A Model Based Safety Assessment for Multirotors

Saad Bin Nazarudeen

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

December 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Saad Bin Nazarudeen**

Entitled: **A Model Based Safety Assessment for Multirotors**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr.

_____ External Examiner
Dr. Otmane Ait Mohamed

_____ Examiner
Dr. Charles Kiyanda

_____ Supervisor
Dr. Jonathan Liscouët

Approved by _____
Martin D. Pugh, Chair
Department of Mechanical and Industrial Engineering

_____ 2022

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

A Model Based Safety Assessment for Multirotors

Saad Bin Nazarudeen

Unmanned Aerial Vehicles (UAVs) must be safe and reliable to prevent fatal accidents in densely populated areas. This research makes the first steps to create a framework which can integrate safety and reliability considerations in the design process. The conceptual design process should consider creating design models coupling sizing with system architecture. Additionally, the multirotor has safety challenges from the propulsor configuration. They lose flight control and show erroneous flight behaviour when propulsors fail. Hence, the design models of multirotor should also incorporate a controllability assessment method to identify and isolate uncontrollable events. For this matter, an appropriate tool should be considered to create such design models.

A combination of OpenAltarica, System Analyst and Python is used to create design models of multirotor in a model-based safety assessment framework. These models are developed by integrating system architecture and controllability assessment following the etiquettes of the process. A case study is used to validate the framework and to demonstrate its ability to explore innovative designs. The reliability analysis confirms that the multirotors are fault-tolerant except quadrotor and some configurations are potentially highly reliable.

The results demonstrate the feasibility of the multirotor system modelling methods in terms of reliability and pave the way to further develop the model-based safety assessment framework with sizing methodologies. The models can also be further enhanced with the addition of a component fault library, additional failure modes and implementation of diagnosability analysis, fault detection and identification analysis. Fault libraries and failure modes can help in foreseeing uncontrollable cases. In contrast, diagnosability analysis, fault detection and identification analysis can integrate detect,

isolate and recover mechanisms, and ensure redundancy optimization effectively. Additionally, the framework should also be combined with multidisciplinary design optimization for sizing. Such design models can contribute to the emergence of UAVs for safety-critical applications.

Acknowledgments

I would like to acknowledge, and give my warmest gratitude to my supervisor, Dr. Jonathan Liscouët for his valuable time and guidance who made this work possible. His guidance and advice encouraged me through all the stages of writing this work. I would also like to thank my committee members for letting my defence be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

I would like to give thanks to Concordia University as a whole for the opportunity given. I would also like to give thanks to the Altarica development team for their timely guidance.

I would also like to give special thanks to my family as a whole for their continuous support and understanding when undertaking my research and writing my project.

Finally, I would like to thank my friends at 975 Old Orchard, for letting me through all the difficulties. I have experienced your warmness day by day. You are the ones who helped me finish my degree.

I will remember all of you for the rest of my life.

Thank you once again.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1 gives a background and safety-critical issues posed by multirotors. It illustrates the problem statement with a state-of-the-art conceptual design which integrates safety and reliability considerations in the design process to identify their limitations. It then presents the scope and the approach to develop a framework that can address the identified limitations.

1.1 Background

Multirotor Unmanned Aerial Vehicles (UAVs) were under active development for more than a decade [1]. They have shown promising applications in urban surveillance, agriculture, media coverage, logistics, deliveries, flying taxis, or flying ambulances that would change our daily lives. Emerging autonomous control has evolved such UAVs into autonomous aerial vehicles (AAV) [2]–[4]. These vehicles can accomplish a preassigned task without human operators, especially high-risk operations like air taxiing, painting skyscrapers, or aerial firefighting. Such operations are safety critical. Failure leading to an uncontrollable UAV can cause a fatal accident if it collides with humans, aircraft, helicopters, or infrastructures. This damage is increasing with its mass, speed, and size. Therefore, designs of such vehicles must be adopted with high safety and reliability considerations.

The state-of-the-art conceptual design methodologies of multirotor focus on propulsor configurations and evaluate through sizing for performance (e.g., overall weight, payload, range, maximum

speed) [5]–[9]. But they lack any safety considerations to avoid failure events. This absence is a major limitation of current design methodologies [10]. However, some of the state-of-the-art reliability evaluation studies on multirotor focus on flight control [11], [12]. It emphasizes the loss of control of the multirotor following a rotor failure and assesses subsequent controllability issues. Some studies create unique designs to avoid physical failure from collision [13], [14]. Thus, it stands out that safety and reliability considerations are not adopted in the sizing of multirotor. This work focuses on the necessity of developing a new methodology and tool to evaluate the reliability of multirotor depending on various architectures coupled with corresponding controllability assessment. And it should also enable the flexibility to integrate a sizing methodology in the future.

For that matter, the research intends to develop a framework that can be integrated into the conceptual design process of multirotor and formalize the reliability calculations for automating safety analysis.

1.2 Problem statement

This section investigates the safety and reliability challenges in the state-of-the-art conceptual design of multirotors by comparing a conventional one with a redundant one using criteria like mass, safety implementations and reliability by a generic approach.

1.2.1 Medium-sized midrange multirotor

A simple medium-sized midrange multirotor is designed to illustrate the problem statement. The design represents medium-sized media drones, mail delivery drones, and surveillance drones that usually operate over heavily populated areas. This design does not represent the typical size of safety-critical applications like air taxis, flying ambulances, and aerial firefighting. The choice of a medium-sized drone is driven by the limitation of the selected sizing tool, which is designed to handle a maximum payload of 10 kg only.

1.2.2 Sizing

The tool named Flyeval [15] is used to size the conventional design architecture of a quadrotor. It doesn't have any means to apply redundancy; therefore, parallel components are added manually to create a redundant design architecture. The weight of each redundant component is added to the total weight, looping back to the performance evaluation and eventually to another sizing loop until converging. This approach avoids developing a new sizing tool, but it does not optimize the redundancies and neglects the effect of failure cases on sizing. Flyeval is proposed in academic research for validating drone sizing methodologies [16], [17]. It allows the user to perform sizing according to preliminary design requirements and select off-the-shelf components from a database. It then generates a mathematical model and calculates the resulting key performance criteria (forward speed, flying range, and hovering time) so that the user can efficiently verify or refine its component selection. The primary mission requirement considered for sizing is to carry a payload of 10 kg within a 5 km range.

1.2.3 Reliability analysis

The typical first rule of safety-critical design standards is: A catastrophic failure condition must not result from a single failure and must be extremely improbable [18]. A catastrophic failure condition is defined as: High impact crash is imminent and unavoidable with the vehicle's destruction due to a complete system failure. Severe injuries or the death of people on the ground is possible. Infrastructures can be damaged heavily. Here, a catastrophic failure condition shall have a probability of occurrence less than or equal to 10^{-7} [18], [19].

Other rules for less critical conditions (hazardous, major, and minor) also apply, but the presented reliability analysis focuses on evaluating the conventional and redundant designs against this first rule only for simplicity and conciseness.

Some studies on fault-tolerant control of quadrotors have proposed emergency landing procedures [20]–[22] to avoid a single rotor failure to lead to a catastrophic condition. It is a significant improvement in the reliability of the quadrotor. However, these procedures do not maintain yaw control. In the studied case, the failure can occur in cruise flight and the control scheme must completely stop

the speeding vehicle and engage a hover mode before any controlled descent. This procedure requires sufficient control of all control axis. That is why the present reliability analysis assumes that a catastrophic failure condition can be avoided only if the UAV keeps full or degraded control of all control axis.

The reliability analysis of the conventional and redundant designs is performed with standard failure mode and effect analysis and reliability block diagrams methodologies [23] and boolean algebra. The reliability computation is based on independent and random failures, leading to constant failure rates and convenient exponential distribution. The probability of failure is expressed as:

$$F(t) = 1 - e^{-\lambda t} \tag{1}$$

Where, λ is the failure rate (h^{-1}) and t is the exposure time (h).

The exposure time corresponds to the maximum flight time of 22 minutes between two charges and is enforced by the assumption of a built-in test procedure confirming the functionality of each component at power-up.

1.2.4 Conventional design

General information

The conventional design of the quadrotor consists of a frame, battery, power distribution board, Flight Controller (FC), Inertial Measurement Units (IMU), and four sets of electronic speed controllers (ESC), rotors, and propellers with no redundancy.

Conventional design reliability analysis

Failure Mode and Effect Analysis (FMEA) is a simple and widely used reliability analysis that considers the effects of each component's single failures and assesses the failure severity [23]. It shows that each component of the conventional quadrotor is susceptible to at least one catastrophic effect. Hence, it violates the typical first rule of safety-critical design standards. It can be concluded that the conventional design is unfit for any safety-critical application.

1.2.5 Redundant design

General

The redundant design is obtained by incorporating redundancies systematically into the conventional design as follows,

(1) A redundant battery and a battery management system are added;

(2) Inertial measurement units and the flight controllers are tripled and managed with a majority voting algorithm. In case of failure, the service will continue with the two agreeing flight controllers;

(3) Quadrotor configuration is modified into a co-axial configuration for redundancy and allowing the development of control failure mitigating strategies;

(4) As the design is modified, a new off-the-shelf frame is selected considering the increase in maximum take of weight due to redundant components. The thickness of the plate of the frame is increased for rigidity. The arm and motor mass are increased to reflect dualization.

Redundant design reliability analysis

Unlike conventional design, the redundant design is robust to single failures. Therefore, the reliability analysis needs to focus on the combination of failures. Fault Tree Analysis (FTA) and Reliability Block Diagram (RBD) analysis [23] are more suited than an FMEA for this purpose. RBD is selected because it has the advantage of providing a physical illustration of the system architecture.

For comparison, an RBD is constructed for both conventional and redundant design according to the ability of the design to perform the emergency landing procedure in case of a complete system shutdown. The reliability of the majority voting redundancies is obtained from a k-out-of-n redundancy calculation as follows:

$$R_{2/3}(t) = 3R_i(t)^2 \cdot (1 - R_i(t)) + R_i(t)^3 \tag{2}$$

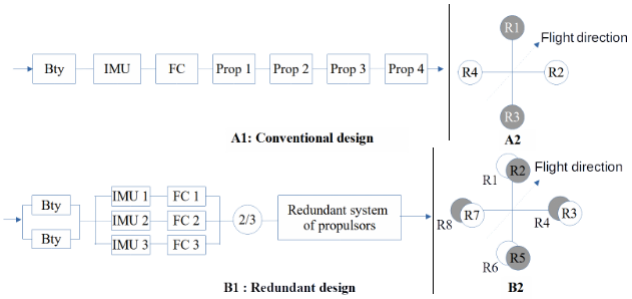


Figure 1.1: RBD of conventional design (A1) and redundant design (B1) with corresponding configurations (A2 and B2)

Where, $R_i(t)$ is the reliability of the identical components of the redundancy, in effect, each assembly of an inertial measurement unit and flight controller.

An evaluation of reliability based on controllability is presented in [11]. This approach applies here to the propulsor system. A propulsor is the assembly of a speed controller, electric motor, and propeller. The reliability of the redundant system of propulsors is calculated from the union of the probabilities of each failure case (no failure, single failures, and double failures) that maintain full or degraded controllability of all the control axis [24]. The co-axial quadrotor can keep control of all the control axis for any single failure. The resulting reliability of the system of propulsors is integrated into the RBD shown in Figure 1.1 and expressed as follows:

$$R_{RSP} = \sum_{i=0}^{k} C_i \cdot R_{prop}(t)^{m-i} \cdot (1 - R_{prop}(t)^i) \tag{3}$$

Where, $R_{RSP}(t)$ is the reliability of the redundant system of propulsors, C_i is the number of controllable cases of multiplicity i and $R_{prop}(t)$ is the reliability of a propulsor, which is the product of the speed controller, motor, and propeller reliabilities.

The failure rates used for RBD calculations in Figure 1.1 represent modern high-end transport category aircraft equipment, which is probably overly optimistic. Therefore, it is interesting to evaluate the impact of derating those failure rates. Table 1.1 shows the reliability of conventional and redundant design for failure rates rating from the toy industry (derating 1000) to high-end transport category aircraft (no derating).

The results in Table 1.1 illustrate the unfitness of conventional design and the potential of a fully

Table 1.1: Emergency landing probability of failure

Failure rate derating	Conventional	Redundant
x1000 (toy industry)	6.42E-02	4.40E-03
x100	6.62E-03	4.65E-05
x10	6.64E-04	4.68E-07
x1 (transport category aircraft)	6.64E-05	4.68E-09

redundant design, but implementation feasibility needs to be demonstrated and several challenges need to be addressed. For example, the analysis assumes an ideal (no failure) battery management system, majority voting algorithms, electric wiring, and airframe. The actual implementation of these components could significantly impact the overall reliability as it can reveal single points of failure.

1.2.6 Conventional and redundant design sizing

Both conventional and redundant designs are sized with the approach described in Section 1.2.2. The conventional design uses a LiPo 12S-44.4V-25C-32000 mAh battery, while the redundant one uses a LiPo 12S-44.4V-25C-62000 mAh. The mass of redundant components and additional fittings, including the harness, is also considered. The frame size remains the same for both designs, but the frame mass of the redundant design increases by +50% to accommodate redundant components, coaxial rotors and doubled battery capacity. It results in an additional mass of +18 kg (82.5% increase) for the redundant design, as illustrated in Figure 1.2. It shows that reliability and safety measures can significantly impact the multirotor mass. However, this limitation is not the scope of the study.

1.2.7 Limitations

The major limitation of the generic approach is that it limits design freedom in the physical architecture of multirotor and is only applicable to specific designs or scenarios. It is limited to adding simple parallel redundancy. Hence, the approach prevents the exploration of innovative design architectures.

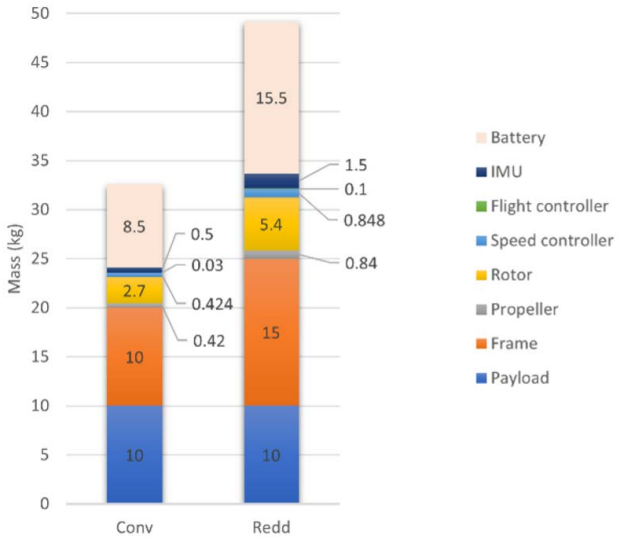


Figure 1.2: Mass comparison of conventional design (Conv) and redundant design (Redd)

On the other hand, physical architecture poses a design limitation. The electronics and propulsion system blocks are connected using a single line. And reliability evaluation of the electronics and propulsor systems are independent of each other as shown in Figure 1.3. Hence, it can be derived as follows.

$$R_{tot} = R_{elec} \cdot R_{prop} \tag{4}$$

Where R_{tot} is the total reliability of the multirotor, R_{elec} is the reliability of the electronics system and R_{prop} is the reliability of the propulsor system. This approach does not represent a full system model. For example, batteries generally power IMUs, FCs and ESCs in multirotor power systems. RBD represents only functional logic, hence the battery is connected in series with other components. Thus, RBD does not represent information flow in the system like power, communication, mechanical connections, control etc.

Another general limitation is the need for recurring safety analysis for each top failure event, for each configuration and for design changes in the architecture. Hence, the safety analysis should be automated to meet such design demands.

These limitations show the necessity of developing a framework to integrate safety and reliability

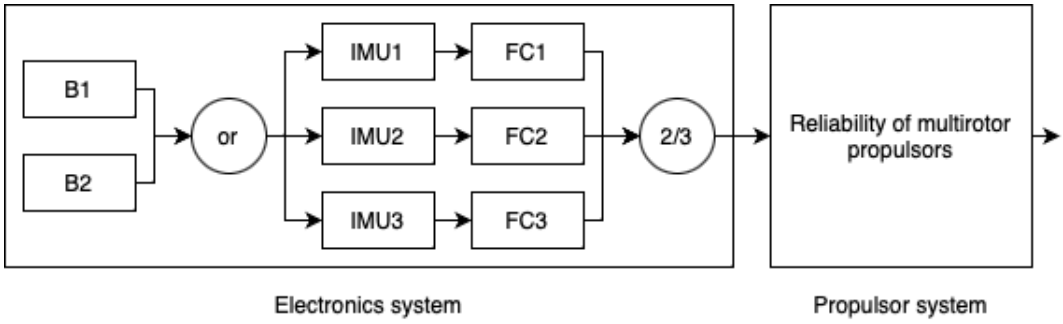


Figure 1.3: RBD of redundant design

considerations into the multirotor design.

1.3 Scope

This research work focuses on developing a framework that provides reliability assessment to various system architectures of multirotor coupling controllability assessments and enables the integration of a sizing methodology in the future.

1.4 Approach

The approach considers the following steps:

- Review the safety assessment methods against the ability to integrate safety considerations addressing the identified limitations of the generic approach.
- Create a safety assessment framework that addresses the identified limitations by allocating networked connections and coupling with corresponding controllability assessments.
- Validate the developed framework with a case study and evaluate the reliability of various multirotor configurations.

1.5 Outline

The work starts with the introduction discussing the background of multirotor and the necessity of safety-critical designs. The problem statement implements a generic approach to state-of-the-art conceptual design in order to identify the limitations. Chapter 2 reviews the classic and model-based safety assessment methods to identify a methodology to address the limitations of the generic approach. It also analyzes available open-source model-based safety assessment tools and selects a suitable tool. Also it introduces the system modelling in Altarica language. It also presents the technical aspects of the language and showcases the coding method. Chapter 3 formalizes the methodology used to develop the model-based safety assessment framework and illustrates system modelling methods. It then follows the model-based safety assessment etiquette to create system models of multirotor capturing both system architecture and controllability assessment. Finally, case studies are presented to validate the developed model-based safety assessment framework. It also compares the reliability of various multirotor configurations to identify the most potential safety-critical multirotor. In chapter 4, the work concludes the findings, contributions and identifies ways to improve the maturity of the design models in the framework for future work.

Chapter 2

Review of Safety Assessment Methods

This section reviews the safety assessment methods to capture the etiquettes of a methodology/tool that can address the limitations of the generic approach in design configuration, architecture and safety analysis. Then, it illustrates the selected tool in order to introduce the principles of system safety modelling to prepare the development of a safety assessment framework.

2.1 Classic safety assessment methods

Classic methods are categorized from the kind of model formalism within the safety assessment. These models are classic as they are informally built by safety engineers/analysts separately from the design process, purely from judgements. This way of coupling safety to design is extremely complex and creates large amounts of data.
Classic models can be categorized into elementary models, boolean formalisms and transition systems.

- Elementary models: They are created by investigating each element of the system and identifying the effects of events associated with that specific element. Failure Mode and Effect Analysis (FMEA) is an example of this kind.
- Boolean formalisms: They make use of boolean operators to architect either connection between elements or hierarchy of event occurrences in the system. Reliability Block Diagrams (RBD) and Fault Trees (FT) are examples of this type.

- Transition systems: They are a graphical method that presents the state identification of each element with respect to event transitions. Markov process and petri nets are good examples of this type.

2.1.1 Failure mode and effect analysis

Failure mode and effect analysis (FMEA) is a systematic methodology designed to identify and categorize known and potential failure modes and their causes and effects on system functionality [23], [25]–[28]. It assesses the risk associated with the identified failure modes, effects, and causes, and prioritizes precautions. It allows the system analyst to identify and carry out corrective actions to address the most serious concerns. An FMEA-based failure propagation model is defined from a local failure to its system effects and subsequent corrective actions. FMEA is presented in a sheet-based tabular form. Major known widely used software are XFMEA from Reliasoft, Reliability Workbench from Isograph, and Relex Reliability Studio 2007 from Crimson Quality.

FMEA presents an overview for evaluating a process by identifying where and how it might fail and assessing the relative impact of different failures. Usually, FMEA requires subject matter experts who can analyze the system to identify weaknesses and propose corrective actions that prevent a negative impact on the system's performance. FMEA doesn't predict failures but identifies existing and potential failures through a subjective and systematic assessment and classifies them according to the priority of risks.

But FMEA needs to be constantly updated. As the complexity of the system increases, a new failure mode will be discovered. This can lead to underestimating the outcome of failure events. FMEAs solely rely on the expertise of the analyst to identify and list failure modes. This task takes a lot of work and is time-consuming depending on the complexity of the system. FMEA is also not suitable to analyze multiple failures and the correlation between them.

2.1.2 Reliability block diagrams

Reliability block diagrams are a graphical representation of a system using component blocks showing how the reliability of each component contributes to the overall performance [23], [25]–[27], [29]. It shows the logical connection between components using functional dependence. For

example, two elements in a series mean that the function of the first component controls the second component. While two elements in parallel mean that the function of the first component and the second component are independent of each other. Hence, the failure of each component is propagated in this logical sense. Thus a system engineer can identify poor reliability areas and make necessary improvements. For a large system, RBD's graphical approach poses a challenge. RBD doesn't require subject matter experts but the system engineer's ability to formulate the functional dependency between components should be well-versed.

2.1.3 Fault tree analysis

Unlike FMEA, Fault tree analysis (FTA) is a graphical tool that identifies the causes of system-level failures [23], [25]–[27], [30], [31]. It is used to ensure that the system withstands the failure tolerance requirements even if multiple failures occur. It uses boolean logic to combine the identified component level failures called basic events that cause the system level failure called the top event to occur. Two major elements in FTA are "events" and "logic gates". These elements connect the events hierarchically using logic gates to identify the cause of the top undesired event. Hence, an FTA-based failure propagation model is defined from a system failure to its root causes. Major known widely used software are BlockSim from Reliasoft, Reliability Workbench from Isograph, and Relex Reliability Studio 2007 from Crimson Quality.

