

Advanced Tools and Model-Based Analysis for Digital and Software-Defined Systems in Safety-Critical Applications

Samuel García Lorente*
Systems Safety Engineer
Anzen Aerospace Engineering
Madrid, Spain
samuelgarcia@anzenengineering.com
*Corresponding author

Daniel Villafañe Delgado
Systems Engineer,
Anzen Aerospace Engineering
Madrid, Spain
danielvilla@anzenengineering.com

Pablo López Negro
Head of Innovation
Anzen Aerospace Engineering
Madrid, Spain
pablolopez@anzenengineering.com

Abstract— Safety-Critical applications rely more and more on Digital and Software-Defined Systems. These systems, not only bring in general enhanced functionalities and performances compared to its predecessors, but also offer additional capabilities from a safety and reliability standpoint, such as autonomous failure detection and self-reconfiguration.

In this paper we explore the use of a Model-Based approach, combined with Markov analysis and formal methods, to verify safety, reliability and availability features of a complex system. The proposed methodology supported with a dedicated tool aims at facilitating the task of systems safety engineers offering an easy-to-use interface to mask the complexity of the mathematical methods and enable its application in industrial projects.

Keywords— digital systems, software-defined systems, safety-critical applications, model-based analysis, Markov, formal methods

I. INTRODUCTION

Digital and Software-Defined Systems are systems where the control and operation are primarily governed by software rather than hardware [1]. Key components of these systems are microprocessors, microcontrollers, digital signal processors and field-programmable gate arrays (FPGAs); that are today widely used in Aerospace applications and other Safety-Critical domains [2].

Digital and Software-Defined Systems offer enhanced functionalities and performances, as operations are controlled with software, and this often enables the implementation of complex logic. Additionally, these systems allow for more flexibility, adaptability and scalability as changes and updates can be implemented through software modifications without needing to alter the hardware [3].

These systems are used in scenarios where the operational complexity is high, and where flexibility, rapid deployment and reconfiguration are essential [4]. Solely in the aerospace industry, these systems are basic building blocks for state-of-the-art aircraft, launchers, satellites and new emerging applications such as Unmanned Aerial Systems and Vehicles

(UAS and UAV), Remotely Piloted Aircraft (RPAS), High Altitude Pseudo-Satellites (HAPS), etc.

These systems have in common that they require some level of autonomy to conduct operations as humans are only partially involved in the control loop. In some cases, like RPAS, with almost continuous monitoring and frequent control, and in others like HAPS, launchers or satellites with just very sporadic control actions. From the safety and dependability standpoint, this means that, up to a certain extent, the system must implement also autonomous means for identifying non nominal conditions and failures and react appropriately to ensure safety and/or availability of the mission. This is often referred to as Failure Detection, Isolation and Recovery (FDIR), in the space domain [5].

In this paper we explore the state-of-art safety analysis methods and tools applied to systems with autonomy features and FDIR capabilities. As it will be presented in the next section, traditional System Safety Analysis techniques are not suitable for analyzing conditions evolving dynamically subjected to autonomous processes occurring on-board. On the other hand, advanced techniques such as Markov Analysis and Formal Methods, especially suited to the task, are complex to apply and have little adoption at industry level. With our work we intend to fill the gap between existing advanced analysis frameworks and industrial Safety Analysis practices, leveraging on the Model-Based approach as an enabler to facilitate the tasks of engineers when dealing with complex digital and software-defined systems.

Section II presents different System Safety Analysis techniques and focusses in particular in the new opportunities enabled by implementing Model-Based and formal analysis approaches. In Section III a methodology and tool leveraging these advanced concepts is presented. Section IV describes a proof-of-concept use case using these tools. Finally, Section V discusses potential applications and provides a brief recapitulation and anticipates some future work to continue this line of investigation.

II. SYSTEMS SAFETY ANALYSIS

System Safety Analyses in aeronautics industry are mainly carried out under the guidance of ARP 4761A [6] “Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, and Equipment”, which is an Aerospace Recommended Practice from SAE International, widely recognized in industry, and defines a set of partially ordered activities that need to be performed in support of the airworthiness process to handle failures or malfunctions that may lead to a hazard [7]. ARP 4761A is applied under the context of the development process described in ARP 4754B [8] “Guidelines for Development of Civil Aircraft and Systems” [9]. ARP 4761A covers safety methods such as Functional Hazard Analysis (FHA), Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA), Markov Analysis (MA), and Failure Propagation Model (FPM), among others.

