2020 AIAA Scitech Forum, Orlando, 6-10 January 2020
DOI:10.2514/6.2020-0550

Enabling Interactive Safety and Performance Trade-offs in Early Airframe Systems Design

Sergio Jimeno[1], Atif Riaz[2], Marin D. Guenov[3], and Arturo Molina-Cristobal[4]
Cranfield University, Cranfield, Bedfordshire, MK43 0AP, United Kingdom

Presented is a novel interactive framework for incorporating both safety and performance analyses in early systems architecture design, thus allowing the study of possible trade-offs. Traditionally, a systems architecture is first defined by the architects and then passed to experts, who manually create artefacts such as Fault Tree Analysis (FTA) for safety assessment, or computational workflows, for performance assessment. The downside of this manual approach is that if the architect modifies the systems architecture, most of the process needs to be repeated, which is tedious and time consuming. This limits the exploration of the design space, with the associated risk of missing better architectures. To overcome this limitation, the proposed framework automates parts of the safety and performance analysis in the context of the Requirement, Functional, Logical, and Physical (RFLP) systems engineering paradigm. Safety analysis is carried out by automatic creation of FTA models from the functional and logical flow views. Regarding performance analysis, computational workflows are first automatically created from the logical flow view, and then executed for a set of flight conditions over the range of the mission in order to determine the most demanding condition. Finally, performance characteristics of the subsystems, such as weights, power offtakes, ram drag etc. are evaluated at the most demanding flight condition, which enables the architect to compare architectures at aircraft level. The framework is illustrated with a representative example involving the design of an environmental control system of a civil aircraft, where the safety and performance trade-off is conducted for multiple ECS architectures.

I. Introduction

System safety is of paramount importance in all industries and particularly in aerospace. Specifically, in relation to (civil) aircraft, there are many failure conditions which may lead to a potentially fatal accident, involving multiple casualties, damage to the aircraft, the surrounding area of the accident and which incur substantial monetary losses. Certification documents, such as the Certification Specifications CS-25 [1], establish safety objectives on the different parts of the aircraft in order to keep the probability of catastrophic accidents to a minimum. Safety can be improved in different ways, e.g., by employing more reliable components or implementing different redundancy strategies. In order to select the best architecture, both performance and safety considerations need to be accounted for. Currently, safety analyses are performed manually, based on informal models and various documents, such as certification documents, which introduces unnecessary subjectivity and lack of consistency to the analyses [2]. A similar situation is encountered when the architecture is sized, and its performance obtained. The process of orchestrating sizing models from the architecture definition is also largely manual and therefore time consuming [3]. The prevalence of manual processes usually requires extensive communication between the architect and other teams of experts. This limits the time for exploration of the design space, with the associated risk of missing better architectures. Thus, computational methods that allow the designers to quickly define and assess architectures are necessary in order to enable the

[1] Research Student, Centre for Aeronautics.
[2] Lecturer, Centre for Aeronautics, AIAA Member.
[3] Professor, Centre for Aeronautics, AIAA Senior Member.
[4] Lecturer, Centre for Aeronautics.

exploration of more alternatives. Furthermore, these methods need to be interactive to keep the architects in charge of the most creative part of the design process, responding to their request and feeding back the relevant results. In turn, automating most repetitive design tasks not only speeds up the design process, but enables the required interactivity.

Recent work by Jimeno et al. [4] discussed the limitations concerning existing methods for automating the safety assessment of architectures and proposed a novel safety framework, focused on automation and interactivity. The framework enables the creation of safety requirements through functional hazard analysis. It also facilitates the assessment of safety by means of an algorithm to automatically create fault trees from the architecture definition. An interactive method for adding redundancy to the architecture is also presented.

Regarding the problem of sizing aircraft systems architectures, several approaches to automate the process have been proposed in recent years. Liscouët-Hanke at al. [5], [6], proposed a methodology where the architecture is composed of power system modules which can be run in sizing mode, to obtain their sizing characteristics; and performance mode, to obtain off-design energy flows, used to calculate aircraft level performance. The interfaces of this power modules are generic enough to allow orchestration of subsystems, which is made manually in terms of power consumers, distributors and generators. Each subsystem is designed following a functional approach to allow different technology choices. The energy flows, and therefore sizes of the system, are modelled as function of time (including different phases) and operation mode (normal, degraded or failure mode). De Tenorio et al. [7], [8] also employ a functional approach in order to select subsystem solutions. The subsystems then are sized in order to satisfy requirements throughout the mission, which consist of three dimensions: flight phase, flight conditions and failed architecture configurations. The execution of subsystem models is scheduled in terms of the functional flows between components. The subsystem sizing problem is posed as an optimization problem where the objective function equals to the weighted of several subsystem attributes, allowing for trade-offs between different attributes. This subsystem level performance is related to aircraft level optimization through the values of the weighting factors, which can be changed by the aircraft level optimizer in order to maximize the aircraft utility. A functional approach to subsystem architecting is employed by Chakraborty et al. [9], leveraging the choice of alternative solutions for each aircraft subsystem. The approach for subsystem sizing consists of steady state equations models representing subsystems and empirical relations. When these equations are solved, the most relevant sizing parameter of the particular subsystem architectures are obtained. The models are orchestrated by linking power consuming subsystems to their respective distribution elements, which in turn are connected to the power generation and distribution devices. In order to obtain the subsystem sizing point, only one or few points of the mission are considered for each subsystem. These points are selected a priori, generally based on previous works. This sizing methodology also considers basic aircraft level resizing rules, based on the subsystems sizing results. The aircraft's new sizes further influence the subsystems results, leading to an iterative process. Judt and Lawson's approach [10], [11], utilize a methodology that generates the full enumeration of architectures given a function tree. This function tree consists on the aircraft functions (boundary functions), for which one or more candidate solutions are proposed, which in turn induce more functions which can be satisfied by different solutions, inducing more functions and so forth. The approach employed for determining the execution subsystem order depends on the interfaces between subsystems which are determined manually and reviewed continuously until a sufficient level of confidence is obtained. The sizing strategy for the subsystems consist of a representative mission plus specific mission failure scenarios for the components that are active in such situations. Bile et al. [3], [12] proposed a framework for automated sizing of airframe systems from the logical view of the architecture, consisting of architecture components and their connections. The proposed framework also accounts for functions as a central element in the architecting process. With this approach, different architectures can be proposed by varying the components and their connections. Components have associated computational models which are orchestrated based on their input-output relations at component level, and source-sink relationships at subsystem level. This approach only considers one mission point for sizing. Several conclusions can be drawn by comparing the aforementioned methods:

- The approaches employed to orchestrating sizing models allow for the automatic sizing of the architecture once its definition process, which is not automated in any case, has been completed. Regarding the level of detail of the architecture represented by the orchestrated models, it is found that in general they represent the subsystems of the architecture. This allows substituting one subsystem for an alternative one as long as a model for each one of them is available. However, it presents the limitation of not being able to process the change of a single component, without manually editing the subsystem model. One exception is the framework proposed by Bile at al. [3], which allows a finer degree of granularity since the models in it are associated to components.
- Concerning the amount of information (one point, several points, or a mission) that is used to find the sizing point of the system, the most restrictive approach is that of Bile et al. as it considers only a single point. The work of Chakraborty at al. [9] has the drawback that extensive previous knowledge is required

to select a priori the sizing points. The rest of the approaches provide a higher level of detail, considering different flight phases, flight conditions and also failed architecture configurations in the case of Tenorio et al. [7], [8]. Still, the information regarding flight conditions is limited to a small number of predefined scenarios and no guidance to choose the failure conditions is provided in any of the approaches.

- It can also be concluded that functions are a key part of the architecting process, especially when selecting alternative architectures.
- The comparison of the above approaches has shown that the potential trade-offs that appear when sizing subsystems have been neglected and only a single solution to the sizing problem is considered, except in the case of de Tenorio et al. [8], who enable trade-offs thanks to their proposed subsystem sizing problem formulation.

With the purpose of overcoming the above mentioned limitations, the work presented in this paper aims to integrate subsystems sizing and performance with the safety framework described in Jimeno et al. [4]. The framework allows functional reasoning as an enabler of the architecting process, which the literature review has been shown to be important, and also provides the necessary safety features. Jimeno et al. proposed a novel sizing process, considering safety, mission and flight conditions variations in the flight envelope in a continuous manner, which goes beyond the most detailed approaches found in the literature. A high level of detail is considered, allowing to perform changes in the architecture at a component level while maintaining a high degree of automation. After the sizing is complete, the aircraft-level performance can be obtained, including the weight, power offtakes, and additional drag of its subsystems. This last step enables the trade-off between different architecture configurations in terms of both safety and performance within in a single framework.

