ELMASAT EXECUTIVE SUMMARY

Abstract

This document gives an overview of the objectives and results of the ELMASAT project (OSIP 2020, ESA contract 4000133652/20/NL/GLC/kk).

DATE: 2022/03/15

REFERENCE: V078L00T00-008

Author(s)	*Function(s) & name(s)*	*Research Engineers*	*E.JENN* [illegible]
Checkers(s)	*Function(s) & name(s)*	*Research engineer*	*G. BRAU* [illegible]
Approver	*Function & name*	*Center Of Competence Leader*	*E. JENN* [illegible]

Revision Table

Issue	Date	modified §	Evolution summary	Modified by
1.0	2022/03/15	All	First version for delivery	E. JENN
2.0	2022/03/24	Headers and footers	Modifications according to ESA review (RID)	E. JENN

Table of Contents

1 Context of the study

Model-based Safety Analysis (MBSA) is an emerging technology. It constitutes the natural sequel of the now well-established Model-Based System Engineering (MBSE) practices.

Academic, commercial, and proprietary MBSA products are already available. This is for instance the case of APSYS' SIMFIANEO or Dassault Aviation's CECILIAS/OCCAS, both of them being based on the Altarica modelling language and being used for the certification of jet avionics products. Those environments enable to model the dysfunctional behaviours of the electrical and electronic architecture of systems in order to perform qualitative and quantitative safety analyses.

These solutions rely on two distinct models: one to support the design activities and another to support the safety and availability analyses. This segregation of concerns *and* models may lead to the "desynchronization" of the two models and, consequently to discrepancies between the system being designed and the system being assessed.

To overcome this issue, solutions such as the COMPASS tool, based on the AADL language and extended with the SLIM language, enable fault and FDIR modelling and analyses to be tightly coupled with the design of the system architecture. However, availability assessment is not covered. Another solution is SOPHIA, built on top of SySML/Papyrus environment by CEA-LIST. The SOPHIA framework embeds specialized profiles and viewpoints for FTA analysis and generation using Altarica in background, for FMEA table generation and for formal verification of properties with the NuSMV language and formal solvers. Yet, no support is provided for availability evaluation in SOPHIA.

In order to support the coupling between MBSE and MBSA, we propose to start from the Capella MBSE environment in order to leverage its relatively large dissemination and openness. Capella is for instance the reference framework for system design at Thales. This environment offers capabilities to extend domain models and create specific "viewpoints" in order to support specific activities. For instance, several viewpoints have been developed for safety analysis. The All4tech Company developed the Safety Architect product enabling to capture information for safety analysis and to interoperate with the Capella and Papyrus environments. Thales Alenia Space (TAS) has also developed a reliability viewpoint for satellite industry needs, allowing to capture reliability data for satellite units and then to compute reliability figures of the system. However, the aforementioned Capella-based viewpoints do not allow performing availability analysis. Furthermore, they do not allow capturing the concepts needed to perform availability analysis in the presence of some specific physical faults.

New trends in space market require miniaturization of equipment, new embedded functionalities, a new design and production process and strong competitiveness. Electronic Commercial-off-the-Shelf (COTS) components such as System-On-Chip (SoC) are key enablers for high processing with wide flexibility. However, those components are generally not designed to be insensitive or tolerant to cosmic radiations (e.g., the width of the transistors is too small, the implementation technology is not tolerant to radiations, etc.). Therefore, a fine analysis is necessary to find the good compromise between protection, mitigation, redundancy and mission availability. We limit the study to Single-Event Effects (SEE) sensitivity analysis.

The traditional design for satellite on-board computer embeds rad-hard component non-sensitive to radiations. In this case, the SoC sub-system is considered as a "black box" because no details about its architecture are required to perform the safety and availability analyses. Our approach intends to "open the black box" and expose some of the internal details of the hardware components in order to account for its sensitivity to SEE, internal failure detection and mitigation. The failure propagation and mitigation is evaluated across the functional chain to compute the availability of the SoC sub-system.

2 Objectives of the study

The main objective of the ELMASAT study is to add to Capella the capability to calculate the availability of a system and provide facilities to support trade-off analyses. In particular, these features will allow assessing a SEE detection and mitigation strategies with respect to availability objectives.

Towards that general goal, the project targeted several major intermediate objectives:

- Build (or reuse) a model of a satellite avionic system to support the development of the tool and validate it
- Build the meta-model capturing the concepts and data supporting availability analysis (fault occurrence, error propagation, failure, consolidation, error detection, mitigation, etc.)
- Revisit the availability calculation algorithm

3 Overview of results

MBSE and MBSA

The ELMASAT availability assessment tool is built upon the Capella MBSE environment. Hereafter, we briefly introduce the main concepts required to understand the MBSE/A approach implemented in the tool. Note that this is only a very partial view of the Capella environment. More information can be found, for instance, in the introductory book "Model-based System and Architecture Engineering with the Arcadia method[1]".