In a typical aeronautical safety assessment [10], the entry point is the aircraft level FHA, where the failure conditions of the aircraft are identified. Then those relevant failure conditions are analyzed by the aircraft level FTAs. The aircraft level FTAs serve as identification of system level failure conditions, which are assessed through the different system level FHAs and FTAs (or equivalent analysis, but FTA is often preferred because is conceptually simple, reasonably easy to understand, and can handle very complex systems [6], in addition to being suitable for analyzing the failure conditions and events in an isolated manner [11]). System level FTAs are also fed by equipment reliability analysis, such as FMEAs, that characterize the equipment level failure modes. These latter values can be scaled upwards to obtain the probability of occurrence of the relevant aircraft failure conditions [9]. A simplification of this process is shown in Fig. 1.

Markov Analysis is another approved method by ARP 4761A that can be applied as part of a safety assessment to determine what failures cause each identified failure condition and can be used as an alternative to FTA [6]. Markov Analysis is useful to model and analyze systems that may experience multiple states and transitions, especially when system

components may repair and return to service. In the scenario where aircraft systems are relying more on FDIR designs the MA application seems more representative of reality than FTAs [12]. Despite that, industry rarely employs Markov models because they are considered too complex to be created and handled [11]. This complexity increases with the number of elements in the system under study and their states [6]. A system has potentially s^e Markov states, where e is the number of elements and s the number of states of those elements, which shows that there is an exponential growth in the Markov state space while in other analyses, such as FTAs, that growth is linear.

In this paper the focus is on the application of MA in a Model-Based framework. This lowers the entry barrier to this analysis for the safety engineer, since the complexity of the mathematical methods is executed in the background, while the safety engineer manipulates the data in a simple way, using models with visual representations (tables or diagrams) as explained in the following sections.

This Markov model-based approach, however, is not covered by ARP 4761A although its latest revision added Model-Based Safety Assessment (MBSA) as an acceptable alternative to FTA. ARP 4761A only covers the Failure Propagation Model as model-based method, but this perception is considered incomplete from this article’s authors perspective, since other safety analysis methods (FHA, FMEA, MA, among others) could be implemented in a model-based framework [9] and be perceived as MBSA.

It is important to note that, although this introductory section emphasizes the system safety process in the aeronautical domain (as outlined in ARP-4754B and ARP-4761A); the concepts, methodology, and tools that are described are applicable to other industrial sectors dealing with safety-critical systems. With minor adaptations, the overall concepts presented in this paper could be extrapolated to automotive (Functional Safety, ISO-26262 [13]), railway (CENELEC, EN 50129 [14]), robotics (Functional Safety, IEC-61508 [15]), and space (ECSS standards, ECSS-Q-ST-40C [16]) applications, among others.

III. METHODOLOGY AND TOOLS

A. Model-Based Systems Engineering Framework

Capella is an open-source Model-Based Systems Engineering (MBSE) framework. Its main asset, and main differentiator with respect to other tools available based on the Systems Modelling Language (SysML), is that it provides a systems engineering methodology, named Arcadia, directly implemented in the tool [17]. This methodology helps engineers with basic concepts, such as *System*, *Capability*, *Function*, *Mode*, *Component*, among others, which are made available together with a set of relationships and visual representations that enable much easier standardization across teams and a steeper learning curve compared to other tools.

Being open-source, Capella could be extended to cover domain-specific use-cases, like in our case, System Safety analysis based on Markov.

