AIRBUS

Technische Universität München

MBSE Assisted Functional Failure Modes and Effects Analysis for Laser Interferometer Space Antenna (LISA) Mission

Master's Thesis

Thesis work to obtain the degree of
M.Sc. Aerospace
at the School of Engineering and Design at the Technical University of Munich.

Submitted by Hakan Yanık
Student ID: 03742436

Submitted on 19.05.2023 in Munich/Germany

Supervised by Prof. Dr. Phil. Alessandro Golkar
TUM - Chair of Pico and Nano Satellites, and Satellite Constellations

Dipl. Ing. Tobias Ziegler
Airbus / Space Systems - Mission & Satellite Chief Engineering Germany

Advisors Christian Greve, Dr. rer. nat.
Jacopo Aurigi, M.Sc.
Jaspar Sindermann, M.Sc.

Dedicated to my hero and role model,

Mustafa Kemal Atatürk

“Science is the most real guide for civilisation, for life, for success in the world. To search for a guide other than science is absurdity, ignorance and heresy.”

…for the Young Generation of the Young Republic,

25.09.1924

ABSTRACT

This master's thesis introduces two methodologies for conducting functional Failure Modes and Effects Analysis (FMEA), namely static model-supported and executable model-based FMEA, that utilize a system architecture model implemented through Model-Based Systems Engineering (MBSE) with SysML (Systems Modeling Language). These two methodologies are investigated and compared with a traditional document-based method, based on a specific set of criteria, in a selected operational scenario within the Laser Interferometer Space Antenna (LISA) Mission.

The thesis establishes a system architecture model that incorporates model elements, traceability links, and executable diagrams to depict functional architecture, interfaces, parameters, behaviours, and constraints with links to functional requirements and technical components. In the static model-supported FMEA, this model assists FMEA activities by automating input creation and providing convenient access and storage of FMEA table and entries. The executable model-based FMEA builds upon this by simulating operational scenarios with failures injected into each function, thereby automatically revealing the effects of these failures on the LISA spacecraft constellation. Through case studies and criteria-based qualitative assessment, the thesis investigates and compares the model-based approaches against a traditional document-based method, highlighting their advantages in enhancing FMEA activities and providing recommendations to address limitations. As a result, the research demonstrates the superiority of the executable model-based FMEA approach for complex space science missions like LISA.

This thesis contributes to the field of space systems engineering by offering a novel MBSE-assisted FMEA methodology and highlighting the benefits of a model-based approach in improving FMEA.

ZUSAMMENFASSUNG

In dieser Masterarbeit werden zwei Methoden zur Durchführung einer funktionalen Fehler- und Einflussanalyse (FMEA) vorgestellt, nämlich die statische, modellgestützte und die ausführbare, modellbasierte FMEA, die ein Systemarchitekturmodell verwenden, das durch Model-Based Systems Engineering (MBSE) mit SysML (Systems Modeling Language) implementiert wurde. Diese beiden Methoden werden in einem ausgewählten Betriebsszenario der Laser Interferometer Space Antenna (LISA) Mission untersucht und mit einer traditionellen dokumentenbasierten Methode verglichen, die auf einem spezifischen Satz von Kriterien basiert.

In dieser Arbeit wird ein Systemarchitekturmodell erstellt, das Modellelemente, Rückverfolgbarkeitsverknüpfungen und ausführbare Diagramme enthält, um die funktionale Architektur, Schnittstellen, Parameter, Verhaltensweisen und Einschränkungen mit Verknüpfungen zu funktionalen Anforderungen und technischen Komponenten darzustellen. In der statischen modellgestützten FMEA unterstützt dieses Modell die FMEA-Aktivitäten, indem es die Eingabeerstellung automatisiert und einen bequemen Zugriff auf und die Speicherung von FMEA-Tabelle und -Einträgen ermöglicht. Die ausführbare modellgestützte FMEA baut darauf auf, indem sie Betriebsszenarien mit Fehlern in jeder Funktion simuliert und so automatisch die Auswirkungen dieser Fehler auf die LISA-Raumfahrzeugkonstellation aufzeigt. Anhand von Fallstudien und einer kriterienbasierten qualitativen Bewertung werden in dieser Arbeit die modellbasierten Ansätze untersucht und mit einer traditionellen dokumentenbasierten Methode verglichen, wobei ihre Vorteile bei der Verbesserung von FMEA-Aktivitäten hervorgehoben und Empfehlungen zur Behebung von Einschränkungen gegeben werden. Als Ergebnis zeigt die Arbeit die Überlegenheit des ausführbaren modellbasierten FMEA-Ansatzes für komplexe wissenschaftliche Raumfahrtmissionen wie LISA.

Diese Arbeit leistet einen Beitrag zum Bereich des Space Systems Engineering, indem sie eine neuartige MBSE-gestützte FMEA-Methodik anbietet und die Vorteile eines modellbasierten Ansatzes zur Verbesserung der FMEA hervorhebt.

TABLE OF CONTENTS

ABSTRACT **3**

TABLE OF CONTENTS **4**

LIST OF FIGURES **6**

LIST OF TABLES **8**

LIST OF ABBREVIATIONS **9**

1 INTRODUCTION **12**

1.1 PURPOSE AND MOTIVATION 12
1.2 THE LISA MISSION 13
1.2.1 Gravitational Wave Measurement Principle in LISA Mission *14*
1.2.2 The LISA Spacecraft Payload *14*
1.2.3 The Laser Acquisition Sequence *16*
1.3 MODEL BASED SYSTEMS ENGINEERING 19
1.3.1 General Overview of MBSE *19*
1.3.2 SysML and Cameo Systems Modeler *19*
1.4 RESEARCH OBJECTIVES 21
1.5 RESEARCH STRUCTURE 21

2 STATE OF THE ART **23**

2.1 FUNCTIONAL FAILURE MODES AND EFFECT ANALYSIS 23
2.2 MBSE ASSISTED FAILURE MODES AND EFFECTS ANALYSIS 24
2.3 LISA MISSION SYSTEM ARCHITECTURE MODEL 26
2.3.1 LISA Mission System Architecture Model Overview *26*
2.3.2 R-MOFLT Architecture Framework *27*
2.3.3 Model Content and Structure *28*
2.4 STATE-OF-THE-ART REVIEW OUTCOMES AND RESEARCH CONTRIBUTIONS 32

3 APPROACH: MODELLING AND CONDUCTING FUNCTIONAL FAILURE MODES AND EFFECTS ANALYSIS IN LISA MISSION **34**

3.1 FUNCTIONAL ARCHITECTURE MODELING 34
3.1.1 Modelling of Technical Components and Functions *35*
3.1.2 Modelling of Functional Interfaces and Flows *38*
3.1.3 Modelling of Functional Modes, Behaviours and Failure Constraints *40*
3.2 EXECUTABLE MODEL SIMULATION SETUP 45
3.2.1 Execution Context *45*
3.2.2 Laser Acquisition Operational Phase Behaviour *46*
3.2.3 Execution Behaviour and Results *48*
3.3 FUNCTIONAL FMEA APPROACHES FOR THE LISA MISSION 50
3.3.1 FMEA Study: Traditional Document Based Analysis *50*
3.3.2 FMEA Study: Static Model-Supported Analysis *54*
3.3.3 FMEA Study: Executable Model-Based Analysis *57*

4 RESULTS AND DISCUSSION **61**

4.1 FMEA COMPARISON 61
4.1.1 Comparison of Examples from Investigated FMEA Results *61*
4.1.2 Discussion and Assessment of Investigated FMEA Methodologies for the LISA Mission *67*
4.1.3 Comparison Conclusion *75*
4.2 THESIS SUMMARY AND CONTRIBUTIONS 76

5 FUTURE WORK **78**

6 APPENDIX 80

6.1 APPENDIX A 80
6.2 APPENDIX B 81

ACKNOWLEDGEMENT 85

REFERENCES 86

LIST OF FIGURES

Figure 1.1: Spectrum of gravitational waves showing astronomical bodies and phenomena the LISA mission is aimed to detect. Courtesy of ESA [10] 13
Figure 1.2: Schematic of a LISA SC with the two MOSAs and the most important instrument units. Modified from Charpigny [5]. 16
Figure 1.3: LISA constellation view showing laser links and locking scheme. 17
Figure 1.4: Overview of SysML diagrams. Courtesy of Object Management Group [3]. 20
Figure 1.5: Flowchart showing the structure of the thesis work. 22
Figure 2.1: LISA System Architecture Model implementation timeline with LISA Mission timeline. Modified from ESA [10]. 26
Figure 2.2: Overview of R-MOFLT methodology, courtesy of Airbus [26]. Usually in (science) space systems with ESA as the customer, artefacts in Mission viewpoint (highlighted in red) are supplied by ESA. The artefacts in Operation viewpoint (yellow) are defined by both ESA and Prime (ADS), still ESA being responsible for the majority. The definition of the so called “solution space” (green), starting with the functional architecture, is on the responsibility of the development team in ADS. 28
Figure 2.3: SOI-Viewpoints of system definition package of LISA Mission System Architecture Model. Modified from Charpigny et.al. [7]. 29
Figure 2.4: Containment tree view of the “Model Execution” package. 30
Figure 2.5: Containment tree view of the “Reliability and Safety” package. It is important to point out the high amount of model elements that need to be generated for only 6 failure cases (FMEA items). 31
Figure 3.1: Example of technical components (left) and functions (right) in the model containment tree view. 35
Figure 3.2: Allocation of functions to technical components. Left: The functional parts are contained inside the technical component LA in the containment tree view. Right: The functional parts are depicted as green rectangles inside the ibd of the technical component. Each functional part has an instance name followed by a function type, divided by a semicolon in between (e.g. “LG : Light Generation”). 36
Figure 3.3: Collecting the technical components inside the Spacecraft-Technical block generates the definition of the spacecraft type with the components and functions inside. 37
Figure 3.4: Profile diagram showing relationships in between functions, systems and requirements. 37
Figure 3.5: Containment tree and ibd view of functional interfaces of “Laser frequency actuation” function. 38
Figure 3.6: Containment tree view showing the functional interface types. Inside, functional flows are defined which are typed by functional flow types. 38
Figure 3.7: Containment tree view of Laser Light flow type 39
Figure 3.8: ibd showing the functional interfaces and functional flows of the Laser frequency actuation function. 40
Figure 3.9: Example of a possible functional architecture of a Laser Assembly. 40
Figure 3.10: State machine diagram of Laser frequency actuation function. Functional modes “Inactive” and “Active” are depicted as the green rectangles with an “M” symbol. Inside the modes, opaque behaviours are defined with the “do/” statements. The mode transitions are depicted as arrows with the signal triggers “1” and “2”. The “activation” guard applied to the transition from inactive mode is shown inside the square brackets. ..41
Figure 3.11: Parametric diagram showing the modelled behaviour and constraints for the “Laser frequency actuation” function belonging to LA. 44
Figure 3.12: Ibd showing the execution context as the instantiation of the Spacecraft-Technical Type three times with laser links established as interface connections. 45
Figure 3.13: Left: Containment tree view of the execution context “Constellation_FMEA”. The part properties depicted with the symbol “P”, represents the three spacecrafts (SC1-2-3) in the constellation. The reference properties depicted with the symbol “R”, represents the three roles (master/reference, slave/transponder 1, slave/transponder 2) defined for the laser acquisition sequence. The value properties depicted with the symbol “V”, is used to assign a role to a specific spacecraft. Right: Classifier behaviour of the execution context as a state machine. 46
Figure 3.14: Functions to be activated during laser frequency locking operational task. 47
Figure 3.15: Activation of a target function implemented as an activity diagram. The first opaque action on top sends the signal that changes the mode of the desired function. The while loop node at the bottom checks and waits until the mode change has happened. The decision node in between is implemented to by-pass the check loop if a failure is deliberately injected to the function. 48
Figure 3.16: Left: Failure is injected to the SC2, left ePMS, Transponder beat note acquisition function by turning its activation value property from “true” to “false”. Right: During execution, model elements with failed parameters are highlighted in red. 49