Within this context, the objective of the presented here work has been the development of a novel interactive framework for incorporating both safety and performance analyses in early systems architecture design, thus allowing the study of possible trade-offs. The scope is limited to early design, under the assumption that the requirements, functions, and logical views of the (systems) architecture are defined, including relations between elements of different domains, such as which components satisfy which functions. The aircraft mission, flight envelope and atmospheric variations that the aircraft is intended to handle successfully are also considered as known. Additionally, computational models for sizing and performance, describing the different logical components of the architecture, are assumed to be available, e.g. provided by the pertinent experts. The physical view, detailing aircraft geometry and subsystems spatial layout, is not considered at this level of detail and analysis requiring geometric or layout information, such as thermal analysis of the aircraft, are therefore out of scope.

The rest of the paper is organized as follows. Section II provides background information and terminology used in the manuscript. The proposed framework is described in Section III. Section IV demonstrates the application of the proposed approach with the help of a representative test case. Finally, summary, conclusions and plans for future work are presented in Section V.

II. Background

In this section, information related to the terminology used in the proposed framework for safety and performance trade-off (Section III) is provided.

A. RFLP (Requirements, Functional, Logical, Physical) Paradigm

RFLP assumes that functional reasoning as part of the systems architecting process is distributed over four notional domains: Requirements, Functional, Logical, and Physical [15]. It is based on the German guideline VDI 2206, “Design methodology for mechatronic systems” [16]. The requirements domain displays all the architecture requirements, which are extracted from the stakeholders’ needs. Requirements can be of functional or performance type, and it is possible to decompose them hierarchically. Functional requirements are mapped to the functions of the system in the functional domain. The functional domain contains all the architecture functions, that is, actions that the system must perform to meet the stakeholders' needs. Functions are linked to the components that satisfy them, and vice-versa. The logical domain consists of components, solutions satisfying the functions, and their interconnections via ports.

In this paper, the RFLP paradigm is augmented with two domains. “Safety domain” for enabling safety assessment of the architecture, as proposed by Jimeno et al. [4], and “computational domain” for providing the capability of automated systems sizing, as proposed by Bile et al. [3]. Additionally, traceability between different view of the architecture is incorporated, as suggested by Guenov et al. [15], enabling a more effective and interactive design process.

B. Safety Domain

The safety domain allows the creation of safety artifacts such as functional hazard analysis (FHA) or fault tree analysis (FTA) from the RFLP definition of the architecture, maintaining links with the elements existing in the requirements, functional and logical domains of the architecture.

Fault Tree Analysis (FTA): FTA is a deductive failure analysis that focuses on one undesired event, the top event, and determines its causes in terms of basic events and their relations expressed through logical gates. According to Ruijters and Stoelinga [13], two kinds of fault trees analysis techniques exists, qualitative and quantitative. Quantitative techniques provide numerical values for failure probabilities or importance measures, which can be used to rank components according to the contribution to system failure. Qualitative techniques provide insight into the structure of the tree and facilitate the detection of vulnerabilities. Some of the most popular qualitative techniques are cut sets and path sets, which are described below.

Minimal Cut Sets: A cut set is a combination of basic events in the fault tree that cause system failure, i.e., the top event in the tree occurs. A cut set is said to be minimal when no subset is a cut set [13], thus it is a minimal set of components that cause system failure. Minimal cut sets can be used to determine the qualitative importance of the basic events and potential common-causes to system failure [14]. Additionally, if the probability of failure of the basic events are known, cut sets can be used to obtain the probability of occurrence of the top event.

Minimal Path Sets: A minimal path set is the opposite of a minimal cut set, that is, a minimal set of basic events such that if any of them occurs, the system remains operational [13]. In other words, the top event of the tree does not occur. Similar to minimal cut sets, minimal path sets provide qualitative information about the reliability of the system and can be used to obtain the probability of failure of the top event.

C. Computational Domain

The computational domain is a notional domain introduced in [3] with its primary purpose to automatically orchestrate the computational models (e.g. steady state equations) associated with each component in the logical view into sizing computational workflows. Indirectly this domain is used to manage the dependencies between parameters, i.e., the numerical values that describe some (behavioral) characteristics of the components, and which can be linked to performance requirements.

Computational Model: A computational model is an executable piece of computer code that describes part of the physical behavior and other relevant characteristics (e.g. weight or cost) of an architectural logical component. Every logical component has one or more such models associated with it. Models have inputs and outputs. Execution of the model updates the values of the outputs according to the values of the inputs.

Computational Workflow: A computational workflow is an ordered set of computational models. The order of models is such that every model only depends on outputs of the models appearing earlier in the order. The workflow can be abstracted as a computational model representing the set of models. Since the subsystems of the architectures are composed of components, and these are represented by computational models, the workflow containing all these models describes the behavior of the subsystem.

III. Proposed Approach

A novel approach is proposed to integrate subsystems sizing and performance with the safety extension to the RFLP paradigm. As shown in Fig. 1, both safety and performance are regarded in the framework as two-way processes, which interact with each other not directly but through the modification of the architecture definition. The architecture definition is used as an input to safety analyses such as FTA. In turn, the safety results are used to update the architecture, for example, by adding new requirements or redundant components to ensure safety compliance. Architectural changes impact the system performance, which might demand even further modification, which in turn affect safety and so forth. The proposed approach enables the assessment of the impact caused by modifications of the architecture definition in the sizing and performance of the subsystems.

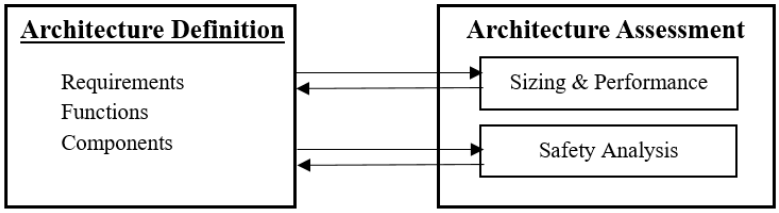

Fig. 1 Architecting process including safety and performance

Fig. 2 shows an overview of the existing RFLP safety framework [4] (in light blue) and the necessary additions and modifications to cater for the proposed methodology. A Mission & Flight Conditions view of the architecture is added; the Computational View is modified to include the information provided related to mission. The Mission & Flight Conditions view contains the necessary information related to the intended mission of the aircraft such as the different flight phases, their characteristics and information about the ambient conditions that can be encountered during these phases.

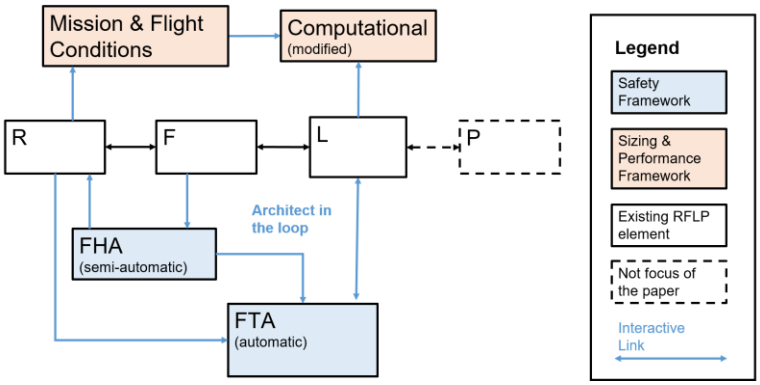

Fig. 2 RFLP Framework augmented with Safety and Computational Domains

The intended use of the framework is described through a series of steps, represented by means of a flowchart in Fig. 3. The process starts with the architecture definition, where the system is described by the requirements, functional and logical views. Next, the safety and performance analyses are performed, and finally a trade-off is conducted with respect to the performance requirements and figures of merit. The rest of this section describes the individual steps in detail.