In our work, we take advantage of Capella and Arcadia concepts such as *System*, *Function*, *Component* and *Mode/State*

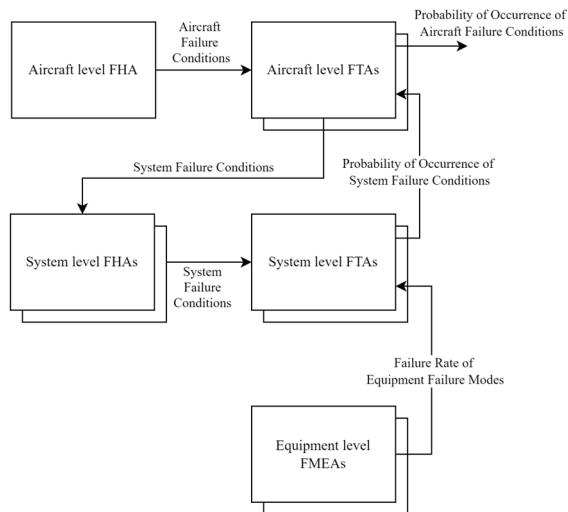


Fig. 1. Safety Analysis Methods used in a simplified traditional aeronautical Safety Assessment

to describe the operation and architecture of a system. With the custom extension developed by us, we allow the user to introduce specific inputs relevant for safety analysis and use the MBSE framework as centralized source of information for design and verification tasks. A visual representation of this framework can be found in Fig. 2.

B. Systems Safety Metamodel

In order to implement a smooth and seamless transition between the Capella models and the Markov Analysis tool, the proposed approach makes use of the State Machine artifact which is natively available in Capella. The metamodel representation of the State Machine and its related model artifacts is shown in Fig. 3. The most important elements are the following:

- ‘State Machines’, highlighted in blue, are *owned* by ‘Components’ (through the artifact named ‘Block’);
- ‘State Machines’ *owns* ‘States’ (through an intermediate artifact named ‘Region’), highlighted in yellow;
- The ‘States’ *implement* ‘State Transitions’ (again through the ‘Region’ artifact), highlighted in pink, which define conditions for moving from one ‘State’ to another. The ‘State Transitions’ can implement additional features such as *Guard* (previous condition to enable the transition), *Trigger* (action that initiates the transition) and *Effect* (consequence of the transition); and
- Additionally, the ‘States’ can be linked to ‘Functions’, highlighted in mauve, to define complex system behaviors.

The Markov Analysis tool uses the ‘State Machine’ artifact as root element for the model transformation required to translate the visual representation into the structured (*formal*) input required to run the mathematical analysis.

This transformation occurs directly in Capella, making use of the Markov Analysis addon integrated into ATICA4CAPELLA Model Based Safety Analysis suite, which is a Capella addon that extends the Model Based Systems Engineering workbench providing additional modeling artifacts to capture specific safety related features and enable advanced analysis directly from the models. ATICA4CAPELLA is part of ATICA [18], a safety engineering framework offering a collection of systems safety and dependability analysis tools

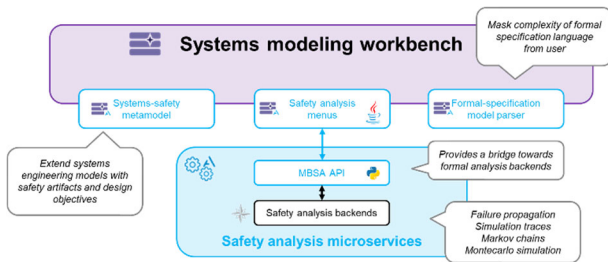


Fig. 2. MBSE framework and custom extensions

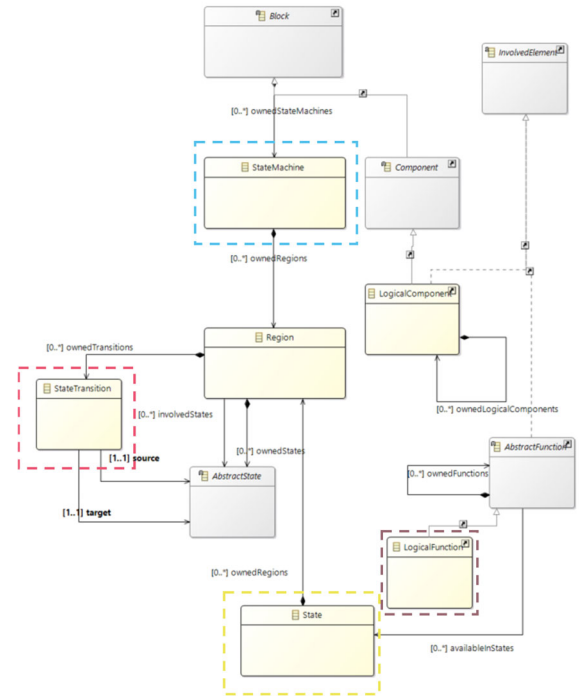


Fig. 3. State Machine metamodel and related model artifacts

developed by Anzen Engineering or derived from open source projects and research studies [19].