Figure 3.17: The Failure Mode table shows detailed info on the injected failure. 49
Figure 3.18: The “Bottom up functional breakdown analysis” relation map is useful to observe the higher level effects of a failure happening inside a function. 49
Figure 3.19: The Failure Effects table shows the propagation of the failure along the constellation. The causes of failure can be seen by looking at the names of the failed constraint properties in the “defining feature” column. 50
Figure 3.20: Context diagram of the traditional document-based FFMEA studies in early phases. 51
Figure 3.21: Process diagram for the traditional document based FMEA.....52
Figure 3.22: Context diagram for static model-supported FMEA 55
Figure 3.23: Process diagram for the static model-supported FMEA. Notice that the input preparation activities are eliminated since they are done by the model automatically.....55
Figure 3.24: Excerpt from example FMEA table filled in using Cameo SRAP profile inside the model.....56
Figure 3.25: Context diagram for executable model based FMEA. 58
Figure 3.26: Process diagram for the executable model based FMEA. Notice that the review loop is eliminated as the model is generating the failure cause and effects information through simulations. A generic review with Systems Engineering might be needed at the end of the study. 58
Figure 4.1: Point based qualitative assessment of approaches according to “Quality of Input” criterion. 69
Figure 4.2: Point based qualitative assessment of approaches according to “Effort for Preparation” criterion. 69
Figure 4.3: Point based qualitative assessment of approaches according to “Information Access and Traceability” criterion. 70
Figure 4.4: Point based qualitative assessment of approaches according to “Quality of Results” criterion. 71
Figure 4.5: Point based qualitative assessment of approaches according to “Effort to Obtain Results” criterion. 72
Figure 4.6: Point based qualitative assessment of approaches according to “Filling the FMEA table” criterion. 73
Figure 4.7: Point based qualitative assessment of approaches according to “Effort for Initial Setup” criterion. 74
Figure 4.8: Point based qualitative assessment of approaches according to “Effort for Initial Setup” criterion. 75
Figure 4.9: Point-based qualitative assessment of traditional document based, static model-supported and executable model-based approaches according to defined criteria. 75
Figure 6.1: Deduction of failure causes and effects from the functional architecture using the static approaches. The proposed method assumes that the upstream functions are possible causes of the primary failure, whereas downstream functions are affected as a result of the primary failure. 80

LIST OF TABLES

Table 1.1: LISA payload instruments and descriptions. 15
Table 1.2: Payload functions for laser link acquisition. 18
Table 3.1: Attributes of Laser Light flow type and their values 39
Table 3.2: The desired format of information of RAMS engineering for the starting of FMEA study is a table that is the list of functions with the entries below. 53
Table 4.1: Input table for the “Laser frequency actuation” function. Compiled either manually using document-based approach or automatically using the model-based approach. 62
Table 4.2: FMEA table for “Laser frequency actuation” failure. First row filled using the static approaches, whereas second row filled using the generic executable approach. Incorrect assumptions of static approaches are given in red colour, whereas additional entries from executable approach are highlighted in blue. 62
Table 4.3: Input table for the alternative architecture “Laser frequency actuation” function. Compiled either manually using document-based approach or automatically using the model-based approach. 63
Table 4.4: FMEA table for alternative architecture “Laser frequency actuation” failure. First row filled using the static approaches, whereas second row filled using the generic executable approach. Incorrect assumptions of static approaches are given in red colour, whereas additional entries from executable approach are highlighted in blue. 64
Table 4.5: Input table for the Internal frequency stabilisation function. Compiled either manually using document-based approach or automatically using the model-based approach. 64
Table 4.6: FMEA table for Internal frequency stabilisation (reference mode) failure. First row filled using the static approaches, whereas second row filled using the generic executable approach. Entries from executable approach are highlighted in blue. 66
Table 4.7: Part of an FMEA table for “Internal frequency stabilisation (reference mode)” failure, filled in using the detailed executable model-based approach. All the 6 instances of the function and where they belong in the constellation are shown in first two columns. The failure effects are captured with their exact location in the constellation (e.g., constellation.sc1.ePMS_L). The cause of failure for the affected functions are also listed next to their respective entries. As anticipated, the functions responsible for stabilizing slave/transponder LAs does not have failure effects as they are not used operationally. 66
Table 4.8: Criteria table showing the comparison criteria with descriptions used for evaluating the different FMEA methodologies. 67
Table 4.9: Description of points of the qualitative assessment. 68
Table 5.1: Recommendations for future work. 78
Table 6.1: Comparison summary table comparing different methodologies of traditional document based, static model-supported and executable model based FMEA approaches with pros [+] and cons [-] indicated. 81

LIST OF ABBREVIATIONS

ad	Activity Diagram
ADS	Airbus Defence and Space
AIT	Assembly-Integration-Test
ALH API	Action Language Helper API
bdd	Block Definition Diagram
CAS	Constellation Acquisition Sensor
CCS	Central Checkout System
CMS	Charge Management System
CST	Cameo Simulation Toolkit
DDMS	Digital Design, Manufacturing and Services
DOORS	Dynamic Object Oriented Requirements System
DWS	Differential Wavefront Sensing
ECSS	European Cooperation for Space Standardization
ePMS	extended Phase Measurement System
ESA	European Space Agency
F_InT	Functional Interface Type
FBD	Functional Break-Down / Functional Breakdown Document
FFMEA	Functional Failure Modes and Effects Analysis
FFT	Fast Fourier Transform
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects and Criticality Analysis
FoV	Field of View
FRS	Frequency Reference System
GRS	Gravitational Referencing Sensor
GRS FEE	Gravitational Referencing Sensor Front-End Electronics
GRSH	Gravitational Referencing Sensor Head
Hz	Hertz
I/O	Input/Output
ibd	Internal Block diagram

IFO	Interferometer
L/R	Left/Right
LA	Laser Assembly
LAS	Laser Acquisition Sequence
LIGO	Laser Interferometer Gravitational-Wave Observatory
LISA	Laser Interferometer Space Antenna
LOA	Long Arm
LS	Laser System
MBSE	Model Based Systems Engineering
MOSA	Movable Optical Sub Assembly
OB	Optical Bench
OBC	On-Board Computer
OBSW	On-Board Software
OMS	Optical Metrology System
par	Parametric Diagram
PDH	Pound-Drever-Hall
PID	Proportional Integral Derivative
PRN	Pseudo Random Number
RAMS	Reliability Availability Maintainability Safety
REF	Reference
R-MOFLT	Requirements-Mission Operational Functional Logical Technical
RX	Receiving
SatSim	Satellite Simulator
SC	Spacecraft
SE	Systems Engineering
SOI	System Of Interest
SRAP	Safety and Reliability Analyzer Plugin
stm	State Machine Diagram
SVF	Software Verification Facilities
SysML	Systems Modeling Language

TDI	Time Delay Interferometry
TEL	Telescope
TM	Test Mass
TM/TC	Telemetry/Telecommand
TX	Transmitting
UML	Unified Modeling Language
USO	Ultra Stable Oscillator
V&V	Validation & Verification

1 INTRODUCTION

This chapter elaborates on the context and the outline of this thesis work. The purpose and motivation to conduct this study is explained in *chapter 1.1.* Information regarding the LISA mission, which the work is applied to, can be found in *chapter 1.2.* An introduction to Model-Based Systems Engineering (MBSE), which is the main theme of the thesis methodology is based on, can be found in *chapter 1.3.* The research objectives of this thesis are summarised in *chapter 1.4.* And finally, the overall outline structure of this thesis document is shown in *chapter 1.5*, with links to the research objectives.

1.1 PURPOSE AND MOTIVATION

For future space missions with complex operational scenarios and stringent performance requirements, it becomes more and more relevant to understand the system behaviour (nominal and in the presence of failures) at early stages in the development lifecycle. Especially for complex space science missions like LISA, where system functions are spread over many system layers and are realised by several elements of the payload, the platform, and even across the different spacecraft in the constellation, an early anticipation of inconsistencies between specifications and system behaviour becomes exceedingly important. The more functional interactions are present in operating a complex system, the harder is it to detect failures and asses their impact on higher level. Moreover, the later fault occurrences appear at higher level in the product development lifecycle, the more their removal will result in cost increase and schedule delay [1, 2].

As an intention to mitigate the aforementioned issues, two main solutions in LISA project have already been adopted:

- In order to cope with complexity and to avoid inconsistency during system definition, Model-Based Systems Engineering (MBSE) solution is implemented as a formalized Systems Engineering (SE) process. Functional modelling with SysML language [3] and Cameo Systems Modeler tool [4] is one aspect of MBSE which is used to describe the system architecture and behaviour in terms of formalized and executable diagrams. This allowed the project team to have a consistent, unambiguous, and conveniently accessible definition of the functions that are needed to do an operational task [5-7].
- To address the identification and analysis of failure cases, Reliability, Availability, Maintainability and Safety (RAMS) discipline have been contacted at an early development phase (Phase A/B1). A strategy is implemented to incorporate failure analyses methodologies, such as functional Failure Modes and Effects Analysis (FMEA), within the presented model-based approach.

However, it was realized that during the early phase development, when detailed information on hardware and software components are still not available, the integration of the RAMS discipline to the MBSE approach is limited. The classical approach to FMEA uses extensive documents review as input to the FMEA study and does not benefit from the modelled system behaviours, operations and traceability links. It is prone to create inconsistencies, repetition of work, and is often limited in identifying failure cases (and their consequences), in particular when system functions are inter-connected between several elements and layers of the system. In addition, once established in a document-based approach, the link between the systems, functions, requirements, and failures needs extensive updates in case of changes in system definition.

On the other hand, using an executable system architecture model to explore failure cases offers the following potential benefits:

- It saves the effort of re-collecting information across multiple documents and experts, in order to perform failure assessments.

- It establishes a consistent link between failures and spacecraft systems, functions, operations, and requirements across different levels of abstraction.
- It enables a better understanding of the behaviour and failure consequences of the investigated spacecraft/satellite (or in case of LISA the constellation consisting of three spacecraft) by injecting intentional failures into the nominal executable behaviours and running operational scenarios. This helps to identify and prioritize the failures that have an impact on the operational tasks instead of a more conservative approach that is independent of spacecraft operations.

Therefore, the main objective of this thesis is to explore and implement approaches that makes use of an (executable) system architecture model to perform a functional FMEA study and to investigate (and if possible, quantify) the benefits of such a model-based approach. The thesis will compare and discuss the proposed model-based approaches with a document-based method in a selected operational scenario within the LISA mission.

1.2 THE LISA MISSION

The Laser Interferometer Space Antenna (LISA) mission is the third large mission (L3) in the ESA Cosmic Vision Science Programme and has a launch date envisaged between 2035 and 2037. The main objective of LISA is to observe gravitational waves in space [8]. Gravitational waves are ripples in the fabric of spacetime that are produced by massive astrophysical events such as the collision of black holes or neutron stars. These waves can provide valuable information about the nature of gravity, the behaviour of matter in extreme environments, and the evolution of the universe. For the first time on 14 September 2015, the gravitational waves were directly measured by ground-based detectors: Laser Interferometer Gravitational-Wave Observatory (LIGO) and the Virgo interferometer [9].