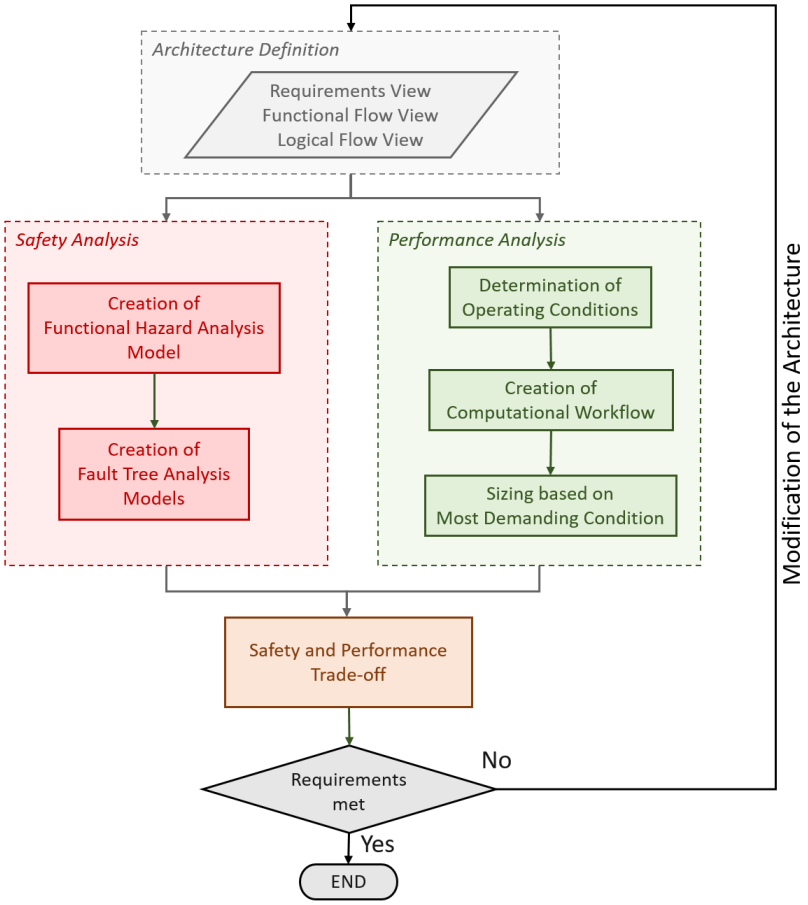

Fig. 3 Flowchart of the Proposed Framework for Safety and Performance Trade-off

A. Safety Analysis

The safety analysis is comprised of two methods. The first method supports partial automation of the functional hazard analysis process, using the functional view as input. The FHA results are then used to update the safety objectives in the requirements view. The second method automatically generates Fault trees from the Logical view. These trees can be evaluated both qualitatively and quantitatively and the contribution to the probability of failure of their components can be ranked through importance measures, helping the architect to focus on the most problematic parts of the architecture.

Creation of Functional Hazard Analysis Model: The first step in the safety analysis is to support the creation of functional hazard analysis (FHA) models. First, a list of functions of the system is obtained to analyze the hazards related to them. This list is compiled automatically by traversing part of the functional view of the architecture in its hierarchical form. In this form, functions are organized using parent-child relationships (with at most one parent), resulting in a data structure known as tree. The scope of the FHA is determined by the selected components to analyze, e.g. the components forming the subsystem under study. The functions that map to the selected components, plus all their children are included in the FHA. More details of the method are presented in an earlier publication from the authors [4].

Creation of Fault Tree Analysis Model: The second step in the safety analysis involves creation of fault tree analysis (FTA) models from the logical flow view. First the algorithm gathers the necessary information to construct the gate that describe a failure in the output. This is assumed to happen when the component that provides the output of any of its inputs fails, hence the type is set to OR. Before constructing the gate, the subtrees rooted on the gates inputs are determined. Finally, these subtrees and the basic event of component fail are put together with the OR gate. The tree connections are created in a similar way as described above, however, instead of receiving a model output receives a model input. This input is used to get the connection to be modelled by the gate. The type of the gate is determined by redundancy type specified in the connection, e.g. AND when any output can be used, k/N for voting systems where k correct inputs out of N are needed, and SPARE gates for modelling dynamic redundancy. Then, the outputs reached are obtained by get-outputs, the outputs will be filtered according to whether they belong to the set of parents and the logic determined by the type of gate. Before constructing the gate, the subtrees rooted on the gates inputs are determined for each one of them. Finally, these subtrees describing the failure of the component inputs are put together with the gate. More details can be found in an earlier publication from the authors [4].

B. Performance Analysis

The performance analysis is comprised of three steps. First the different configurations of the architectures are identified. Next, the computational workflows representing the subsystems in each of their configurations are created. Finally, the systems are sized by taking the most demanding flight condition.

Determination of configurations: The study of the different configurations that a system can adopt is fundamental for the sizing of the architecture subsystems. This is due to the fact that some components are only active in certain configurations, so ignoring them will provide insufficient information for their sizing. Additionally, some configurations demand more from some components than others, specially failed configurations where the surviving component need to compensate the loss of the failed component. Considering these situations is fundamental for sizing. For every configuration the active components and their connections are used to automatically assemble a computational workflow.

The determination of different design configurations, according to different flight phases, depends exclusively on the architect's knowledge about the system being designed. However, FTA results can provide guidance to the architect to determine degraded scenarios. The minimal path sets obtained from FTA indicate the smallest number of components that, if working properly, enables the system to perform its intended functions. Therefore, these combinations are good candidates to determine degraded scenarios. If the architect wishes to consider additional scenarios, minimal cut sets can also provide valuable information. Due to the nature of minimal cut sets if any of the elements of the sets goes back to its operational state the whole system becomes operational. Following this strategy, every set can provide as many degraded conditions as the number of elements it contains.

Depending on the architecture definition and number of redundant components, the number of different degraded scenarios that may be considered can grow quickly and soon become intractable. Good engineering judgment from the architect is necessary to select only the most significant combinations, which is not necessarily an easy task

considering only a priori information. Enabling interactive links between the FTA results and the logical components of the architecture is proposed as a method to visually aid the designer with this task.

Creation of computational workflows: The process of creating a computational workflow for the subsystems of the architecture is based on the approach described in Bile et al. [3]. In this approach workflows are assembled at two levels:

- At subsystem level, individual workflows are created for each one of subsystems based on input-output relations. One computational workflow needs to be created for each subsystem in each of its configurations. This is because every configuration includes different components and interconnections which translate into different workflows of models.
- At system level, those workflows are ordered in a sequence determined by the source-sink relationships between subsystems. A subsystem that demand any kind of flow (e.g. electric power) from another subsystem is executed before the latter.

Sizing based on most demanding condition:

The proposed framework uses a set of flight conditions over the range of mission in order to determine the most demanding condition. Full factorial design of experiment is employed in order to consider a wide range of all possible combinations. The idea is to find the most demanding condition as it will represent the sizing point of the component. This is the logic that is followed for quantities that are sized based on extreme values, such as the peak power a battery needs to be able to provide. For this purpose, the valid region of the ambient variables (altitude, density, pressure, speed, etc.) space, where the sizing point will be searched for, needs to be defined manually by the architect or a relevant expert. A special sizing case is that of the sizing of magnitudes that are consumed during the mission, such as the necessary capacity of a battery or the amount of fuel. For sizing this kind of components, integration over mission profiles of the design mission or missions needs to be considered. The problem consists of finding the maximum necessary size of the component in a region of the ambient variables, e.g. which ambient conditions lead to a maximum power demand for a battery. The extent of this region depends on the flight phase. The maximum demand for a sizing variable is obtained for every flight condition in failure-free and failed conditions. This is valuable information that can not only determine the final size for the subsystem components but help the architect to understand the behavior of the system throughout its intended mission.

In order to compare different architectures at top (aircraft) level, it is necessary to obtain the aircraft performance taking into account the weight, additional drag and power offtakes of the aircraft subsystems. If the tool used to compute aircraft level performance does not allow the detailed consideration of some of drag or secondary power consumptions, , the performance penalties caused by subsystems drag and secondary power extraction can be calculated following the methodology developed by Chakraborty [19]. In brief, it obtains the mission performance with the secondary power outtakes set to zero and no additional subsystem drag considered, and then iterates through every mission step in reversed order (starting at landing, finishing at take-off), adding the contribution of secondary power to the amount of fuel burned.

C. Safety and performance trade-offs

The whole process of safety and performance analysis described above can be repeated for many different architectures. If the results are not satisfactory, e.g. the requirements are not met, new architectures or modifications to the existing ones can be proposed. This will trigger the sizing and performance process for these new architectures, providing a new set of results. The process will be repeated as many times as necessary until the architect is satisfied.

IV. Demonstration

In this section, the approach for aircraft safety and performance trade-off is demonstrated through a use case. The objective of this use case is to highlight and discuss the capabilities and benefits of the presented methodology, not to come up with the best aircraft design. An in-house developed software, named “AirCADia Architect” [20], is used for (1) synthesizing the airframe systems architectures, and (2) analyzing the safety and performance characteristics of the synthesized architectures. Moreover, publically available computational models are employed for the performance estimation of aircraft configurations and subsystems architectures. The rest of the section is divided into two parts: the application use case is described first, then the individual steps of the proposed methodology are applied on the case study.