FTA helps in highlighting the critical components related to the system failure. It prioritizes the action items to solve the problem. But a large complex system with too many gates and events is difficult to be considered under a fault tree. Due to FT's graphical approach, another disadvantage is it examines only one top event at a time. It also has difficulties to capture common cause failures, time-related and other delay factors. Like FMEA, FTA also requires experienced subject matter experts who also know of logic gates.

2.1.4 Markov process

Markov process is a system analysis that describes the sequence of possible events from the state of the previous events [23], [25]–[27], [32]. It is a stochastic method for randomly changing systems. It means that the next state is dependent on the current state and is independent of the

past state. Markov models represent all states of a system as observable. Thus it shows all possible states, between states and the transition rate (probability of moving from one state to another per unit of time).

The simplest way to show a Markov model is through a Markov chain. It can be expressed using a transition matrix or a graph. A transition matrix is used to indicate the probability of moving from each state to another state. Generally, the current states are listed in rows, and the next states are represented as columns. Each cell then contains the probability of moving from the current state to the next state. For any given row, all the cell values must then add up to one. A graph consists of circles, each of which represents a state, and directional arrows to indicate possible transitions between states. The directional arrows are labelled with the transition probability. The transition probabilities on the directional arrows coming out of any given circle must add up to one.

A major disadvantage of the Markov process is state explosion. The number of possible states of a system with an 'N' number of components is 2^N. It also does not represent a functional system architecture. This makes it difficult for both design and safety engineers to collaborate.

2.1.5 Advantages and disadvantages of classic safety assessment methods

The advantages of classic safety assessment methods are as follows,

- Early identification and elimination of potential failures.
- Prioritize and implement corrective actions.
- Document failure propagation path and system safety history.

The major disadvantages of classic safety assessment methods are listed for clarity.

- Structural difference between systems specifications and models.
- Expertise to read safety data and knowledge in safety-related softwares.
- Inability to automate safety analysis following recurring design changes.
- Misunderstandings between safety analysts and designers due to discrepancies between working hypotheses. Hence, it causes difficulties in model checking the system models.

- One model usually represents one major safety goal.

Additionally, the mentioned disadvantages of classic methods prevent addressing the limitations of the generic approach in order to develop a framework.

2.2 Model-based safety assessment methods

2.2.1 MBSA background

As the importance of safety criticality grows with system complexity and size, the effort to handle reliability and safety analysis with the design aspects is getting convoluted. Thus developed the idea, to create system formal models based on system behaviour mathematically. At first, a nominal model will be developed with the logic to fulfill all system functionalities. Faulty or degraded behaviours can be injected into the model and create an extended system model [33]–[37]. Finally, the framework can automate the generation of required safety artifacts like FMEA tables and FTs from the extended system model. The model matures with time by integrating more design decisions and the safety artifacts automatically update with the model. This process is termed as Model-Based Safety Assessment (MBSA). The formal models in MBSA methods give both the design team and safety team a means to develop the system collaboratively. Thus unlike the classic methods, MBSA methods take safety initiatives from the earlier phases of the system development.

State-of-the-art MBSA frameworks are mostly for industrial production systems. These approaches are developed by combining Model-Based Systems Engineering (MBSE) with safety analyses, for ex. Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [38], Modularized component fault trees [39], [40], and Safe component model [41]. Extensive overviews of these methods are given by Aizpurua and Muxika [42], Sharvia et al [43], and Lisagor et al [44]. These MBSA approaches showcase the changes in the system model on the system's safety and the error propagation of each component separately. Höfig et al [45] present two case studies that show why these approaches are useful for complex industrial cases. MBSA methods were introduced for various metamodels, ex., MathWorks's MATLAB/Simulink [46], [47], Architecture Analysis & Design Language (AADL) [48], EAST-ADL [49], and the universal modelling language (UML) [50].

Eventually, MBSA started emerging without a systems engineering perspective. Nordmann presents

a SysML-based MBSA approach that combines SysML Internal block diagrams and Activity diagrams of system overview with component fault trees and shows how it supports safety experts when performing FTA and FMEA [51]–[53]. Bozzano captures an approach to the formal modelling of the system, the formalization of the requirements and the formal verification and validation of the system characteristics. Also, created an MBSA framework using the xSAP platform to automatically extract minimal cut sets for the triple modular generator in the avionics domain [54]. Nannapaneni proposes a method to automate reliability block diagrams from a formal framework using a Generic Modeling Environment with a domain-specific modelling language and PyEDA package available in the python environment [55]. Even though there are these MBSA abstracts captured, none of these are openly present as ready-to-use tools.

An overview and classification of approaches to model-based safety assessment is given in [56], [57]. Most of them focus on the automated generation of fault trees or FMEA or both. AltaRica [58] is a formal language for specifying complex, hierarchical systems in terms of automata and equations, which allows for modelling the functional and dysfunctional behaviour of a system. The FSAP/NuSMV-SA [36] approach supports failure mode definition based on a library of commonly used failure models, automatic fault injection, and automatic fault tree construction, based on a model checker. HiP-HOPS is a tool that supports the annotation of hierarchical system models with failure logic information, which can then be used for the automatic generation of fault trees. Later, It is integrated with Simulink models using Matlab [59]. Some MBSA tools worth mentioning that can automate both FMEA and FTA are [60]–[62]. Another MBSA tool developed using SysML can auto-generate FMEA tables [63], [64]. Safety analysis linked to a system dynamic behaviour is also proposed using Modelica [65]. SmartIflow is a component-oriented modelling language developed from information flow that shares many features like AltaRica [66].

However, a detailed analysis of open-source MBSA tools is required. But before that, an overview of the MBSA process should be understood to enable the discussion.

2.2.2 MBSA general overview

In general, MBSA methods can be categorized into a seven-step process which is handled collaboratively by design engineers and safety engineers as detailed below.

(1) Model capturing: This is the first step in MBSA methods where the design team provides a formal model to the system under development.

(2) Requirement capturing: Both the design and the safety engineer provide the properties of the system under development.

(3) Failure Mode capturing: The safety engineer identifies failure modes of the components of the system model under development.

(4) Failure injection: The safety engineer injects failure modes of the system into the design model and creates an extended system model.

(5) Formal assessment of system model:

- Formal Verification: The design engineer checks the model against a set of pre-defined requirements, under the hypothesis of nominal behaviour.
- Assess Safety: The safety engineer checks the model against a set of pre-defined requirements, under the hypothesis that the component may fail.

(6) Automatic generation of safety artifacts: automated generation of minimal cut sets, fault trees and FMEA tables from the extended system model.

(7) Diagnosability Analysis, Fault Detection, and Identification Analysis: Optimized redundancy allocation and evaluation of the amount and quality of the level of observers (sensors) of a system for diagnosing faults.

This way of developing a system poses several advantages when compared to classic modelling,

- Assist in system model construction from the earlier phase.
- Supports a tight integration between the design and the safety teams.
- Automates (some of) the activities related both to the verification and to the safety analysis of systems in a uniform environment.
- Compatible to incrementally develop based on iterative releases of the system model at different levels of detail.

2.2.3 Open-source MBSA tools comparison

Open-source MBSA tools are compared against their ability to address the limitations of the generic approach. The selected tool should be able to accommodate any design configuration, and all kinds of system architectures and automate safety analysis for different top events. There are some other criteria to find distinct approaches of MBSA. The first one is model construction. There are extended system models, standalone system models and generic models. Extended system models are models created by injecting fault data into nominal models, stand-alone models are models created solely for safety analysis while generic models are a hybrid of the other two. The extended system models are more advanced than the standalone models but the computational effort is significantly high. The second criterion is modelling logic. There are failure logic modelling and failure effect modelling. Failure logic modelling (FLM) describes component failure propagation by variation in output values depending on the input values and failure modes. Failure effect modelling (FEM) describes the connection between components using information flow like power flow between the electrical system components. Most of the MBSA approaches use the combination of both modelling logic. The third criterion is the system abstraction level. It shows the level of captured system behaviour like nominal, degraded, and failed. However, higher abstraction puts a toll on computational effort, making it the last criterion.

With the limitations of the generic approach and the deduced criteria, MBSA tools can be assessed to find a suitable one for this work. The comparison is presented in the table 2.1.

Altarica creates standalone safety assessment models. Components in a system are represented by a set of flow and state variables, transitions, events, and equations to express the behaviour. State variables describe the state of the component and the flow variables are used to exchange information between components. Altarica uses a combination of both modelling logic (FEM and FLM). Components exchange information about failure modes and flow properties. Altarica also supports bidirectional flows thus making it possible to model undirected connections. System analyst is a graphical interface of Altarica developed separately [67]. It can create mini-versions of an extended system model for Altarica.

The fully integrated Simulink version of HiP-HOPS uses safety models, which utilize the structure

Table 2.1: Open-source MBSA tools comparison

	Config. and Arch.	Auto. Saf Analysis	Model constr.	Model logic	Sys. abst. lvl	Comp. effort
Altarica	Any	Yes	standalone	FEM/FLM	4	Medium
System analyst	Any	Yes	extended	FEM/FLM	4	Medium
HiP-HOPS	Any	Yes	generic	FLM	5	High
Simulink-NuSMV	Any	Yes	generic	FLM	3	Low
SLIM	Any	Yes	standalone	FEM/FLM	1	Low
SAML	Any	Yes	generic	FEM/FLM	1	Low
SmartIflow	Any	Yes	generic	FEM/FLM	2	Medium

of existing design models. HIP-HOPS is based on FLM as the components only exchange information about signal deviations at input ports.

The Simulink-NuSMV model creates an extended system model injected with a fault model using FEM. This model is translated into the input language of the NuSMV model checker. Simulink gives a graphical representation of the NuSMV model. Computation tree logic is used for modelling. This logic combines both linear-time and branching-time operators. Linear logic operators describe events along a single computation path while branching logic operators quantify the paths that are possible from a given state. This modelling approach also does not have state variables to express state-dependent behaviour and makes use of directed connections.

System Level Integrated Modeling Language (SLIM) framework is based on NuSMV and computation tree logic is used for requirement specification. SAML (Safety Analysis and Modeling Language) is another MBSA tool worth mentioning.

SmartIflow creates component-based models also using computation tree logic. Each component communicates via ports and connections by manipulating and checking abstracted physical flows as well as by exchanging key-value pairs that are called properties. It makes use of a combination of FEM/FLM modelling logic and uses undirected connections.

It can be concluded that the Altarica language can create models that address the limitations of the generic approach and provides a good measure in other criteria while comparing it with other tools. Also, the System analyst GUI is an additional advantage.

2.3 General overview of Altarica

Altarica is an object-oriented modelling language dedicated to the safety analysis of complex system architectures in the avionics, automotive and nuclear industries. Object-oriented languages are programs that can model objects which contain data and code to interact with each another. Generally, data is in the form of attributes or properties of the elements in the model and code is in the form of procedures or methods which represents the behaviour. The main purpose of Altarica is to fill the gap between the system specifications, safety models and traditional safety analyses. Altarica is integrated into modeling environments like Cecilia OCAS by Dassault Aviation, Simfia v2 by Airbus-Apsys and Safety Designer by Dassault Systèmes. The Altarica language creates models that imitate the functioning of each element in a system and establishes the connection between them. Altarica models can provide clarity about how a system works, degraded system functions, how it may fail and the outcomes of the failure. Thus, it ensures that the considered system is safe to operate. Altarica model contains blocks, classes, object properties, connections, observers etc. Blocks are used to define the system and subsystem functionalities. Classes define each element in a system and provide a means to accommodate object properties. Major object properties defined are element states, events related to states, failure rates and repair rates. The element states are represented by state variables. The change of state occurs when an event occurs (defined as transitions), thus updating the values of the state variables. Connections between elements are established using assertion by means of flow variables. These variables circulate the state information between elements of a model, thus establishing remote interaction between elements at the far ends of the physical architecture. These remote interactions portray the consequences of failures of individual components into the system as a whole [68]–[72].

Altarica is reinforced using guarded transition systems which is a mathematical model of computation gathered under the generic term of finite-state machines or finite-state automata. Finite state machines can be simulated to be in one of the finite number of states at any given time. It can change from one state to another in response to some inputs.

2.4 Guarded transition systems

Guarded Transition System (GTS) is a states/events formalism dedicated to safety analyses. GTSs can be considered as a generalization of block diagrams, petri nets, and the Arnold–Nivat model of parallelism. Block diagrams give the concept of flow or network allowing the interactions between components of a system. Petri nets give the idea of representing states using variables and changes of states using events and transitions. Arnold–Nivat model of parallelism gives the composition to create hierarchical models. While Altarica is a convenient way to describe and structure guarded transition systems [73].

Mathematically, a guarded transition system is defined using a quintuple $\langle$V, E, T, A, $\iota\rangle$, where:

- V is a set of variables. It can take its value in as a certain set of constants (Booleans, integers, reals or a set of symbolic constants). V is the union of two subsets - State variables - S (defines component's operative states concerning safety, eg: working, repairing, and failed) and Flow variables - F (defines connections, eg: input and output of components).
- E is a set of events.
- T is a set of transitions that defines the event. A transition T is a triple $\langle$e, G, P$\rangle$, also denoted as a function e : G $\rightarrow$ P; where e is an event of E, G is called the guard of the transition, which is a boolean formula built on variables of V, and P is called the action of the transition that changes the value of state variables or a post-condition. The transitions can be enabled or disabled whenever necessary. It is called the firing process.
- A is an instruction, called assertion, that calculates the values of flow variables from the values of state variables.
- ι is a function that gives the initial/default value of state variables and the flow variables.

To understand the semantics of GTS in detail, the following sections consider a quadrotor depicted in Figure 2.1, made of,

- Batteries, B1 and B2 supplying power to Electronic Speed Controller ESCs E1, E2, E3 and E4

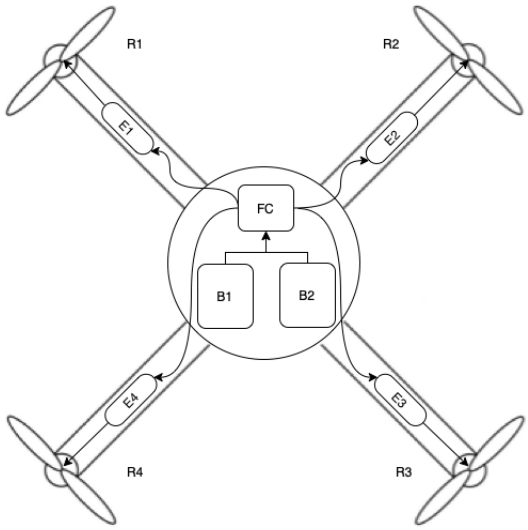


Figure 2.1: Simplified view of a quadrotor

which regulates the speed of rotors R1, R2, R3 and R4. Each rotor is connected to the corresponding propellers. One combination of ESC, rotor and propeller can be called a propulsor.

- Flight controller FC is connected to ESCs to control the rotors.

The goal of the system is to keep the quadrotor in hover flight by supplying power to all four rotors.

2.5 Definition of GTS using component description

Consider the component ESC of the quadrotor. The state diagram of ESC is depicted in Figure 2.2.

ESC can have two states – working or failed. The initial state of the ESC is working. In a failure event, a change of state occurs from working to failed. If the ESC is working, then the variable 'output' of the component will be equal to the variable 'input' else there is no output. The following part describes how the attributes of component ESC is converted into a GTS model.

- The set of variables, V is a disjoint union of state variables, S and flow variables, F. It can be represented as V = S $\uplus$ F.

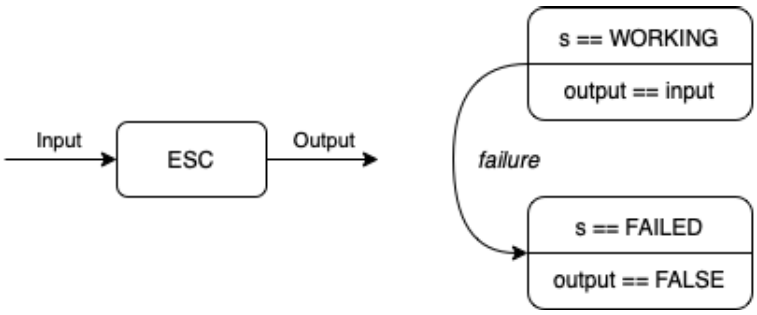


Figure 2.2: State diagram of ESC

- the state variables S can take in various operative states of the defined component, for example, working, failed, repaired, stopped etc. It can also take in boolean conditions. It can be represented as,

  S = { s, c } with

  dom(s) = { WORKING; FAILED }

  dom(c) = { TRUE; FALSE },

- the flow variables F defines connections between components, for example, input and output. It can be represented as,

  F = { input; output } with

  dom(input) = dom(output) = { TRUE; FALSE }

- The set of events E can contain the events like start, stop, and failure of components.

  E = { failure }

- Transition T defines the event. Here in event failure, state variable s change from working to failed.

  failure: s == WORKING $\rightarrow$ s := FAILED

- The assertion A is a block of instructions that contains two conditional assignments: Output only exists when state variable s is working, and input is connected. Thus, both assignments are connected with the Boolean operator 'and'.

  output := (s == WORKING) and (input)

```
domain ComponentState { WORKING, FAILED }

class ESC
    ComponentState s (init = WORKING);
    parameter Real lambda = 1.0e-5;
    Boolean input, output (reset = FALSE)
    event failure (delay = exponential(lambda));
transition
    failure: s == WORKING -> s := FAILED;
assertion
    output := (s == WORKING) and (input);
end
```

Figure 2.3: GTS of ESC

- Finally, the initial or default variable assignment ι is as follows: the initial state of state variable s is working and flow variables input and output are false.

  (s = WORKING, input = FALSE, output = FALSE)

Following Figure 2.3 shows an Altarica syntax representing the GTS of the ESC.

The state values are saved in a domain called ComponentState. There are two states, defined as WORKING and FAILED. Component ESC is modelled into a class named ESC. The variable 's' represents the state of the ESC. The attribute init gives the initial value to state 's' as WORKING. Class can take parameter values using parameter Real. Failure rate, lambda is defined so. Variables input and output represent the in and out connection of the ESC. They are the flow variables. Initial values of these variables are given by the attribute reset. The state variable 's' is changed using transition by initiating an event. Event failure defines the transition of the state from WORKING to FAILED. These variables are connected to state variables using assertion.

The transitions are subjected to firing which means an event has occurred. The event failure mentioned above is only fireable when the state variable s is WORKING. The firing changes the value of s to FAILED and then updates the values of flow variables.

Likewise, all components of the quadrotor can be created into distinct classes using the GTS model in Altarica.

```
block Quadrotor
    Battery B1, B2;
    ESC E1, E2, E3, E4;
    FlightController FC;
    Rotor R1, R2, R3, R4;
    Propeller P1, P2, P3, P4;
end
```

Figure 2.4: GTS of quadrotor system

2.6 Composition of GTS

2.6.1 Independency

The composition of two (or more) guarded transition systems is also a guarded transition system. Let $G_1 : \langle V_1, E_1, T_1, A_1, \iota_1 \rangle$ and $G_2 : \langle V_2, E_2, T_2, A_2, \iota_2 \rangle$ be two independent GTSs.

Then $G = G_1 + G_2$,

$$G = \langle V, E, T, A, \iota \rangle \text{ such that,}$$

$$V = V_1 \cup V_2$$

$$E = E_1 \cup E_2$$

$$T = T_1 \cup T_2$$

$$A = A_2; A_1$$

$$\iota = \iota_2 \cdot \iota_1$$

Larger models can thus be obtained by composing smaller models. For example, consider the block created from a quadrotor system in Figure 2.4. It consists of two batteries, four ESCs, one flight controller and four rotors. So, the GTS of a quadrotor is a composition of GTSs representing the battery, ESC, flight controller and rotors.

The block Quadrotor contains two instances of class battery named B1 and B2, four instances of ESC named E1, E2, E3 and E4, one instance of Flight controller FC and four instances of Rotor named R1, R2, R3 and R4. The resulting GTS contains all variables, events, transitions, assertions, and initial values from each instance of GTS. All named objects are distinguished by prefixing the name of the class. For example, variable 'input' of class 'Battery' instance B1 will be termed as B1.input.

```
assertion
    B1.input    := true;
    B2.input    := true;
    FC.input    := B1.output or B2.output;
    E1.input    := FC.output;
    E2.input    := FC.output;
    E3.input    := FC.output;
    E4.input    := FC.output;
    R1.input    := E1.output;
    R2.input    := E2.output;
    R3.input    := E3.output;
    R4.input    := E4.output;
    R1.output   := P1.input;
    R2.output   := P2.input;
    R3.output   := P3.input;
    R4.output   := P4.input;
end
```

Figure 2.5: System flow variables

2.6.2 Connection

Connections between each class in the block Quadrotor are defined by the flow variables using assertion in a GTS as in Figure 2.5. B1.input and B2.input are equated to true as they are the source of power for other components. The input of flight controller FC.input is connected to B1.output and B2.output with a boolean operator 'or' as the FC is powered by both batteries. One of the batteries should always work to power FC. FC regulates the ESCs to control the speed of rotors. Thus, the input of ESCs E1.input, E2.input, E3.input and E4.input is connected to FC.output and the input of rotors R1.input, R2.input, R3.input and R4.input are connected to the output of corresponding ESCs. Each rotor contains one propeller. Thus propeller inputs are connected to the corresponding rotor output. The dependency of both state and flow variables of each component can be seen in Figure 2.6.

The diagram shows combined status of transition and assertion in the block Quadrotor, meaning how the state variables and flow variables in the block Quadrotor are connected to each other. Each output of the component is only possible when the component state is working and the component input is true. Simply stating, a component's functionality is defined by its working condition and the input component connected to it.