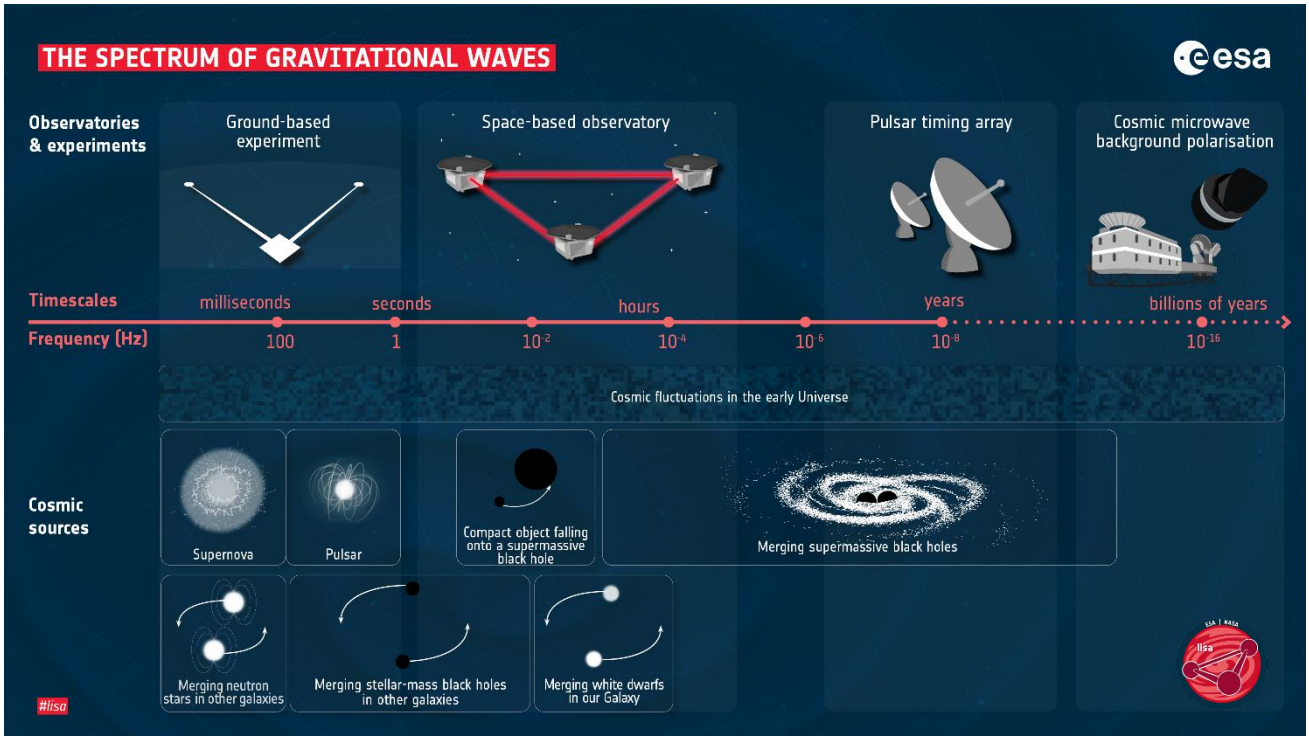

Figure 1.1: Spectrum of gravitational waves showing astronomical bodies and phenomena the LISA mission is aimed to detect. Courtesy of ESA [10]

LISA is designed to detect gravitational waves with a frequency range of 0.1 mHz to 1 Hz, which is lower than the frequency range observed by ground-based detectors and cannot be measured on-ground due to omni-present seismic noise. This low frequency range makes LISA particularly well-suited for observing supermassive black holes, which are thought to be located at the centres of most galaxies (see *Figure 1.1*).

The space segment of the LISA mission is a constellation of three identical spacecraft, flying in an equilateral triangular formation with 2.5 million kilometres arm length in between. Each spacecraft is equipped with two cubic test masses isolated from the outside environment in a free fall. By exchanging bi-directional laser beams in between each other along the constellation arms, the spacecrafts form a Michaelson interferometry setup. This setup enables measuring the relative test-mass distance to picometer accuracy, which, due to the extreme stiffness of space-time, is required to detect the gravitational waves at lower frequencies than the ground-based detectors [11]. The laser links and free-falling test masses (see *Figure 1.2*) are one of the factors that makes this mission unique from other space missions, as it requires highly collaborative functioning of all three spacecrafts and their payload and platform systems to achieve the science objectives.

1.2.1 Gravitational Wave Measurement Principle in LISA Mission

The principle of gravitational wave measurement in the LISA mission is based on the use of laser interferometry to detect the tiny changes in the distances between free flying test masses located inside the three spacecrafts, arranged in a nearly equilateral triangle in space [8]. The measurement principle of LISA is based on the fact that a passing gravitational wave causes a small, but measurable change in the relative distances between the test masses that act as end mirrors of a large laser interferometer that is spanned by the three spacecraft. The lasers are used to measure the distances between the spacecraft by sending (TX) and receiving laser (RX) beams, which are reflected by the test masses and interfered with each other. The change in distance is detected by comparing the phase difference between the laser beams at different times. The phase difference is measured by interferometry, which involves recombining received laser beam with the local beam to produce an interference pattern. When a gravitational wave passes through the LISA constellation, the phase of the beat note signal (a sinusoidal modulation of the detected laser intensity) changes as a function of time, allowing the tiny changes in distance to be detected. The LISA mission uses a combination of interferometry and time-delay interferometry (TDI) to measure gravitational waves. TDI is a technique that combines data from multiple interferometers to remove the effects of laser frequency noise and spacecraft motion arising from the difference in length of the interferometric inter-spacecraft links. By making use of the measured absolute distances between different spacecraft and synthesizing interferometric signals in post-processing, TDI can replicate the effect of a Michelson interferometer with equal arm lengths [11].

1.2.2 The LISA Spacecraft Payload

The LISA spacecraft (SC) payload is mainly based on an Optical Metrology System (OMS) and a Gravitational Reference Sensor (GRS) System to achieve the interferometric gravitational wave measurement goal. The OMS is composed of:

- Telescope (TEL)
- Optical Bench (OB), also hosting the Constellation Acquisition Sensor (CAS)
- Laser System (LS), including Laser Assembly (LA) and Frequency Reference System (FRS)
- Extended Phase Measurement System (ePMS).

Another central part of the LISA payload is the Gravitational Reference Sensor (GRS) System consisting of:

- GRS Head (GRSH)
- GRS Front-End Electronics (GRS-FEE)
- Charge Management System (CMS)

The roles of the listed payload elements are described in the *Table 1.1*. Some core payload elements are mounted in the so-called Movable Optical Sub Assembly (MOSA), comprising TEL, OB and GRSH. This assembly is movable in one rotational degree of freedom to be able to change the angle between the two arms of a spacecraft, thereby tracking the seasonal dynamics of the LISA constellation. There are 2 MOSAs (Left and Right) in each spacecraft and most of the payload elements are only responsible for the operation of their own local MOSA unit, which can be seen in the schematic in Figure 1.2.

Table 1.1: LISA payload instruments and descriptions.

Payload Instrument	Description
Laser Assembly (LA)	Generates, controls and modulates laser light. Contains the Laser Heads which feed laser light to the OB.
Frequency Reference System (FRS)	Serves as a reference for stabilizing the frequency of the generated laser light. It is based on an optical cavity, consisting of two mirrors with a highly stable spacer in between. The eigenmodes formed between the mirrors serve as the optical frequency reference. The frequency of the laser light is stabilized using the Pound-Drever-Hall (PDH) locking technique [12].
Optical Bench (OB)	Hosts 3 interferometers (IFO) called Reference (REF IFO), Long-Arm (LOA IFO) and Test-Mass (TM IFO) interferometers, mechanisms, photoreceivers and photodetectors. • REF IFO: Interferes laser lights from local LA and the LA of the neighbouring MOSA. It is used to lock the local lasers feeding the two MOSAs to one-another. • LOA IFO: Interferes laser light from local LA and RX light from LA of the remote spacecraft at the other end of the long-arm. It is used to measure the distance fluctuations in-between the two spacecrafts. • TM IFO: Interferes laser light from local LA and laser light reflected from test-mass. It is used to measure the distance fluctuations between the free-floating test-mass and the local spacecraft.
extended Phase Measurement System (ePMS)	Extracts the interferometric signals from the IFOs and provides control signals for the lasers, based on the frequency of the interferometric beat signals.
Telescope (TEL)	Serves as a laser beam expander/compressor to collect the RX light and emit TX light.
Constellation Acquisition Sensor (CAS)	Detects the laser beam from the remote spacecraft to support the alignment of two spacecrafts' MOSAs onto each other, which is necessary for the laser link acquisition.

GRS Head (GRSH)	Hosts the test-mass in an electrode housing mounted in a dedicated vacuum chamber, isolating the test-mass from external disturbances such as solar radiation pressure.
GRS Front-End Electronics (GRS-FEE)	Controls and detects the test-mass attitude and position electrostatically.
Charge Management System (CMS)	Controls the test-mass charge.

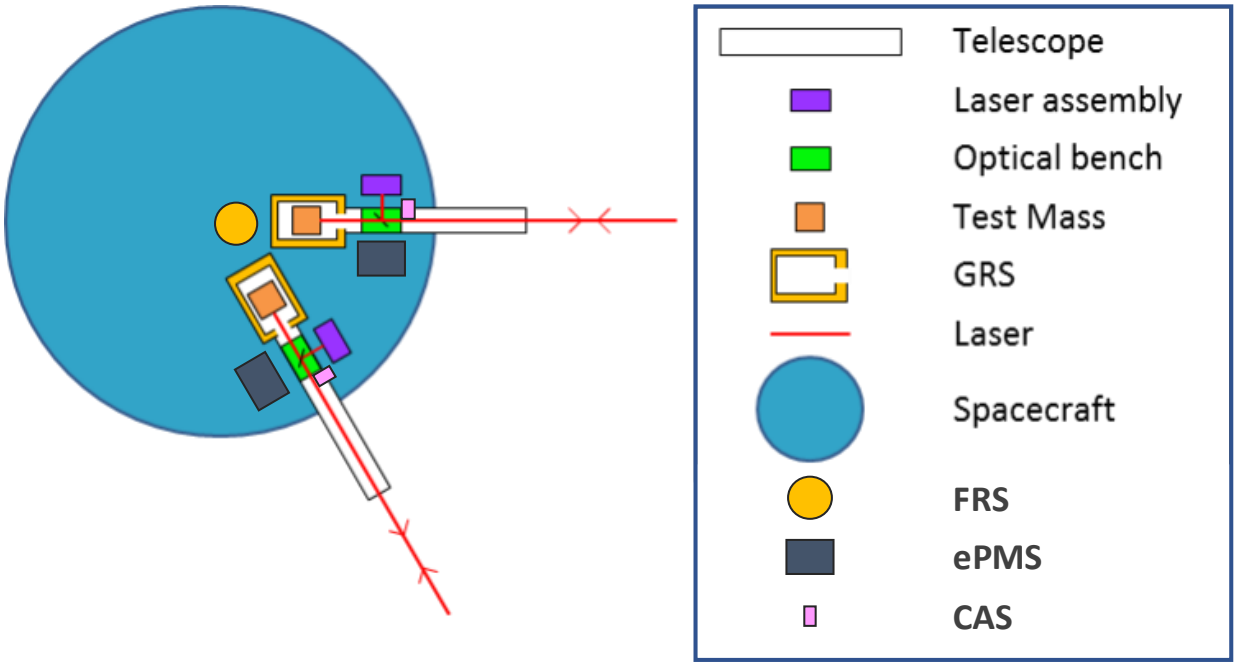

Figure 1.2: Schematic of a LISA SC with the two MOSAs and the most important instrument units. Modified from Charpigny [5].