A. Use Case Description

In order to demonstrate the proposed methodology, a use case involving the design of Environmental Control System (ECS) is considered. A trade-off between safety and performance characteristics is conducted for multiple (conventional and electric) ECS architectures. The schematic diagram of the ECS is shown in Fig. 4. Two options are considered for air supply. In the first case, referred to as conventional (bleed-air) ECS architecture, air supply units take hot air bled from the stages of engine compressors. In the second case, referred to as electric (bleed-less) ECS architecture, air supply units take ram air through the dedicated air inlets, and compress it using compressors powered by electric motor. The air from the supply units is then passed through the ozone converters in order to remove ozone particles. Next, the air conditioning packs are used to cool the air at the required temperature. There are many different architectures proposed for the air conditioning packs, however in the current research work, only the bootstrap air cycle machine is considered. In both conventional (bleed-air) and electric (bleed-less) ECS architectures, ram air from separate air inlets is used as coolant in air conditioning packs. Next, the fresh conditioned air is mixed with some portion of recirculated air (from the cabin) in the mixing manifold, and is passed to the cabin. In the present work, the aircraft is assumed to be comprised of three separately controlled temperature zones: flight deck, and two cabin zones A and B. The required temperature of the cabin supply air may be different for each zone, due to different target temperatures and internal heat loads. Therefore, the air conditioning pack provides the air with lowest required cabin supply temperature among all the cabin zones, and the higher temperature requirements for the remaining cabin zones are fulfilled by mixing the cold air from air conditioning pack with the trim air from the hot air manifold.

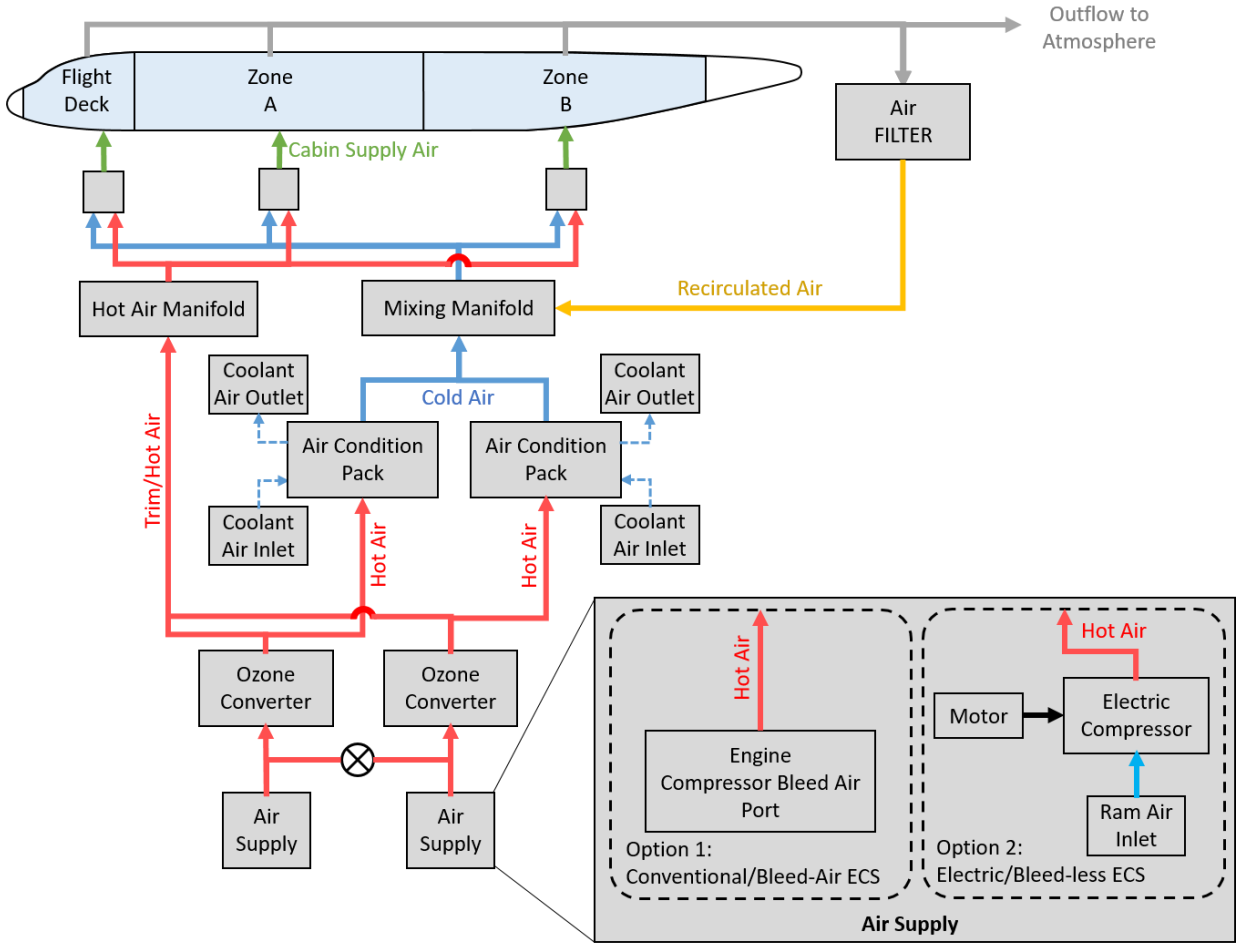

Fig. 4: Schematic Diagram of Environmental Control System (ECS)

In order to perform safety analysis, only the functional and logical flow views of ECS architectures are required. However, for the performance analysis, information about the whole aircraft and its mission is required, so that different ECS architectures can be assessed at aircraft-level (in terms of block fuel, range, etc.). For this purpose, a single-aisle conventional (tube-and-wing) configuration, similar to Airbus A320 and Boeing 737, is considered. In

addition, the mission considered for the performance analysis includes taxi-out, take-off, climb, cruise, descent, landing, and taxi-in segments.

B. Proposed Approach Implementation

This section is concerned with the implementation of the proposed approach for aircraft safety and performance trade-off, described in Section III. The process starts with defining the requirements view. Here, AirCADia Architect is used to generate the requirements hierarchy of a typical environmental control system (ECS), which includes functional, performance, and safety requirements, as shown in Fig. 5. These safety requirements (shown in pink color) and performance requirements (shown in blue color) will be considered during safety and performance trade-off. The target values for safety requirements are obtained through functional hazard analysis (FHA), as discussed in reference [4]. Note that in addition to safety and performance requirements (shown in Fig. 5), other key parameters of the ECS such as mass, power offtake, ram drag etc. will also be considered during trade-off, which are discussed later.