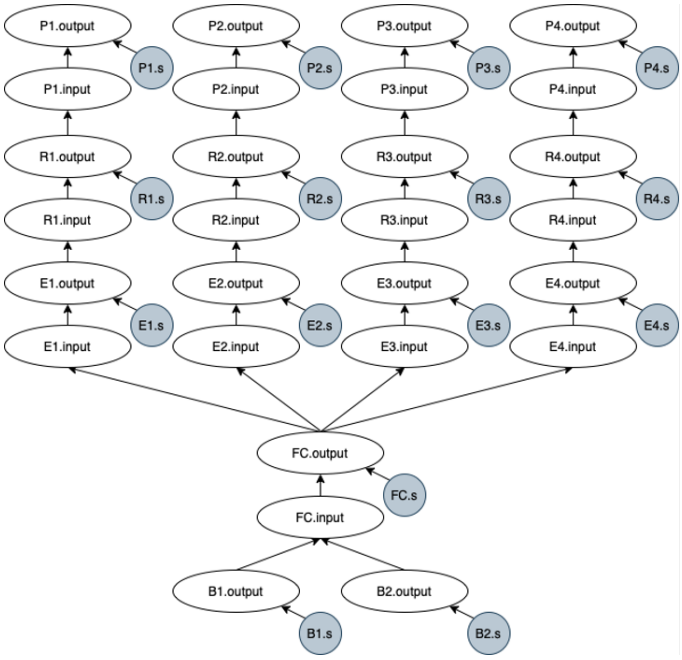


Figure 2.6: Dependency of variables

2.6.3 Synchronization

Transitions of GTS are asynchronous, which means a set of events cannot occur simultaneously. Two transitions cannot be fired once. So, a synchronization mechanism should be used to activate a set of events step by step. Generally, a transition is represented as e : G $\rightarrow$ P. It is considered as a triple, t = $\langle e, G, P \rangle$. Let σ be a variable assignment before the firing of transition. Then, a transition is fireable only if σ(G) = TRUE in a given state. Once a transition e : G $\rightarrow$ P is fired, a new variable assignment will be calculated from σ and all variables needs to be updated: First, state variables are updated by applying the instruction P to σ creating a new variable assignment; Second, flow variables are propagated using the assertion. For example, let's consider the event failure.

Event, E = {failure};

Transition T of event E is,

failure: s == WORKING $\rightarrow$ s := FAILED.

Here, σ(G) is s == WORKING. Only if σ(G) = TRUE in a given state, a failure event can occur converting the state from s == WORKING to s == FAILED else not.

Assertion A as shown in 2.3 defines that

output := (s == WORKING) and (input)

Output is only available when the state is WORKING, and input is TRUE. So, after the failure event occurs or the firing of the transition, the state is no longer WORKING and is FAILED. Thus after updating the state variables, Output is no longer available, and the component is completely FAILED.

2.7 Summary

It can be concluded that the Altarica language can create models that address the limitations of the generic approach. Therefore, the safety assessment framework will be developed with Altarica language and System Analyst GUI. The chapter also presents the Altarica language technically by creating a simple quadrotor system.

Chapter 3

Developed MBSA Framework

This section presents the developed MBSA framework describing from the system modelling methods to the safety analysis. It elaborates on the MBSA processes to create system models of multirotor configurations. It presents case studies to validate the MBSA framework.

3.1 MBSA Framework

As specified before, MBSA framework is developed using OpenAltarica, System Analyst and Python to model the multirotor systems. It is illustrated in Figure 3.1. The framework follows three processes,

- System modelling
- Flattening
- Assessment tools

3.1.1 System modelling methods

Three major modelling methods can be used to create multirotor system architecture in an MBSA framework.

- Code-based modelling

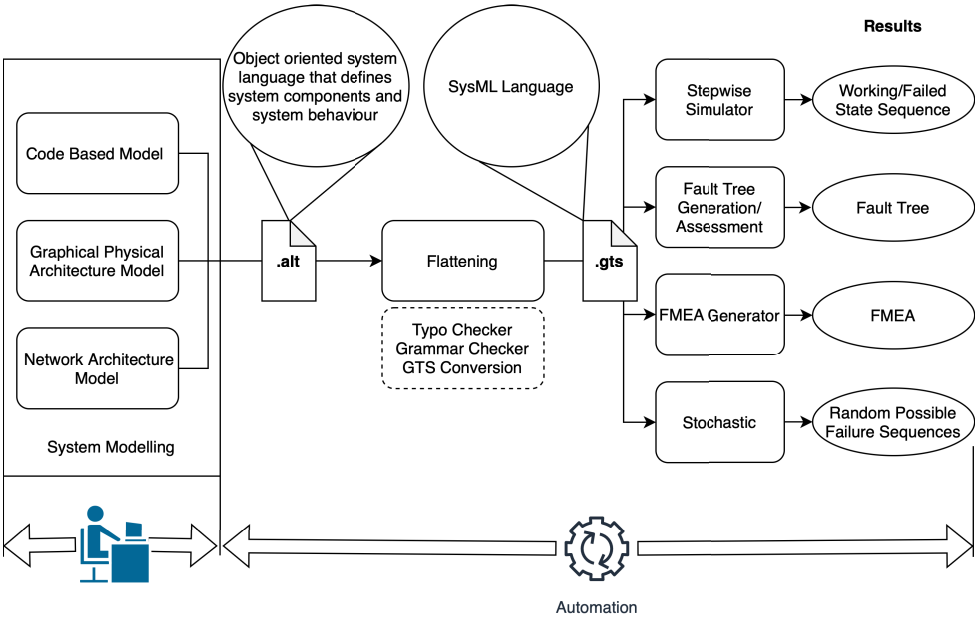


Figure 3.1: Developed MBSA framework

- Network modelling
- Graphical modelling

Code based modelling

Code based modelling is based on the OpenAltarica code framework. Here the engineers create the system using altarica code as shown in chapter 4. System components are defined using classes and their attributes are defined using class parameters, the states of each component are defined using state variables. The relationship between components is defined using flow variables. State variables and flow variables are linked in such a way that flow variables respond to the state variables whenever the component state varies. For ex, if a component state changes from WORKING to FAILED, the flow variables OUTPUT is no longer available from that component. Without the OUTPUT from the component, the next component's state stays not WORKING. Thus, the system is made into a flow of components where they respond to each other's state.

This altarica code can thus be fed into the MBSA framework to auto-generate conventional reliability assessment methods. The limitation of this method is that code should be updated for every

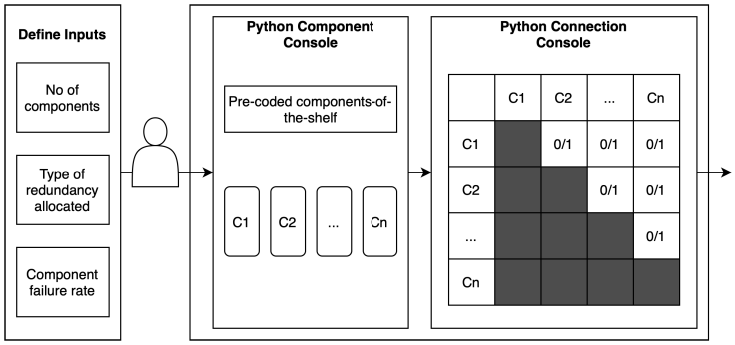


Figure 3.2: Network architecture

latest change in the design stages.

Network architecture modelling

Network modelling is developed from the idea that a system and its components behave like a network. For this purpose, altarica code is automated using python. Each component is pre-coded into a python capsule which behaves like a component-of-the-shelf. The required amount of components and their attributes are manual inputs depending on the size of the system. The system model is defined in a tabular format that represents the connection between components. It is controlled using binary codes, 0 for no connection and 1 for connection. The python then automates the altarica code. This is shown in Figure 3.2.

This altarica code can also be fed into the MBSA framework to auto-generate conventional reliability assessment methods. The limitation of this method is the complexity of coding from the size of the system and the number of components.

Graphical modelling

Graphical modelling is developed by System Analyst which can create altarica code using a graphical user interface. To create a system architecture, each component can be defined using functional blocks and the connections between components can be defined using the input and output of components.

This modelling method can auto-generate conventional reliability assessment methods.

3.1.2 Flattening

Flattening is a model-checking process to convert the Altarica model/code written in object-oriented language to a universal language mentioned earlier called the Guarded Transition System (GTS). Before conversion, it also applies a typo-checking tool and a grammar checker. This makes it possible to detect typo and syntax errors and retrieve the error locations in the model.

3.1.3 Assessment tools

The assessment tools in Altarica can generate safety artifacts from GTS code based on the system model. It includes stepwise simulation, fault tree assessment, FMEA generator and stochastic simulation.

- Stepper: The stepper converts state and flow variables from the GTS model and creates the system architecture in a tree format. It shows the operation sequence of the system as well as the state of each component. It also can show state transitions, and active and lost input and output connections of components. It also gives means to manually enable and disable components to foresee the system behaviour. This helps in validating the system architecture logically.

- Fault tree Assessment: GTS model can be compiled into a fault tree. Minimum cut sets and top event probability can be calculated.

- FMEA Generator: GTS model can generate an FMEA table depending on the list of failure modes entered.

- Stochastic simulation: Stochastic models are used to estimate the probability of various outcomes while allowing for randomness in one or more inputs over time. The models result in probability distributions, which are mathematical functions that show the likelihood of different outcomes. The Monte Carlo simulation is one example of a stochastic model. it can simulate how a portfolio may perform based on the probability distributions of individual stock returns. It can thus produce the random/every possible sequence of events that can occur in a system.

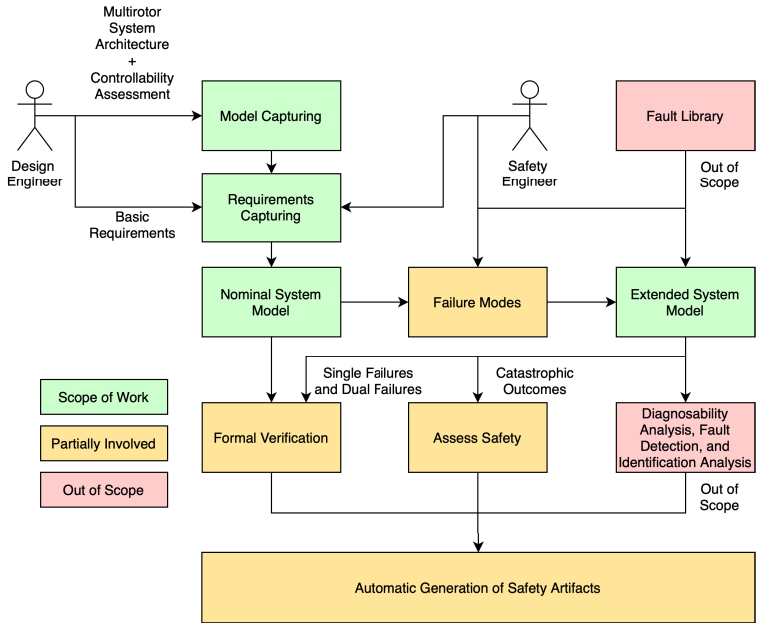


Figure 3.3: MBSA process with the scope of study

3.2 Model based safety assessment of multirotor configurations

As the framework is presented, this section applies the MBSA process to initialize the multirotor system model. It will be limited to the scope of the current study to emphasize the acceptable complexity of the model. Figure 3.3 illustrates the MBSA process.

- Model capturing: For capturing the multirotor model, the system architecture model and controllability assessment model of corresponding propulsor configurations should be created. System behaviour and component properties should also be collected.
- Requirement capturing: Design requirements should be collected.
- Failure Mode capturing: Failure modes of each component should be identified.
- Failure injection: Failure modes of the system should be injected into the design model and create an extended system model.
- Formal assessment of system model:

- Formal Verification: Check the model against requirements, under the hypothesis of nominal behaviour. Redundancies should be created to withstand single failures and dual failures.
- Assess Safety: Check the model against a set of requirements, under the hypothesis that the component may fail. In case of a complete failure, the flight control system should be assumed to initiate an emergency landing to avoid a complete failure of the multirotor. In case of a susceptible failure, the flight control system will continue operation with degraded functionalities.

- Automatic generation of safety artifacts: Automatically generate the fault trees and FMEA tables from the extended system model of the multirotor.

- Diagnosability Analysis, Fault Detection, and Identification Analysis: This part is not in the scope of the current work. It requires a mature design of the system.

3.2.1 Model capturing

Here, the work creates a formal model for multirotor configuartions. Some examples of such configurations - quadrotor, coaxial quadrotor, two configurations of hexarotor (PNPNPN and PPN-NPN) and octarotor configurations are shown in the figure 3.4. It is discussed in two steps. The first step is to create the system architecture and the second is to assess the controllability of each multirotor configuration when one or more propulsors fail.

System architecture of multirotor configurations

A system architecture is a conceptual model of a system that defines the physical structure, system functions, driving parameters and system behaviour. The system architecture of a multirotor is shown in figure 3.5.

A multirotor consists of various systems. Each system has various components and distinct functionalities. It can be listed as follows,

- Power System: The power system represents the power storage and provides the required

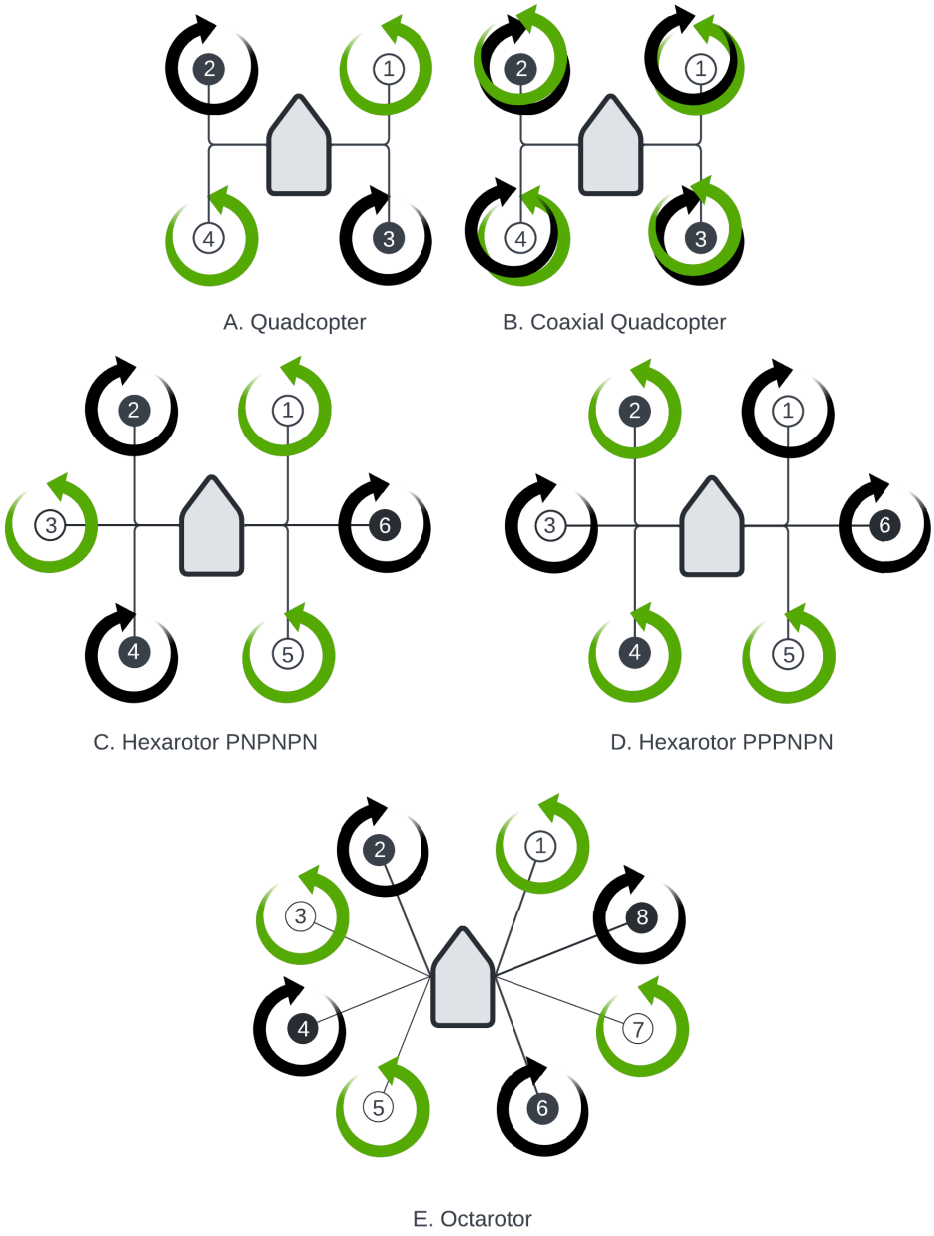


Figure 3.4: Multirotor configurations

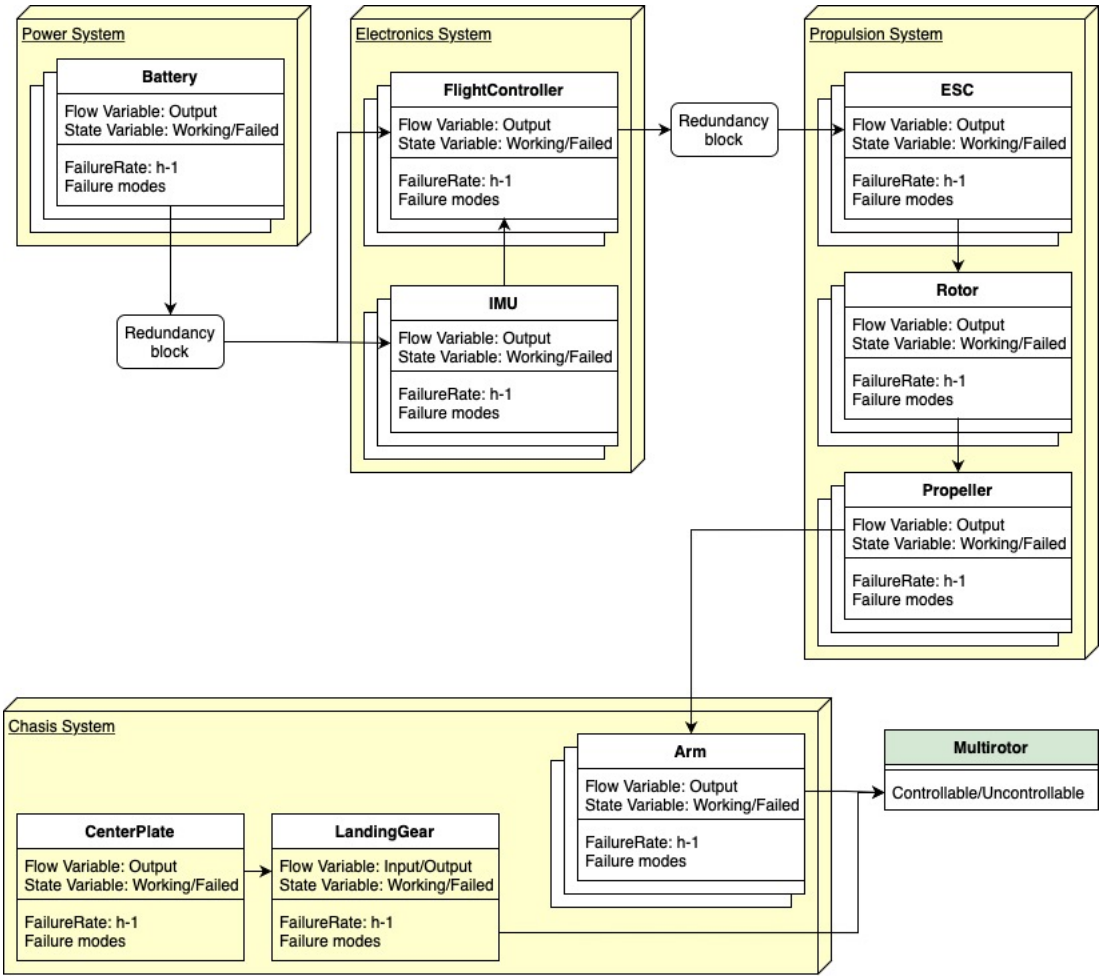


Figure 3.5: System model of a multirotor

power to system components. It contains batteries and supplies power to the electronics system and propulsion system. Depending on the requirements, the system can contain redundant batteries.

- Electronics System: The electronics system can be sub-categorized as the flight control system and the communication system. It contains the major electronic circuits like flight controllers and onboard sensors called Inertial measurement units (IMU). The flight controller collects information from the sensors and controls the propulsion system. Depending on the requirements, the system can contain redundant flight controllers and sensors.

- Propulsion system: The propulsion system represents all the propulsors and provides flight power. A propulsor is a single combination of an electronic speed controller (ESC), rotor and propeller. Multirotors are classified into quadrotor (4), coaxial quadrotor (4 pairs of 2 rotors), hexarotor (6) and octarotor (8) depending on the number of propulsor configurations. ESCs control the speed of the rotor attached to a propeller depending on signals from the flight controller.

- Chassis system: The chassis system is the skeleton of the multirotor and provides structural support. It contains the number of arms depending on the configuration, a center plate to hold the entire body of the multirotor and landing gears for safe takeoff and landing. It can have more components, but this work intends to limit the complexity of the system.

Controllability assessment of multirotor configurations

Controllability assessment is to make the multirotor susceptible to the challenges facing the flight control system of the multirotor. A major challenge in multirotor flight control design is the angular momentum generated from the torque of the rotor. This is handled by contra-rotating (pairing it with a rotor with the counter-rotation). The total number of rotors rotating clockwise and counterclockwise should always be the same for a multirotor to maintain complete stability. But this is not possible when a propulsor failure occurs - Failure of ESC, failure of the rotor, failure of propeller or loss of an arm can all have the same effect. Consequently, the multirotor will lose control of the control axes. However, each multirotor configuration has different behaviour when

propulsors fail.

