An Integrated Process for FDIR Design in Aerospace

Benjamin Bittner[1], Marco Bozzano[1], Alessandro Cimatti[1], Regis De Ferluc[2], Marco Gario[1], Andrea Guiotto[3], and Yuri Yushtein[4]

[1] Fondazione Bruno Kessler, Trento, Italy
[2] Thales Alenia Space, France
[3] Thales Alenia Space, Italy
[4] European Space Agency (ESA), ESTEC, Noordwijk - The Netherlands

Abstract. The correct operation of complex critical systems increasingly relies on the ability to detect and recover from faults. The design of Fault Detection, Isolation and Recovery (FDIR) sub-systems is highly challenging, due to the complexity of the underlying system, the number of faults to be considered and their dynamics. Existing industrial practices for FDIR are often based on ad-hoc solutions, that are conceived and developed late in the design process, and do not consider the software- and system-level RAMS analyses data (e.g., FTA and FMEA). In this paper we propose the FAME process: a novel, model-based, integrated process for FDIR design, that addresses the shortcomings of existing practices. This process aims at enabling a consistent and timely FDIR conception, development, verification and validation. The process is supported by the FAME environment, a model-based toolset that encompasses a wide range of formal analyses, and supports the FDIR design by providing functionality to define mission and FDIR requirements, fault propagation modeling, and automated synthesis of FDIR models. The FAME process and environment have been developed within an ESA-funded study, and have been thoroughly evaluated by the industrial partners on a case study derived from the ExoMars project.

1 Introduction

The design of critical systems in aerospace is a very complex and highly challenging task, as it requires assembling heterogeneous components, implemented either in hardware or in software, and taking into account their interactions. Moreover, safety- and mission-critical systems have to obey several sorts of functional and non-functional requirements, including (real-)time constraints, safety and dependability requirements. The correct operation of such systems increasingly relies on the ability to detect and recover from faults. The design of Fault Detection, Isolation and Recovery (FDIR) sub-systems is highly challenging, due to the complexity of the systems to be controlled, the number of faults and failure combinations to be considered, their effects and interactions.

Currently, there is no defined FDIR development process for aerospace coherently addressing the full FDIR lifecycle, and including the corresponding

verification and validation perspective. Current approaches are poorly phased: no dedicated approach to FDIR development exists, which can be employed starting from the early system development phases, and is able to take into account the design and RAMS data from both software and system perspective. Existing practices are often based on ad-hoc solutions, that are conceived and developed late in the design process. In particular, results of Software and System (Hardware) RAMS activities (e.g., FTA and FMEA), become available late in the process, leading to late initiation of the FDIR development, which has a detrimental effect on the eventual FDIR maturity.

Furthermore, no underlying, unifying model for various fault identification methods is available, making it difficult to ensure that the FDIR design is complete. There is a conflict between the bottom-up and top-down approaches: FMEA (bottom-up) can not be completed until system design has sufficient levels of details, whereas FTA (top-down) does not guarantee that every possible component failure mode which contributes to system failure has been considered. Finally, FDIR complexity limits the possibility to effectively determine the propagation of failures in terms of time.

To address these shortcomings, we propose a novel and comprehensive process for FDIR design, called the FAME process. This process aims at enabling a consistent and timely FDIR conception, development, verification and validation. It enables the specification and analysis of failure propagation using fault propagation models, the possibility to specify a set of relevant and decomposable requirements for FDIR, and to model, or synthesize, FDIR components that comply with the requirements. Finally, it enables verification of the effectiveness of FDIR. The FAME process is supported by a model-based toolset for FDIR development and verification: the FAME environment. The process and environment have been thoroughly evaluated by the industrial partners on a case study derived from the ExoMars project.

This work has been carried out as a response to an invitation to tender of the European Space Agency on the topic of FDIR development and verification. This study builds upon the previous COMPASS study [8,14], whose goal was to develop a comprehensive methodology and toolset for model-based development and verification of aerospace systems, supporting verification capabilities ranging from requirements analysis to functional verification, safety assessment, performability evaluation, diagnosis and diagnosability.

The paper is structured as follows. In Sect. 2 we present the FAME process, and in Sect. 3 we discuss tool support. In Sect. 4 we present the evaluation on the case study. In Sect. 5 and Sect. 6 we discuss related work and conclude.