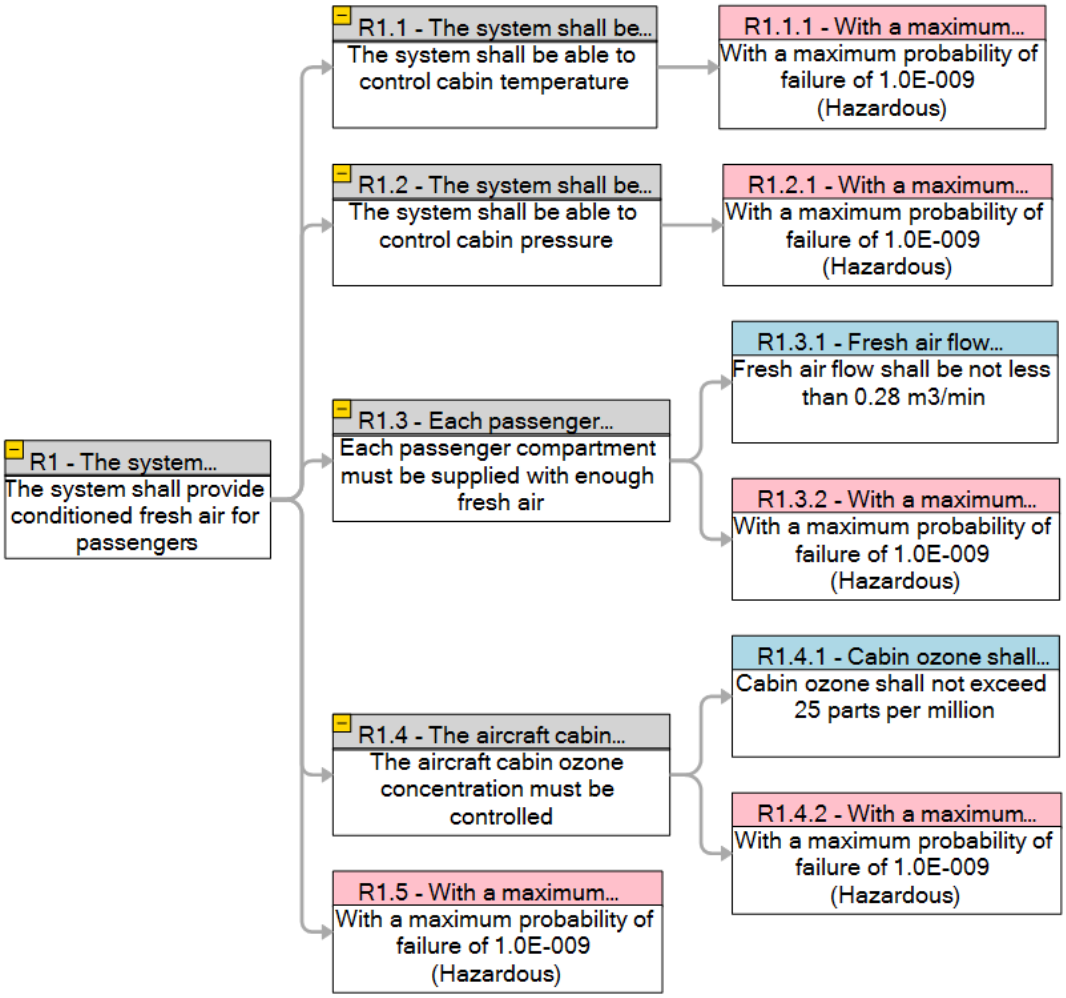

Fig. 5: Requirements View

After defining the requirements, the next step is to generate the functional and logical architectures. Here, the concept of functional-logical zigzagging [20] is employed to generate the functional and logical flow views. The logical flow view of a simplified electrical environmental control system, generated in AirCADia Architect, is shown in Fig. 6. It features a single air conditioning pack with a single heat exchanger, which is fed with high pressure air by an electric compressor. Once the functional and logical flow views have been defined, the ECS architecture can be analyzed from performance and safety perspectives simultaneously.

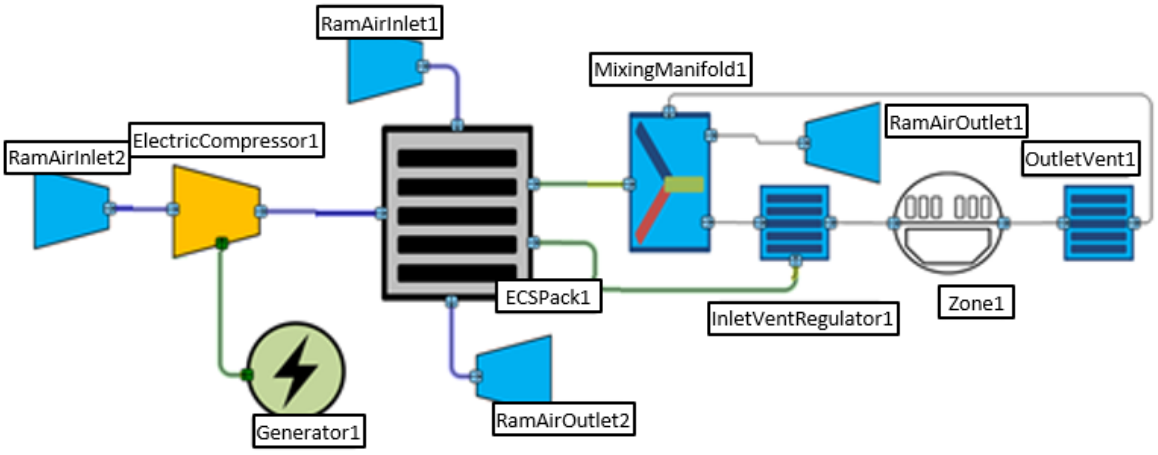

Fig. 6: Logical Flow View for a Simplified Environment Control System (ECS)

In order to size the components of ECS, the three-step process described in Section III is applied.

Step 1: Identify all the configurations of the architecture. In this case, for the simplified ECS architecture (shown in Fig. 6), there is only one configuration present due to the lack of redundant components.

Step 2: Create the computational workflow for each configuration. Here, a method from Bile et al. [3] is employed to automatically/dynamically generate the computational workflows. The computational workflow of the simplified ECS is shown in Fig. 7.

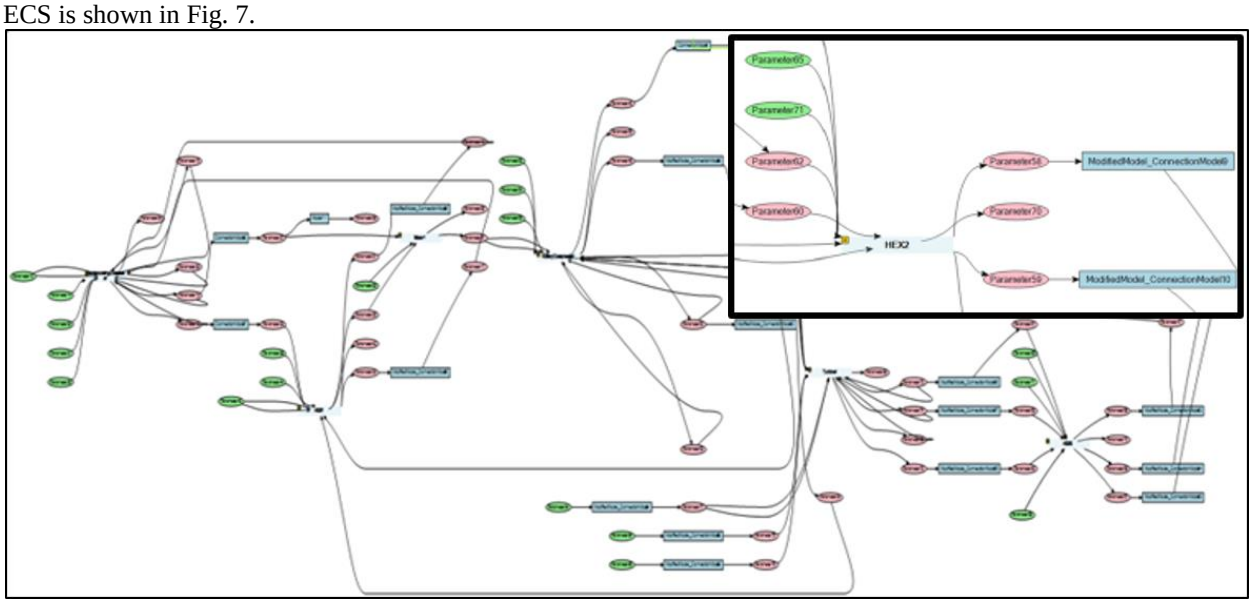

Fig. 7: Computational Workflow of the Simplified ECS

Step 3: Size the system based on the worst case flight condition. This step involves identifying all the flight conditions and then executing the computational workflow for each of the flight conditions. The one which requires the maximum power offtake becomes the worst case, and is used for sizing the components of the ECS. The collection of different flight conditions can be obtained from the specified mission. The assumed mission speed schedules for climb, cruise and descent segments are given in Table 1. The climb segment is flown with constant speed of 250 kts CAS for altitude up to 10000 feet, and 280 kts CAS above 10000 feet until transition. For the cruise segment, constant altitude of 35000 feet and Mach number of 0.78 are considered. In practice, however, the cruise segment is flown in steps of constant altitude (i.e. step-cruise) due to air-traffic management. Finally, the descent segment is flown with the fixed speed of 250 kts CAS. In addition, the flight conditions on ground (for takeoff/landing) are considered from normal (ISA), cold (ISA-55°C) and hot (ISA+35°C) days.

Table 1: Speed Schedules for Mission Segments

Climb Segment Speed Schedule			Cruise Segment Speed Schedule			Descent Segment Speed Schedule		
Altitude [ft]	Speed CAS [kts]	Speed Mach	Altitude [ft]	Speed CAS [kts]	Speed Mach	Altitude [ft]	Speed CAS [kts]	Speed Mach
1500	250	0.388	35000	264.4	0.78	35000	250	0.741
5000	250	0.413				30000	250	0.668
7500	250	0.432				25000	250	0.604
10000	250	0.452				20000	250	0.547
15000	280	0.555				15000	250	0.497
20000	280	0.610				10000	250	0.452
25000	280	0.672				7500	250	0.432
30000	280	0.742				5000	250	0.413
31000	280	0.757				1500	250	0.388
32463	280	0.780						
35000	264.4	0.780						

A full-factorial design of experiment study was conducted in the AirCADia software by considering different combinations of altitude, speed, and ISA temperature deviations. The results of executing the computational work for all the different combinations of flight conditions are shown as scatter plots and parallel coordinates plot in Fig. 8. It can be identified that the values for the fresh air mass flow rate varies from 1.55 kg/s to 4.81 kg/s. Similarly, the required power offtake varies approximately from 15.5 kW to 226 kW. As the maximum power offtake required by the ECS is ~226 kW, therefore the worst-case flight condition for the sizing the simplified ECS architecture corresponds to it, i.e. 30000 ft altitude, 0.78 Mach speed, and ISA + 35°C temperature. This condition results in the total mass of the simplified ECS equal to ~997 lb.