Each multirotor configuration has different behaviour when propulsors fail. Assuming the overall weight can be lifted by the rest of the propulsors, various controllable and uncontrollable cases in single and dual failures of each configuration should be derived to evaluate the reliability of the propulsion system and should reflect in the safety analysis. The current study has collected and identified controllable and uncontrollable cases of each configuration [24], [74]. The controllable cases will enable the system model to identify the recoverable actions to withstand single or dual failures of propulsors.

Analyzing each configuration, all configurations are insusceptible to single failures except quadrotor. Investigating through dual failures, coaxial quadrotors have 12 controllable cases, hexarotors (PNPNPN) have 9 controllable cases, hexarotors (PPNNPN) have 3 controllable cases and octarotors are insusceptible to dual failures. And all configurations are susceptible to triple failures except octarotors with 8 controllable cases. This is shown in the table 3.1.

The MBSA framework should enable this data to be reflected in the safety analysis. The system model should incorporate a controllability model from controllable and uncontrollable cases. For this matter, Figure 3.6 shows the controllability assessment model developed for this study.

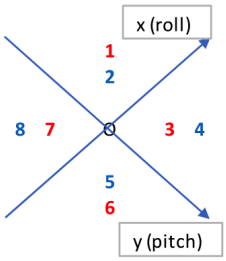
In the controllability model, propulsors are connected to a state monitor that checks the operating status of components. As the multirotors are insusceptible to single propulsor failure except quadrotor, a failure sequence is necessary. Controllable and uncontrollable cases of each configuration from controllability assessment should be fed to a checker. And the checker can compare it with the generated failure sequence to it. This is done using boolean algebra and observers from Altarica. An observer can ensure the status of a component's state variable (WORKING or FAILED). If the identified failure sequence is controllable, safety analyses can reflect the precautionary measures. If contrary, safety analyses can suggest emergency measures.

3.2.2 Requirement capturing

Requirement capturing is about identifying the major constraints of the design and defining its properties. The selected design model represents a multirotor that operates over heavily populated

Co-axial quadrotor (X-config)

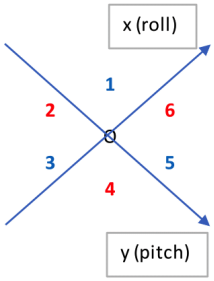
Double failures								
Propulsor	1	2	3	4	5	6	7	8
1		X	X				X	
2				X				X
3				X		X		
4					X			
5						X		X
6							X	
7								X
8								



Clockwise
Counter clockwise

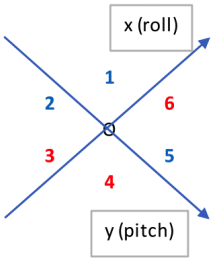
Hexarotor PNPNPN

Double failures						
Propulsor	1	2	3	4	5	6
1			C	C	C	
2				C	C	C
3					C	C
4						C
5						
6						



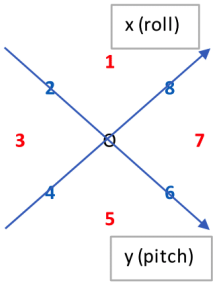
Hexarotor PPNNPN

Double failures						
Propulsor	1	2	3	4	5	6
1			C	C		
2				C		
3						
4						
5						
6						



Octarotor

Double failures								
Propulsor	1	2	3	4	5	6	7	8
1		C	C	C	C	C	C	C
2			C	C	C	C	C	C
3				C	C	C	C	C
4					C	C	C	C
5						C	C	C
6							C	C
7								C
8								



Octarotor

Triple failures								
Propulsor	1	2	3	4	5	6	7	8
1			5		7		3	
2				6		8		4
3					7			
4						8		
5								
6								
7								
8								

Table 3.1: Controllable cases

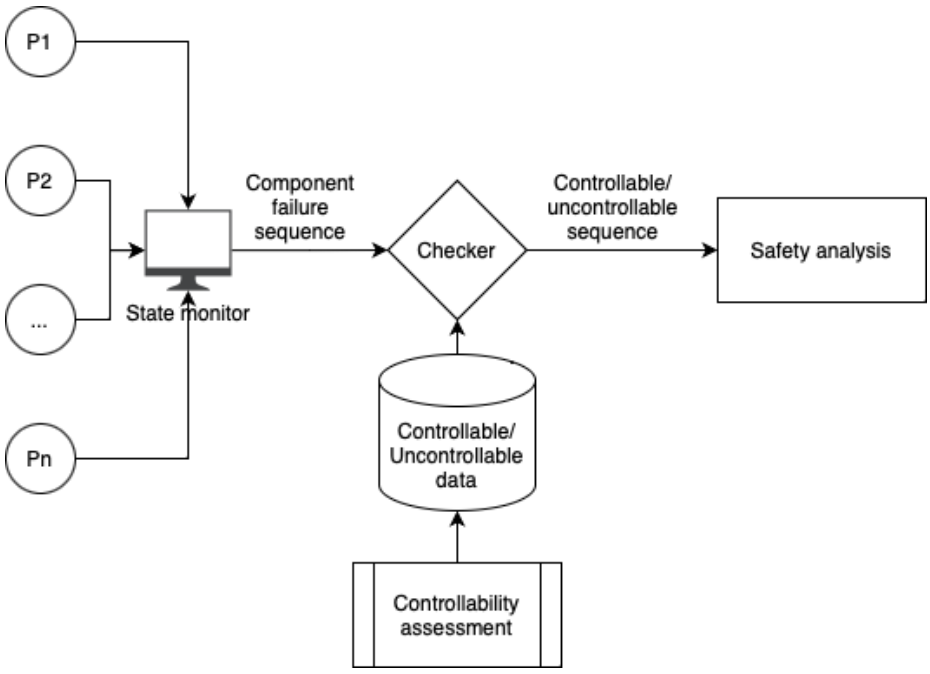


Figure 3.6: Controllability assessment model

areas with heavy loads. The requirement is to follow the first rule of safety-critical design standards - A catastrophic failure condition must not result from a single failure and must be extremely improbable [[18], Pt. VTOL.2510(a)]. A catastrophic failure condition is defined as: High impact crash is imminent and unavoidable with the vehicle's destruction due to a complete system failure. Severe injuries or the death of people on the ground is possible. Infrastructures can be damaged heavily. A catastrophic failure condition shall have a probability of occurrence less than or equal to 10^{-7} [18], [19]. Ability to enable emergency landing procedures should be maintained under said failure condition. Hence, the procedure requires sufficient control of all control axis.

3.2.3 Failure mode capturing

For simplicity, the work is limited to failure modes that cause complete system shutdowns as shown in Table 3.2

Table 3.2: Failure modes

Failure mode	Root causes
Battery failure	Low capacity and voltage
Sensor failure	Low calibration, low power, circuit short
Flight controller failure	Low power, circuit short
ESC failure	Low power, circuit short
Rotor failure	Low power, circuit short
Propeller failure	Collision, fracture
Arm failure	Collision, fracture
Center plate failure	Collision, fracture
Landing Gear failure	Collision, fracture

3.2.4 Failure injection

After identifying the failure modes of the system components, failure rates of each failure mode should be injected into the design model and create an extended system model. This section should be further developed with failure modes representing partial failures of components. Thus, all failure modes of components can be injected into the extended system model as the design of the system matures.

3.2.5 Formal assessment of system model

- Formal Verification: Formal verification can be conducted on both the nominal model and the extended model. The nominal model should be verified that the system model is functional. The extended model should be verified under the presumption that the system model is functional even after a failure. Redundancies should be created to withstand single failures. To avoid single failures, a simple single parallel redundancy can be applied, for example. Battery. To avoid multiple failures and to protect critical components like flight controllers, a majority voting monitoring redundancy can be applied.

- Assess Safety: This step is to identify cases assuming that a critical component may have failed or an uncontrollable case is identified. In case of such an event, the flight control system should be assumed to initiate an emergency landing. This can only be possible by maintaining the major control axes of the multirotor.

3.2.6 Automatic generation of safety artifacts

MBSA framework can generate failure sequences, minimal cut sets, fault trees and FMEA tables from the extended system model.

3.2.7 Diagnosability analysis, Fault detection and Identification analysis

Safety critical systems should be fault tolerant in presence of faults. It should be able to operate fully functional or maintain major functions. Fault tolerance typically requires proper architectures (Redundant), mechanisms to monitor, detect and isolate faults, and recover mechanisms. Diagnosability analysis should verify the amount and effectiveness of system architectures and evaluate the amount and quality of the level of observers (sensors) of a system for diagnosing faults. Fault detection and Identification analysis should assess the ability to detect the fault and apply recovery mechanisms at specified mission time. This part is out of the scope of the current study and should be applied as the design stage matures.

3.3 Case studies

This section presents case studies to validate the developed MBSA framework against the limitations of generic approach. Further, it shows the overall reliability results of various propulsor configurations.

3.3.1 Hexarotor system model

The hexarotor system model is developed from functional logic. Thus, it contains a component structure to achieve maximum reliability and a controllability assessment block to avoid uncontrollable cases. The component structure is adopted with redundancy techniques. The component structure is listed below,

- Battery: This is the sole power source of the system. A simple active redundancy is applied to avoid a single failure.

- Flight controller: It is the decision-making component of the system. A triple redundancy is applied with the majority voting algorithm.
- IMU: IMU is a collection of sensors that support the flight controller with necessary data. Hence, each flight controller is equipped with its own data source.
- ESC: Each ESC is connected to all rotors, but only supports one at a time. In case of an ESC failure, another ESC in a different propulsor can take over the rotor.
- Rotor: Hypothesis - All configurations except the quadrotor can handle a single rotor failure. But all dual failures cannot be handled using redundancy. Hence the controllability block assesses the case and checks whether the resulting scenario is controllable or not.
- Propeller: Hypothesis - All configurations except the quadrotor can handle a single propeller failure. But dual failures cannot be handled using redundancy. Hence the controllability block assesses the case and checks whether the resulting scenario is controllable or not.
- Arm: Hypothesis - All configurations except the quadrotor can handle a single arm failure or loss of the arm. But dual failures cannot be handled using redundancy. Hence the controllability block assesses the case and checks whether the resulting scenario is controllable or not.
- Central plate: Should be reinforced with mechanically strong elements.
- Landing gear: Should be reinforced with mechanically strong elements.

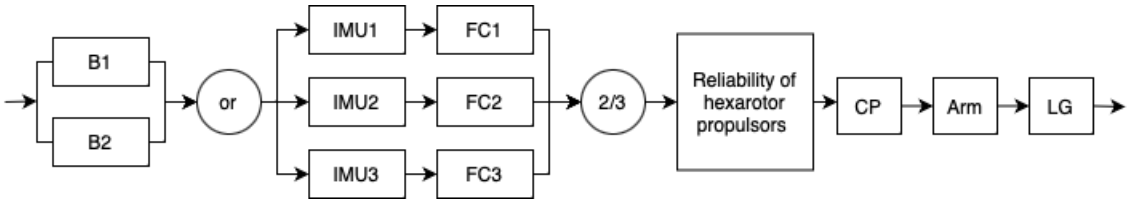


Figure 3.7: RBD of Hexarotor configuration

In order to validate this approach, an RBD shown in Figure 3.7 is developed adopting a similar scenario for hexarotor and is compared with that of a system model for MBSA framework. The

Component	Failure rate (per hour)
Battery	$1\ 10^{-6}$
Inertial measurement unit (IMU)	$3\ 10^{-6}$
Flight controller	$5\ 10^{-5}$
Electronic speed controller (ESC)	$1\ 10^{-5}$
Rotor	$1\ 10^{-6}$
Propeller	$1\ 10^{-8}$
Center plate	$1\ 10^{-6}$
Arm	$1\ 10^{-6}$
Landing gear	$1\ 10^{-6}$

Table 3.3: Failure rates

failure rates are summarized in Table 3.3 based on the orders of magnitude of modern high-end transport category aircraft equipment. Figure 3.8 represents the full system model equipped with said techniques of hexarotor in System Analyst.

The overall probability of failure generated from MBSA framework for the top event - complete loss of flight control shows a 10% error with that of RBD. Thus, the methodology used to develop the MBSA framework can be considered valid.

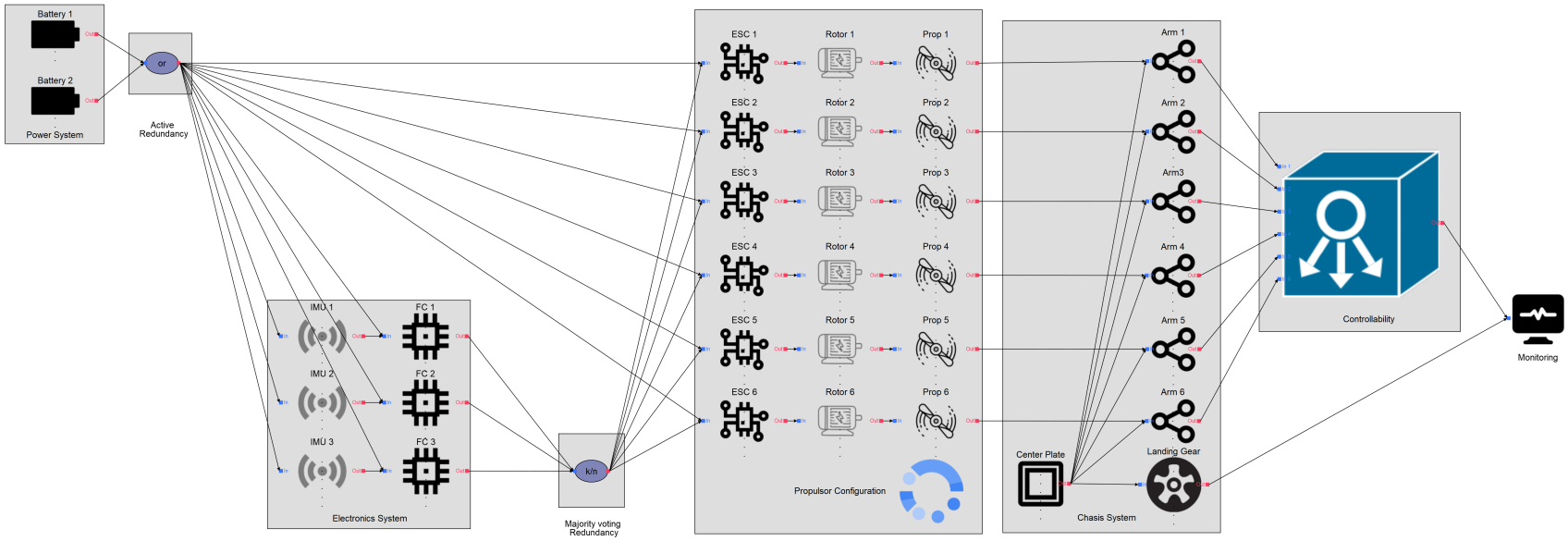


Figure 3.8: Full system model of a hexarotor's MBSA

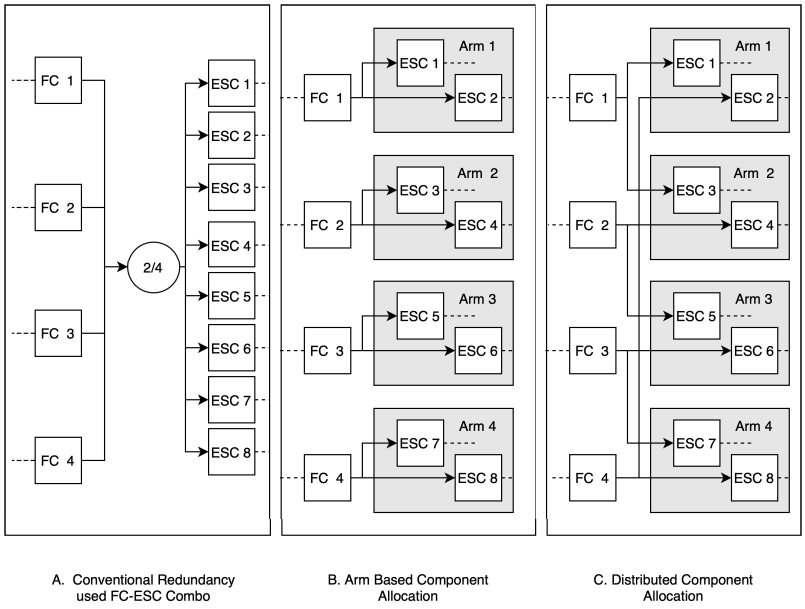


Figure 3.9: Component allocation for FC-ESC combo

3.3.2 Component allocations

The MBSA framework can solve the design limitation in the system architecture of multirotors from the single connection between the electronics and the propulsor system by introducing component allocations. Using the developed MBSA framework, multirotors can adopt two kinds of component allocation, especially in the connection between the flight controller and the ESC, namely arm based and distributed. This can avoid two scenarios. First, failure of the electronics system causing a complete shutdown of the propulsor system and second, failure of the same-side rotating propulsors. In arm-based allocation, each flight controller is allocated to a certain arm. This kind is suited for coaxial multirotor as it have to operate two sets of propulsors. It is assumed that the flight controllers will operate in a swarm behaviour for providing collaborative control. While distributed component allocation is derived to avoid common cause failures of same-side rotating propulsors. Most of the uncontrollable cases occur when two clockwise or counterclockwise rotors fail. It can also occur in a complete loss of the arm. This can be avoided by a flight controller that controls different sets of propulsors in multiple arms with one clockwise and the other counterclockwise. Subsequently, failure of this flight controller will not cause the loss of complete flight control.

Also, both ways can reduce the frequency of failures in flight controllers significantly as they only support two arms. These allocations can make the electronics systems and propulsor systems more reliable.

A simple case study was conducted in the coaxial quadrotor to show the potential of such architecture as shown in Figure 3.9. The results are shown in Figure 3.4 with derating failure rate. Distributed control allocation has high reliability while 3 out of 4 redundancy and arm-based control allocation show similar reliability. However, the complexity and cost of applying a majority voting algorithm can be avoided by arm-based control allocation.

Hence, the MBSA framework expands design freedom and allows a designer to try and judge different design scenarios in terms of reliability.

Failure rate derating	3 out of 4	Arm based	Distributed
x1 (transport aircraft)	5.10E-08	5.10E-08	5.20E-09
x10	5.10E-07	5.10E-07	5.20E-08
x100	5.10E-06	5.10E-06	5.20E-07
x1000 (toy industry)	5.10E-05	5.10E-05	5.20E-06

Table 3.4: Probability of failure comparison after component allocation with a mission time of 22 min

3.3.3 Reliability results

This section compares the reliability of various multirotor configurations - quadrotor, coaxial quadrotor, hexarotor (PNPNPN and PPNNPN) and octarotor.

As mentioned before, the overall probability of failure requirement to be accepted by the standards is 10^{-7}. This value can be generated from the auto-generated fault trees from the MBSA framework presented in the Appendix A of the discussed multirotor configurations and is shown in Figure 3.10. All multirotors are insusceptible to single failures except quadrotors. The overall probability of failure of the quadrotor also reveals that it does not fulfil the safety requirements. The coaxial quadrotor and hexarotor with PNPNPN configuration are above the specified reliability limit and are acceptable for safety-critical operations. However, quadrotor with redundant electronics is below the reliability limit as they are still susceptible to single propulsor failures. Hexarotor with PPNNPN configuration is above the specified reliability limit but has limited controllable cases in dual failure

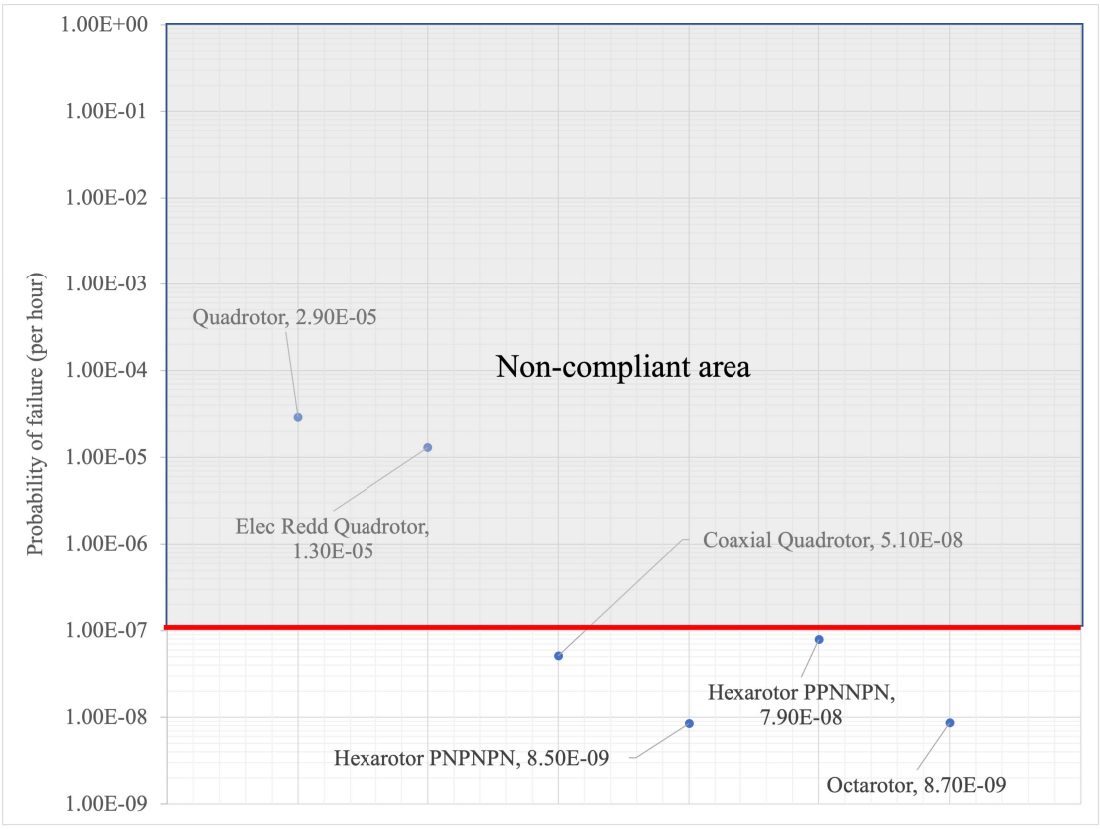


Figure 3.10: Reliability values of various multirotor configurations

sequences. However, octorotor is the best performer due to the higher number of controllable cases and it withstands single, dual and triple propulsor failures.