Capella

An example of a system architecture described in Capella is given on Figure 1. This diagram shows the architectural elements involved in availability assessment: *physical functions* (in green), mapped on *physical* behavioural *components* (in blue), themselves mapped onto *physical nodes* (in yellow). Basically, functions specify *what* is to be done while physical components describe *how* it is done (or implemented). Functions exchange items via their input and output ports and though *connections* between ports (represented by an edge). In a complete model, the functional communication links would themselves be "mapped" onto physical links; this is not represented here.

A set of functions implement one or several *functional chains.* A functional chain represents a path crossing several functions involved in the realisation of one, higher level, function.

Note that, in the context of the project, we only consider the physical view of the system. However the Capella modelling environment and its associated engineering methodology (ARCADIA) address other *perspectives* of the engineering process, including those supporting the operational analysis, system analysis, and logical analysis.

Availability and error propagation

In the tool, availability is defined and computed with respect to one or several functional chains, and a functional chain is considered "available" if a certain combination of the functions that constitute the chain is available. The availability of a function is defined with respect to its outputs. A function is available if its outputs are non-erroneous (no failure).

Errors propagate in the system via two channels. First, errors propagate via functional exchanges between functions and, inside a function, via the functional dependencies between its inputs and its outputs. This corresponds to the situation where a function receives and processes erroneous data sent by other functions. Second, errors propagate from the physical node (for instance, the one hosting the function) to the function. For instance, this corresponds to the case where the processor executing some software function is faulty.

In the ELMASAT tool, propagation of errors *between functions* exploits the connections between functional ports introduced before. Inside a function, propagation is modelled using logical equations expressing the relationship between the occurrence of an error on a functional output and the occurrence of an error on functional inputs. This flexible scheme allows modelling situations where a function's output is erroneous if any of its input is erroneous (worst-case) or the case where a specific combination of its inputs are erroneous.

Propagation of errors between physical nodes and functions is expressed explicitly by indicating that a function *depends on a component* using modelling extensions introduced by the viewpoint. These modes of propagation are illustrated on Figure 2.

Faults and error modelling

For an error to occur, some fault must be activated[2]. In the ELMASAT tool, focus is placed on hardware faults due to radiations. The modelling environment allows capturing the probability of occurrence of the event (the fault) and the probability for the fault to be activated and become an error.

[1] Jean-Luc Voirin, "Model-based System and Architecture Engineering with the Arcadia Method", ISBN 978-1785481697. Even though this book focuses on the method (Arcadia), it also describes the modelling environment (Capella) that supports the method. Other information can also be found on the Web (see e.g. the Wikipedia article).

[2] "An error is that part of the system state which is liable to lead to subsequent failure: an error affecting the service is an indication that a failure occurs or has occurred. The adjudged or hypothesized case of an error is a fault" (Laprie, Dependability: basic Concepts and Terminology, Springer-Verlag, 1992)