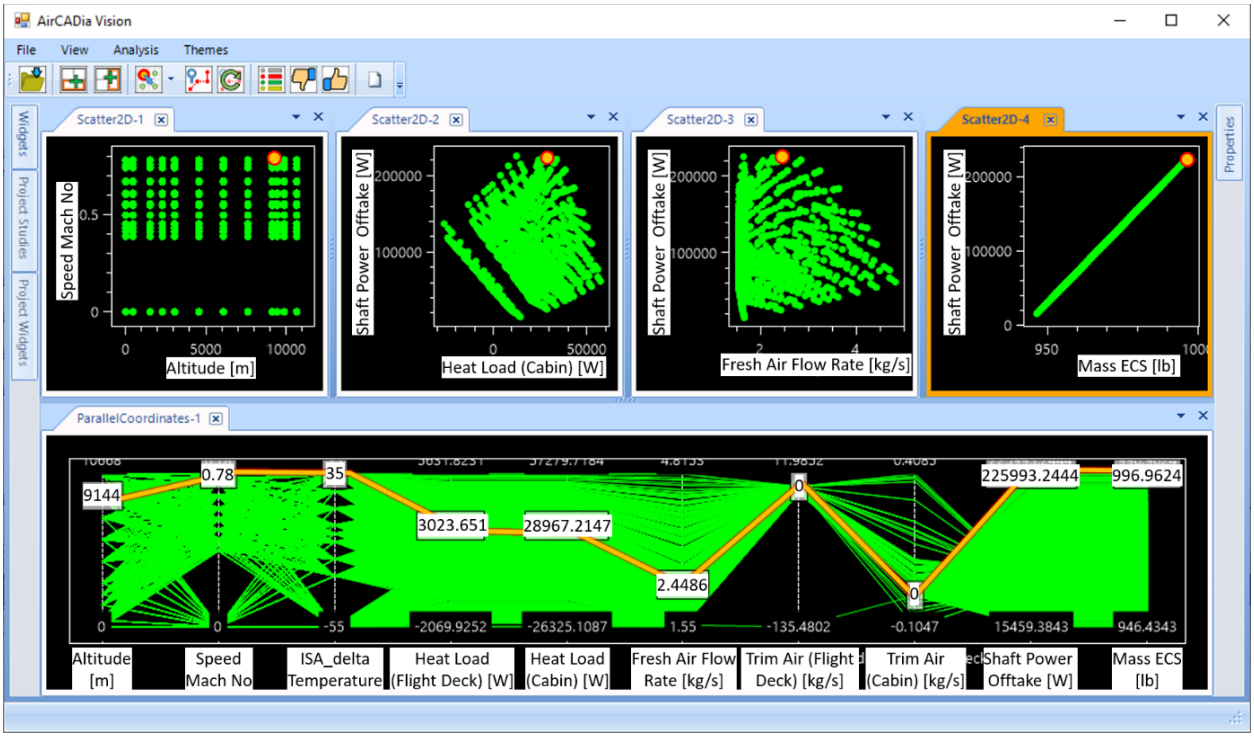

Fig. 8: Performance Characteristics of the simplified ECS

Next, the generated ECS architecture is analyzed for safety assessment. The functional and logical flow views of the simplified ECS is used to automatically generate the fault tree analysis (FTA) models for different failure conditions by employing the method proposed in reference [4] . The resulting FTA model for condition "failure to control cabin pressure" is shown in Fig. 9.

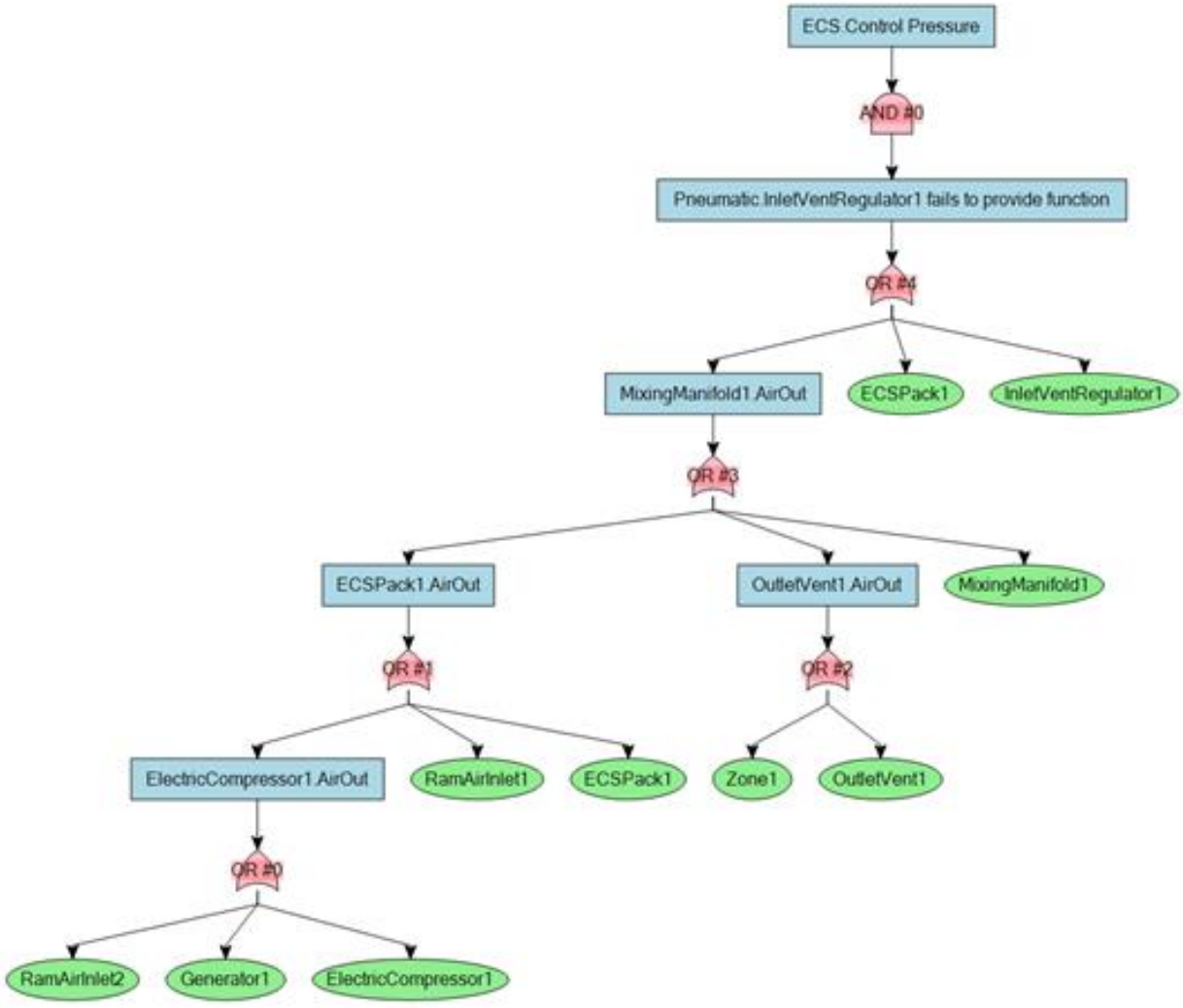

Fig. 9: Fault Tree Analysis Diagram

The results of the fault tree analysis for the condition "failure to control cabin pressure" are shown in Table 2. The relative probabilities of failure assigned to each element (shown in column 3) are for demonstration purposes. The minimal cuts are displayed in descending importance according to their relative probability (probability of the sets divided by the probability of the top event). The calculated probability of failure to control the cabin pressure is 3×10^{-5}, which is greater than the required target value of 10^{-7}, i.e. the safety of the system is not satisfactory.