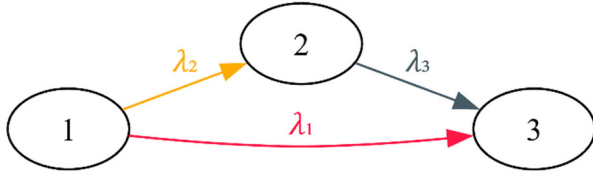
C. Markov Analysis Backend

ATICA4CAPELLA employs *markovsolver4safety* [20] as its backend for conducting Markov Analysis. This open-source Python package, developed by Anzen Engineering, accepts structured inputs, including states information and system or component models representation. It then calculates the probability of being in each of the states by a given time, based on specified initial conditions.

Specifically, the states’ information is introduced to the script using a structured representation (dictionary) with the following elements:

- Markov states’ names;
- For each Markov state, the combination of elements (constituent components) states that makes up the corresponding Markov state (e.g., C1 = ‘operating’, C2 = ‘failed’, C3 = ‘standby’, ...); and
- Optionally, the consequence of each Markov state (e.g., ‘nominal operation’, ‘total loss’, ‘operating without redundancy’, ...).

The model is completed with a second input, a nested dictionary, containing the information of the elements in the system or component, their states and their transitions to other states, along with the parameters of those transitions. Fig. 4 shows an example of the Markov Chain that can be obtained with *markovsolver4safety* (top), and the inputs required by it to model the case study (bottom). The variable “data” represents



```
data = {
  "State": ["1", "2", "3"],
  "Machine": ["Operating", "Degraded", "Failed"]
}

model_representation = {
  "Machine": {
    "Operating": {
      "Failed": {'value': 0.01, 'color': '#FF1744', 'symbol': '\\lambda_1'},
      "Degraded": {'value': 0.05, 'color': '#FFAB00', 'symbol': '\\lambda_2'},
      "Operating": {}
    },
    "Degraded": {
      "Failed": {'value': 0.07, 'color': '#455A64', 'symbol': '\\lambda_3'},
      "Degraded": {}
    },
    "Failed": {
      "Failed": {}
    }
  }
}

mc = MarkovChain(data, model_representation)
```

Fig. 4. Example of a Markov Chain (top) and the input needed to model it in markovsolver4safety (bottom)

the Markov states' information, while "model_representation" contains all the information of the system to perform the analysis.

These two inputs are combined to get automatically the state transition matrix, M , that conforms the ordinary differential equation (ODE) system (1), where $P_1(t)$, $P_2(t)$, ..., $P_n(t)$ represent the probability of be in the Markov state $1, 2, \dots, n$ at a time t .

$$\frac{d}{dt} \begin{bmatrix} P_1(t) \\ P_2(t) \\ \vdots \\ P_n(t) \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \dots & m_{nn} \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ \vdots \\ P_n(t) \end{bmatrix} \quad (1)$$

Then, the ODE system (1) is solved for a given time under specific initial conditions. By default, it is considered that $P_1(0)$ is equal to one (fully operative Markov state) and the rest are equal to zero, but other initial conditions may be specified. If consequences have been initially described in the input data, the results may be directly presented grouped by the specified consequences.

IV. USE CASE EXAMPLE

A. Systems model and inputs to Safety Analysis

The tool and methodology have been tested on an example system model. As shown in Fig. 5, this model is composed of

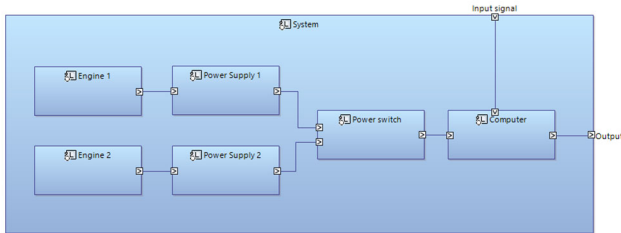


Fig. 5. Example system model

two redundant engines and power supplies that feed a computer, which computes some sort of data (input signal) and gives an output (output signal) to other system.