2 The FAME Process

The proposed FDIR design process is schematically illustrated in Fig. 1. This picture shows the input and outputs of the activities, and illustrates how the "classical" design process can be improved, and supported, using formal technologies. The main novelties provided by the formal environment are:

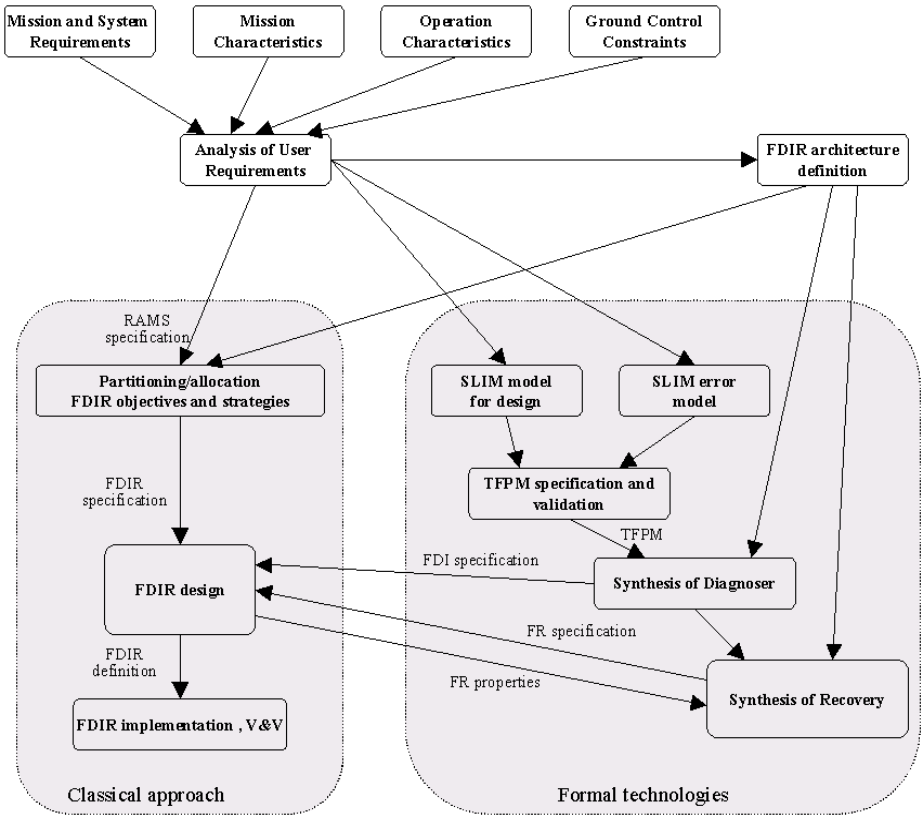

Fig. 1: Process Overview

- Formal modeling of both nominal and error models (i.e., taking into account faulty behaviors) in SLIM [9] (System-Level Integrated Modeling Language – a variant of AADL).
- Formal verification and validation capabilities, including RAMS analysis (e.g., Fault Tree Analysis, FMEA) carried out on the formal model.
- Modeling of fault propagation using so-called TFPM (Timed Failure Propagation Models).
- Formal definition of requirements for FDIR.
- Automatic synthesis of fault detection and fault recovery components (encoded in SLIM), starting from the TFPM and the FDIR requirements.

In this view, formal modeling and synthesis of FDIR is carried out in parallel with the classical process. The FAME process is technology independent, therefore, SLIM models could be replaced with other formal models, if available. Fig. 2 describes the process.