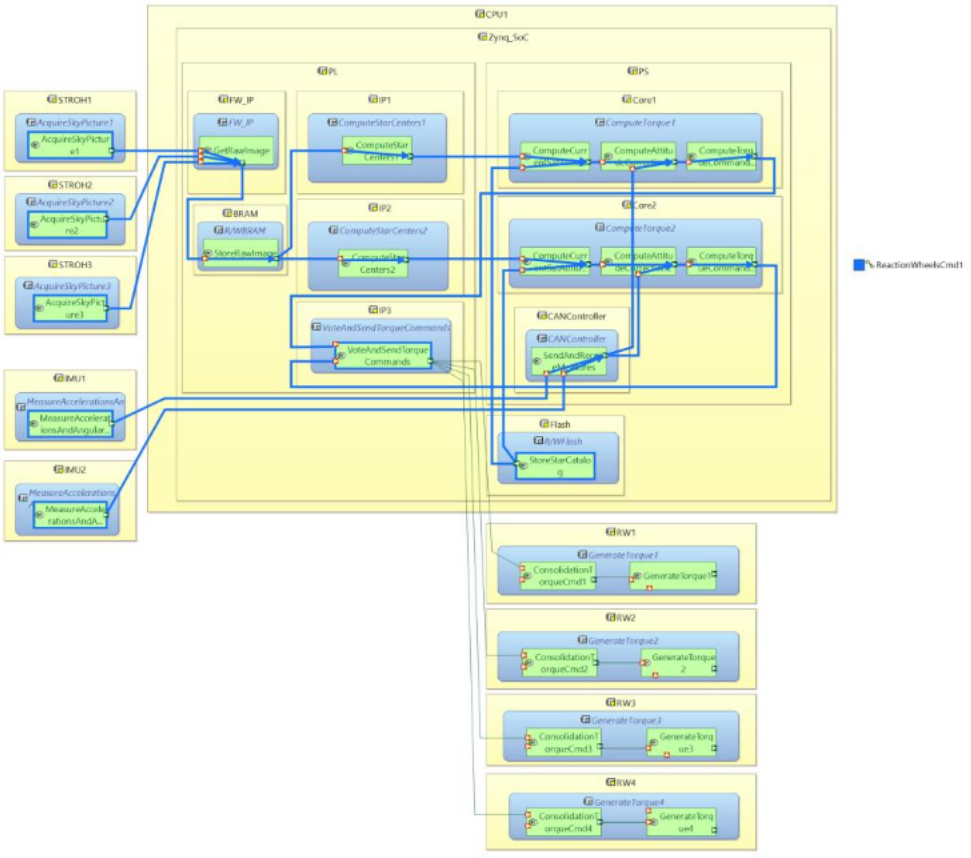
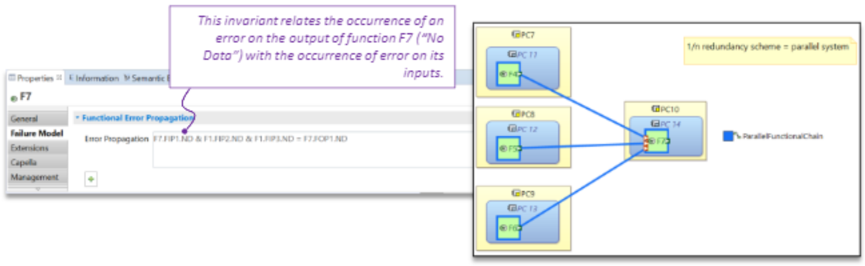


Figure 1. A Capella model ('Reaction Wheels Command' functional chain, CPU1)


This invariant relates the occurrence of an error on the output of function F7 ("No Data") with the occurrence of error on its inputs.

F7

Functional Error Propagation

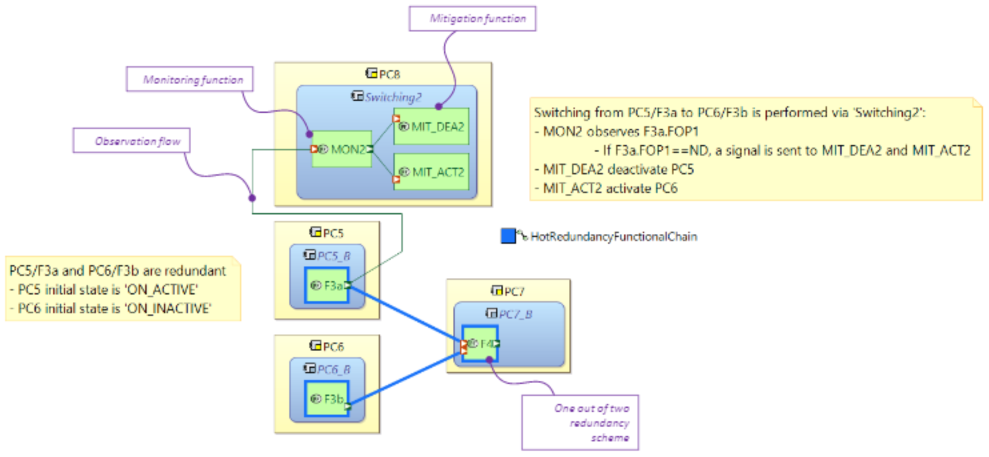
Error Propagation: F7.FIP1.ND & F1.FIP2.ND & F1.FIP3.ND = F7.FOP1.ND

Figure 2. Intra-function error propagation modelling

Error detection and mitigation, consolidation schemes

Without any error mitigation means, an error affecting a functional chain (so the service delivered by the system) would eventually lead to a system failure and, consequently, to the unavailability of the system. In that case, the (mean) unavailability would be $\frac{1}{2}\lambda T_m$ where λ is the failure rate and T_m the duration of the mission. Without any error detection and mitigation means, it is likely that non rad-hard COTS components will not allow reaching availability objectives due to a very high value of λ.