Each element has its own operating states:

- Engine: operating and failed.
- Power Supply: operating and failed.
- Power Switch: always operating.
- Computer: Operating, failed and erroneous

Also, they have allocated a rate of transitioning from one state to another (when possible). Both operating states and their transitions are defined in the state machines. Fig. 6 shows the state machine applicable to both the engine and power supply elements, while Fig. 7 shows the state machine of the computer element.

A few considerations of the modeling are stated below.

- It is assumed that the failure of the power switch is negligible, and therefore, it is always operating.
- FDIR design is considered for the computer, so it is possible to transit from the *erroneous* state back to the *operating* state, as shown in Fig. 7.

For the sake of simplicity, the transition from operating to failed in all elements has the same failure rate allocated.

From the system perspective, three operational modes have been defined, depending on the space state of its components: *Operating*, *Total loss of function* and *Malfunction*. Table I shows a partial representation of the system states table, i.e., combination of operating states (automatically generated by the

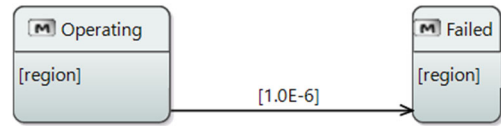


Fig. 6. Example state machine diagram of the engines and power supplies components

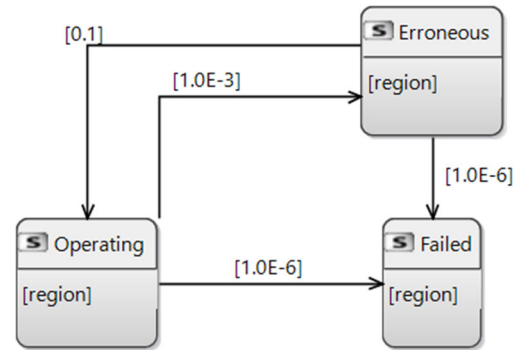


Fig. 7. Example state machine diagram of the computer component

TABLE I. SYSTEM STATES TABLE

State ^a	Operational Mode	Engine 1	Engine 2	Power Supply 1	Power Supply 2	Computer	Power Switch
State_000000	Operating	0	0	0	0	0	0
State_000010	Total loss of function	0	0	0	0	1	0
State_000020	Malfunction	0	0	0	0	2	0
...	...	...	...	...	...	...	...
State_111110	Total loss of function	1	1	1	1	1	0
State_111120	Total loss of function	1	1	1	1	2	0

^a. States could be 0 (operative), 1 (failed) or 2 (erroneous)

tool) and their relationship with the operational modes. In total there are 48 combinations.

The elements' state machine and the system states table are the input to perform the Markov model-based analysis.

B. Objective of the Safety Analysis

Considering that the total loss of function of this system has a safety goal of having a maximum probability of occurrence per flight hour equal to 10^{-4} , which comes from a higher-level safety analysis, and that the mission time is equal to 1000 flight hours, the objective of this Markov model-based analysis is to compute the probability of finding the system in the operational mode after that time.

C. Results and Discussion

After running the Markov analysis with the model presented above and considering the mission duration, the results are presented in Table II.

The results of Table II show the probability of being in each of the operational modes after the mission time, however, to compare them against the safety goal, it is needed to divide that value by the mission time. For the operational mode of interest, *total loss of function*, it is obtained a probability of occurrence per flight hour equal to 10^{-6} , showing compliance against the safety goal.

V. CONCLUSIONS AND FUTURE WORK

A. Summary and conclusions

This paper explores the use of models as enablers for advanced safety verification analysis, particularly Markov analysis. Traditionally, Markov Analysis has been disregarded by most of the safety engineers in the aeronautical industry due to its complexity, even losing ground to other analyses, such as FTAs, although MA application was more appropriate in the field of complex cyber physical systems with reconfiguration capabilities. *ATICA4CAPELLA* and *markovsolver4safety* are

pieces of open-source software developed by Anzen Engineering team, to use in conjunction with Capella, but can be extrapolated to other Model-based tools, with the aim to reduce the barrier to entry into Markov Analysis for Safety Engineers. This approach intends to encourage them to perform the proper assessment in systems that require a dynamic analysis, such as FDIR systems, instead of using a more classic static analysis like FTAs.