3.4 Summary

The chapter presents the developed MBSA framework and also elaborates on the processes used to create system models of multirotor configurations. The developed MBSA framework provides much wider design freedom in modelling the physical architecture of multirotor using various system modelling methods. The reliability of propulsors is evaluated by assessing the controllability of each configuration separately by identifying controllable/uncontrollable cases. As the safety analysis is automated, different top failure events can also be assigned to create safety data for numerous

scenarios. Case studies are presented to validate the framework. It also shows the overall reliability results of various propulsor configurations and identifies a potential one for safety-critical applications.

Chapter 4

Conclusion

As there is a lack of a comprehensive UAV design approach with safety and reliability considerations in the open literature, this research made the first steps to add such a scenario in the design process by developing an MBSA framework.

A generic approach is created by applying typical design for reliability modifications in the state-of-the-art conceptual design to identify the limitations in design configuration, architecture and safety analysis. The review of safety assessment models paved the way to develop an Altarica-System Analyst-Python-based framework. The developed MBSA framework can integrate system models from the physical architecture with networked connections and corresponding controllability assessments. The framework can also create innovative system architectures and automate fault tree generation for various multirotor configurations. The reliability analysis confirms that the multirotors developed using the framework are fault-tolerant and some configurations are potentially highly reliable.

Some aspects of the MBSA processes are omitted to reduce the system complexity and computation effort. In future works, the models can be further enhanced with the addition of a component fault library, additional failure modes, diagnosability analysis, fault detection and identification analysis. Fault libraries and failure modes can help in foreseeing uncontrollable cases, while diagnosability analysis, fault detection and identification analysis can integrate detect, isolate and recover mechanisms, and optimize redundancy effectively. Additionally, the framework should also be combined with multidisciplinary design optimization for sizing. Such design models can contribute to the

emergence of UAVs for safety-critical applications shortly.

Appendix A

Fault trees

Following pages show the fault trees generated using the MBSA framework for various multi-rotor configurations.

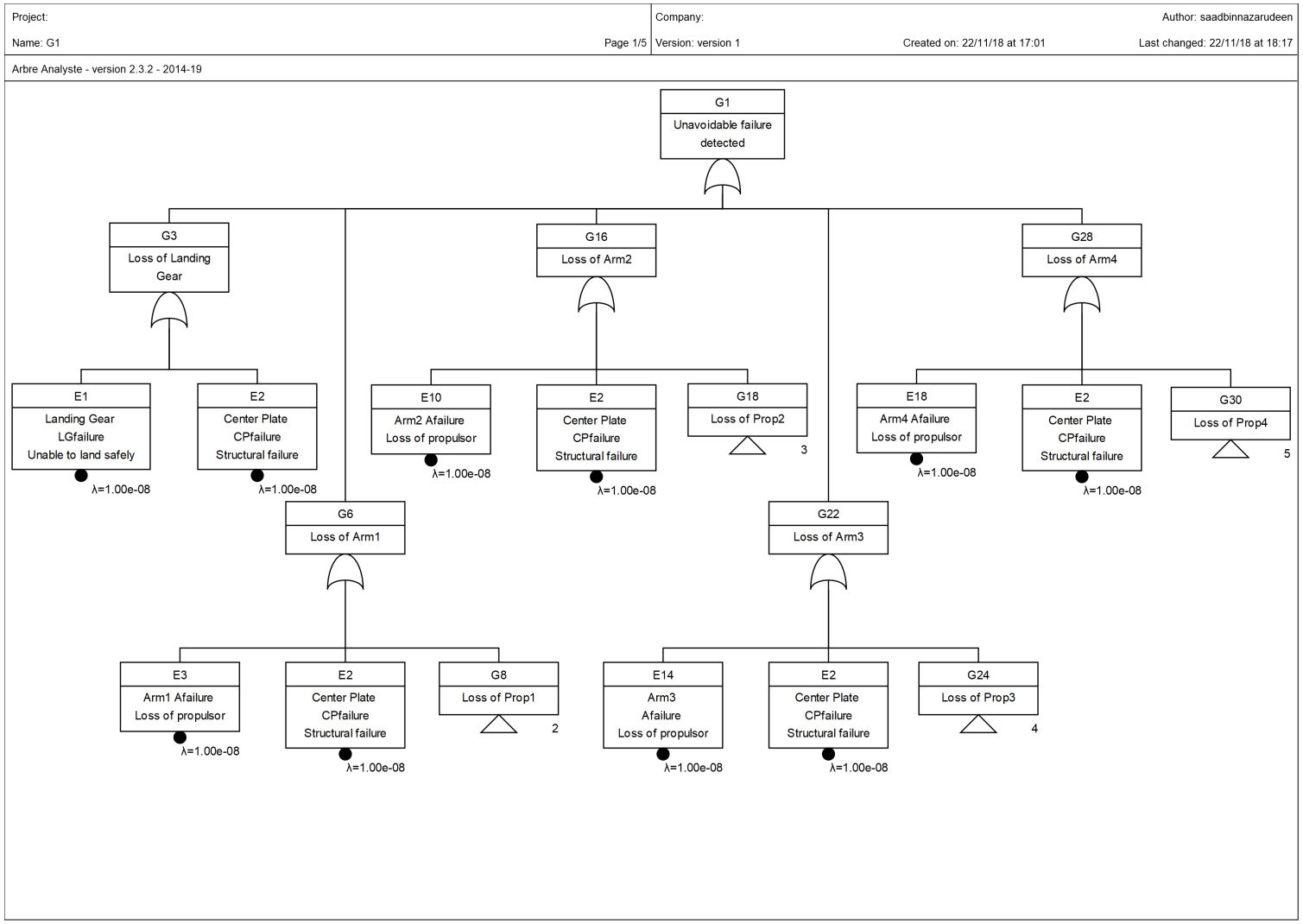


Figure A.1: Quadrotor main fault tree

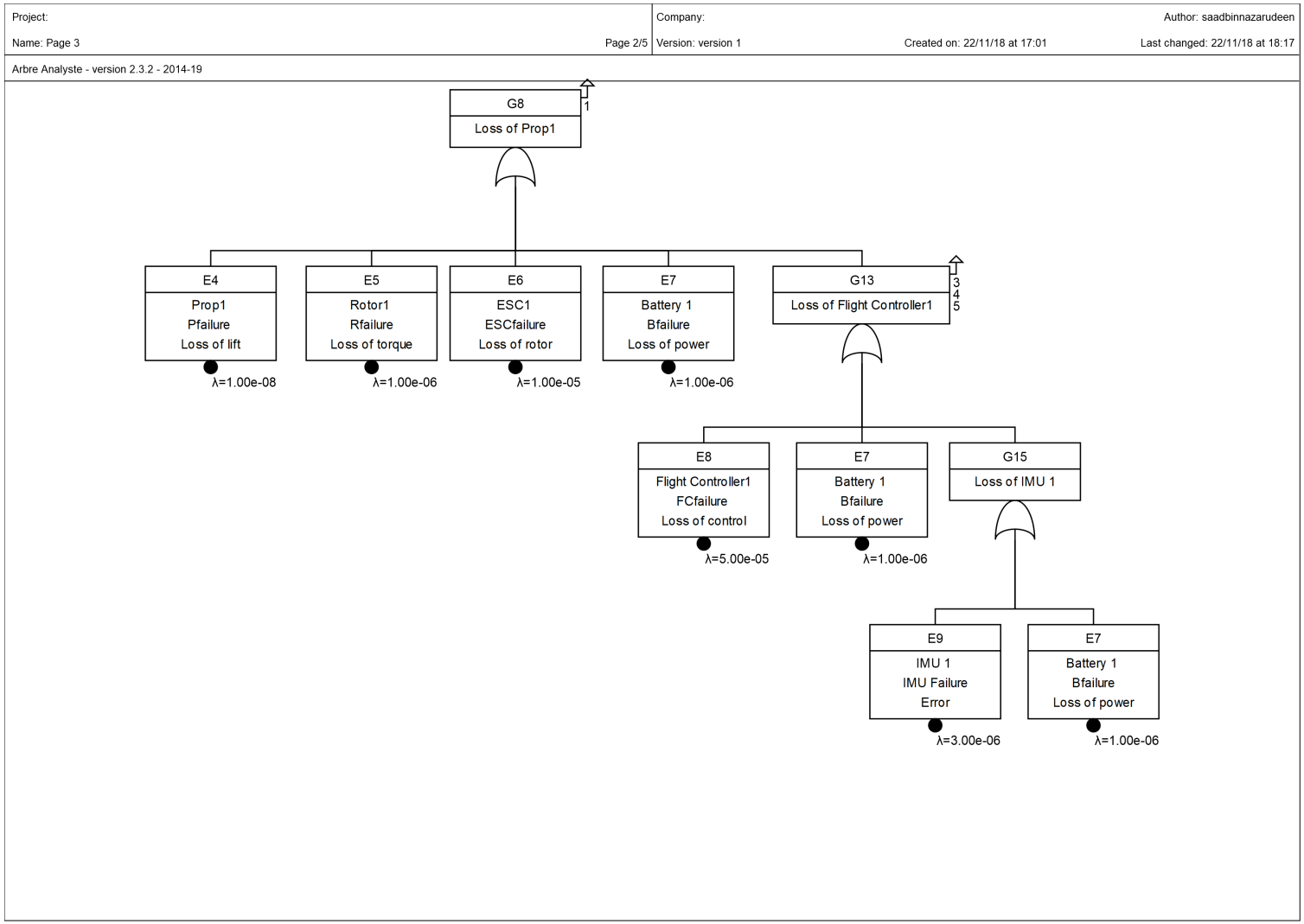


Figure A.2: Quadrotor fault tree extension G8 (similar to G18, G24, G30)

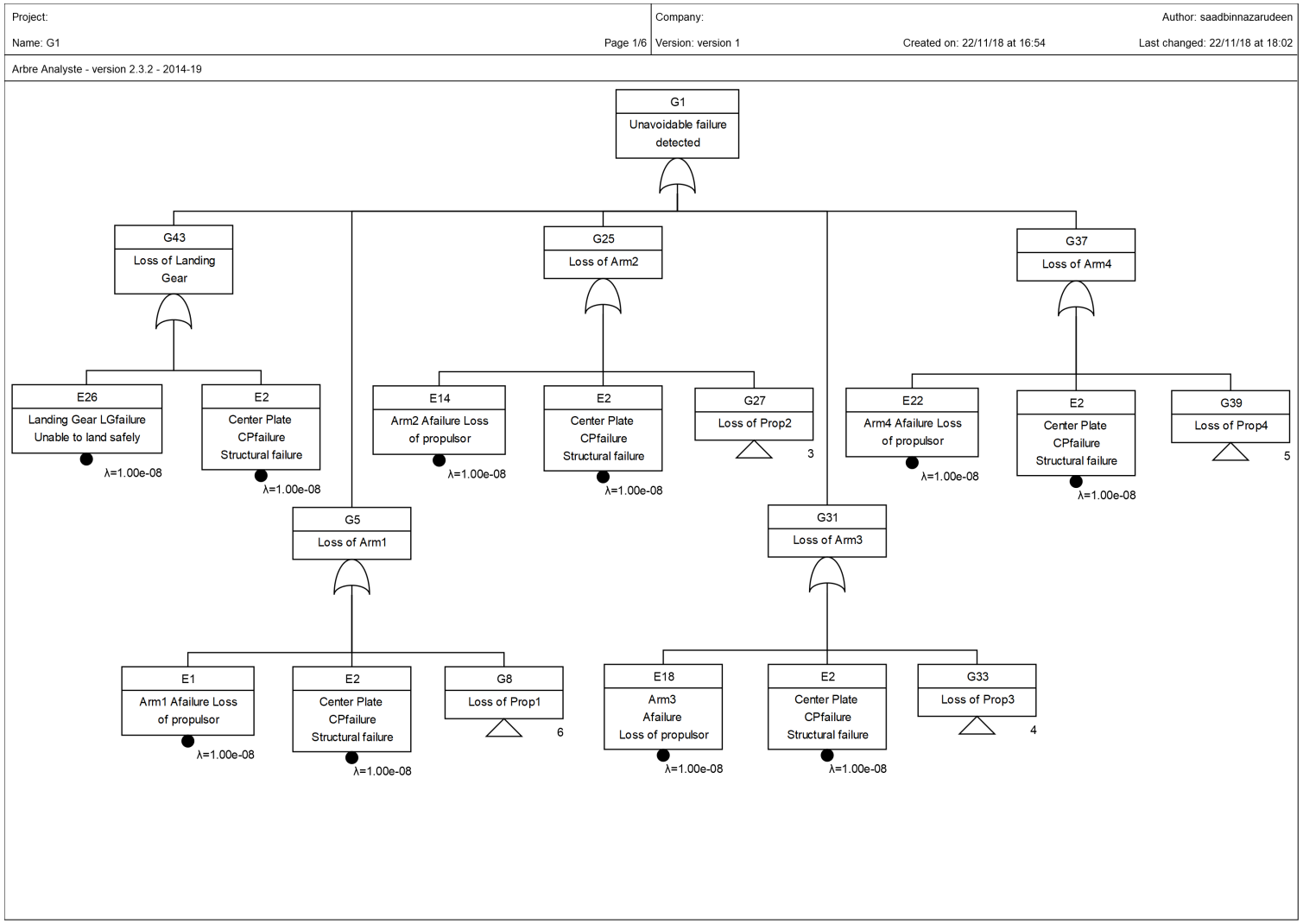


Figure A.3: Quadrotor with redundant electronics main fault tree

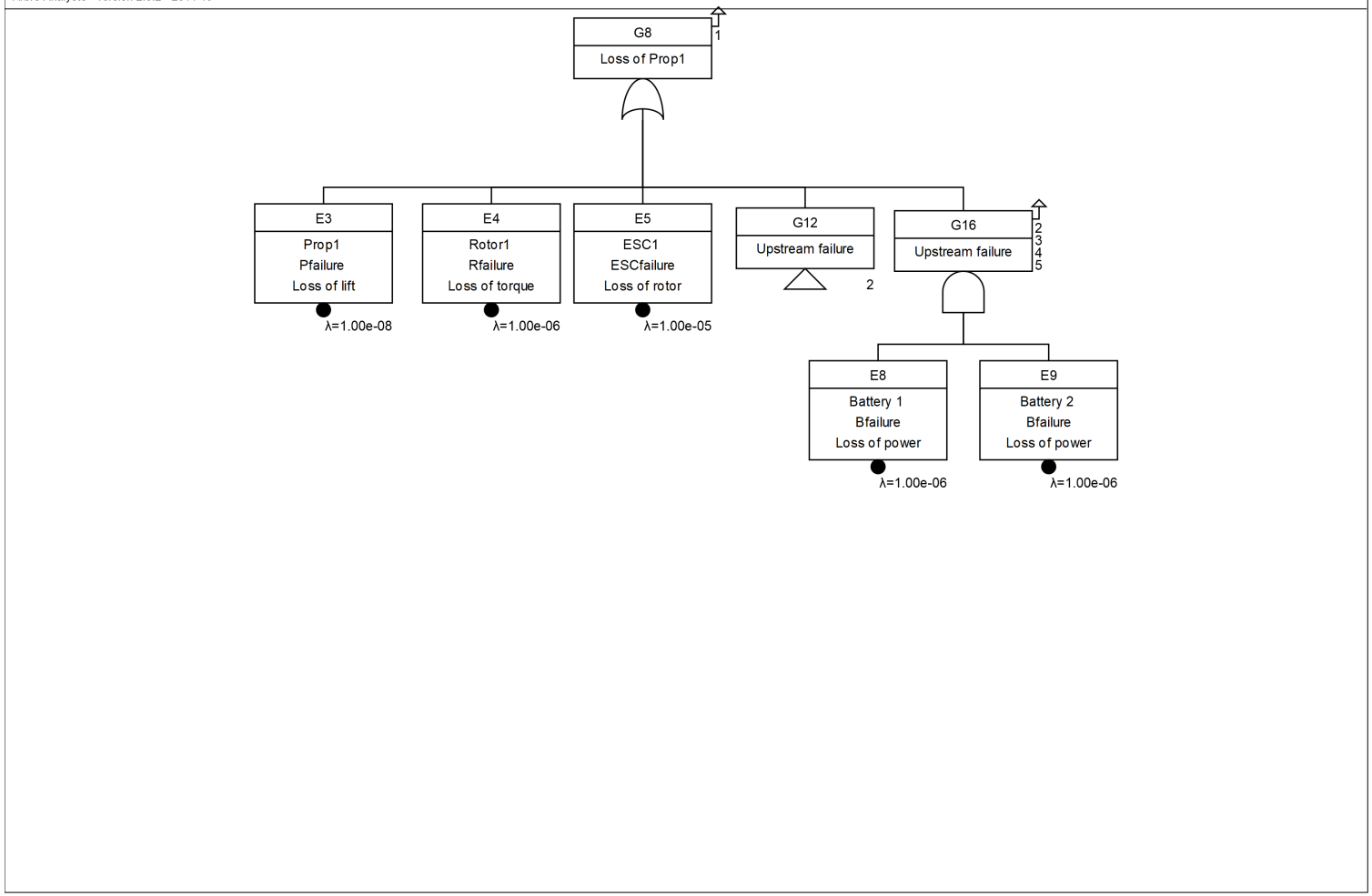


Figure A.4: Quadrotor with redundant electronics fault tree extension G8 (similar to G27, G33, G39)