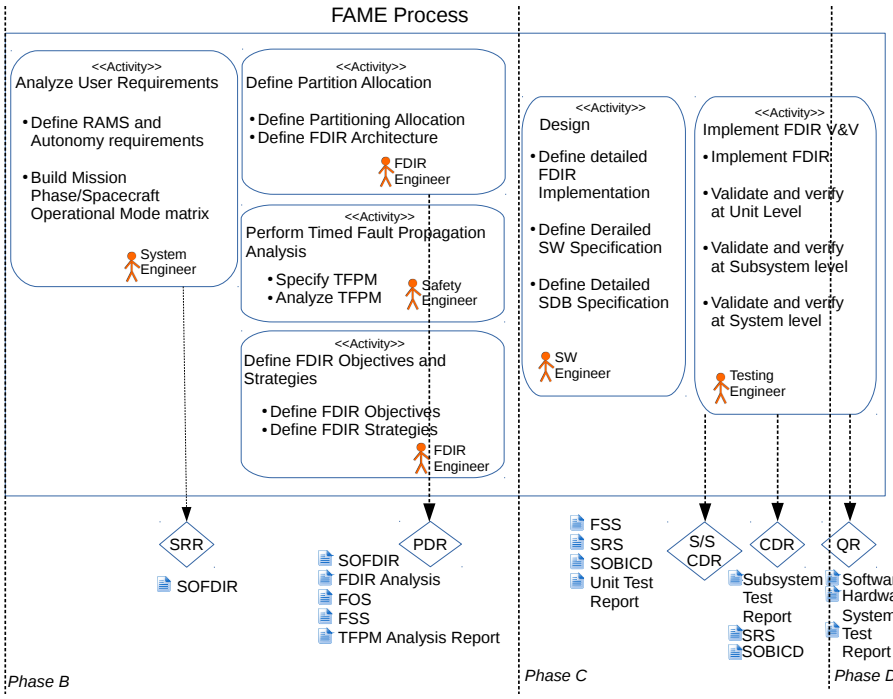

Fig. 2: FAME Process

The process is instantiated onto the ECSS standards [15]. In particular, it makes reference to lifecycle phases B, C, D, milestones such as SRR (System Requirements Review), PDR (Preliminary Design Review), CDR (Critical Design Review) and QR (Qualification Review), and to artifacts such as SOFDIR (System Operations and FDIR Requirements), FOS (FDIR Objectives Specification), FSS (FDIR Strategies Specification) and so on. The process is composed of the following steps:

Analyze User Requirements The purpose of this activity is the collection and the analysis of all the user requirements that impact the design of FDIR. The requirements are analyzed to derive the FDIR objectives, and define their impact on the spacecraft design from system level down to unit level. The output of this activity is a document, containing the specification of the spacecraft FDIR operational concept. It starts at the beginning of System Phase B and terminates before System SRR. This activity includes the following tasks: Define RAMS and Autonomy requirements (identification of critical functions and redundancy schemes, classification of failures into failure levels, identification of FDIR levels, identification of pre-existing components to be re-used, definition of fail-safe and fail-operational strategies), and Build Mission Phase/Spacecraft Operational Mode matrix (identification of spacecraft modes, FDIR modes, and their association).

Define Partitioning/Allocation In this step, RAMS and Autonomy Requirements are allocated per Mission Phase/Spacecraft Operational Mode. Moreover, the spacecraft FDIR architecture is modeled, including all the involved sub-systems (e.g., avionics, payload), taking into account the distribution of the FDIR functionalities. FDIR components can be distributed, hierarchical or a combination of both, thus providing enough flexibility to cover a wide range of architectural solutions. The output of this activity is the FDIR Analysis, describing the reference FDIR architecture and RAMS and Autonomy Requirements allocated per Mission Phase/Spacecraft Operational Mode. This activity starts after System SRR and terminates at System PDR. It includes the following tasks: Define Partitioning/allocation (definition of FDIR approach and autonomy concepts along different mission phases/operational modes) and Define Architecture (identification of functional decomposition, sub-system HW/SW partitioning, sub-system functions and redundancy, integration of pre-existing FDIR functionalities, definition of FDIR levels, FDIR catalogue, perform FDIR analysis for each failure).

Perform Timed Fault Propagation Analysis In this step, a fault propagation is specified using a TFPM (Time Failure Propagation Model). Inputs to this activity are the results of RAMS analyses, such as fault trees and FMEA tables, and of Hazard Analysis. This activity starts at SRR and terminates at PDR. It includes the following tasks: Specify TFPM (taking into account the definition of the fault propagation model, starting from mission and system requirements, RAMS and hazard analysis, and specification of the observability information) and Analyze TFPM (whose goal is to check the correctness/completeness of the TFPM with respect to the system model, and analyze the suitability of the TFPM as a model for diagnosability, taking into account its observability characteristics).

Define FDIR Objectives and Strategies The goal of this step is to specify FDIR Objectives (at system-level) and FDIR Strategies (at sub-system level), starting from RAMS and Autonomy Requirements, and exploiting the previous results on FDIR and fault propagation analysis. This activity starts after System SRR and terminates at System PDR (it can be done in parallel with the previous activity). It includes: Define FDIR Objectives (including the definition of objectives, such as required behavior in presence of failures) and Define FDIR Strategies (representing the FDIR functional steps to be performed, given the fault observation and the objective).

Design This step is concerned with the design of the various FDIR sub-systems, and the corresponding software and data base, on the basis of the FDIR Reference Architecture. This activity starts at System PDR and terminates at Sub-System CDR. It includes: Define detailed FDIR implementation (identification of parameters to be monitored, ranges, isolation and reconfiguration actions), Define Detailed SW Specification (analyze the suitability of the available PUS services, and extend them as needed with new functionalities) and Define Detailed Spacecraft Data Base (SDB) specification (insert monitoring information, define the recovery actions, and the link between monitoring and recovery actions).