Authors interpretation of the MBSA concept is stated, completing the approach presented in ARP 4761A: MBSA is not just one method (FPM), but a set of methods in which FPM is included.

Markov Analysis is one of those considered traditional analysis that can be included in that set, and it has been proven that performing it in a model-based framework facilitates the tasks of the safety engineer when dealing with complexity, opening a new path: using state machines as inputs and masking all the calculation underneath the model-based tool. This way the barrier entry is reduced, since modelling a state machine is relatively easy through a table or a simple diagram.

This approach allows the safety engineer to model the system under study element by element, avoiding the exponential growth of the complexity of bigger systems. In addition, Markov Chain diagrams are not so necessary when there are machine states and system diagrams, and all the transitions between the Markov states are automatically obtained.

B. Future work

Despite being able to obtain a functional Markov model-based approach there is room for improvement in the tool.

From ATICA4CAPELLA side, currently it is not possible to model simultaneous failures (e.g., directly transit from a Markov state where the two components of a system are operational to other Markov state where those two components are failed) although the solver has that capability. Also, when generating automatically the Markov states from the system elements' state machines there are some states that could be incompatible (e.g., a standby component being operational while the main component is still operational). For those cases, the user has to manually delete the incompatible combinations, which can be improved by allowing the user to model those restrictions to avoid that extra manual work. Another extra manual work for the user, once all the Markov states are generated, is writing the operational modes corresponding to each Markov state (like "Operating", "Total Loss", etc.) one by one. It is planned to include a query where the user can easily enter the desired

TABLE II. PROBABILITY OF OCCURRENCE OF SYSTEM OPERATIONAL MODES

Operational Mode	States	Probability of Occurrence at 1000 hours
Operating	[State_000000, ..., State_101000]	0.989
Total loss of function	[State_000010, ..., State_100120]	0.001
Malfunction	[State_000020, ..., State_101020]	0.01

natural language, e.g., “each state with one computer failed is considered as ‘Total Loss’ despite the rest of the elements” and automatize this part of the process. This is especially interesting for complex systems where the resulting Markov State table can be huge. In case the Capella user wants to obtain the Markov Chains diagrams, ATICA4CAPELLA can retrieve the graph information from the solver backend and draw the Markov Chains within the model. However, this feature is still not implemented.

On the backend solver side, it is planned a rework of the input format to enable a direct import of sysml files that represents the model under study, creating a smoother bridge between the MBSE tool and the solver. Currently, the only metric that the solver returns is the probability of being in a specific state (or failure condition), which is the Cumulative Distribution Function (CDF). In the future more metrics are planned to be output, such as the Probability Density Function (PDF), the Hazard Rate and the Cumulative Hazard Rate, as outlined in ARP 4761A. Last, current implementation only supports Homogeneous Markov Models (i.e., transition rates are modelled with exponential distributions), but it is planned to include the capability to model also Non-Homogeneous Markov Models and use time-dependent transition rates (modelled with Weibull distributions).

All these future lines of work will improve considerably the tool, enhancing even more the user experience.

C. Long term vision

This Markov approach, regardless of being implemented in a MBSE framework, does not imply the entire implementation of a MBSE philosophy by the safety engineer to execute it. In this sense, the proposed framework offers both the possibility of extending existing system models (architectural models, product and functional breakdown...) with a safety analysis layer; or creating directly the safety models specifically for simplifying the Markov Analysis process.

Together with other safety analysis methods currently being implemented in the ATICA suite, the Markov Analysis solver is especially well suited to the verification of digital and software-defined systems implementing complex behavioral logics. The Model Based representation facilitates usage, masks complexity of mathematical methods and enables easier traceability along design and development phases where system (design) models and safety analysis evolve in parallel.

ACKNOWLEDGMENT

This work has been funded with support from the Spanish Technological Development and Innovation Centre (CDTI) through the Technical Aerospace Program 2023-2025 (PTA).