The ELMASAT tool models error detection and mitigation means using specific functions (see Figure 3). The error detection function "observes" (or "monitors") the outputs of any function, detects the occurrence of an error, and triggers a mitigation function. The mitigation function operates on the physical node to eliminate the error (i.e., "repair" the node). Since those functions are similar to those contributing to deliver the operational services, they can be deployed on any physical node, including the one that is monitored. This allows in particular to account for the failure of the execution structure of error detection and monitoring.



Switching from PC5/F3a to PC6/F3b is performed via 'Switching2':
- MON2 observes F3a.FOP1
  - If F3a.FOP1==ND, a signal is sent to MIT_DEA2 and MIT_ACT2
- MIT_DEA2 deactivate PC5
- MIT_ACT2 activate PC6

PC5/F3a and PC6/F3b are redundant
- PC5 initial state is 'ON_ACTIVE'
- PC6 initial state is 'ON_INACTIVE'

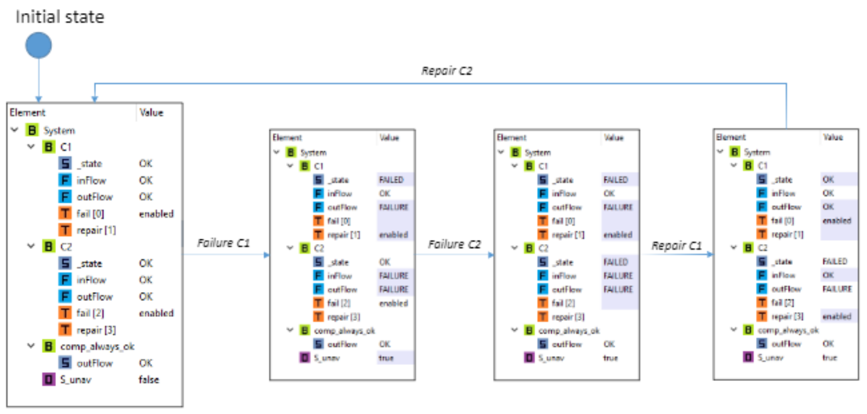
Figure 3. Error detection and mitigation

The ELMASAT tool also supports the modelling of various redundancy schemes, including n-out-of-m in the cold and hot modes (see Figure 3). Again, this is achieved using dedicated functions embedding the consolidation logic.

Availability evaluation and trade-off analysis

Once the attributes required to model the fault occurrence process have been added, possibly with the architectural elements to express error monitoring and mitigation, the availability of functional chains can be computed.

The ELMASAT tool implements two calculation schemes: one calculating the availability "analytically" by traversing the structure of the model and accounting for the contribution of each component, and another one based on stochastic discrete event simulation. In the latter case, the design model is transformed to an Altarica model[3] that can be first validated using interactive simulation (see Figure 4) and evaluated using stochastic simulation.



Element	Value
System	
C1	
_state	OK
inFlow	OK
outFlow	OK
fail [0]	enabled
repair [1]	
C2	
_state	OK
inFlow	OK
outFlow	OK
fail [2]	enabled
repair [3]	
comp_always_ok	
outFlow	OK
S_unav	false

Element	Value
System	
C1	
_state	FAILED
inFlow	OK
outFlow	FAILURE
fail [0]	
repair [1]	enabled
C2	
_state	OK
inFlow	FAILURE
outFlow	FAILURE
fail [2]	enabled
repair [3]	
comp_always_ok	
outFlow	OK
S_unav	true

Element	Value
System	
C1	
_state	FAILED
inFlow	OK
outFlow	FAILURE
fail [0]	
repair [1]	enabled
C2	
_state	FAILED
inFlow	FAILURE
outFlow	FAILURE
fail [2]	
repair [3]	
comp_always_ok	
outFlow	OK
S_unav	true

Element	Value
System	
C1	
_state	OK
inFlow	OK
outFlow	OK
fail [0]	enabled
repair [1]	
C2	
_state	FAILED
inFlow	OK
outFlow	FAILURE
fail [2]	
repair [3]	enabled
comp_always_ok	
outFlow	OK
S_unav	true

Figure 4. Interactive simulation of an Altarica model

Once the availability of the system can be evaluated, trade-off analyses can be performed.

A trade-off analysis is aimed at evaluating and comparing multiple architectures with respect to a set of metrics. In the current version of the tool, metrics include the one brought by the availability viewpoint developed specifically in the project, and the "basic viewpoints" provided with Capella (price, mass, and performance).