Table 2: Result of Fault Tree Analysis (FTA) for Simplified ECS

Minimal Cut Set	No. of Components	Relative Probability
Generator	1	0.333
ECS Pack 1	1	0.333
Electric Compressor 1	1	0.333
Zone 1	1	3.33e-6
Outlet Vent 1	1	3.33e-6
Mixing Manifold 1	1	3.33e-6
Ram Air Inlet 1	1	3.33e-6
Ram Air Inlet 2	1	3.33e-6
Inlet Vent Regulator 1	1	3.33e-6
Probability of Failure: $P(\text{Failure to control the cabin pressure}) = 3 \times 10^{-5}$		

In order to improve the safety of the electric environmental control system, two new ECS architectures with redundant components were created in AirCADia Architect. The first is a double-redundant architecture, where one extra air conditioning pack, electric compressor and motor are added. The second architecture is a triple-redundant with two extra options for the critical components. The double- and triple- redundant architectures are shown in Fig. 10 and Fig. 11, respectively. The objective is to identify the impact of redundancy on the safety and performance characteristics.

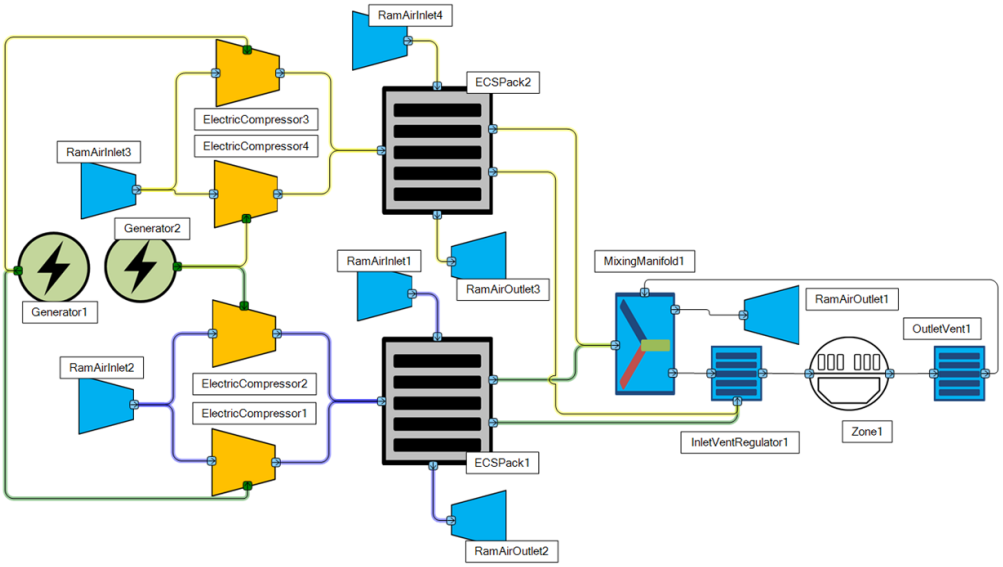

Fig. 10 Double-Redundant Architecture just meeting the safety target

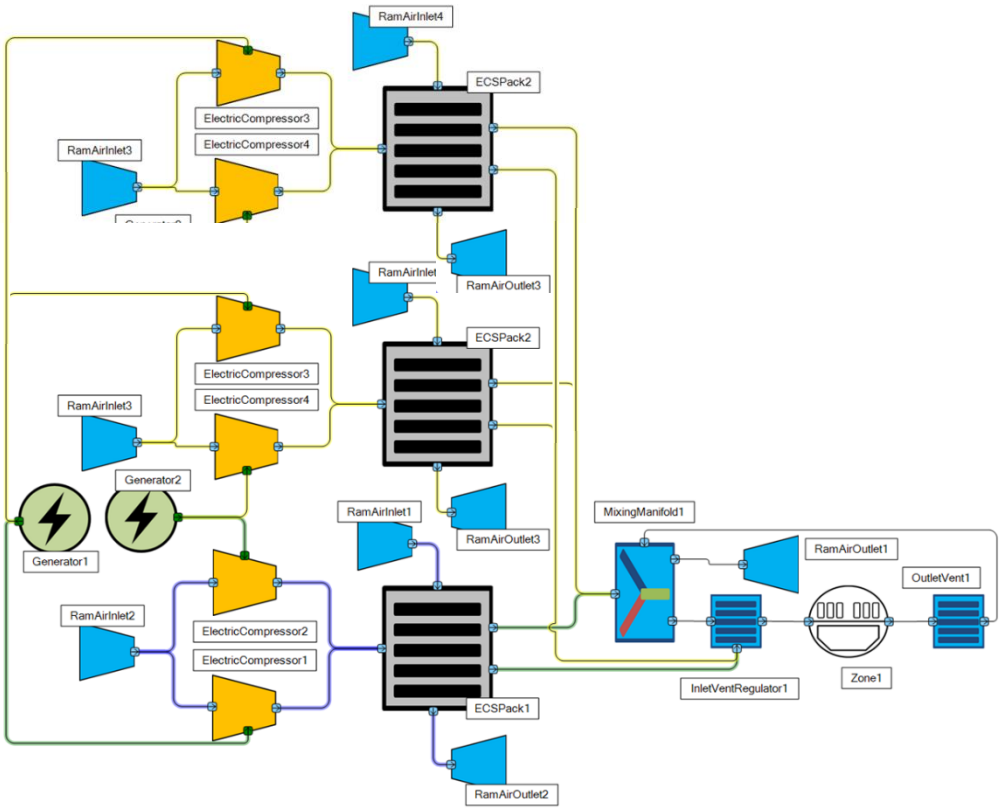

Fig. 11 Triple-Redundant architecture

The redundant architectures result in multiple configurations. For instance, the double-redundant architecture has two configurations, as shown in Fig. 12. In order for the two air conditioning packs to be functioning independently (in case of failure), these should be sized in order to satisfy the total load from the aircraft cabin. Similarly, the other redundant components, including electric compressor and motor need to be sized for the total load. This leads to creation of two computational workflows (one for each configuration). In this particular case, the two computational workflows are identical. However, if the two configurations employ different means, then the two workflows would be different. For instance, if electric power is generated by three components, engine generator, battery, and ram air turbine, then the three resulting computational workflows would be different. Each workflow will be used to size their respective components.

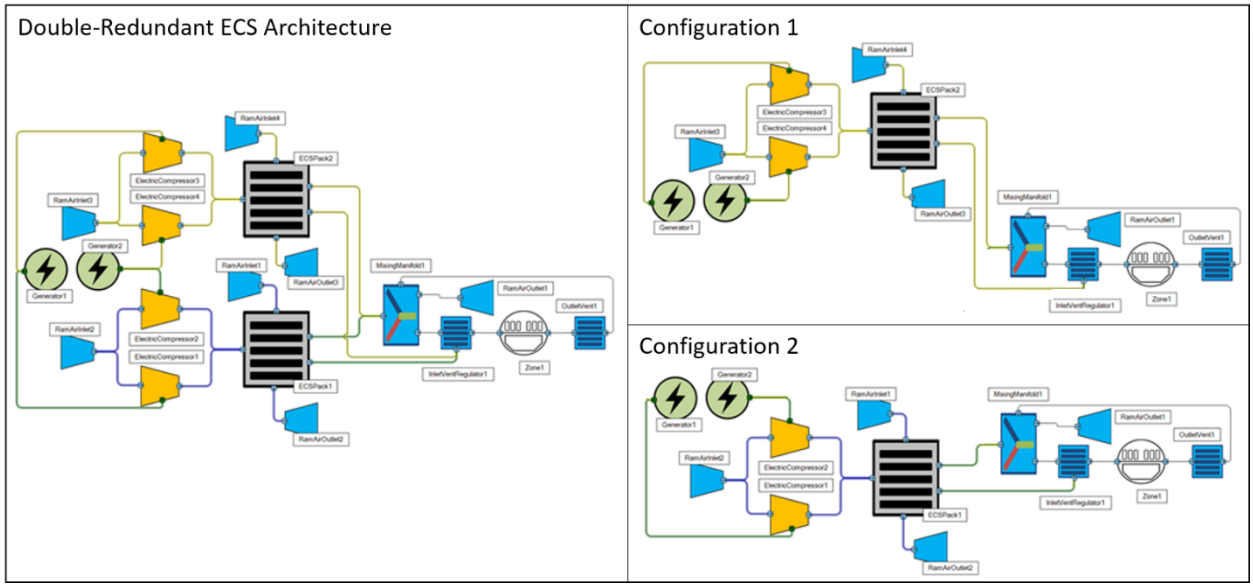

Fig. 12: Multiple Configurations resulting from Redundant Architecture

The safety and performance characteristics of the two redundant architectures were assessed using the same methods which were applied to the simplified ECS architecture. First, the fault tree analysis models were generated automatically for the condition "failure to control cabin pressure" in order to determine the probability of failure. Next, the two architectures were evaluated for performance characteristics (power offtakes, mass, drag). The results of the two architectures are shown in Table 3.