1.2.3 The Laser Acquisition Sequence

One of the most important and complex operations in the LISA mission is the acquisition of the laser links between the three spacecrafts in the constellation, involving various payload instruments and functions (see *Table 1.2*). The laser links enable detection of the gravitational waves using laser interferometry and also enable data transfer via modulation of phase of the laser light. This auxiliary data retrieved from the laser links consists of science telemetry data, absolute ranging information and clock noise signals that are used to post process the science data to make it available for analysis.

In order to enable this data exchange, first, the bi-directional laser beams have to be physically aligned such that the MOSAs on the two opposing spacecrafts point to one another. This process is called the geometric acquisition and is a prerequisite to detect interferometric signals. During geometric acquisition, each spacecraft searches for the remote spacecraft using a scanning manoeuvre. Once it detects the laser beam from the remote spacecraft on its CAS, the position of the RX spot on the CAS detector is used to adapt the attitude of the spacecraft and its MOSA. This results in both spacecrafts on each end to align themselves according to CAS readouts to establish the laser link. CAS is a camera with a large Field of View (FoV) and designed to be strictly co-aligned with the narrow FoV of the interferometric photoreceivers. Therefore, when the RX laser beam spot position is properly aligned on the CAS, it also aligns on the photoreceivers. This enables a spatial co-

alignment of the RX beam with the local TX beam on the LOA IFO photodetectors, allowing interferometric signals to be detected. This process is done for all three spacecrafts of the constellation to complete the geometric acquisition operation.

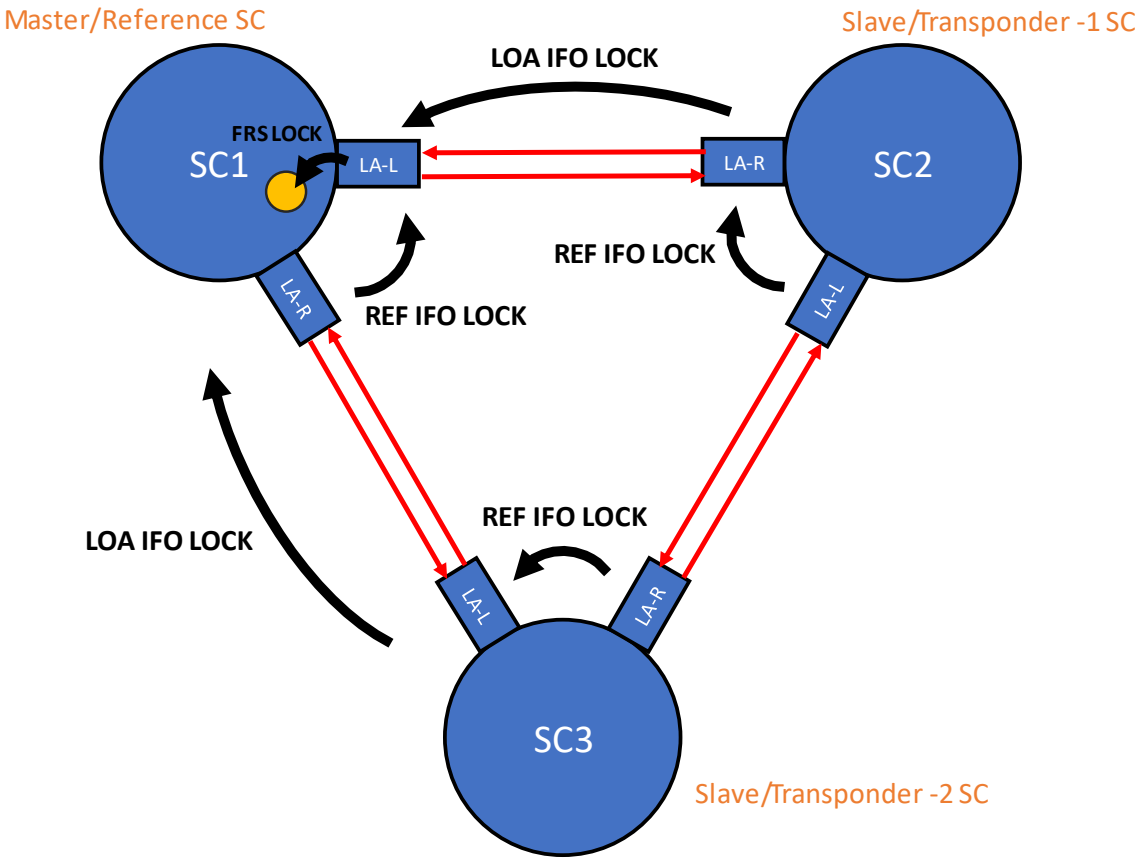

Figure 1.3: LISA constellation view showing laser links and locking scheme.

However, the geometric acquisition is not enough to establish a datalink or science measurement in the constellation. The laser beam frequency is too noisy to enable exchange of data or extracting picometer length measurements, therefore must be stabilized. This is done by the process called laser frequency locking onto a stable frequency source. Each spacecraft is equipped with an optical cavity in its FRS, establishing an appropriate reference for LAs to stabilize their laser frequencies. In the constellation, the FRS of only one of the spacecrafts is active at a time, which becomes the “master/reference spacecraft”. This spacecraft hosts the LA stabilized onto the FRS unit, which is called the “master/reference LA” of the constellation. Every LA in the constellation eventually must align its laser frequency according to this reference. This alignment, called laser frequency locking, is a Proportional-Integral-Derivative (PID) control-actuation loop in the LA, which replicates the laser frequency of the reference it is following, with a desired frequency offset. Once the control loop achieves the desired laser frequency, and the corresponding PID control loop is closed, the laser is termed to be "frequency locked".

There are three different frequency locking ways in the constellation.

- In the master/reference spacecraft, one of the LA frequency locks onto the optical cavity inside the FRS. This LA is now referred as the master/reference LA and the process is called “master/reference locking”.

- The LA can frequency lock onto the neighbouring LA, by comparing the beat note between the local and the neighbouring laser. This process is called as “slave/transponder mode - REF IFO locking”, because it uses control signals which are provided from the ePMS and derived from the interferometric signal of the reference interferometer.
- The LA can frequency lock onto the LA inside the remote spacecraft on the other end of the laser link, by comparing the beat note between the local and the remote laser. This process is called as “slave/transponder mode - LOA IFO locking”, because it uses control signals which are provided from the ePMS and derived from the interferometric signal of the long-arm interferometer.

Once the locking chain for all LAs involve the reference/master LA, the laser light frequency for these LAs become stabilized. An example of a locking sequence is explained below and in *Figure 1.3*. Note that a dot notation is used in the explanation as “SC.LA”, implying the owner of the element from right to left of the dot.

1. Inside the SC-1, LA-L frequency locks onto the optical cavity inside the FRS. Therefore, SC-1 becomes the “master/reference SC” and SC-1.LA-L becomes the “master/reference LA”.
2. Inside the SC-1, LA-R frequency locks onto LA-L using the slave/transponder mode - REF IFO locking process.
3. SC-2.LA-R frequency locks onto SC-1.LA-L using the slave/transponder mode - LOA IFO locking process. Since SC-2 is the first spacecraft to establish the LOA IFO lock with the master/reference spacecraft, it becomes the “slave/transponder – 1 SC”.
4. SC-3.LA-L frequency locks onto SC-1.LA-R using the slave/transponder mode - LOA IFO locking process. Since SC-3 is the second spacecraft to establish the LOA IFO lock with the master/reference spacecraft, it becomes the “slave/transponder – 2 SC”.
5. Inside the SC-2, LA-L frequency locks onto LA-R using the slave/transponder mode - REF IFO locking process.
6. Inside the SC-3, LA-R frequency locks onto LA-L using the slave/transponder mode - REF IFO locking process.

This example locking sequence deems all of the 6 LAs to become stable in frequency, therefore the science measurement and data exchange can happen, completing the laser acquisition sequence.

Table 1.2: Payload functions for laser link acquisition.

Payload Instrument	Function
CAS	Principle task of acquisition (Providing centroiding information for the RX beam)
FRS	Internal frequency stabilization (reference mode)
LA	Light Generation Intensity Control Phase modulation Power stabilization on OB External frequency control (transponder mode) Laser frequency actuation
ePMS	IFO parameter extraction from carrier beatnotes Retrieval of PRN code delay and data Transponder beat note acquisition Extraction of DWS angles from LOA and TM IFO Demodulation of sideband-sideband beatnotes and clock noise retrieval

	PRN code generation Generation of upscaled USO frequencies FFT peak detection algorithm
OB	Laser Beam acceptance for emission to remote SC Laser Beam acceptance for local interferometry Laser Intensity Measurement on OB Long-arm interferometer Reference interferometer Support for RX beam centroiding
TEL	Beam Expander/Compressor

1.3 MODEL BASED SYSTEMS ENGINEERING

This chapter explains the model-based systems engineering approach with a general overview in *section 1.3.1*, and description of the tools and language that is used in the thesis work in *section 1.3.2*.

1.3.1 General Overview of MBSE

Model-Based Systems Engineering (MBSE) is a methodology that uses models to represent and analyse complex systems, and to support decision-making throughout the system lifecycle [13]. MBSE is based on the idea that a model is a more efficient and effective means of communicating information about a system than traditional documents, diagrams, or other forms of textual description. In MBSE, the system, its components and functions are represented using models that can be visual, mathematical, or both. The models can be established using a variety of modelling languages and tools, including SysML, UML, Arcadia/Capella, MATLAB, Simulink, etc.

In MBSE approach, the models can be used to define, simulate, and analyse system architectures, and behaviours under various conditions and to test the system's performance and/or functionality early in its lifecycle. Once established, these models become the main artefacts used to communicate information about the system to various stakeholders, including engineers, managers, customers, and regulators.

One of the key advantages of MBSE is that it promotes a holistic approach to engineering with the philosophy of “single source of truth”. By representing the system and its components as interconnected models, MBSE allows engineers to design, analyse, and communicate the system definition as a whole and to identify and resolve potential conflicts or issues early in the design process. The connected way of design and communication through models ensures consistency and high level of awareness in the project team, leading to more efficient and effective system designs and reducing the risk of costly errors or delays later in the development process [14].

1.3.2 SysML and Cameo Systems Modeler

SysML (Systems Modeling Language) is a graphical modeling language used for system engineering applications [3]. It is an extension of the Unified Modeling Language (UML) and is specifically designed to support the modeling of complex systems. SysML provides a standard set of graphical symbols and diagrams that can be used to represent the different aspects of a system and its components. The aspects are classified as structure, behaviour, requirements, and parametrics. Some of the common SysML diagrams include (see *Figure 1.4*):

- Block Definition Diagram (bdd): provides a high-level structural view of the system and its components. Commonly referred as Blackbox view.

- Internal Block Diagram (ibd): represents the internal structure of a system and the interactions between its components. Ibd provide a more detailed view of the system than bdd and commonly referred as Whitebox view.
- Activity Diagram (ad): represents the behaviour of a system, including its processes and activities. They are particularly useful for modelling the flow of information and control between different components of a system.
- State Machine Diagram (stm): represents the behaviour of a system or component using discrete events reflected as states and transitions. The states represent different conditions or modes that the system or component can be in, while the transitions represent events or actions that cause the system to move from one state to another.
- Parametric Diagram (par): represents the relationships between the parameters of a system or component. It shows how changes in one parameter can affect other parameters and ultimately the overall behaviour of the system through constraints.

