by different reaction times, handle more or less abstract data representations and have different type knowledge of the system state (global or local). Ground high-level objectives can be further on-board detailed into a sequence of commands for the subsystems and can be autonomously adapted during their execution according to context changes, such as new objectives and altered on-board resource profile. FDIR functions are allocated to the three levels. A first set is paired to the satellite subsystems

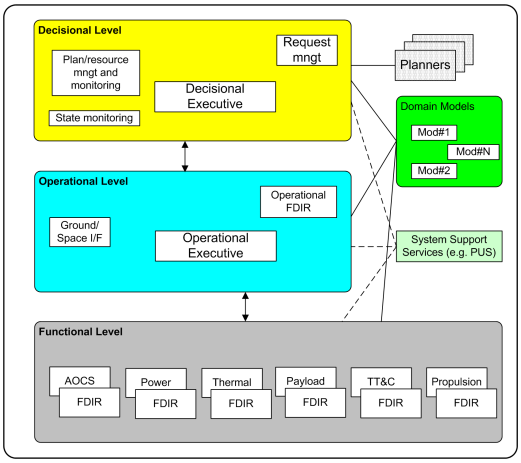

Fig. 4. OBSW architecture of autonomous systems

and is allocated to the bottom of the architecture, that is the functional level. This level serves as basis to operational level, where system-level FDIR functions are deployed. At the decisional level, FDIR monitors the plan execution and detects inconsistencies according to the reports coming from the operational level.

Finally, special attention has to be paid to the verification and validation (V&V) of advanced FDIR systems since optimization and model-based algorithms are at their core ([10]). FDIR systems tend to be very sensitive to the environment, therefore one should explore the behavior of the system over a range of plausible environments in order to demonstrate the robustness of the system. All together these things constitute a quantum leap in the V&V. It is recommended confirming the consistency, the completeness and the correctness of FDIR system implementation both on model-level and on code-level (e.g. the data generated out of the models and the reasoning engine itself). While reasoning engines can be verified by adopting classical V&V techniques in order to demonstrate that their search mechanisms are correct, the domain models need more advanced V&V techniques, which are largely based on formal or analytical methods. They refer to the use of techniques from logic and discrete mathematics and can be categorized as follows:

- Runtime Monitoring: to evaluate the code while it runs or to scrutinize the artifacts (e.g. event logs) of running code.
- Static Analysis: to detect runtime errors and to assess code proprieties without executing it.
- Model Checking ([11]): to evaluate the system behavior and its state evolution by means of its representative model and a set of proprieties expressed via an executable specification language.
- Theorem Proving: to prove system requirements by means of logical induction over the execution steps of the program.
- Compositional Verification ([12]): to decompose the proprieties of a system into proprieties of its components, so that if each component satisfies its respective propriety, then so does the entire system. This method is a key step in achieving scalability in the verification of large software systems.

V. Conclusion and Future Work

Based on real projects carried out at OHB System AG, this paper describes programmatic and technical key-elements for the design of spacecraft health management and monitoring systems. The current industrial approaches have experienced some shortcomings which can be remarkably improved by integrating conventional practises with innovative solutions, e.g. qualitative/quantitive model-based mechanisms. In the near future space missions will indeed require high level of on-board autonomy, including the capability of handling faults with negligible ground intervention. OBSW and formal verification will play a relevant role in the implementation of enhanced on-board autonomy and FDIR systems. Explicit representation of spacecraft states and behavioral models in the OBSW will offer a unique opportunity to verify the correctness and the consistency of information that nowadays is usually implicitly embedded in the algorithms. This will reduce notably the gap between concepts and assumptions that reside in the mind of system engineers and what is actually implemented and will also lead to easier portability from mission to mission, as only the models need to be updated with domain specific knowledge.

References

[1] *ECSS-Q-ST-30-02. Space product assurance - Failure modes, effects and criticality analysis*. European Cooperation for Space Standardization Standard, 2001.

[2] *Space product assurance - Software Dependability and safety*. European Cooperation for Space Standardization Standard, 2012.

[3] 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*, 2012.

[4] J. Bos, D. Zorita, A. Bacchetta, G. Chlewicki, D. Guichon, and I. Rasmussen, "ACMS FDIR system for the Herschel/Planck satellites," in *Proceedings of the 6th International ESA Conference on GNC Systems. Loutrakis, Greece*, 2005.

[5] F. SalarKaleji and A. Dayyani, "A survey on fault detection, isolation and recovery (FDIR) module in satellite onboard software," in *International Conference on Recent Advances in Space Technologies (RAST)*, 2013, pp. 545 – 548.

[6] *ECSS-E-ST-40C: Space Engineering - Software*. European Cooperation for Space Standardization Standard, 2009.

[7] *ECSS-E-70-41A: Ground systems and operations - telemetry and telecommand packet utilization*. European Cooperation for Space Standardization Standard, 2003.

[8] I. Hwang, S. Kim, Y. Kim, and C. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 3, pp. 636–653, 2010.

[9] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.

[10] G. Brat, E. Denney, D. Giannakopoulou, J. Frank, and A. Jonsson, "Verification of autonomous systems for space applications," in *Proceedings of Aerospace Conference*, 2006.

[11] M. Feather, L. Fesq, M. Ingham, S. Klein, and S. Nelson, "Formal methods for the validation of fault tolerance in autonomous spacecraft," in *Proceedings of Aerospace Conference*, 2004, pp. 682 – 697.

[12] S. Bensalem, M. Bozga, T.-H. Nguyen, and J. Sifakis, "Compositional verification for component-based systems and application," in *Software, IET*, 2010, pp. 181 – 193.