Table 3: Comparison of different ECS Architectures in terms of Safety and Performance Characteristics

	Safety	Performance		
ECS Architectures	Probability of Failure to Control Pressure	Power Offtake [kg/s or kW]	Mass [lb]	Ram Air Drag Coefficient
Architecture 1: Conventional Bleed-Air	2.2×10^{-8}	1.78 kg/s	1572	-
Architecture 2: Simplified ECS	3.0×10^{-5}	226 kW	997	0.000176
Architecture 3: Double Redundant ECS	8.9×10^{-8}	226 kW	1662	0.000176
Architecture 4: Triple Redundant ECS	6.1×10^{-11}	226 kW	2326	0.000176

It can be observed from Table 3 that all the three electric ECS architectures (i.e. Architectures 2, 3, and 4) require power offtake of 226 kW with additional ram air drag coefficient of 0.000176. However, the mass and probability of failure to control pressure for the three electric ECS architectures are different. On one hand, increasing the number of redundant components reduces the probability of failure to control pressure, but on the other hand it increases the total mass of the ECS system. The three electric ECS architectures can be compared against the conventional ECS architecture (i.e. Architecture 1) in terms of mass and probability of failure to control pressure. However, conventional (bleed-air) and electric (bleed-less) architectures cannot be directly compared because the two architectures require different types of power offtakes (conventional ECS requires bleed air power offtake and electric ECS requires shaft power offtake. This implies that the two different types of ECS architectures should be compared at aircraft level.

Table 4 shows the aircraft level parameters for the four ECS architectures. It can be observed that all the three electric ECS architectures provide more range (column 3 in Table 4) compared to the conventional ECS architecture. The gross weight of the triple-redundant ECS architecture (Architecture 4) is greater than the conventional architecture (Architecture 1) due to extra ram air compressors, however, the range is still better than the conventional architecture. Please note that the thermal analysis was not conducted during these studies, which may reduce the performance of electric ECS architectures.

Table 4: Comparison of different ECS Architectures in terms of Top-Level Aircraft Parameters

Architectures	Gross Weight [lb]	Range [nmi]	Takeoff Field Length [ft]	Landing Field Length [ft]	Approach Velocity [kts]	Flyover Noise [EPNdB]	Sideline Noise [EPNdB]	Nitrogen Oxide Emissions [lb]
Architecture 1	174146	2871.5	7463.84	5674.74	143.0	83.49	86.80	1054.83
Architecture 2	173207	3089.1	7359.75	5652.22	142.6	83.36	86.86	955.43
Architecture 3	173932	3076.1	7418.70	5669.60	142.9	83.38	86.85	960.24
Architecture 4	174651	3063.2	7477.60	5686.56	143.2	83.42	86.84	965.01

The presented methods for automatically generating the safety models (such as fault tree analysis) and computational workflows from the functional and logical flow views enable the architects to make better informed decisions, by conducting the trade-off between safety and performance characteristics at the early design stage.

V. Summary and Conclusions

Presented is a framework for aircraft subsystems sizing which also computes the effect of systems size on the resulting aircraft level performance. An essential part of the framework is safety analysis [4]. Thus integrated safety and performance analyses are enabled in early systems architecture design, and the study of possible trade-offs. The framework utilizes the Requirement, Functional, Logical, and Physical (RFLP) paradigm as a means of describing the architecture. RLFP is augmented here with a Mission & Flight Conditions domain, a Configurations domain and a Computational domain to facilitate the subsystems sizing process. The methodology consists of several steps. First, the various configurations that the architecture can adopt are determined. This includes failed configurations, which are determined with the help of the safety results provided by Fault Tree Analysis. For each one off them a workflow containing the associated computational models is assembled. Then, the systems are sized based on the worst case flight condition. It involves identifying all the flight conditions and then executing the computational workflow for each of the flight conditions of a configuration. The flight condition which requires the maximum power offtake becomes the worst case flight condition, and is used for sizing the components of the system. Finally, the aircraft level performance is evaluated while accounting for the masses, drag contributions and power outtakes of the subsystems, enabling safety and performance trade-offs. The concept is illustrated with a representative example, where an electrical environmental control system is sized following the suggested methodology and the aircraft level performance and possible trade-offs are studied. One of the main advantages of the proposed method is that instead of assessing systems at the single point in the mission segment, it enables systems sizing of extreme cases for safety consideration.

Future work will include incorporation of the physical view, detailing aircraft geometry and subsystems spatial layout, in order to conduct ‘transversal studies’ such as thermal analysis.

References

[1] EASA, “Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes CS-25,” no. Amendment 20, 2017.

[2] A. Joshi, S. Vestal, and P. Binns, “Automatic Generation of Static Fault Trees from AADL Models,” *DSN Workshop on Architecting Dependable Systems*, 2007.

[3] Y. Bile, A. Riaz, M. D. Guenov, and A. Molina-Cristobal, “Towards Automating the Sizing Process in Conceptual (Airframe) Systems Architecting,” in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018, pp. 1–16, doi:10.2514/6.2018-1067.

[4] S. Jimeno, A. Molina-Cristobal, A. Riaz, and M. Guenov, “Incorporating Safety in Early (Airframe) Systems Design and Assessment,” in *AIAA Scitech 2019 Forum*, 2019, no. January, pp. 1–18, doi:10.2514/6.2019-0553.

[5] S. Liscouët-Hanke, J.-C. Maré, and S. Pufe, “Simulation Framework for Aircraft Power System Architecting,” *Journal of Aircraft*, vol. 46, no. 4, pp. 1375–1380, Jul. 2009, doi:10.2514/1.41304.

[6] S. Liscouët-Hanke, “A model-based methodology for integrated preliminary sizing and analysis of aircraft power system architectures,” University of Toulouse, 2008.
[7] C. de Tenorio, D. Mavris, E. Garcia, and M. Armstrong, “Methodology for Aircraft System architecture sizing,” in *26th Congress of International Council of the Aeronautical Sciences*, 2008.
[8] C. de Tenorio, M. Armstrong, and D. Mavris, “Architecture Subsystem Sizing and Coordinated Optimization Methods,” in *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, 2009, no. January, pp. 1–11, doi:10.2514/6.2009-37.
[9] I. Chakraborty, D. N. Mavris, M. Emeneth, and A. Schneegans, “An integrated approach to vehicle and subsystem sizing and analysis for novel subsystem architectures,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 230, no. 3, pp. 496–514, Mar. 2016, doi:10.1177/0954410015594399.
[10] D. M. Judt and C. Lawson, “Development of an automated aircraft subsystem architecture generation and analysis tool,” *Engineering Computations*, vol. 33, no. 5, pp. 1327–1352, Jul. 2016, doi:10.1108/ec-02-2014-0033.
[11] D. M. Judt and C. Lawson, “Methodology for Automated Aircraft Systems Architecture Enumeration and Analysis,” in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, no. September, pp. 1–17, doi:10.2514/6.2012-5648.
[12] Y. Bile, “Component-Driven Computational Design of Complex Engineering Systems,” Cranfield University, 2018.
[13] E. Ruijters and M. Stoelinga, “Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools,” *Computer Science Review*, vol. 15–16, pp. 29–62, Feb. 2015, doi:10.1016/j.cosrev.2015.03.001.
[14] “ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,” Society of Automobile Engineers, 1996.
[15] M. D. Guenov, A. Riaz, Y. Bile, A. Molina-Cristóbal, and A. S. J. van Heerden, “Computational Framework for Interactive Architecting of Complex Systems,” *Systems Engineering - Submitted*, 2019.
[16] “Design methodology for mechatronic systems (VDI 2206),” Verein Deutscher Ingenieure, 2004.
[17] C. De Tenorio, “Methods for Collaborative Conceptual Design of Aircraft Power Architectures,” Georgia Institute of Technology, 2010.
[18] L. A. McCullers, “Flight Optimization System - Release 8.23 - User’s Guide.” 2011.
[19] I. Chakraborty, “Subsystem Architecture Sizing and Analysis for Aircraft Conceptual Design,” Georgia Institute of Technology, 2015.
[20] M. Guenov, A. Molina-Cristóbal, V. Voloshin, A. Riaz, A. S. van Heerden, S. Sharma, C. Cuiller, and T. Giese. "Aircraft systems architecting-a functional-logical domain perspective." in *16th AIAA Aviation Technology, Integration, and Operations Conference*, p. 3143. 2016.

2020-01-05

Enabling interactive safety and performance trade-offs in early airframe systems design

Jimeno, Sergio

AIAA

Jimeno S, Riaz A, Guenov MD, Molina-Cristobal M. (2020) Enabling interactive safety and performance trade-offs in early airframe systems design. In: 2020 AIAA SciTech Forum, 6-10 January 2020, Orlando, Florida, USA

https://doi/10.2514/6.2020-0550