REFERENCES

- [1] N. Kabbara, M. O. Nait Belaid, M. Gibescu, L. R. Camargo, J. Cantenot, T. Coste, V. Audebert, and H. Morais, "Towards Software-Defined Protection, Automation, and Control in Power Systems: Concepts, State of the Art, and Future Challenges," *Energies*, vol. 15, no. 24, Art. no. 9362, 2022, doi: 10.3390/en15249362.
- [2] P. Galkin, R. Umiaarov, O. Grigorieva and D. Ageyev, "Approaches for Safety-Critical Embedded Systems and Telecommunication Systems Design for Avionics Based on FPGA," 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 2019, pp. 391-396, doi: 10.1109/PICST47496.2019.9061421.
- [3] A. Darabseh and N. M. Freris, "A Software Defined architecture for Cyberphysical Systems," 2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, Spain, 2017, pp. 54-60, doi: 10.1109/SDS.2017.7939141.
- [4] B. Bittner, M. Bozzano, A. Cimatti, R. De Ferluc, M. Gario, A. Guiotto, and Y. Yushtein, "An Integrated Process for FDIR Design in Aerospace," in *Model-Based Safety and Assessment*, F. Ortmeier and A. Rauzy, Eds. Cham: Springer International Publishing, 2014, pp. 82-95.
- [5] A. Zolghadri, "Advanced model-based FDIR techniques for aerospace systems: Today challenges and opportunities," *Progress in Aerospace Sciences*, vol. 53, pp. 18-29, 2012, doi: 10.1016/j.paerosci.2012.02.004.
- [6] Guidelines and Methods for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment, SAE ARP 4761A, 2023.
- [7] B. Gallina and L. Fabre, "Benefits of security-informed safety-oriented process line engineering," *IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, Prague, Czech Republic, 2015, pp. 1-24, doi: 10.1109/DASC.2015.7311637.
- [8] Guidelines for Development of Civil Aircraft and Systems, SAE ARP 4754B, 2023.
- [9] S. G. Lorente, D. A. Alonso and T. K. Vetter, "Safety 4.0: Model-Based Approach Toward Perpetual Flight with an Autonomous, Solar-Powered Aircraft," 7th International Conference on System Reliability and Safety (ICSRs), Bologna, Italy, 2023, pp. 195-205, doi: 10.1109/ICSRs59833.2023.10381297.
- [10] N. G. Leveson, C. H. Fleming, J. Thomas, and C. Wilkinson, "A Comparison of SAE ARP 4761 and STPA Safety Assessment Processes", [Online]. Available: <http://sunnyday.mit.edu/papers/ARP4761-Comparison-Report-final-1.pdf>. [Accessed: August 20, 2024].
- [11] A. Gomes, A. Mota, A. Sampaio, F. Ferri, and E. Watanabe, "Constructive model-based analysis for safety assessment," *Int. J. Softw. Tools Technol. Transfer*, vol. 14, no. 6, pp. 673–702, Dec. 2012, doi: 10.1007/s10009-012-0238-x.
- [12] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," *Reliability Engineering & System Safety*, vol. 71, no. 3, pp. 249-260, 2001, doi: 10.1016/S0951-8320(00)00077-6.
- [13] ISO 26262, "Road vehicles – Functional safety," ISO Standard, 2018.
- [14] EN 50129, "Railway applications - Communication, signaling and processing systems - Safety related electronic systems for signaling," European Committee for Electrotechnical Standardization, 2018.
- [15] IEC 61508, "Functional safety of electrical/electronic/programmable electronic safety-related systems," International Electrotechnical Commission, 2010.
- [16] ECSS-Q-ST-40C, "Space product assurance – Safety," Rev. 1, European Cooperation for Space Standardization, Feb. 2009.
- [17] P. Roques, "MBSE with the ARCADIA Method and the Capella Tool," in *Proc. 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, Jan. 2016.
- [18] Anzen Engineering, ATICA Suite. [Online]. Available: <https://www.anzenengineering.com/digital-tools/>. [Accessed: September 13, 2024].
- [19] M. Bozzano, H. Brujntjes, A. Cimatti, J.-P. Katoen, T. Noll, and S. Tonetta, "COMPASS 3.0," in *Tools and Algorithms for the Construction and Analysis of Systems: 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I*, Springer-Verlag, Berlin, Heidelberg, 2019, pp. 379–385, doi: 10.1007/978-3-030-17462-0_25.
- [20] S. G. Lorente, *Markov Solver for Safety*, V0.1.0. [Online]. Available: <https://github.com/samuelglorente/markovsolver4safety>. [Accessed: September 10, 2024].