The contents of these SysML diagrams, called model elements, are interconnected. This interconnectedness allows for the easy traceability of requirements, behaviours, and functions of the system being modelled. By linking the model elements, SysML enables users to access a wealth of additional information that can be used to verify that the system is meeting its requirements and to identify any issues that may arise during the development process. This level of traceability also allows for the efficient management of complex systems, making SysML a powerful tool for engineers and systems architects alike.

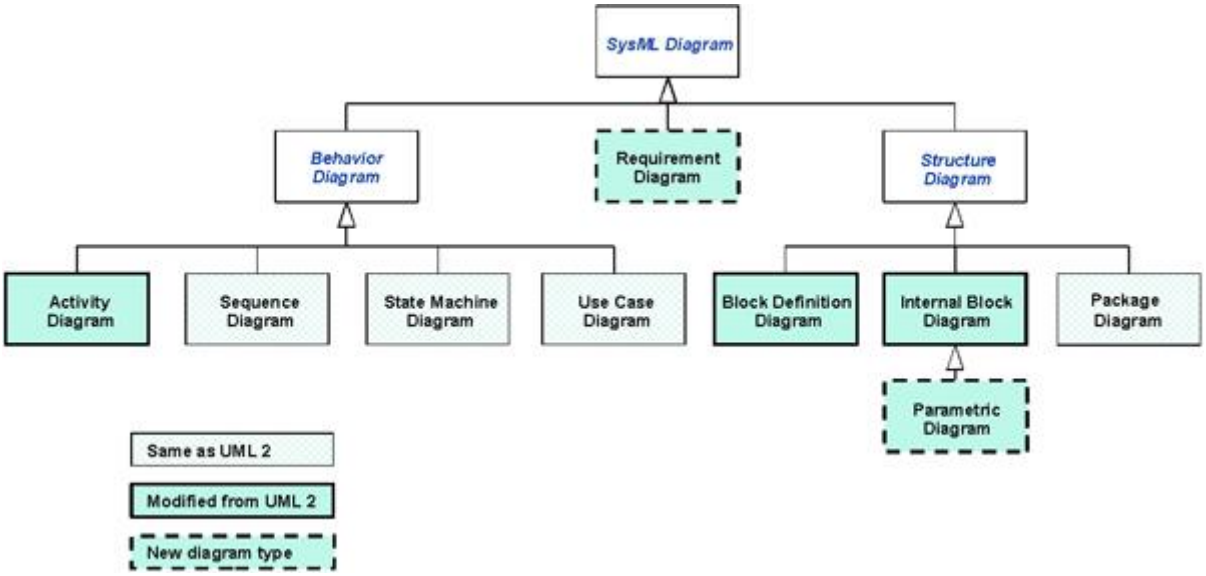

Figure 1.4: Overview of SysML diagrams. Courtesy of Object Management Group [3].

Cameo Systems Modeler is a powerful modelling tool that is widely used in the field of model-based systems engineering (MBSE) and supports the Systems Modeling Language (SysML). It is a part of the CATIA product family from Dassault Systèmes and adopted in Airbus Defence and Space (ADS) as the main tool for early phase system architecture definition. The main capabilities of the tool involves collaborative modelling, simulation and analysis, traceability, and reporting and documentation [4]:

- Collaborative Modelling: Cameo Teamwork Cloud tool provides collaborative modelling capabilities by allowing multiple users to access and work simultaneously on the same model

stored in a cloud server. This can help to improve teamwork and collaboration, as well as reduce the risk of errors and inconsistencies that may arise from working on multiple versions of the same model.

- Simulation and Analysis: Cameo Simulation Toolkit provides the ability to simulate and analyse models, allowing engineers to test the behaviour of a system swiftly without the effort of exporting to external software. This can help to identify potential issues and conflicts early in the design process.
- Traceability: Cameo Datahub tool provides traceability features that enable engineers to track the relationships between different elements of a model and requirements management software (e.g., IBM DOORS [15]). As an example, a system function in a Cameo model can be traced to a specific customer requirement in DOORS. This traceability feature allows to ensure that the requirement is properly addressed in the design and that any changes made to the requirement are reflected in the system function.
- Reporting and Documentation: Through report wizard, it is possible to generate reports and documentation from models, making it easier to communicate the design to stakeholders.

1.4 RESEARCH OBJECTIVES

The main objective of this thesis is to explore and implement approaches that makes use of an (executable) system architecture model to perform a functional FMEA study and to investigate (and if possible, quantify) the benefits of such a model-based approach. The thesis will compare and discuss the proposed model-based approaches with a document-based method in a selected operational scenario within the LISA mission. In particular, the thesis intends to find answers to the research questions stated below:

1- How can a traditional document based functional FMEA process be improved using the same input information but benefiting from a system architecture model implemented in MBSE environment, instead of documents?

2- How can behavioural analysis and critical operational scenarios be integrated into a functional FMEA study performed using the executable system architecture model?

3- What would be the difference between traditional (document based) FMEA, and model based FMEA approaches in terms of:

   - Analysis results
   - Documentation process
   - Process in case of update in system design assumptions
   - Impact of operation scenario on failure cases

4- From the investigated approaches, which approach would be the suitable one to be implemented in a complex space mission project like LISA? What would be the benefits?

1.5 RESEARCH STRUCTURE

The research structure of this thesis work is explained in this chapter and summarised in *Figure 1.5*.

Chapter 2 provides an elaboration on the state-of-the-art information in the literature. It investigates Functional Failure Mode and Effects Analysis (FMEA) utilized for functional failure mode detection and recovery analysis in the European space industry. Additionally, the chapter explores research related to MBSE assisted FMEA, including different approaches to the problem and identified gaps.

Furthermore, the LISA Mission System Architecture Model is introduced, with an examination of its capabilities and gaps throughout its deployment history. Lastly, the chapter explains the state-of-the-art outcomes and research contributions aimed at filling the gaps in the literature.

Chapter 3 delves into the methodology pursued to implement an early phase functional FMEA study by leveraging the LISA Mission system architecture model. It describes a proposed approach for the modelling of executable functional architecture, followed by an explanation of the execution and simulation implementation of this model to answer *Question 2-* of the research objectives. Subsequently, the chapter presents three different approaches for conducting a Functional FMEA study within a defined scope in the LISA mission, namely the traditional document-based approach, the static model-supported approach, and the executable model-based FMEA approach, elaborating the improvements using the model and answering the *Question 1-* of the research objectives .

Chapter 4 is dedicated to presenting the results of the work performed to compare the three FMEA methodologies (traditional document-based, static model-supported, and executable model-based), answering *Questions 3- and 4-* of the research objectives. Additionally, the chapter highlights the contributions of the thesis work to the investigated problem.

Chapter 5 focuses on potential areas for future work related to the thesis topic. It provides a compilation of recommendations for the further development of the thesis, along with suggestions for improving the modelling process and the integration of RAMS into MBSE methods and procedures.

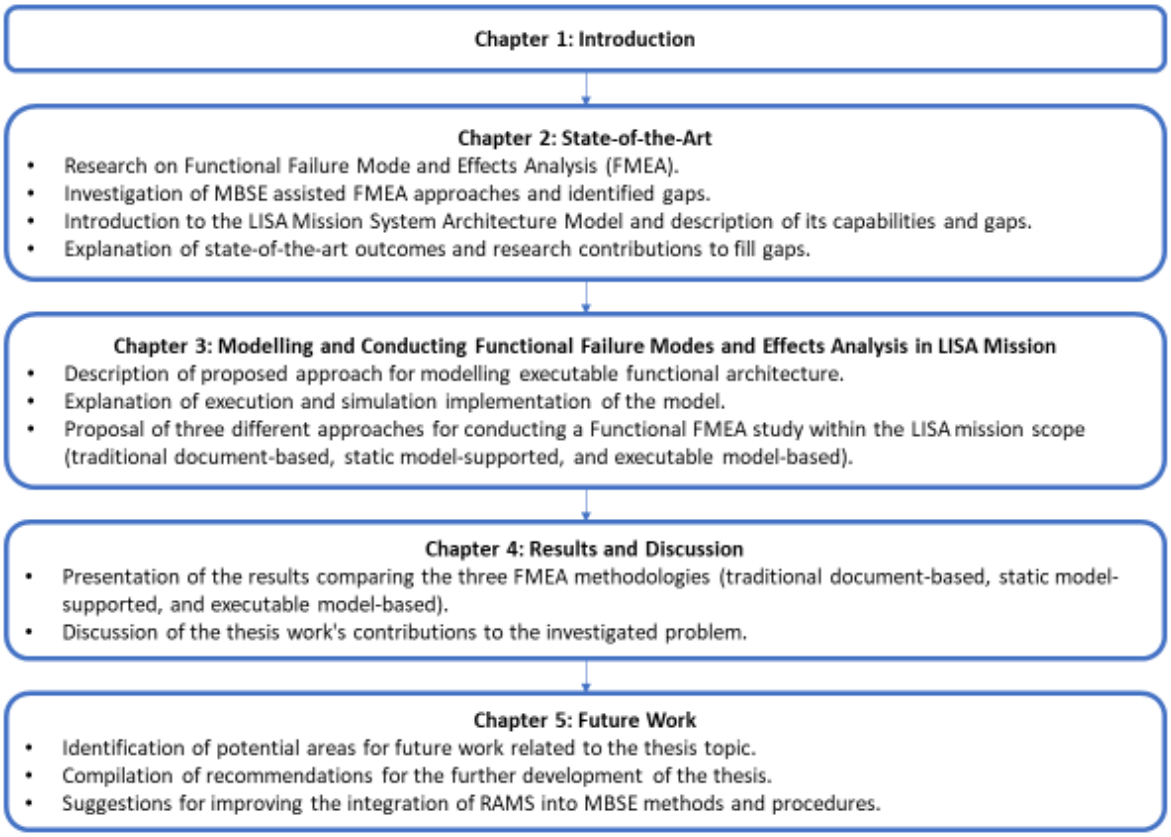

Figure 1.5: Flowchart showing the structure of the thesis work.

2 STATE OF THE ART

This chapter elaborates on the state-of-the-art information in the literature. In *chapter 2.1* Functional Failure Mode and Effects Analysis is researched with the process depicted for ECSS-Q-ST-30-02C standard [16], which is used for functional failure mode detection and recovery analysis in the European space industry. In *chapter 2.2,* research related to MBSE assisted FMEA is investigated with the description of different approaches to the problem as well as identified gaps. In *chapter 2.3,* the LISA Mission System Architecture Model is introduced, with the investigation of its capabilities and gaps throughout its deployment history. Finally in *chapter 2.4,* the outcomes of the literature review and the research contributions aimed to fill in the gaps in the state-of-the-art are explained.

2.1 FUNCTIONAL FAILURE MODES AND EFFECT ANALYSIS

Functional Failure Mode and Effects Analysis (FFMEA) is a methodology used in systems engineering to identify potential functional failure modes of a system with their causes and effects [17]. Functional FMEA is a type of FMEA that focuses on the functional requirements of the system. This approach is particularly useful when the system design properties are not known, or uncertain, especially in the early phases of the development lifecycle. Since the functional description is one of the most stable definition of a system, it is of high interest for engineers to analyse the potential failure cases on this baseline before the system is designed/built, to avoid and mitigate any fundamental errors in later phases [1].