Phase	Tool Functionality	Rationale
Analyze User Requirements	System Modeling & Fault Extension	Formal system modeling – nominal and faulty behavior (in SLIM); automatic model extension
	Formal Analyses	Derive requirements on FDIR design (input for following phases)
	Mission Modeling	Definition of mission, phases, and spacecraft configurations
Define Partitioning/Allocation	System Modeling	Modeling of context, scope, and FDIR architecture
	Formal Analyses	Derive and collect FDIR requirements
Define FDIR Objectives and Strategies	FDIR Requirements Modeling	Modeling of FDIR objectives and strategies, definition of pre-existing components to be re-used, and FDIR hierarchy
Perform Time Fault Propagation Analysis	Formal Analyses	Derive information on causality and fault propagation (input for TFPG modeling)
	TFPG Modeling	TFPG modeling, editing, viewing
	TFPG Analyses	TFPG behavioral validation, TFPG effectiveness validation, TFPG synthesis
Design	FDIR Modeling/Synthesis	Formal modeling and automatic synthesis of FDIR
	Formal Analyses	FDIR effectiveness verification
Implement FDIR, V&V	*Contract-based testing*	*Support for automatic generation of test suites*

Table 1: Tool support for FAME process

Implement FDIR, V&V The last step is concerned with the implementation of FDIR in hardware and/or software, and its verification and validation with respect to the specifications. This activity starts at Sub-System PDR and terminates at System QR. It includes the following tasks: Implement FDIR and Validate and verify (this is typically carried out using a testing campaign, and repeated for Unit level, sub-system level and system level).

3 Tool Support

The FAME process is supported by a tool – called the FAME environment. Table 1 schematically illustrates tool support for the different process phases. Each phase of the FAME process is mapped with one or more tool functionalities – for each functionality, an explanation of its purpose is provided (the *contract-based testing* functionality is listed in italics, as it is part of future work).

The FAME environment is built on top of COMPASS [8], a framework for model-based design and verification, that provides several verification capabilities, including simulation, property verification, RAMS analysis (FTA, FMEA), diagnosability and FDIR analysis. The tool is freely distributed within the ESA member states. In the rest of this section, we briefly describe the capabilities that are specific of the FAME process, and in particular we focus on the underlying technological solutions. We refer to [5,16] for more details.

Mission and FDIR Specifications FAME provides the possibility to specify the Mission Phase/Spacecraft Operational Mode matrix, including information on phases, operational modes, and spacecraft configurations. Phase/mode

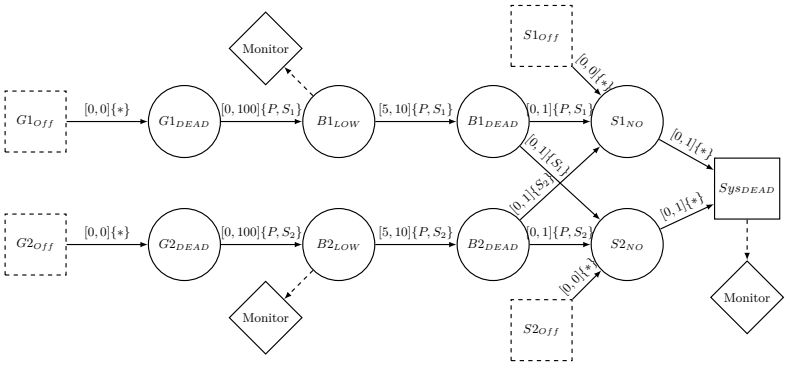

Fig. 3: An example of a TFPG

pairs can be tracked by FDIR to contextualize its strategies. FDIR requirements may be linked to specific phase/mode combinations, and they include the specification of spacecraft configurations as recovery targets, the specification of alarms that need to be fired, and the integration of existing FDIR components.

Fault Propagation Analysis Timed Failure Propagation Graphs (TFPGs) are used to model temporal interactions of failures and their effects [1]. TFPGs model how failures propagate, affecting various monitored and unmonitored properties, and are thus an abstract view of the underlying system. An example is shown in Fig. 3. A TFPG consists of basic failure events, discrepancy nodes representing off-nominal system properties, and edges connecting these nodes, representing possible propagation paths. The edges are constrained by system modes (they are enabled only in those modes). Finally, edges contain lower and upper time bounds for the failure propagations. FAME supports loading, syntactic verification, displaying, editing of TFPGs, and definition of nodes using basic expressions over system variables (to interpret the behaviors of the system in terms of the TFPG). FAME allows checking whether the system exhibits failure propagations that are not captured by the TFPG (*behavioral validation*), namely that all links between discrepancies and timing information are correct. If wrong values are present, a counter-example is produced to guide the user in the refinement process. If no counter-example is found, the analysis guarantees that the timing values are correctly specified. The tool also allows checking the TFPG adequacy as a model for diagnosis, using diagnosability analysis [12]. This is called *(diagnosability) effectiveness validation*. This analysis enables the identification of the failure modes that are not diagnosable. Finally, a (prototype) function based on FTA is available which enables the automatic derivation from the system model of a basic TFPG without precise timing and mode information.