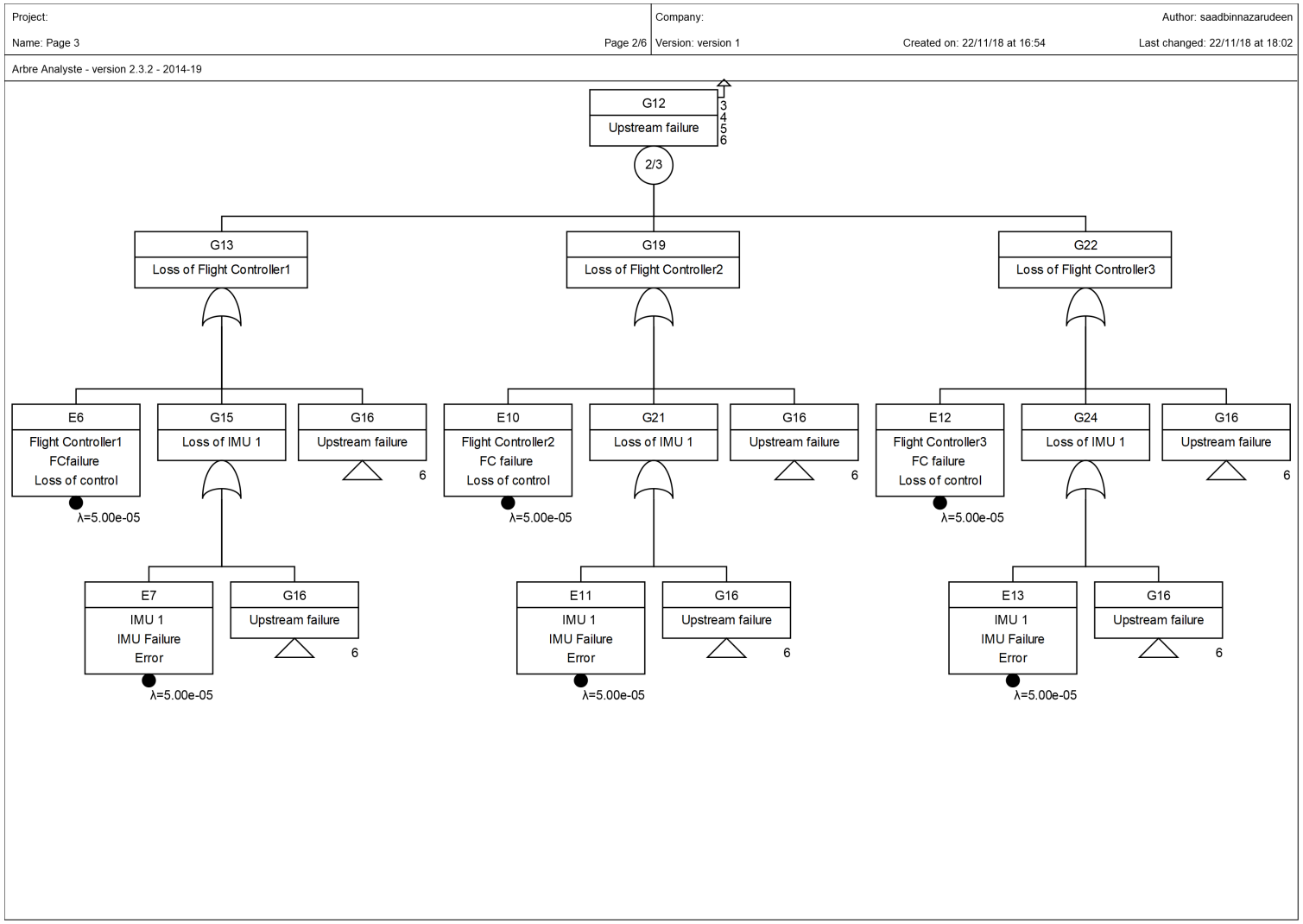


Figure A.5: Quadrotor with redundant electronics fault tree extension G12

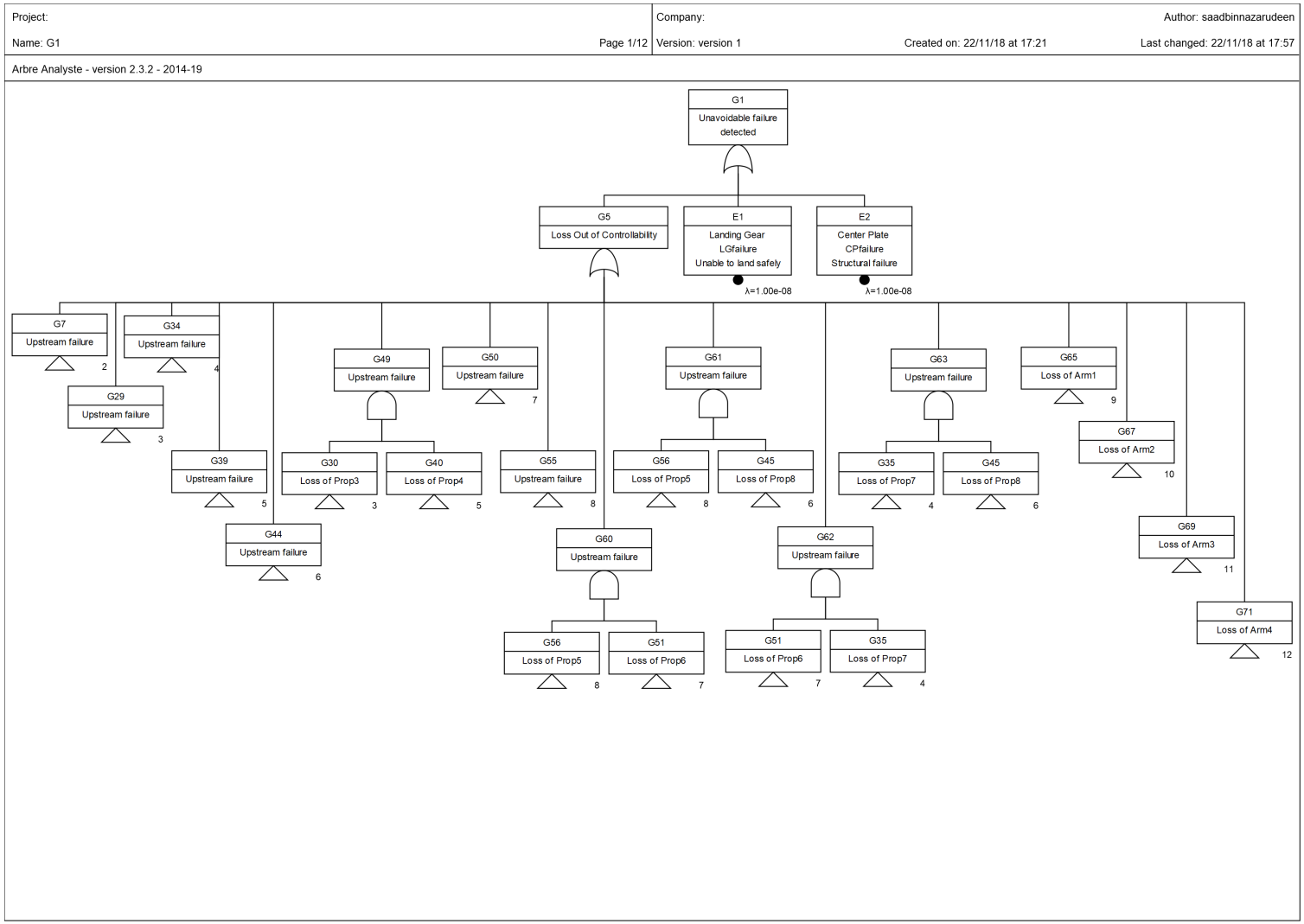


Figure A.6: Coaxial quadrotor main fault tree

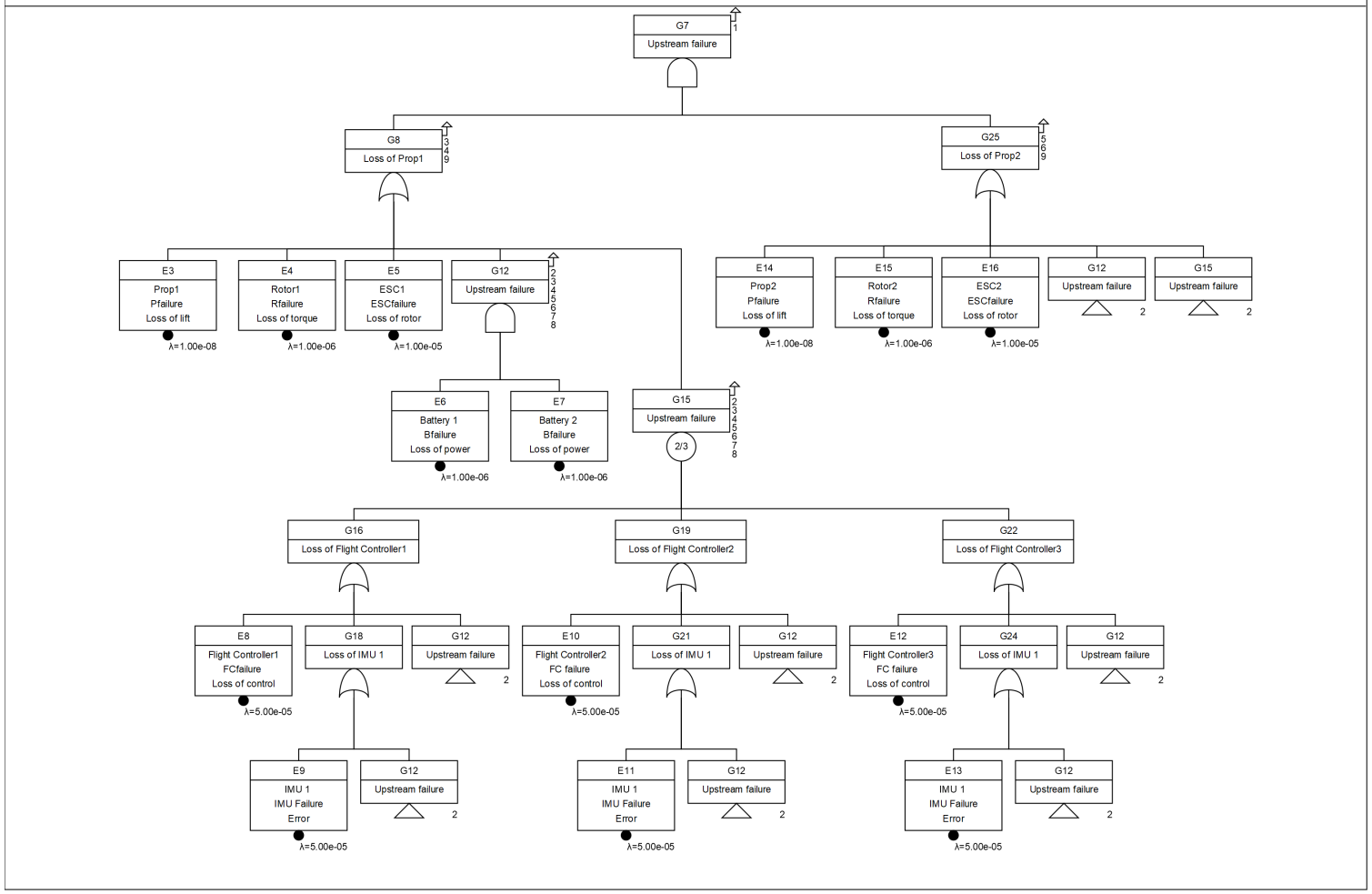


Figure A.7: Coaxial quadrotor fault tree extension G7 (similar to)

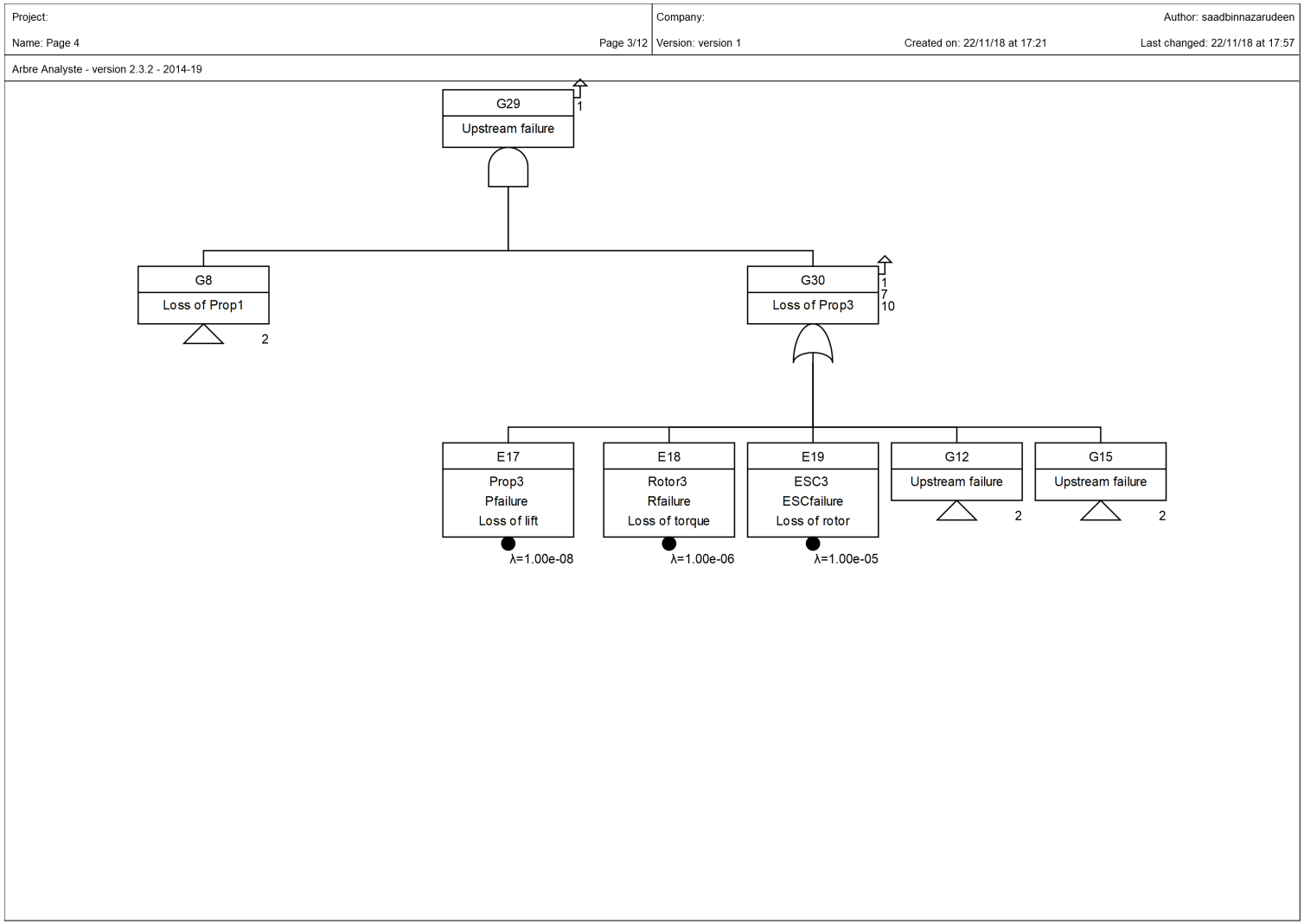


Figure A.8: Coaxial quadrotor fault tree extension

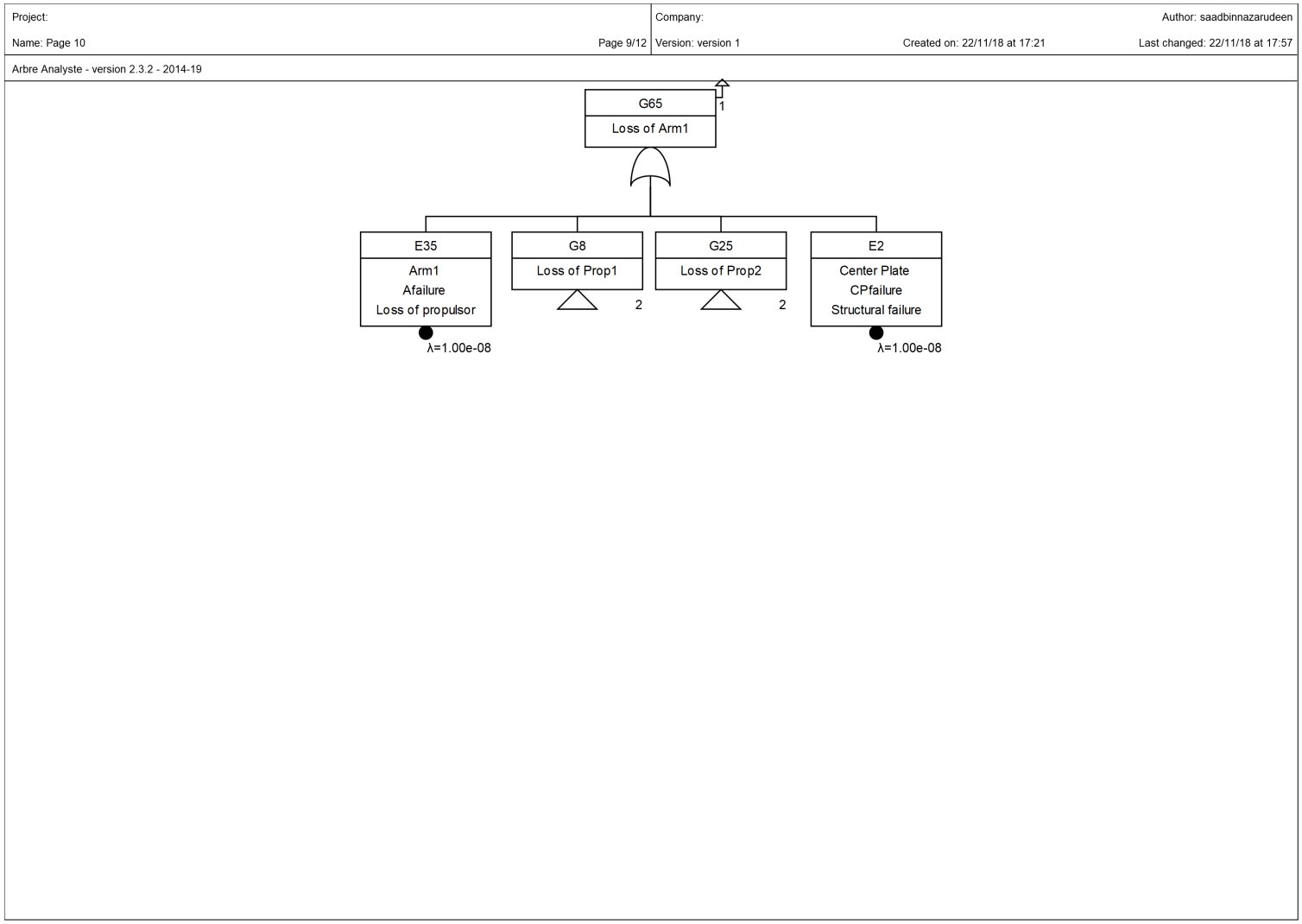


Figure A.9: Coaxial quadrotor fault tree extension

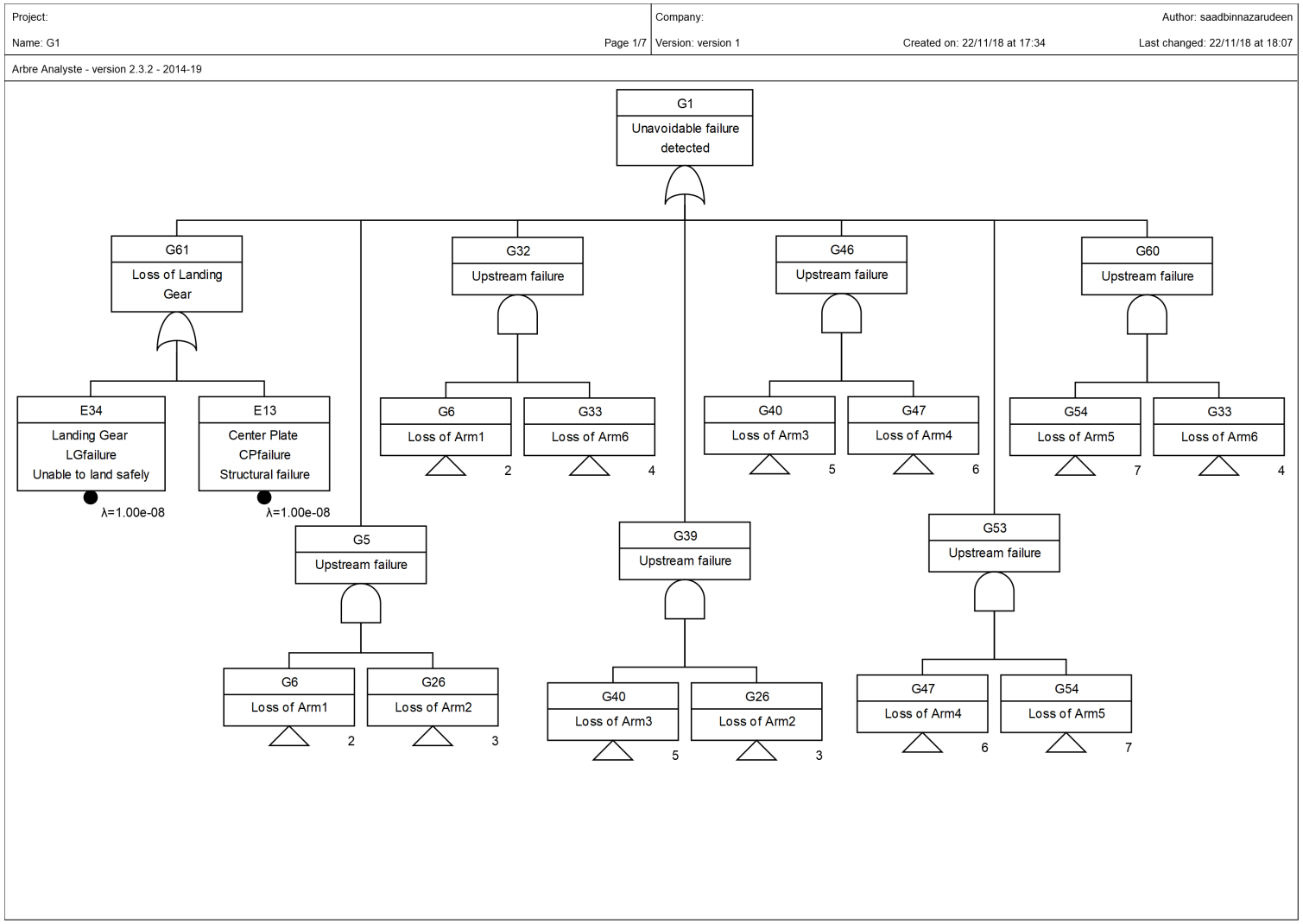


Figure A.10: Hexarotor (PNPNPN) main fault tree

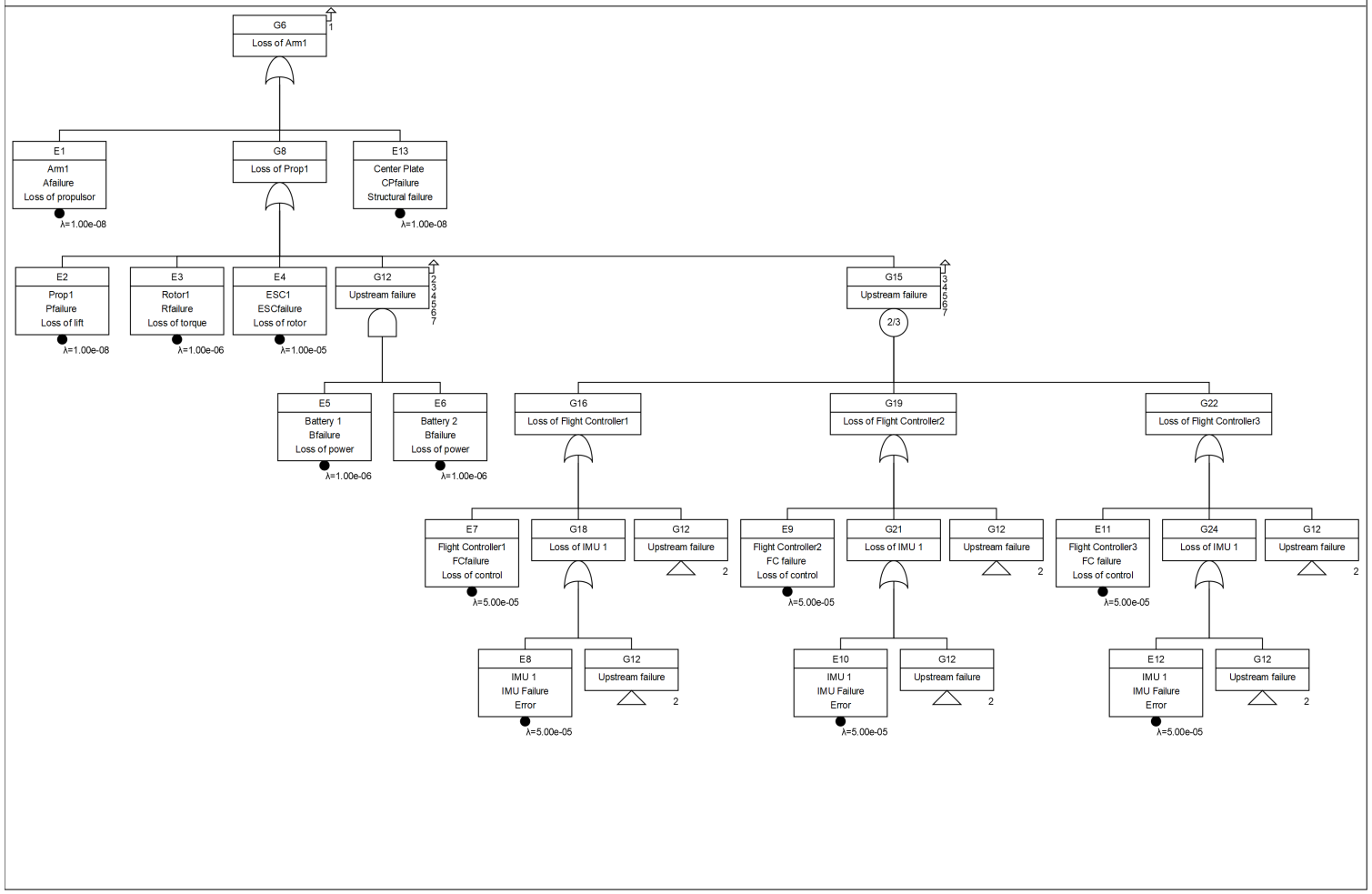


Figure A.11: Hexarotor (PNPNPN) fault tree extension G6 (similar to G33, G40, G26, G47, G54, G33)