In the context of space applications, functional FMEA is often used to assess the reliability and safety of spacecraft and their subsystems. As an example, the ECSS-Q-ST-30-02C is a standard that provides guidelines for performing Failure Modes and Effects Analysis (FMEA) [16]. The FMEA requirements section of this standard outlines the general requirements for conducting FMEA, which can be summarized as follows:

1. Identify the system functions: The first step in conducting FMEA is to identify the functions of the system and break them down into their lowest level (“leaf”) functions and components. The description, interfaces, interrelations and dependencies, as well as operational/mission link has to be defined.
2. Identify potential failure modes: For each function, identify all potential failure modes that could occur. Burge [17] defines potential failure modes with proverbs to the function itself such as Over/Under/No/Intermittent/Unintended functioning.
3. Analyse the causes/effects of failure: Analyse the causes and effects of each potential failure mode on the system, including its impact on safety, reliability, and mission success. This analysis can be done with the help of the functional architecture and the functional breakdown.
4. Integrate findings: According to causes and effects of each failure mode, integrate them along the functional hierarchy and architecture to create a consistent definition of failure cases along the system abstraction layers.
5. Rank failure modes: Rank the potential failure modes according to their severity, occurrence probability, and detection capability, to prioritize those that require mitigation.
6. Identify mitigation measures: For each identified failure mode, identify and implement mitigation measures that reduce the risk of the failure occurring or minimize its effects.

It is also important to note that the analysis is done on the same system layer (horizontal) and also along the system layer hierarchy (vertical). For example, horizontal investigation of a system layer can lead to the identification of failure propagations along various functions and items in one level, even if they are not part of the same hierarchical breakdown branch (see *section 4.4b, 4.6b and 4.6c in* [16]). On the other hand, the vertical integration of the failure modes establishes a consistent traceability of the failures to the higher levels of abstraction, where severity and impact of a failure on

operations and eventually the mission is more apparent (see *section 4.5 in* [16]). This way of integration enables the consistency in between levels of abstractions and full coverage of the system definition for the failure cases.

2.2 MBSE ASSISTED FAILURE MODES AND EFFECTS ANALYSIS

As the traditional systems engineering processes are transitioning toward the model-based approach for creating systems definitions, it is more and more desired to integrate the FMEA activities into this way of development methodology. Especially in the aerospace systems domain, integration of MBSE and FMEA is an interesting topic of research for the development teams due to the complexity involved. A selection of studies that offer distinct solutions regarding MBSE and FMEA integration are investigated in this chapter.

Biggs et.al. [18] established a data model and a standard for the integration of safety analysis to SysML language, creating the baseline of the Cameo Safety and Reliability Analyzer Plugin (SRAP) [19]. The standard makes use of additional model elements, representing failure modes, effects, failure causes etc. and allows users to fill up FMEA tables inside the model. The use of additional model elements for safety and reliability analysis creates a separate viewpoint in the model, decoupling it from the model elements used for the definition of the systems. The use of this decoupled approach enables:

- direct linking of the failure analysis elements to system definition elements without altering the properties of the system definition,
- separation in model structure for experts to conveniently focus on their own related artefacts (e.g., safety and reliability experts work more on the SRAP model elements than the ones used for system definition).

The links between the system elements and FMEA elements gives a strong traceability and overview that is not present in paper-based format. However, there are significant gaps identified in the proposed approach:

- The standard does not offer any automation in terms of deduction of failure causes and effects, nor making use of the modelled system behaviours and/or architecture.
- The standard requires a model element to be generated for each property in the failure analysis. Every different kind of failure mode, effect, cause, etc. must be reflected as a new model element, instead of a new property/attribute to one existing FMEA element. This creates high effort and clutter in the model, if the failure analysis involves specific failure modes, effects, and causes per each system definition (see *Figure 2.5*).

Francesco [20] performs several RAMS analyses including FMEA with an MBSE approach for the A320 flight control system with alternative architectures. The analysis makes use of state machines for the system elements that include both nominal and failure states with their transitions and respective behaviours inside, as failure causes and effects. The use of failure state also effects the overall system behaviour since by execution the failed system will also not behave as intended. However, there are several gaps identified with the proposed solution in the research:

- The approach does not identify the propagated effects of a specific failure along the entire system architecture.
- Although modelled elements are mapped to the FMEA table entries by description, a solution on how to compile the entries automatically using the model is not offered.

Girard et.al. [21] makes use of SysML modelling and an external software SmartIflow, to generate an automated FMEA study. The procedure in modelling is the use of state machine diagrams for each system:

- The states represent nominal and failure modes.
- The transitions to failure modes correspond to system faults.
- The behaviours defined inside the failure mode states correspond to the failure effects.

The SysML model is then imported to smartIflow software, which runs a simulation called “unfolding” to generate an FMEA table. The approach traverses through every possible state transition that can happen until ending in a final state and records this sequence. The sequences that end in a failure mode state corresponds to a so called “failure chain” and therefore are represented in the FMEA table. While offering a comprehensive identification of possible failure cases, the gap identified in the proposed solution is that it does not include the effect of the operation of the systems on the identified failure chains.

Schummer and Hyba [22], proposes a data path and anomaly tracing approach to identify failure propagations using a SysML model and queries implemented in Neo4j database tool [23]. The system architecture is modelled using an ibd with:

- SysML: Part Property representing systems,
- SysML: Proxy Port representing system interfaces,
- SysML: Itemflow representing exchanged items and telemetry parameters.

The model is then transferred to Neo4j database for analysis. The queries search through the data extracted from itemflow information from the model, that shows where a specific item/parameter traverses through the system elements in the architecture. This allows to identify potentially effected elements due to faulty exchanges through system interfaces. While offering a convenient solution to trace failure propagations, the approach does not take into account how a faulty input affects the system behaviour. For example, the faulty parameter could:

- cause the failure of the system therefore fully deeming system output invalid/faulty,
- or without affecting the system functioning it can still be carried on as a part of the system output,
- or without affecting the system functioning it can be cancelled out in the system output.

These behavioural aspects are significant in identifying the impacts of failures on overall system, therefore important to address.

Mhenni et. al. [24] offer an Functional FMEA table generation approach by extracting Activity diagram data from the model, which is then completed by a safety expert. In this approach, the system functions are modelled as nested activity diagrams, with SysML: Action Input/Output Pin depicting functional interfaces and SysML: Object Flow reflecting exchanged functional parameters, forming a functional behavioural chain. The FMEA table is then generated by applying the following steps:

- All the lowest level “leaf” functions are extracted from the diagrams and listed into a table.
- Generic failure modes (e.g., “Fails to perform”, “Operates inadvertently” etc.) are listed for the functions.
- Failure causes and effects are listed as input-output pins and/or upstream-downstream activities.

Automatic filled table is then evaluated and finalized by the safety expert. The approach offers a practical solution to automatically compile an FMEA table from modelled behaviour data. However, several gaps have been identified in the approach, including:

- The connection between failure modes and failure effects is not addressed by the model.

- The approach does not capture the effects of relatively complex modelled functional behaviours such as feedback loops, conditional behaviours etc. on a failure case.

2.3 LISA MISSION SYSTEM ARCHITECTURE MODEL

This chapter explains the implementation of the MBSE model, named as the LISA System Architecture Model, within the project team in ADS. The overview of the model, involving the history and strategy is explained in *section 2.3.1.* The Airbus R-MOFLT methodology used in the model is described in *section 2.3.2.* Information on the content of the model can be found in *section 2.3.3.*

2.3.1 LISA Mission System Architecture Model Overview

LISA is among the largest planned science missions within the next years. It is characterized by a complex mission architecture that requires the interaction of numerous elements at many levels in a system of systems. In order to manage this complexity, LISA team in ADS employed the MBSE approach as a part of their development process. Due to the complexity of the project, a behavioural analysis capability has been set up to allow understanding and analysis of the implemented artefacts and their impacts on higher levels in the system abstraction layers and different viewpoints.

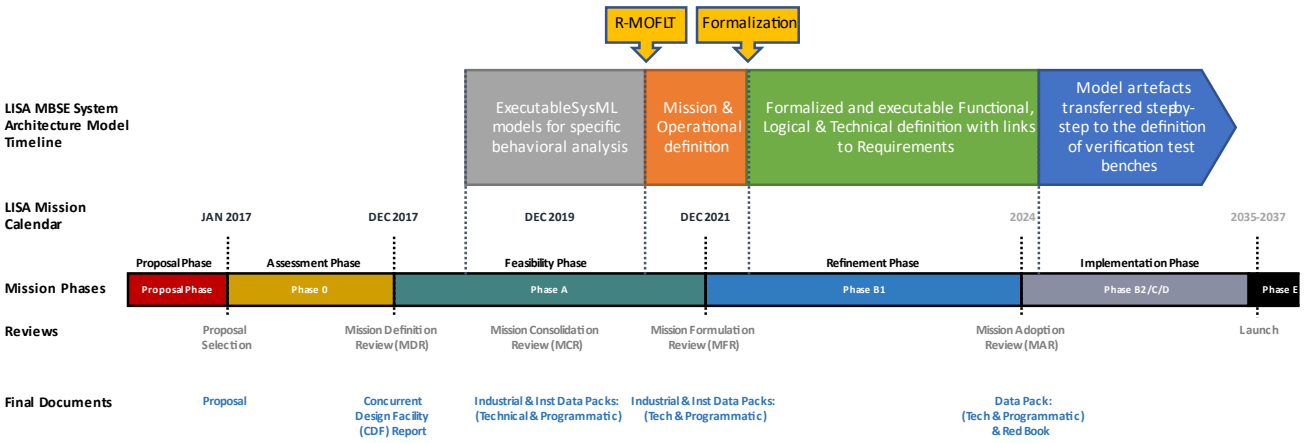

Figure 2.1: LISA System Architecture Model implementation timeline with LISA Mission timeline. Modified from ESA [10].

The evolution of the model through mission timeline can be observed in the *Figure 2.1*. The first models focused on analysis and simulations for specific cases [5-7] and did not implement formalized system definitions. This resulted in the obsoletion of the models once the analysis is finalized. Therefore, the project team pursued a more generalized use case by transferring documentation into the model to create a centralized definition of the system using the Airbus R-MOFLT methodology. With this approach, the model became the formalized documentation source (or interface) for a defined scope of operations, functions, systems and associated requirements, and this definition is used for simulations and deliverables as a by-product. From there on, the activities on the model pursued this aim especially in late A and B1 phases: documenting and generating (inputs to) deliverables using the model where the simulations are implemented as a support to experts and/or for the automation of artefact generation. It is important to note that the goal and the scope of the model needed to be clearly defined at an early stage, otherwise the aim to generate a holistic system description could have become an endless effort [25]. In the case of the LISA system architecture model, the scope was set as:

- Functional description of the complex payload and how it interacts with the platform and constellation.
- Ability to trace system functions to functional requirements.
- Generating inputs for critical operations definition.
- Generating inputs for Functional FMEA (the topic of this thesis work).

The implementation of the executable LISA system architecture model in this set scope has therefore addressed the gaps in the existing SE process and methodologies for the LISA project team. As a future outlook, the model is envisaged to be used until phase B2, where the mission-operation-function-system and requirement definitions are finalized. Then, it will be used to transfer the system definition to the suppliers and so called “verification benches” either via document excerpts or as a model export if desired. The verification benches comprise:

- High-fidelity numerical simulators: e.g., flight dynamics, environment models, detailed equipment models including Telemetry/Telecommand (TM/TC) interfaces.
- Software Verification Facilities (SVF): where an emulated On-Board Computer (OBC) is used with real On-Board Software (OBSW) for software testing.
- Satellite Simulators (SatSim): In the SatSim, a "copy" of the SVF is connected to the Central Checkout System (CCS) operated by Assembly-Integration-Test (AIT) team. It is at the same time used for training.
- Full Electrical Functional model: Verification is finally done on a full Electrical Functional model, where specific equipment mathematical models are replaced by real hardware models.