FDIR Synthesis The FAME environment supports fully automated synthesis of diagnosis (FD) and recovery (FR) components, starting from an extended SLIM model, a TFPG, an FDIR specification, and an optional sampling rate. FD synthesis [7] creates a diagnoser that generates a set of alarms for each specified failure, by monitoring the available sensors. The FR model can be synthesized

using conformant planning [13]; it provides for each specified alarm and phase/-mode pair a recovery plan, i.e., a sequence of actions that guarantee to achieve the target under any circumstance. After synthesis, the FD and FR components are connected (in such a way that a generated alarm triggers the corresponding recovery plan) and combined with the original SLIM nominal model.

4 Case Study

The evaluation of the FAME process and environment was performed by Thales Alenia Space on a sub-set of the Trace Gas Orbiter (TGO) of the ExoMars project. The ESA ExoMars system will be launched in 2016 and will arrive at Mars approximately 9 months later in 2016. It is composed of a spacecraft that will carry an Entry and Descent Module (EDM) demonstrator. During the transit from Earth to Mars, the TGO will carry and provide power and other services to the EDM. The release of the EDM will take place prior to the critical Mars Orbit Insertion (MOI) manoeuvre by the TGO. After capture by Mars gravitation field, the TGO will orbit around Mars and provide support to the EDM. Once the EDM surface operations are completed, the TGO will start a science data acquisition phase. Near the end of this phase, the 2018 mission should arrive at Mars so that the emphasis may shift to the Rover support.

The case study was chosen since it provides an opportunity for evaluating all the aspects of the approach. In particular, it presents a complexity level that is representative of the classical complexity level in this domain. An FDIR design for the ExoMars TGO has already been developed, although the design has not yet been completely frozen. This provides the opportunity of comparing the results obtained with the FAME process with the existing results and, at the same time, provide feedback to the ExoMars team on the FDIR design.

The ExoMars mission can be divided into several mission phases, but in our case study we only consider the Mars Orbit Insertion (MOI). In this phase, several operational modes are used, including nominal modes (Routine - `ROUT`), safe modes (`SAFE_1`, `SAFE_2`) or degraded modes (Manouver in critical conditions - `MAN_C`) used by the FDIR when reconfiguration is required. The main functional chain considered during the case study is the Guidance, Navigation and Control (GNC) function which encompasses sensors, control software, and actuators. The goal of this sub-system is to maintain the correct spacecraft attitude. The faults that were considered are those related to the units that can lead to the main feared event considered in this study: the loss of spacecraft attitude.

4.1 Evaluation Criteria

In order to evaluate the FAME process and environment, we defined a set of questions, that are intended to evaluate the process, the technological choices and the prototype environment independently.

Process Is the process suitable for an industrial project, and coherent with the applicable standards, and the project lifecycle? What are its benefits with

Item	Fault	Failure Detect Method	Local Effect	System Effect
IMU_001	Sensor signal is too low	EXT	Continuous self-reset of IMU	No measure is sent
IMU_002	Sensor output is biased	EXT	None	Biased output from sensor channel
IMU_003	Sensor output is erroneous	INT	Loss of RLG dither control	Erroneous output from sensor channel

Fig. 4: FMECA Table

respect to the current industrial process? Are there any blocking points, issues, or concerns which could limit its application? Was any insight gained from modeling and specifying the requirements in a more formal way? How do the results obtained with FAME compare with the results obtained by the ExoMars team?

Technology Is TFPG formalism suitable for handling space domain FDIR issues? Can all relevant constraints be expressed in the modeling language?

Environment Is the FAME environment suitable for industrial needs ? Is the GUI efficient? Does the prototype help in modeling the FDIR-related aspects of the system? Did the synthesized TFPG resemble the manually designed one?

4.2 FAME Process Applied to the Case Study

We used the system architectural and behavioral information, and information concerning the mission phases and operational modes, as inputs to model the nominal system. Modeling was done in the SLIM [9] language.

Feared Event Analysis and FMECA The first activity for safety analysis is the Feared Event Analysis. We are interested in the feared events coming from the units realizing the acquisition of the spacecraft attitude: the Inertial Management Units (IMU). We consider three possible failure modes: *No measure*, *Biased measure* and *Erroneous measures*. The only failure mode that is not diagnosable is biased measure, while absence of measure and erroneous measures are detectable either by rate control, or by cross-checking with other readings. Using documentation and FMECA from IMU equipment supplier, the IMU FMECA Items are analyzed and those having impact on the system are selected. Others are discarded with justification. Fig. 4 gives a selection of three FMECA items of the IMU equipment. We can see that the local and system effects can be matched with the failure modes identified in the previous feared event analysis.

Error model and Fault injection Using the FMECA information, the system can be enriched with faulty behaviors. Thus, an *error model* is defined for the IMU component; the FMECA items are translated to error events, the failure modes are translated to error states, and the local / system effects are used to define the fault injections [10,8].