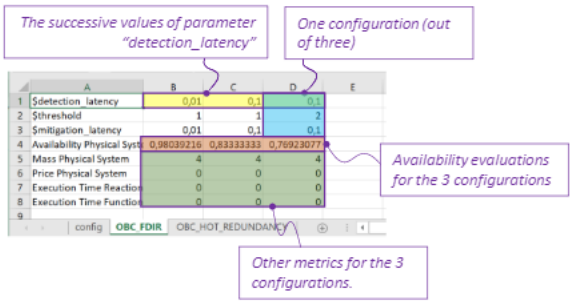
Usually, a trade-off analysis consists to build some Pareto front involving a pair of parameters and a constraint. A typical case is the one where the constraint is an availability objective, and the parameters are any pair in {price, mass,

[3] Altarica is a high-level modelling language dedicated to dependability analyses. With respect to other classical formalisms such as Stochastic Petri Nets, Markov chain, Fault Trees, etc., the structure of an Altarica model reflects more directly the structure of the system under assessment. In the context of the ELMASAT project, this strongly facilitates the transformation from the design model to the analysis model. In ELMASAT, we use the Altarica V3 language and the Altarica Wizard tool (both available from the Altarica Association at http://www.altarica-association.org/).

performance}, but other problems may consider constraints concerning any other of the four metrics. The Pareto front is established by evaluating a set of configurations.

In order to facilitate the sampling of the configuration space, parameters of the model (e.g., fault rate, coverage factor, delays, etc.) can be associated with “symbolic” parameters. In the tool, the actual valuation of those parameters is given in a dedicated Excel sheet: one configuration is one vector (or column) of the sheet (see Figure 5). The tool uses the Excel file as the specification of the evaluations to be performed and collects the generated results in the same file. This way, the user benefits from all the features of Excel to build a series of configurations (i.e., compute configuration vector *n* from configuration *n-1*), compute specific metrics for those configurations, and generate diagrams.

It is important to note that a “configuration” refers to a set of values of the numerical parameters characterizing the model, not to the architecture of the model itself. Therefore, if the exploration space covers several architectural alternatives, several models must be built and evaluated separately, and results must be consolidated by hand.


The successive values of parameter “detection_latency”
One configuration (out of three)
Availability evaluations for the 3 configurations
Other metrics for the 3 configurations.
config
OBC_FDIR
OBC_HOT_REDUNDANCY

	A	B	C	D	E
1	$detection_latency	0,01	0,1	0,1	
2	$threshold	1	1	2	
3	$mitigation_latency	0,01	0,1	0,1	
4	Availability Physical Syst	0,98039216	0,83333333	0,76923077	
5	Mass Physical System	4	4	4	
6	Price Physical System	0	0	0	
7	Execution Time Reaction	0	0	0	
8	Execution Time Function	0	0	0	
9					

Figure 5. A configuration using 3 parameters

4 Conclusion and future work

The Availability Viewpoint extends the Capella MBSE environment with the modelling and calculation capabilities to evaluate availability from a physical architectural model.

In the current version of the tool, focus is placed on hardware faults due, in particular, to radiations (Single Event effects). The viewpoint captures the propagation of errors through the dependencies between the functions and their supporting hardware resources (execution unit, memory, …) and though the functional flows. The viewpoint provides support to model and evaluate consolidation and reconfiguration schemes.

Two modes of availability computations are implemented. The first one uses the analytical formulation of availability for each component (e.g., $U = \frac{\lambda \times \tau}{1+\lambda \times \tau}$ for a repairable component) and the application of unavailability computation formulae for modular architectures, and the second one uses stochastic simulation.

The tool provides support to perform simple trade-off analyses thanks to the capability to use symbolic parameters in the model and explore multiple valuations of these parameters in an automatic manner.

Among the possible extensions, we can mention:

- Extended trade-off capability

  In the current version of the tool, trade-off analyses rely on three main features: the capability to use symbolic parameters, the capability to estimate quantities provided by other viewpoints such as mass or performance, and the capability to evaluate a series of models. However, the architecture of the model is **not** a configuration parameter. So, if the user wants to investigate multiple architectures differing structurally (e.g. a 2 out of 3 architecture *vs.* a dual architecture with self-checking pairs), s/he needs to build *different models*.
- Link with parameters databases

  In the current version, values of parameters are given either “internally” in the modelling environment or “externally” using symbolic parameters valuated in an Excel sheet. A simple extension would be to retrieve those parameters from a database. (Note that this could be easily implemented in the current version using VBA in Excel.)

More fundamentally, the tool is limited to the evaluation of availability. It could be extended to cover other RAMS analyses or, conversely, be integrated in a broader environment. This objective is addressed in another ESA project, “Integration of Reliability Engineering in Model Based Systems” (ITT 10766).

END OF DOCUMENT