The model artefacts will be used as a baseline to define the verification plan and the verification bench contents, being a part of a consistent definition chain between requirements, design, analysis, and verification activities.

Still, the model is always a key part to communicate with various stakeholders (e.g., suppliers, customer etc.) and as a reference throughout the development lifecycle. It acts as a user interface for the experts to access desired information conveniently, and to train newcomers to the project in a more visualized and holistic view to the systems.

2.3.2 R-MOFLT Architecture Framework

The Airbus R-MOFLT (Requirements - Mission, Operation, Functional, Logical and Technical) framework is a baseline in the definition of system architecture models used for developments of complex space systems in early phases (Phase 0/A/B1). The framework provides the methodology and the tool to support the development of systems with an MBSE approach:

- Methodology: Step-by-step and top-down development process to create a system definition using MBSE.
- Tool: Profiles and automations for Cameo Systems Modeler to customize the model elements and add features according to the needs of the project, supporting the methodology.

The methodology is based on the fact that a system definition consists of multiple viewpoints (Requirements, Mission, Operation, Functional, Logical, Technical) that can be implemented fully or partly to the whole spectrum of system abstraction levels of interest. The viewpoints and abstraction levels are interconnected and show different aspects of the system to different stakeholders, e.g., subsystem functional integration team is interested more on the Functional viewpoint on subsystem level, whereas operations team is on the Operation viewpoint on system level, and RAMS team focuses more on the Functional viewpoint on multiple system levels for FFMEA. The methodology

therefore guides the experts to fill in some key artefacts in all the viewpoints for the whole abstraction levels in scope, creating an organized and holistic system definition. The key artefacts for the viewpoints can be seen in *Figure 2.2*. The Requirement viewpoint is integrated to the model elements in all levels and in all viewpoints using model relationship links, enabling full traceability to system specifications.

R-MOFLT methodology focuses more on the consistent and full system definitions rather than executable models. The major aim is to make diagrams easy to read and understand rather than compatible for execution. The R-MOFLT models use majority of the SysML diagrams for its viewpoints:

- activity and state machine diagrams for behaviour definitions
- internal block diagram for architecture definitions
- block definition diagram for structure and hierarchical definitions

Usage of parametric diagrams are not common in the methodology, as they are more specialized for simulation purposes.

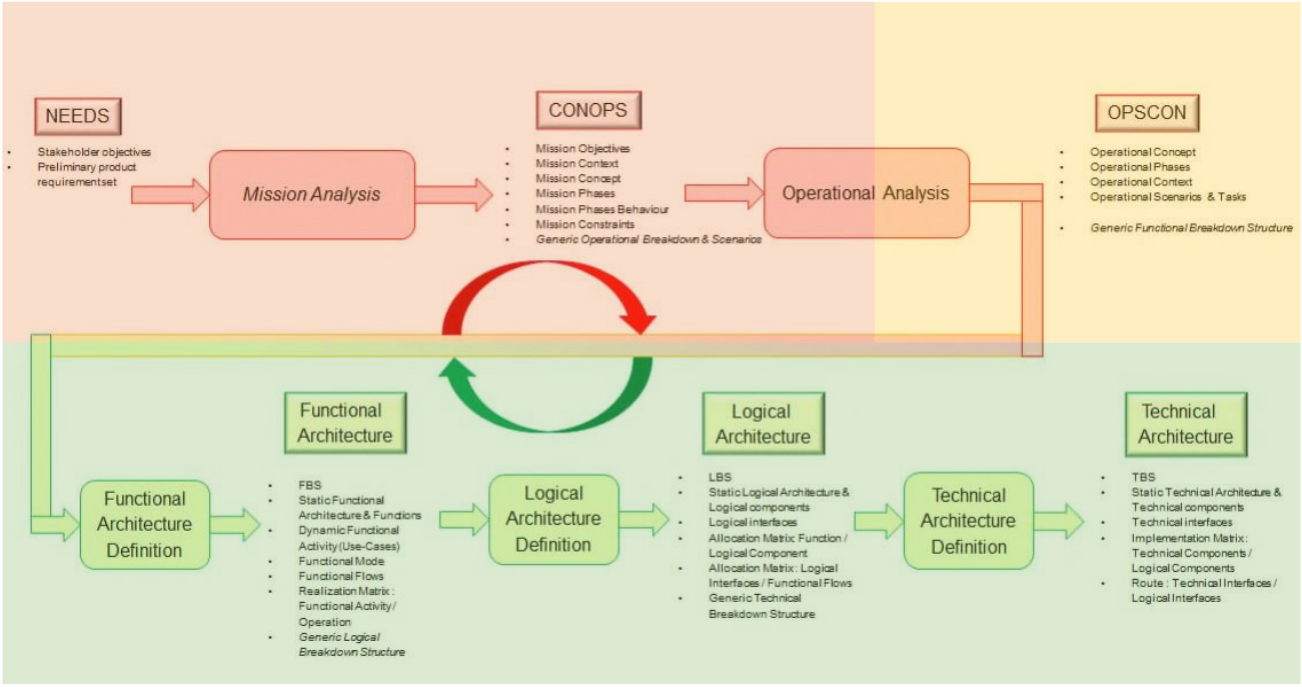

Figure 2.2: Overview of R-MOFLT methodology, courtesy of Airbus [26]. Usually in (science) space systems with ESA as the customer, artefacts in Mission viewpoint (highlighted in red) are supplied by ESA. The artefacts in Operation viewpoint (yellow) are defined by both ESA and Prime (ADS), still ESA being responsible for the majority. The definition of the so called “solution space” (green), starting with the functional architecture, is on the responsibility of the development team in ADS.

2.3.3 Model Content and Structure

LISA system architecture model consists of 2 main parts implemented as separate SysML: Package dedicated to model definition and model execution. Another part was added to the model named as “Reliability and Safety” for the contents of this thesis.

The model definition package (named “LISA_model”) involves all the model elements, relations, diagrams, scripts and tables that create the system definition according to the R-MOFLT

methodology. The model definition consists of 4 system layers or System of Interest (SOI)-Viewpoints in MOFLT terms (see *Figure 2.3*).

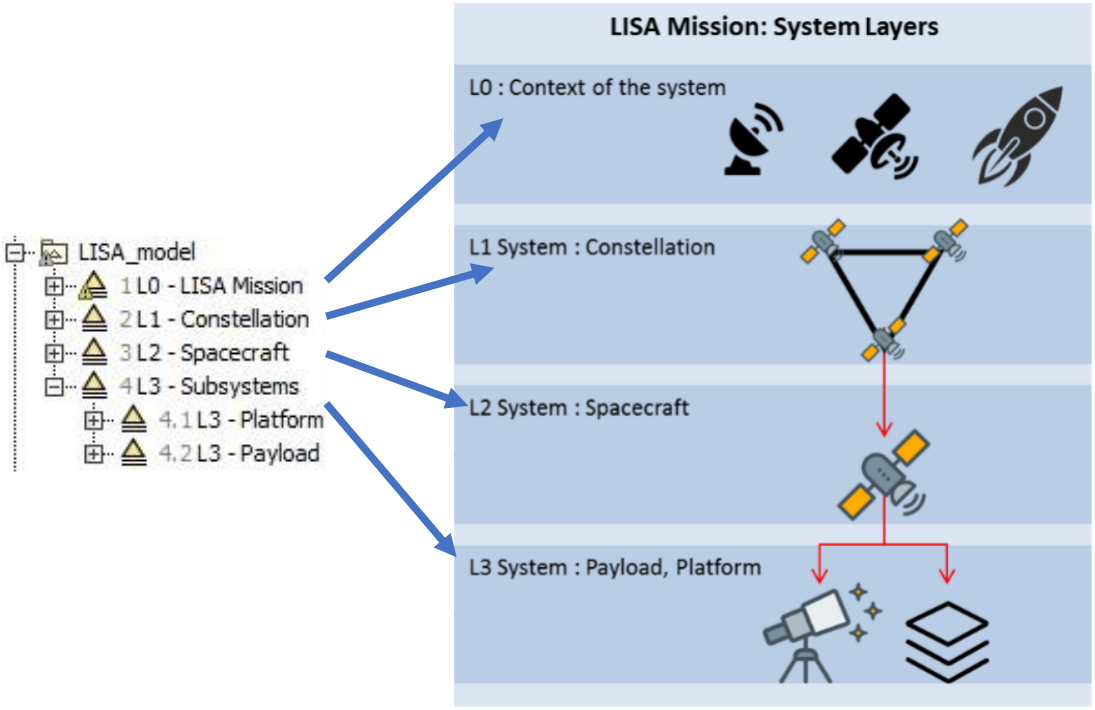

Figure 2.3: SOI-Viewpoints of system definition package of LISA Mission System Architecture Model. Modified from Charpigny et.al. [7].

- L0 – LISA Mission, is mainly dedicated for the Mission viewpoint. It involves the LISA mission objectives, mission concept, mission phases and stakeholder analysis.
- L1 - Constellation, is dedicated to the definition of the constellation formed by 3 spacecrafts. For this SOI viewpoint, Operational view is of interest. Operational concepts, phases and behaviours of the constellation are modelled in this layer including the Laser Acquisition Sequence.
- L2 – Spacecraft, is dedicated to the definition of the LISA spacecraft. For this SOI viewpoint, functional, logical and technical definitions are of interest. For example, the integration of Spacecraft-Technical architecture is done in Technical view, where all the technical components in the planned scope (mostly payload instruments) are integrated inside the spacecraft. As an important note, since the spacecrafts are identical in the mission, only one definition is enough, saving time and effort for the development team. This still does not prevent the model to use the same definition but impose different behaviours on them in the constellation level.
- The final levels are L3 – Payload and L3 – Platform. For these SOI viewpoints, functional, logical and technical definitions are of interest. As an example, functions belonging to Payload and its sub-elements are stored in this level, as well as the payload functional architecture.

The model execution package (see *Figure 2.4*) uses the elements in the model definition package to create simulation contexts according to the desired analysis from the project team. This package involves all the model elements, diagrams and scripts that enables the simulations to be configured

and run. Each analysis is stored in their corresponding packages and dedicated to a different combination of the MOFLT viewpoints to analyse the effects of:

- nominal system behaviour in specific viewpoints. (e.g., “Execution – Operational/Logical” package)
- faulty system behaviour in specific viewpoints. (e.g., “Execution – LAS Functional/Technical FMECA” package)

The rest of the packages are dedicated to the signals used in the simulations and the configuration elements.

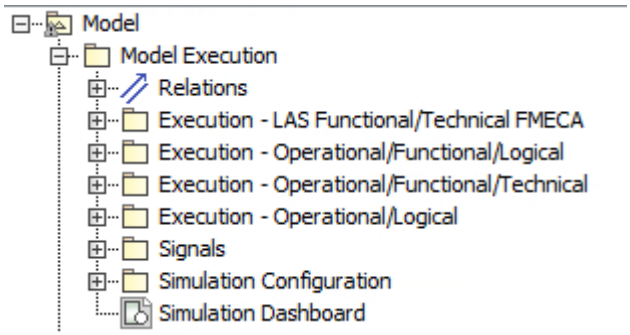

Figure 2.4: Containment tree view of the **“**Model Execution” package.