Failure Propagation Modeling Sub-sets of system failures have been considered in order to simplify the analysis, e.g., in cold redundancy the failures of nominal equipment are analyzed independently of the failures of the redundant equipment. We start by considering the IMU equipment: in any mission phase where this unit is used, the failures propagate into the system (at this stage no

FDIR prevents the propagation), and impact the spacecraft attitude, leading to the feared event we are interested in: loss of the spacecraft attitude. This fault propagation is modeled in the SLIM model by implementing for each function the effect on its outputs given erroneous input values. Since fault propagation may not be immediate, it is important to consider timing information. Function implementations therefore introduce delays in this propagation. Given the nominal SLIM model, the error models and the fault injections, and based on the consolidated failure propagation analysis performed using the TFPG, we can perform fault tree analysis for the case study - the results are as expected.

Analysis and Specification of FDIR user requirements For our case study, a set of requirements coming from the ExoMars TGO project are analyzed to produce FDIR objectives, strategies, and specifications – that are in turn provided to the FAME environment. The complete FDIR specification for the case study defines one alarm for each of the selected faults (Fig. 4) and for each of the IMUs, thus providing a total of 6 alarms in the specification. Each alarm is associated with a recovery requirement for each possible phase and mode combination. In total, this provides us with 24 recovery requirements, 12 for each IMU unit – the nominal one (IMU1) and the redundant one (IMU2).

Timed Fault Propagation Graph Modeling Instead of building a global TFPG that would cover all failures of the system for all modes, the approach we adopted is to build several TFPGs covering the failures of respectively the nominal and the redundant IMU. With the current FAME environment only one TFPG is taken into account for the FDIR specification and the FD and FR synthesis. Therefore, independent FD and FR blocks are obtained for the two equipment units, leading to a decentralized FDIR. From the SLIM model enhanced with timing aspects and the error model, the TFPG model for IMU_1 failures is defined manually or synthesized using the toolset. The modeled TFPG is shown in Fig. 5. On this TFPG we can see the three failure modes of the nominal IMU equipment propagating in the system. On the propagation path, discrepancies are induced; some of them are observable, whereas some are not. Associations between discrepancies modes and system model define the relation between nodes in the TFPG and the original system. The TFPG is validated using behavioral and effectiveness validation analysis provided by the tool, verifying that the diagnosability of the system is well captured in the TFPG.

Fault Detection and Recovery Synthesis The fault detection synthesis is run based on the fault detection specification. The result is an FD SLIM module that encodes a finite state machine with 2413 states. The fault recovery synthesis is run based on the fault recovery specification, only considering IMU_1. The result is an FR SLIM module with recoveries (6 recoveries out of 9 are found). The missing recoveries identify situations in which there is no strategy that can guarantee the recovery (note that it might be possible to find strategies that would work under certain circumstances, however the tool focuses on finding solutions that always work).

FDIR verification In order to verify the resulting FDIR, both the SLIM system model extended with fault injections and the synthesized SLIM FDIR

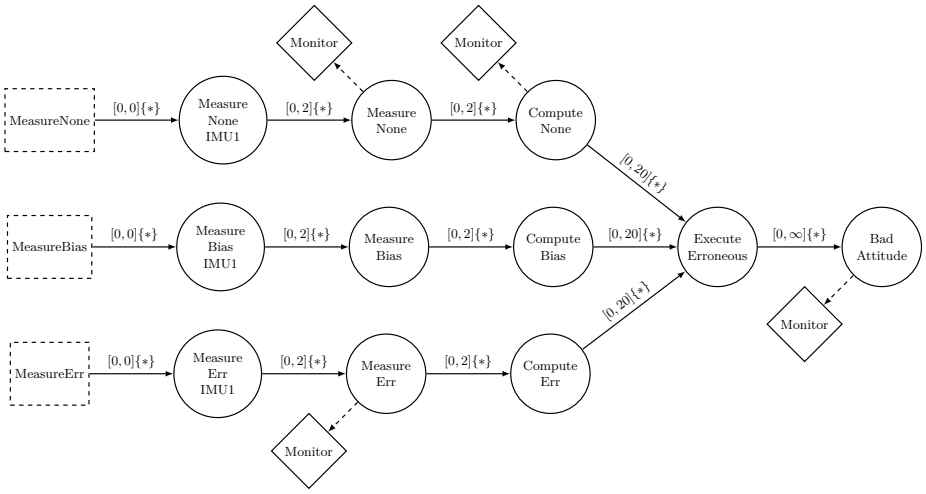

Fig. 5: Case study TFPG

components have to be loaded into the COMPASS toolset for performing FDIR analysis (Fault Detection verification and Fault reconfiguration verification). As the synthesized FD model has numerous states, a lot of memory is required to load it in the toolset, which may require a powerful machine. The FD component is hard to verify by looking at the synthesized SLIM file, whereas the FR components can be checked easily in order to understand the content of the reconfiguration sequences.