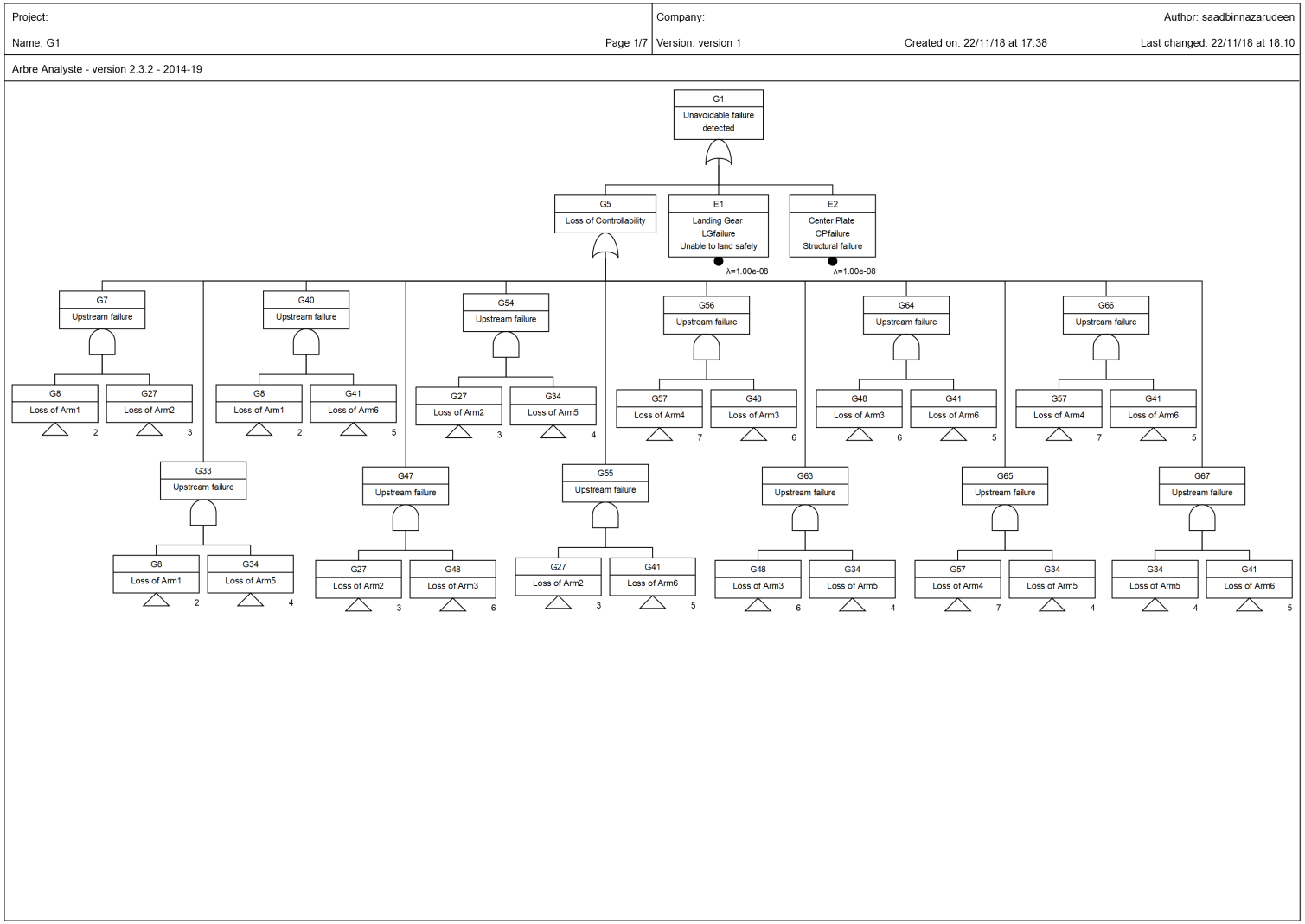


Figure A.12: Hexarotor (PPNNPN) main fault tree

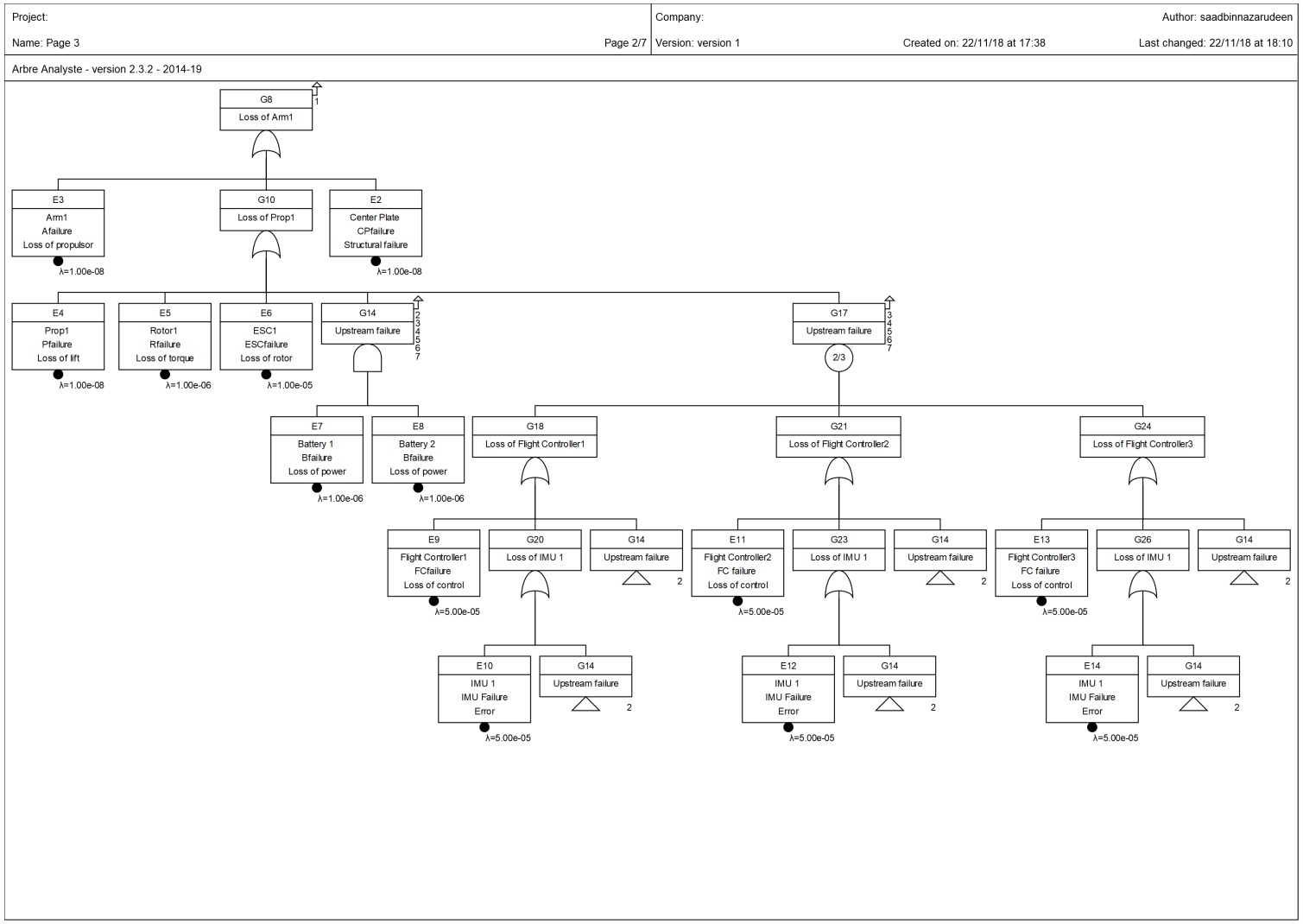


Figure A.13: Hexarotor (PPNNPN) fault tree extension G8

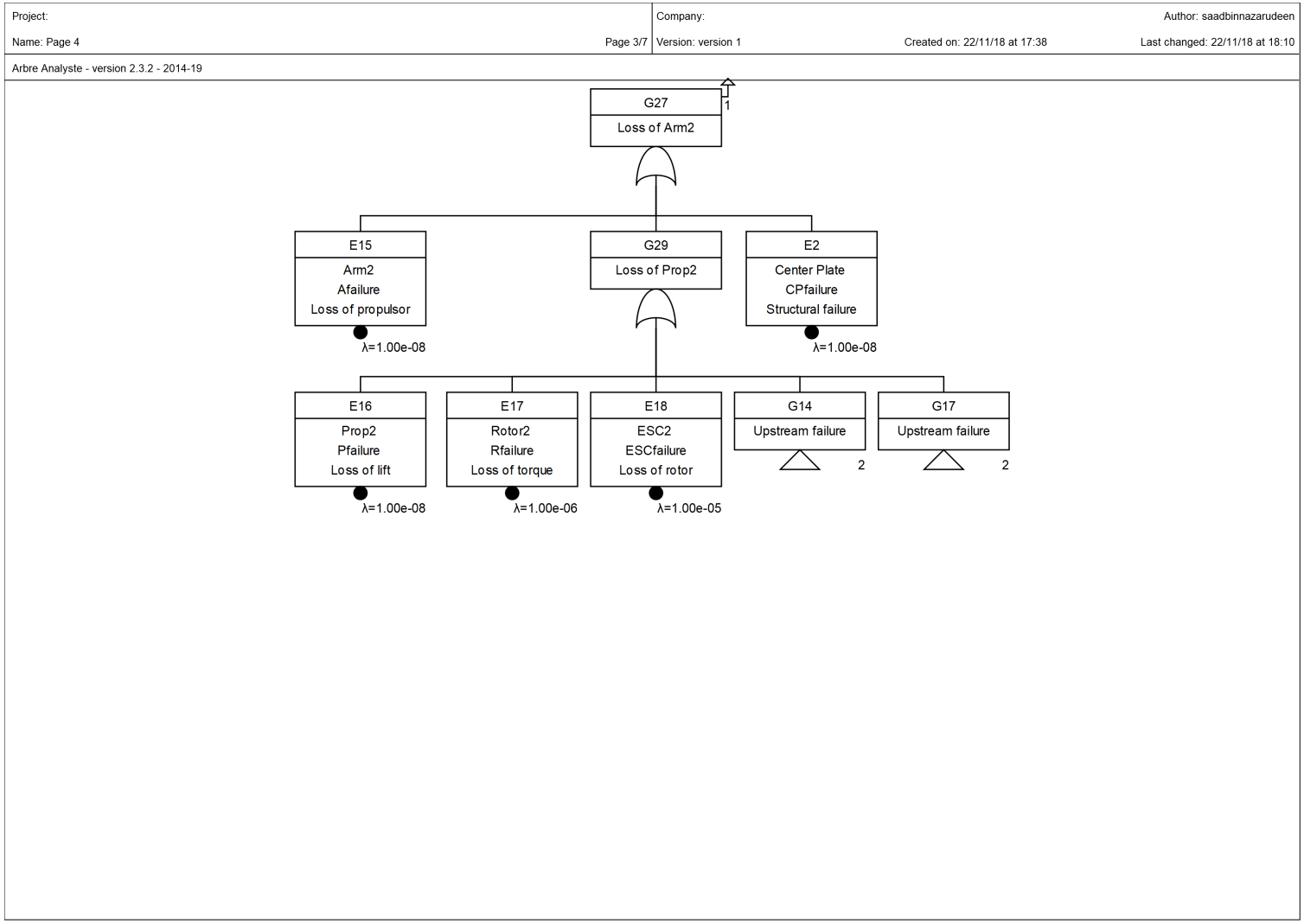


Figure A.14: Hexarotor (PPNNPN) fault tree extension G27 (similar to G34, G41, G27, G48, G54, G57)

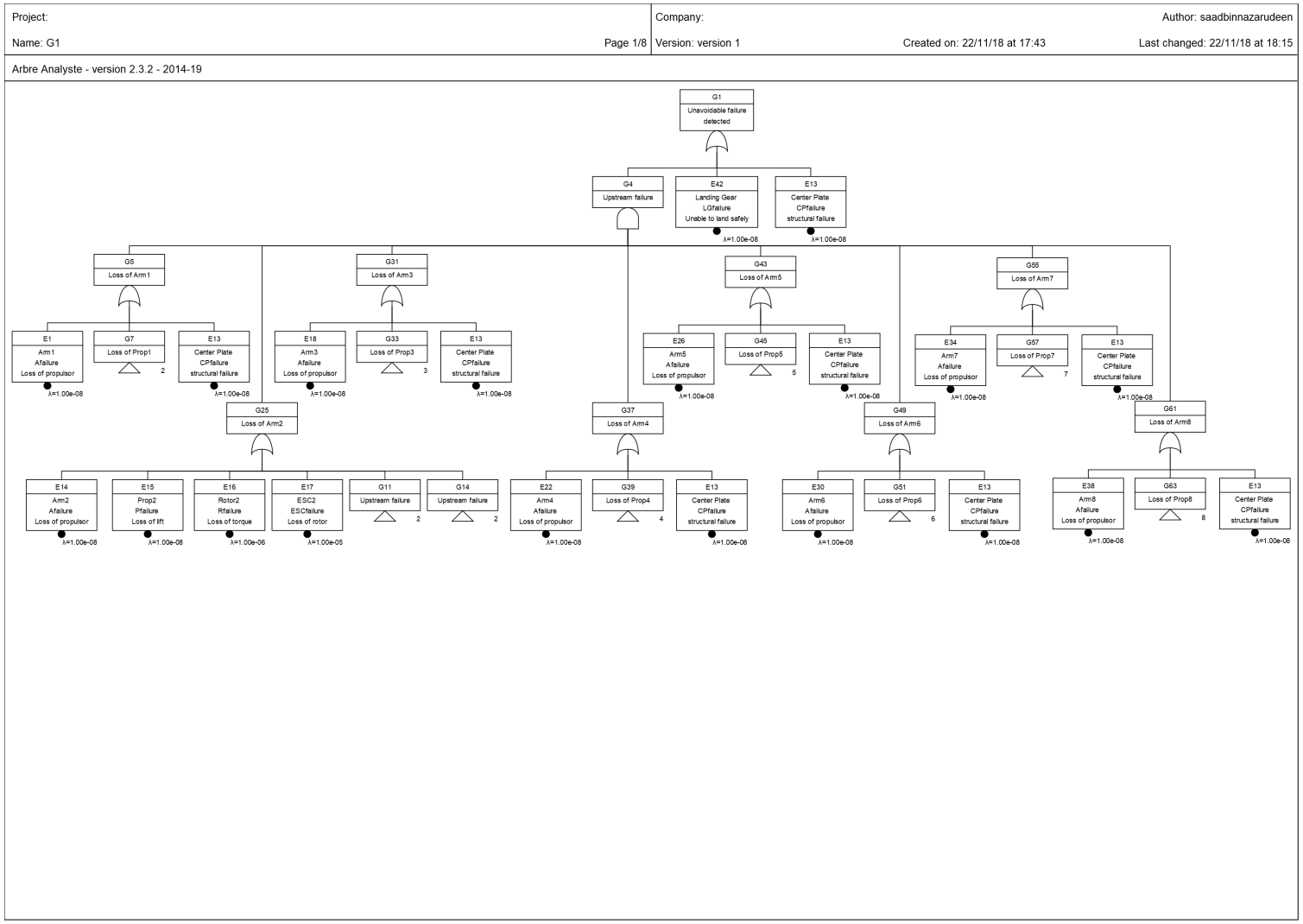


Figure A.15: Octarotor main fault tree

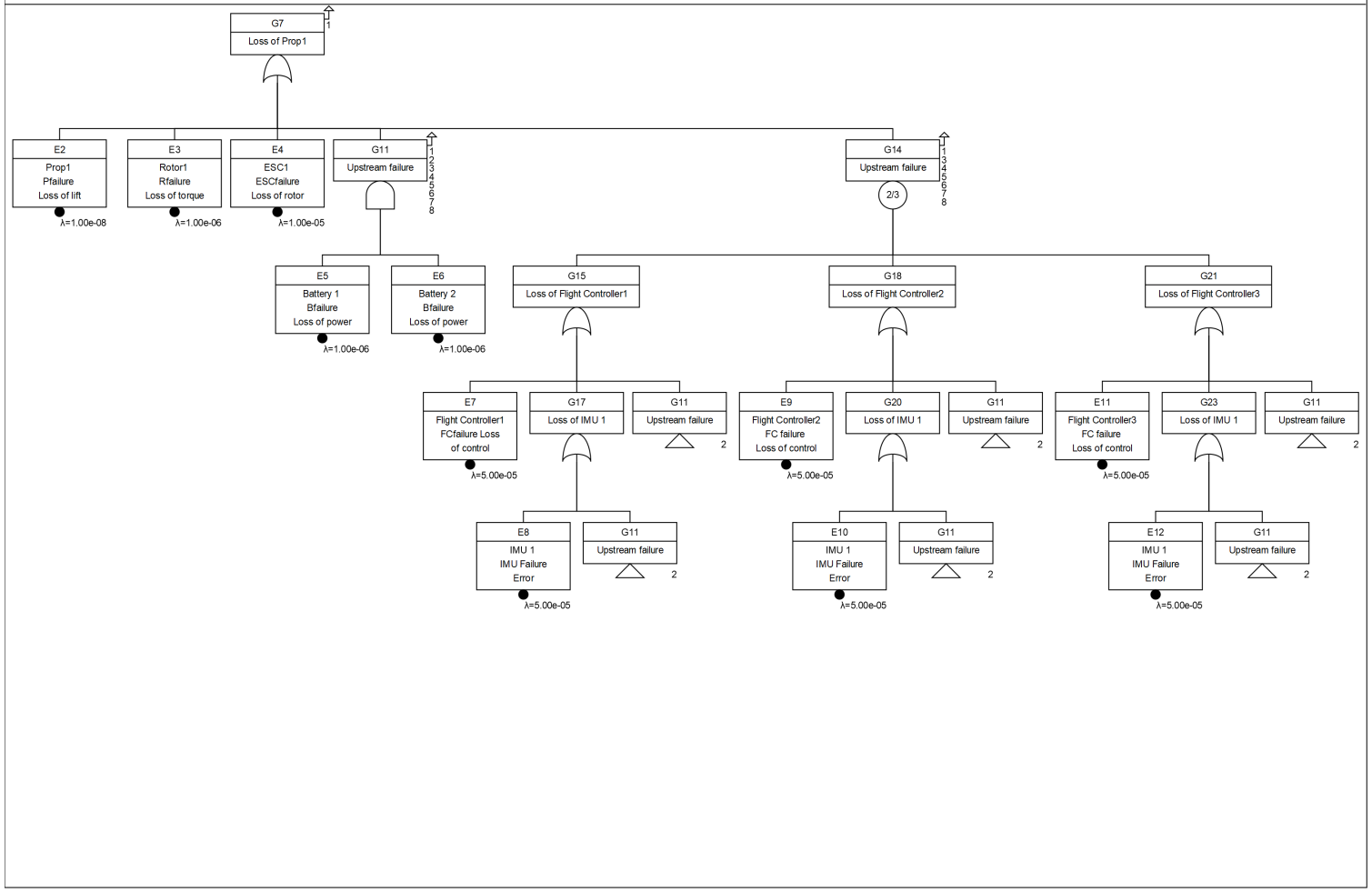


Figure A.16: Octarotor fault tree extension G7

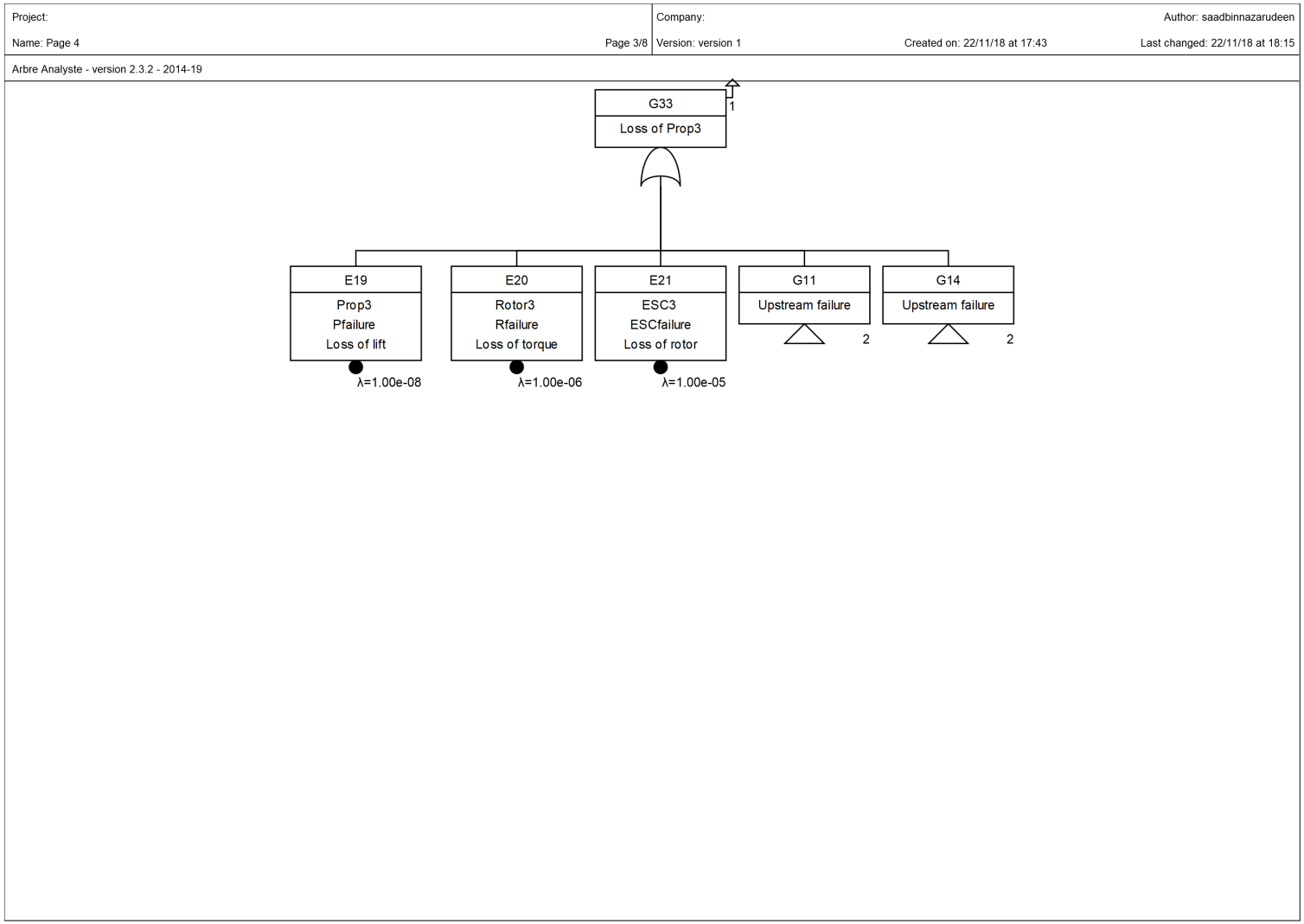


Figure A.17: Octarotor fault tree extension G33 (similar to G45, G57, G39, G51, G63, G57)

Bibliography

[1] Q. Quan, *Introduction to Multicopter Design and Control*. Springer Singapore, 2017. DOI: 10.1007/978-981-10-3382-7.