4.3 FAME prototype toolset evaluation

The case study allowed us to evaluate the FAME prototype environment, and to provide some answers for our evaluation criteria, as discussed below.

Process We believe that the FAME process is suitable for an industrial project, and coherent with the applicable standards and project lifecycle. This is particularly true when dealing with single failure tolerant systems, since the analysis of fault propagation is performed in isolation for each failure or group of failures. If multiple failure tolerance is required, this approach should be consolidated to find the best way to analyze the failure combinations. Projects can benefit from FAME in the initial phases where FDIR is not yet defined. Time spent for analysis of mission requirements can be reduced by using FAME tool. The FAME process can be inserted easily in the current industrial process, provided that the users receive training on TFPGs. The synthesized SLIM models can be used as a starting point to implement HW and SW. Moreover, compared with the current industrial process, FAME permits to rely on early analysis of the robustness of the system and early design of FDIR. Formal modeling also allows to clarify the design, and to consider the system as a whole. Formalization is desired in order to prevent (mis-)interpretation, and forces the definition

of clear requirements. Traditionally, time in failure propagation is often a major concern; the FAME process takes a step towards simplifying this type of analysis. Compared to the ExoMars project, the FDIR specification was similar to the real one; moreover, fault propagation analysis was not addressed at the same level, and FAME produced richer results. Fault Detection and Reconfiguration cannot be compared with the real project results, since they represent a completely novel approach relying on synthesis and time aware diagnosers.

Technology SLIM enables a good characterization of the system; however, adding external constraints might be required in order to perform efficient model analysis (e.g., FR synthesis). The TFPG formalism seems to be adequate to handle the description of failure propagation during the FDIR design. Although the timing information for transitions is well understood, the mode information is not yet well understood from the user perspective. This has to be investigated further, in order to understand how this piece of information be exploited to model TFPGs per each failure (or sub-set thereof). This "mode" information will certainly be useful when dealing with TFPG combination. Unfortunately, the application of the FAME process to the industrial world may be limited by the state-space explosion when introducing time on complex models. This might impact the possibility of performing certain analyses or the automated synthesis process. It would be important to explore existing techniques from the formal methods community (e.g., contract-based design [11]) to deal with this problem.

Environment The prototype environment is able to support the FAME process. Moreover, the structure of the synthesized TFPG was exactly the same as the one designed manually. Some limitations should be addressed in order to simplify the application of the process. The TFPG editing capabilities of the prototype FAME environment could be improved. The environment provides both textual editing and graphical view. Unfortunately, both methods are not user-friendly when dealing with big graphs. The environment provides little support to the traceability of requirements; this however could help improve the design cycle. Finally, another possible limitation could come from the incompatibility of formalisms used by the eco-system of tooling already in-use in the space domain. For instance, TAS uses an in-house modeling tool (Melody Advance) to model the system from early stages of the development process to later stages. Modeling is a time- and cost-consuming activity, hence a project can not afford to develop several models of the same system in different languages. It is recommended that SLIM models used in the FAME process are not created from scratch, but derived from existing models of the system. This is partly achievable, and a connection with Melody Advance seems possible.

5 Related Work

There are several previous works on model-based formal verification and failure analysis. The TOPCASED project [4] provides a toolchain that translates AADL and its Behavior Annex into Fiacre, and from Fiacre into timed Petri nets, which can be then model checked using the TINA toolbox. In [19], the authors present

an ontology-based transformation of AADL models into the Altarica formal language; they can detect lack of model elements and semantically inconsistent parts of the system, but timed or hybrid extensions remain out of scope. Another AADL-based tool is ADeS [2] for simulation of system architectures. In [18], the authors present a framework for model based safety analysis, that covers both probabilistic and qualitative analysis; transformations into state-of-the-art model checkers, including PRISM and NuSMV, are available. Finally the FAME tool is based on the SLIM variant of AADL, however similar technologies are also available for the NuSMV family of languages [20,21].

[17] contains an interesting account of different notations for evaluating component-based systems and failure propagation, namely FPTNs, HiP-HOPS, CFTs and SEFTs. While TFPGs contain explicit temporal information, such as nondeterministic propagation timings and dependency on system modes, other models (e.g., SEFTs) enable the definition of probabilistically distributed delays, they provide FTA-like notation using temporal gates, and they can be evaluated quantitatively. A more detailed comparison will be done as part of future work.

As regards the FDIR modeling and process, in [3] the authors explore a few alternatives for modeling and verification of FDIR sub-components, using languages such as Altarica, SCADE, SLIM and the COMPASS tool; the FAME process and tool address some of the questions the authors raise, and also provide fault propagation analysis. Finally, [6] analyzes some of the issues for the development and validation of FDIR; FDIR functions, mechanisms and are expressed in AADL models and verified using the Uppaal model checker.

6 Conclusions and Future Work

In this paper we have presented a novel, model-based, dedicated process for FDIR development, verification and validation in aerospace. This process allows for a consistent and timely FDIR conception, development, verification and validation. The process is based on formal model-based approaches that enforce a high level of rigor and consistency; it can be integrated with the system and software development lifecycle, and it enables to effectively use the available mission and system requirements and the RAMS analysis data. The process has been successfully evaluated within an industrial setting.

As part of our future work, we will consider the extension of our framework in a few directions. In particular, we will investigate the specification of FDIR requirements, and synthesis of FDIR, in presence of decentralized or distributed architectures, where coordination is needed between different FDIR sub-components. In this context, FDIR decomposition can be driven by notions such as scope, context, and level of authority of FDIR. In the same area, we will explore the possibility to decompose, possibly hierarchically, the specification of fault propagation using multiple failure propagation models.

References

1. S. Abdelwahed, G. Karsai, N. Mahadevan, and S.C. Ofsthun. Practical implementation of diagnosis systems using timed failure propagation graph models. *Instrumentation and Measurement, IEEE Transactions on*, 58(2):240–247, 2009.
2. ADeS, a simulator for AADL. `http://www.axlog.fr/aadl/ades_en.html`.
3. E. Bensana, X. Pucel, and C. Seguin. Improving FDIR of Spacecraft Systems with Advanced Tools and Concepts. In *Proc. ERTS*, 2014.
4. B. Berthomieu, J.P. Bodeveix, P. Farail, M. Filali, H. Garavel, P. Gaufillet, F. Lang, F. Vernadat, et al. Fiacre: An Intermediate Language for Model Verification in the TOPCASED Environment. In *Proc. ERTS*, 2008.
5. B. Bittner, M. Bozzano, A. Cimatti, R. de Ferluc, M. Gario, A. Guiotto, and Y. Yushtein. FAME: A Model-Based Environment for FDIR Design in Aerospace. 2014. In Proc. IMBSA 2014.
6. J.-P. Blanquart and P. Valadeau. Model-based FDIR development and validation. In *Proc. MBSAW*, 2011.
7. M. Bozzano, A. Cimatti, M. Gario, and S. Tonetta. A formal framework for the specification, verification and synthesis of diagnosers. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
8. M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, T. Noll, and M. Roveri. Safety, dependability, and performance analysis of extended AADL models. *The Computer Journal*, doi: 10.1093/com, March 2010.
9. M. Bozzano, A. Cimatti, V. Y. Nguyen, T. Noll, J.-P. Katoen, and M. Roveri. Codesign of Dependable Systems: A Component-Based Modeling Language. In *Proc. MEMOCODE '09.*, 2009.
10. M. Bozzano and A. Villafiorita. The FSAP/NuSMV-SA Safety Analysis Platform. *Software Tools for Technology Transfer*, 9(1):5–24, 2007.
11. A. Cimatti, M. Dorigatti, and S. Tonetta. OCRA: A tool for checking the refinement of temporal contracts. In *ASE*, pages 702–705, 2013.
12. A. Cimatti, C. Pecheur, and R. Cavada. Formal Verification of Diagnosability via Symbolic Model Checking. In *Proc. IJCAI*, pages 363–369. Morgan Kaufmann, 2003.
13. A. Cimatti, M. Roveri, and P. Bertoli. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*, 159(1):127–206, 2004.
14. The COMPASS Project. `http://compass.informatik.rwth-aachen.de`.
15. European Cooperation for Space Standardization. European cooperation for space standardization web site. Available at `http://www.ecss.nl/`.
16. The FAME Project. `http://es.fbk.eu/projects/fame`.
17. L. Grunske, B. Kaiser, and Y. Papadopoulos. Model-Driven Safety Evaluation with State-Event-Based Component Failure Annotations. In *Proc. CBSE*, pages 33–48, 2005.
18. M. Güdemann and F. Ortmeier. A Framework for Qualitative and Quantitative Formal Model-Based Safety Analysis. In *Proc. HASE*, pages 132–141, 2010.
19. K. Mokos, G. Meditskos, P. Katsaros, N. Bassiliades, and V. Vasiliades. Ontology-Based Model Driven Engineering for Safety Verification. In *Proc. SEAA*, pages 47 –54. IEEE, 2010.
20. The nuXmv model checker. `https://nuxmv.fbk.eu`.
21. The XSAP safety analysis platform. `https://es.fbk.eu/tools/xsap`.