[2] E. Lygouras, A. Gasteratos, K. Tarchanidis, and A. Mitropoulos, "ROLFER: A fully autonomous aerial rescue support system," *Microprocessors and Microsystems*, vol. 61, pp. 32–42, Sep. 2018. DOI: 10.1016/j.micpro.2018.05.014.

[3] I. Nizar, Y. Illoussamen, H. El Ouarrak, E. Hossein Illoussamen, M. Grana (Graña), and M. Mestari, "Safe and optimal navigation for autonomous multi-rotor aerial vehicle in a dynamic known environment by a decomposition-coordination method," *Cognitive Systems Research*, vol. 63, pp. 42–54, Oct. 2020. DOI: 10.1016/j.cogsys.2020.05.003.

[4] C. A. Ochoa and E. M. Atkins, "Fail-safe navigation for autonomous urban multicopter flight," in *AIAA Information Systems-AIAA Infotech @ Aerospace*, Grapevine, Texas: American Institute of Aeronautics and Astronautics, Jan. 9, 2017. DOI: 10.2514/6.2017-0222.

[5] M. Gatti, "Complete preliminary design methodology for electric multirotor," *Journal of Aerospace Engineering*, vol. 30, no. 5, p. 04 017 046, Sep. 2017. DOI: 10.1061/(ASCE)AS.1943-5525.0000752.

[6] T. Du, A. Schulz, B. Zhu, B. Bickel, and W. Matusik, "Computational multicopter design," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–10, Nov. 11, 2016. DOI: 10.1145/2980179.2982427.

[7] M. Biczyski, R. Sehab, J. F. Whidborne, G. Krebs, and P. Luk, "Multirotor sizing methodology with flight time estimation," *Journal of Advanced Transportation*, vol. 2020, pp. 1–14, Jan. 20, 2020. DOI: 10.1155/2020/9689604.

[8] W. Ong, S. Srigrarom, and H. Hesse, "Design methodology for heavy-lift unmanned aerial vehicles with coaxial rotors," in *AIAA Scitech 2019 Forum*, San Diego, California: American Institute of Aeronautics and Astronautics, Jan. 7, 2019. DOI: 10.2514/6.2019-2095.

[9] S. Delbecq, M. Budinger, A. Ochotorena, A. Reysset, and F. Defay, "Efficient sizing and optimization of multirotor drones based on scaling laws and similarity models," *Aerospace Science and Technology*, vol. 102, p. 105 873, Jul. 2020. DOI: 10.1016/j.ast.2020.105873.

[10] S. B. Nazarudeen and J. Liscouët, "State-of-the-art and directions for the conceptual design of safety-critical unmanned and autonomous aerial vehicles," in *2021 IEEE International Conference on Autonomous Systems (ICAS)*, Aug. 2021, pp. 1–5. DOI: 10.1109/ICAS49788.2021.9551158.

[11] D. Shi, B. Yang, and Q. Quan, "Reliability analysis of multicopter configurations based on controllability theory," in *2016 35th Chinese Control Conference (CCC)*, Chengdu: IEEE, Jul. 2016, pp. 6740–6745. DOI: 10.1109/ChiCC.2016.7554418.

[12] G.-X. Du, Q. Quan, B. Yang, and K.-Y. Cai, "Controllability analysis for multirotor helicopter rotor degradation and failure," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 5, pp. 978–985, May 2015. DOI: 10.2514/1.G000731.

[13] J. Zha, X. Wu, J. Kroeger, N. Perez, and M. W. Mueller, "A collision-resilient aerial vehicle with icosahedron tensegrity structure," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Oct. 24, 2020, pp. 1407–1412. DOI: 10.1109/IROS45743.2020.9341236.

[14] D. Floreano, S. Mintchev, and J. Shintake, "Foldable drones: From biology to technology," presented at the SPIE Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring, M. Knez, A. Lakhtakia, and R. J. Martín-Palma, Eds., Portland, Oregon, United States, May 1, 2017, p. 1 016 203. DOI: 10.1117/12.2259931.

[15] *Flight evaluation of multicopter*. https://flyeval.com/.

[16] X. Dai, Q. Quan, J. Ren, and K.-Y. Cai, "An analytical design-optimization method for electric propulsion systems of multicopter UAVs with desired hovering endurance," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 1, pp. 228–239, Feb. 2019. DOI: 10.1109/TMECH.2019.2890901.

[17] X. Dai, Q. Quan, J. Ren, and K.-Y. Cai, "Efficiency optimization and component selection for propulsion systems of electric multicopters," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 10, pp. 7800–7809, Oct. 2019. DOI: 10.1109/TIE.2018.2885715.

[18] *MOC SC-VTOL- Proposed Means of Compliance with the Special Condition VTOL*, European union aviation safety agency. Jul. 24, 2020. [Online]. Available: https://www.easa.europa.eu/document-library/product-certification-consultations/special-condition-vtol.

[19] *Light Unmanned Aircraft Systems - SC Light-UAS 01*, European union aviation safety agency. Jul. 20, 2020. [Online]. Available: https://www.easa.europa.eu/sites/default/files/dfu/special_condition_light_uas.pdf.

[20] V. Lippiello, F. Ruggiero, and D. Serra, "Emergency landing for a quadrotor in case of a propeller failure: A backstepping approach," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA: IEEE, Sep. 2014, pp. 4782–4788. DOI: 10.1109/IROS.2014.6943242.

[21] A. Lanzon, A. Freddi, and S. Longhi, "Flight control of a quadrotor vehicle subsequent to a rotor failure," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 580–591, Mar. 2014. DOI: 10.2514/1.59869.

[22] Y. V. Morozov, "Emergency control of a quadrocopter in case of failure of two symmetric propellers," *Automation and Remote Control*, vol. 79, no. 3, pp. 463–478, Mar. 2018. DOI: 10.1134/S0005117918030062.

[23] S-18 Aircraft and Sys Dev and Safety Assessment Committee, "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," SAE International. DOI: 10.4271/ARP4761.

[24] J. Liscouët, F. Pollet, J. Jézégou, M. Budinger, S. Delbecq, and J.-M. Moschetta, "A methodology to integrate reliability into the conceptual design of safety-critical multirotor unmanned aerial vehicles," *Aerospace Science and Technology*, vol. 127, p. 107 681, Aug. 1, 2022. DOI: 10.1016/j.ast.2022.107681.

[25] E. Dubrova, *Fault-Tolerant Design*. New York, NY: Springer New York, 2013. DOI: 10.1007/978-1-4614-2113-9.

[26] J.-P. Signoret and A. Leroy, *Reliability Assessment of Safety and Production Systems: Analysis, Modelling, Calculations and Case Studies* (Springer Series in Reliability Engineering). Cham: Springer International Publishing, 2021. DOI: 10.1007/978-3-030-64708-7.

[27] H. Hecht, *Systems reliability and failure prevention* (Artech House technology management and professional development library). Boston, MA: Artech House, 2004, 230 pp.

[28] N. Scientific and T. I. Program, "Failure modes and effects analysis (FMEA) - a bibliography," NASA Langley Technical Report Server, Technical Report, 2000.

[29] M. Čepin, "Reliability block diagram," in *Assessment of Power System Reliability*. London: Springer London, 2011, pp. 119–123. DOI: 10.1007/978-0-85729-688-7_9.

[30] N. Scientific and T. I. Program, "Fault tree analysis - a bibliography," NASA Langley Technical Report Server, Technical Report, 2000.

[31] M. Čepin, "Fault tree analysis," in *Assessment of Power System Reliability: Methods and Applications*, M. Čepin, Ed., London: Springer, 2011, pp. 61–87. DOI: 10.1007/978-0-85729-688-7_5.

[32] M. Čepin, "Markov processes," in *Assessment of Power System Reliability: Methods and Applications*, M. Čepin, Ed., London: Springer, 2011, pp. 113–118. DOI: 10.1007/978-0-85729-688-7_8.

[33] M. Bozzano, A. Villafiorita, O. Akerlund, *et al.*, "ESACS: An integrated methodology for design and safety analysis of complex systems," Jun. 15, 2003.

[34] A. Joshi, S. Miller, M. Whalen, and M. Heimdahl, "A PROPOSAL FOR MODEL-BASED SAFETY ANALYSIS," in *24th Digital Avionics Systems Conference*, vol. 2, Washington, DC, USA: IEEE, 2005, pp. 8.C.2–1–8.C.2–13. DOI: 10.1109/DASC.2005.1563469.

[35] M. Bozzano and A. Villafiorita, *Design and Safety Assessment of Critical Systems*. Nov. 12, 2010. DOI: 10.1201/b10094.

[36] M. Bozzano and A. Villafiorita, "Improving system reliability via model checking: The FSAP/NuSMV-SA safety analysis platform," in *Computer Safety, Reliability, and Security*, S. Anderson, M. Felici, and B. Littlewood, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2003, pp. 49–62. DOI: 10.1007/978-3-540-39878-3_5.

[37] M. Bozzano, A. Cimatti, J.-P. Katoen, *et al.*, "Spacecraft early design validation using formal methods," *Reliability Engineering & System Safety*, vol. 132, pp. 20–35, Dec. 1, 2014. DOI: 10.1016/j.ress.2014.07.003.

[38] Y. Papadopoulos and J. A. McDermid, "Hierarchically performed hazard origin and propagation studies," in *Computer Safety, Reliability and Security*, M. Felici and K. Kanoun, Eds., red. by G. Goos, J. Hartmanis, and J. van Leeuwen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, vol. 1698, pp. 139–152. DOI: 10.1007/3-540-48249-0_13.

[39] B. Kaiser, P. Liggesmeyer, and O. Mäckel, "A new component concept for fault trees.," presented at the Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software (SCS'03), vol. 33, Jan. 1, 2003, pp. 37–46.

[40] B. Kaiser, "Extending the expressive power of fault trees," in *Annual Reliability and Maintainability Symposium, 2005. Proceedings.*, Alexandria, VA, USA: IEEE, 2005, pp. 468–474. DOI: 10.1109/RAMS.2005.1408407.

[41] D. Domis and M. Trapp, "Integrating safety analyses and component-based design," in *Computer Safety, Reliability, and Security*, M. D. Harrison and M.-A. Sujan, Eds., vol. 5219, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 58–71. DOI: 10.1007/978-3-540-87698-4_8.

[42] J. Aizpurua Unanue and E. Muxika Olasagasti, "Model-based design of dependable systems: Limitations and evolution of analysis and verification approaches," *International Journal On Advances in Security*, vol. 6, pp. 12–31, Jul. 15, 2013.

[43] S. Sharvia, S. Kabir, M. Walker, and Y. Papadopoulos, "Chapter 12 - model-based dependability analysis: State-of-the-art, challenges, and future outlook," in *Software Quality Assurance*, I. Mistrik, R. Soley, N. Ali, J. Grundy, and B. Tekinerdogan, Eds., Boston: Morgan Kaufmann, Jan. 1, 2016, pp. 251–278. DOI: 10.1016/B978-0-12-802301-3.00012-0.

[44] O. Lisagor, T. Kelly, and R. Niu, "Model-based safety assessment: Review of the discipline and its challenges," in *The Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety*, Guiyang, China: IEEE, Jun. 2011, pp. 625–632. DOI: 10.1109/ICRMS.2011.5979344.

[45] K. Hofig, A. Joanni, M. Zeller, *et al.*, "Model-based reliability and safety: Reducing the complexity of safety analyses using component fault trees," in *2018 Annual Reliability and Maintainability Symposium (RAMS)*, Reno, NV: IEEE, Jan. 2018, pp. 1–7. DOI: 10.1109/RAM.2018.8463058.

[46] Y. Papadopoulos, M. Walker, D. Parker, *et al.*, "Engineering failure analysis and design optimisation with HiP-HOPS," *Engineering Failure Analysis*, The Fourth International Conference on Engineering Failure Analysis Part 1, vol. 18, no. 2, pp. 590–608, Mar. 1, 2011. DOI: 10.1016/j.engfailanal.2010.09.025.

[47] J. I. Aizpurua, E. Muxika, Y. Papadopoulos, F. Chiacchio, and G. Manno, "Application of the d3h2 methodology for the cost-effective design of dependable systems," *Safety*, vol. 2, no. 2, Mar. 25, 2016, Publisher: MDPI. DOI: 10.3390/safety2020009.

[48] Z. Mian, L. Bottaci, Y. Papadopoulos, and S. Sharvia, "Model transformation for multi-objective architecture optimisation of dependable systems," in *Advances in Intelligent Systems and Computing*, vol. 307, Journal Abbreviation: Advances in Intelligent Systems and Computing, May 10, 2015, pp. 91–110. DOI: 10.1007/978-3-319-08964-5_6.

[49] M. Walker, M.-O. Reiser, S. Tucci-Piergiovanni, *et al.*, "Automatic optimisation of system architectures using EAST-ADL," *Journal of Systems and Software*, vol. 86, no. 10, pp. 2467–2487, Oct. 1, 2013. DOI: 10.1016/j.jss.2013.04.001.

[50] R. Adler, D. Domis, K. Höfig, *et al.*, "Integration of component fault trees into the UML," vol. 6627, Oct. 3, 2010, pp. 312–327. DOI: 10.1007/978-3-642-21210-9_30.

[51] P. Munk, A. Abele, E. Thaden, *et al.*, "Semi-automatic safety analysis and optimization," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18, New York, NY, USA: Association for Computing Machinery, Jun. 24, 2018, pp. 1–6. DOI: 10.1145/3195970.3199857.

[52] *Systems modeling language (BDD SysML) version 1.3*. [Online]. Available: https://sysml.org/.res/docs/specs/OMGSysML-v1.3-12-06-02.pdf.

[53] A. Nordmann and P. Munk, "Lessons learned from model-based safety assessment with SysML and component fault trees," in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '18, New York, NY, USA: Association for Computing Machinery, Oct. 14, 2018, pp. 134–143. DOI: 10.1145/3239372.3239373.

[54] M. Bozzano, A. Cimatti, M. Gario, D. Jones, and C. Mattarei, "Model-based safety assessment of a triple modular generator with xSAP," *Formal Aspects of Computing*, vol. 33, no. 2, pp. 251–295, Mar. 2021. DOI: 10.1007/s00165-021-00532-9.

[55] S. Nannapaneni, A. Dubey, S. Abdelwahed, S. Mahadevan, and S. Neema, "A model-based approach for reliability assessment in component- based systems," *Annual Conference of the PHM Society*, vol. 6, no. 1, 2014, Number: 1. DOI: 10.36001/phmconf.2014.v6i1.2394.

[56] O. Lisagor, T. Kelly, and R. Niu, "Model-based safety assessment: Review of the discipline and its challenges," in *The Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety*, Jun. 2011, pp. 625–632. DOI: 10.1109/ICRMS.2011.5979344.

[57] A. Joshi, M. Whalen, and M. Heimdahl, "Model-based safety analysis final report," Feb. 1, 2006.

[58] A. Arnold, G. Point, A. Griffault, and A. Rauzy, "The AltaRica formalism for describing concurrent systems," *Fundamenta Informaticae*, vol. 40, no. 2, pp. 109–124, Nov. 1, 1999.

[59] Y. Papadopoulos and M. Maruhn, "Model-based synthesis of fault trees from matlab-simulink models," Aug. 1, 2001, pp. 77–82. DOI: 10.1109/DSN.2001.941393.

[60] F. Grigoleit, S. Holei, A. Pleuss, *et al.*, *The qSafe Project - Developing a Model-based Tool for Current Practice in Functional Safety Analysis*. Aug. 21, 2017. DOI: 10.29007/1lp8.

[61] F. Grigoleit, S. Holei, A. Pleuss, *et al.*, "The qSafe project – developing a tool for current practice in functional safety analysis," in *Kalpa Publications in Computing*, vol. 4, EasyChair, Jan. 6, 2018, pp. 297–312. DOI: 10.29007/1lp8.

[62] K. Hofig, A. Joanni, M. Zeller, *et al.*, "Model-based reliability and safety: Reducing the complexity of safety analyses using component fault trees," in *2018 Annual Reliability and Maintainability Symposium (RAMS)*, Jan. 2018, pp. 1–7. DOI: 10.1109/RAM.2018.8463058.

[63] C. Tommasi, N. Jankevicius, and A. Armonas, "Model-based reliability and safety analysis, fosters agility in design of mission-critical systems," in *CIISE*, 2017.

[64] C. Stigliani, D. Ferretto, C. Pessa, and E. Brusa, "A model based approach to design for reliability and safety of critical aeronautic systems," in *CIISE*, 2016.

[65] L. Rogovchenko-Buffoni, A. Tundis, M. Z. Hossain, M. Nyberg, and P. Fritzson, "An integrated toolchain for model based functional safety analysis," *Journal of Computational Science*, vol. 5, no. 3, pp. 408–414, May 1, 2014. DOI: 10.1016/j.jocs.2013.08.009.

[66] P. Hönig, R. Lunde, and F. Holzapfel, "Model based safety analysis with smartIflow †," *Information*, vol. 8, p. 7, Jan. 3, 2017. DOI: 10.3390/info8010007.

[67] *System Analyst*. http://system-analyst.fr/.

[68] M. Batteux, T. Prosvirnova, and A. B. Rauzy, "AltaRica 3.0 in ten modelling patterns," *International Journal of Critical Computer-Based Systems*, vol. 9, no. 1, p. 133, 2019. DOI: 10.1504/IJCCBS.2019.098809.

[69] Y. Papadopoulos, K. Aslansefat, P. Katsaros, and M. Bozzano, Eds., *Model-based safety and assessment: 6th International Symposium, IMBSA 2019, Thessaloniki, Greece, October 16-18, 2019: proceedings*, Lecture notes in computer science 11842, Meeting Name: IMBSA, Cham, Switzerland: Springer, 2019, 380 pp.

[70] M. Batteux, T. Prosvirnova, and A. Rauzy, "AltaRica 3.0 assertions: The whys and wherefores," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 231, no. 6, pp. 691–700, Dec. 2017. DOI: 10.1177/1748006X17728209.

[71] T. Prosvirnova and A. Rauzy, "Automated generation of minimal cut sets from AltaRica 3.0 models," *International Journal of Critical Computer-Based Systems*, vol. 6, no. 1, p. 50, 2015. DOI: 10.1504/IJCCBS.2015.068852.

[72] P.-A. Brameret, A. Rauzy, and J.-M. Roussel, "Automated generation of partial markov chain from high level descriptions," *Reliability Engineering & System Safety*, vol. 139, pp. 179–187, Jul. 2015. DOI: 10.1016/j.ress.2015.02.009.

[73] A. B. Rauzy, "Guarded transition systems: A new states/events formalism for reliability studies," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 222, no. 4, pp. 495–505, Dec. 1, 2008. DOI: 10.1243/1748006XJRR177.

[74] D. Shi, B. Yang, and Q. Quan, "Reliability analysis of multicopter configurations based on controllability theory," in *2016 35th Chinese Control Conference (CCC)*, Chengdu: IEEE, Jul. 2016, pp. 6740–6745. DOI: 10.1109/ChiCC.2016.7554418.