The reliability and safety package (see *Figure 2.5*) involves all the model elements, relations, diagrams, and tables that are dedicated to the RAMS activities that uses the Cameo SRAP profile. These contents are linked to and make use of the elements in the model definition package. There are key model elements from the Cameo SRAP profile used for the FMEA analysis in this thesis work:

- FMEA item: It is the main model element to store failure related properties of a function (in a functional FMEA study) or a logical/technical component (in a system FMEA study). There are one FMEA item per failure mode of each function and are linked to the functions using the "FMEA Item" property (e.g., if a function has two failure modes, there must be two FMEA items linked to the function for each failure mode). FMEA item model elements are stored in a dedicated package called “FMEA Items”.
- Failure mode: Failure mode model elements are linked to the FMEA item of each function. Generic failure modes can be defined, and assigned to the system functions (e.g., “No Function”) to keep the number of failure mode model elements manageable. They are stored in the “Failure Modes” package.
- Cause of failure: Cause of failure model elements are linked to the FMEA item of each function. They are only defined as verbal descriptions and are not connected to any functions, functional interfaces or flows defined in the model. Therefore, they must be generated and named manually for each analysed failure case. They are stored in the “Failure Causes” package.
- Local/Final effect of failure: Local/Final effect of failure model elements are linked to the FMEA item of each function. They are only defined as verbal descriptions and are not connected to any functions, functional interfaces or flows defined in the model. Therefore, they must be generated and named manually for each analysed failure case. They are stored in the “Final Effect of Failures” and “Local Effect of Failures” packages.

The FMEA/FMECA tables are stored inside the “FMECA Tables” package and their columns are customized according to desired standard/practice. These tables list the FMEA items and show all linked FMEA elements to them. The tables can be filled in with dragging and dropping dedicated

elements into the column entries of the table, which would automatically generate a link to the FMEA item. Finally, the tables, diagrams, relation maps, and matrices that are used to show dependencies in between the failure analysis artefacts are stored inside the “Dependency Diagrams” package. It allows to collect and identify most critical or interdependent failure cases using queries in the model and particularly useful to the engineers.

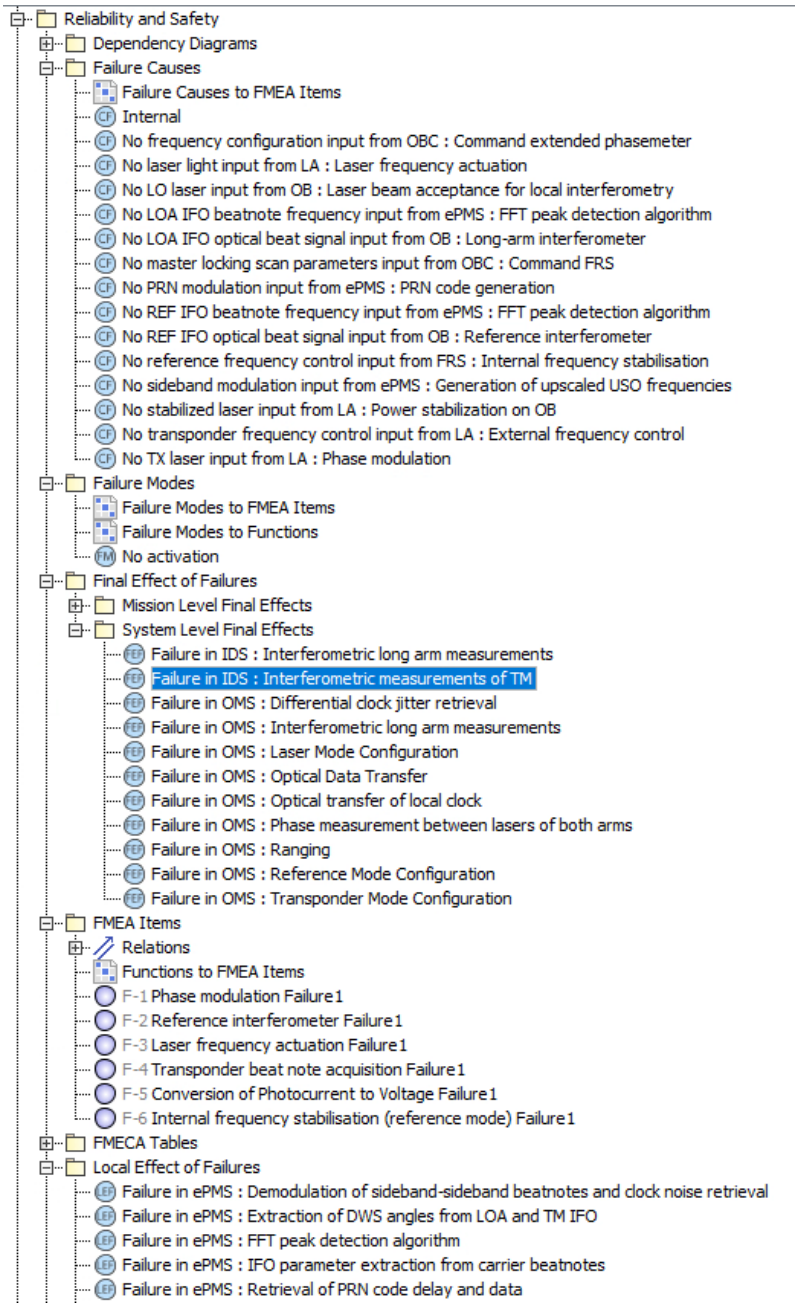

Figure 2.5: Containment tree view of the **“**Reliability and Safety” package. It is important to point out the high amount of model elements that need to be generated for only 6 failure cases (FMEA items).

2.4 STATE-OF-THE-ART REVIEW OUTCOMES AND RESEARCH CONTRIBUTIONS

This chapter outlines the state-of-the-art review outcomes and research contributions made in addressing the identified gaps in the field of functional Failure Mode and Effects Analysis (FMEA) and Model-Based Systems Engineering (MBSE) applications. The outcomes and gaps were identified through an investigation of various approaches for functional FMEA, MBSE assisted FMEA applications, Airbus R-MOFLT methodology, and the LISA system architecture model in the preceding chapters.

The following sections describe the identified gaps in the literature that the thesis aims to fill.

- Inclusion of Operations in FMEA: The existing approaches lack the consideration of operational aspects in the FMEA analysis. To account for the complexity of operations in the LISA constellation, the proposed approach of the thesis incorporates operations in the FMEA analysis. This ensures that the potential failure modes associated with operations are considered, leading to a more comprehensive analysis.
- Effects of Exchanged Interface Flows on the Function: The influence of exchanged interface flows on system functions is not adequately addressed in the existing approaches. Understanding how these interface flows affect the system functions and overall failure propagation is crucial for accurate FMEA analysis.
- Inclusion of Functional Behaviours and Architecture together in FMEA: Existing approaches in the literature tend to primarily focus on either the functional architecture without explicitly considering individual functional behaviours, or they concentrate solely on the functional behaviours without adequately addressing the functional architecture. This limited focus leads to an incomplete understanding of the system's overall behaviour and the impact in case of failures. To overcome this limitation, the proposed approach integrates both behavioural and architectural aspects in the FMEA analysis by considering functional behaviours, constraints, interfaces, and flows.
- Detailed Modelling and Analysis Descriptions: Through the literature review, it has become evident that there is a significant lack of detailed instructions on how to realize modelling and analysis steps. Existing descriptions tend to focus on the overall approach but often omit crucial details related to software configuration and the specific processes required to achieve desired results (e.g., obtaining an automatically filled FMEA table in Cameo). Recognizing this gap, the aim of this thesis is to provide comprehensive end-to-end descriptions of the modelling approach, including detailed instructions on software configuration and the step-by-step process for achieving desired results. By addressing this deficiency, the thesis seeks to enhance the understanding and replicability of the modelling process.

In terms of state-of-the-art outcomes, several methods have successfully addressed the needs of using Model-Based Systems Engineering (MBSE) for Failure Mode and Effects Analysis (FMEA) applications, which are also considered for implementation in this research.

- Usage of ECSS-Q-ST-30-02C Standard: The approach and process outlined in the ECSS-Q-ST-30-02C standard [16] (see *chapter 2.12.2*), which is a norm in the European space industry, will be employed for conducting the FMEA analysis in this thesis. This standard will serve as a guiding framework for the FMEA methodology and ensure compliance with industry practices and requirements.
- Usage of Viewpoints: The MOFLT methodology (see *section 2.3.2*) separates the model structure into viewpoints, allowing experts to focus on their relevant artifacts. This approach is employed for FMEA topics, specifically using the "reliability and safety" viewpoint package in the model (see *section 2.3.3*).
- Usage of Cameo SRAP: The concept of using dedicated model elements for safety and reliability analysis and establishing a formal link to the model elements used for system

definition, is found beneficial and will be applied in this thesis work. The approaches from Biggs et. al. [18] (see *chapter 2.2*) and the Cameo SRAP plugin [19] (see *section 2.3.3*) will be utilized in the LISA system architecture model.

- Separation of Nominal and Non-Nominal Modes by States and State Machines: The approach proposed by Francesco [20] and Girard et. al. [21] (see *chapter 2.2*) shows promise in enhancing flexibility in behaviour modelling, providing concise representation and facilitating convenient switching and control over nominal and non-nominal functional behaviours. Hence, it will also be employed in the modelling approach of this thesis.
- Usage and Tracking of Interchanged Parameters: The approach presented by Schummer and Hyba [22] (see *chapter 2.2*) involves utilizing and tracking interchanged parameters between systems to assess their propagation across the system elements. This enables a better understanding of how faulty parameters propagate in the overall system architecture. This approach will be utilized for functions, functional interfaces, and functional architecture in this thesis work.
- Usage of I/O and Architecture Information for Failure Cause/Effects: To enhance the analysis of failure causes and effects, the proposed approach by Mhenni et. al. [24] (see *chapter 2.2*) leverages functional inputs/outputs and functional architecture information. This approach assists in identifying and evaluating potential failure causes and their impact on system functions during the FMEA analysis. Therefore, it is employed in the methodology of this thesis work.

3 APPROACH: MODELLING AND CONDUCTING FUNCTIONAL FAILURE MODES AND EFFECTS ANALYSIS IN LISA MISSION

This chapter elaborates on the methodology pursued to implement an early phase functional FMEA study by making use of the LISA Mission system architecture model. Upon the investigation of the state-of-the-art and the iteration with the project team in ADS, some key constraints are identified for the approach:

- Usage of Cameo Systems Modeler and SysML: The approach shall use the tool and language that has already been adopted by ADS LISA system engineering team for early phase system architecture development. This limitation is to reduce the effort of transferring knowledge along multiple software, which enables to analyse architectural implementations swiftly and conveniently. It also enables collaboration, support and development along other project teams in Airbus.
- Compatibility and integrability to MOFLT functional architecture modelling process: As the standard is being defined by the Airbus group-wide Digital Design, Manufacturing and Services (DDMS) activity, it is important to ensure compatibility with the company engineering strategy. This would involve utilizing existing templates, manuals, and support, as well as benefiting from profile automations and tools (e.g., report generator). Furthermore, aligning with the MOFLT process enables building upon existing work and know-how of the project team, reducing the training and learning effort.
- No implementation of performance analysis or high-fidelity physical/mathematical relations: The approach shall not implement any performance analysis or high-fidelity physical/mathematical relations to reflect the system behaviour. This role is clearly allocated to detailed analysis models and simulators developed by discipline experts in standard tools and processes that handle way better than Cameo in this case.
- Minimum modelling effort specific to FMEA: To reduce the modelling effort required for conducting the FMEA study, the approach shall make use of existing model elements dedicated to system definition as much as possible. Although FMEA is a key goal, the model is not solely established for conducting FMEA activities, as described in *section 2.3.1* of the model strategy. Adapting a modelling method with a majority of elements that are only useful for FMEA activities would result in higher effort and clutter in the model. Therefore, any modelling effort and model elements specific to FMEA that are not related to system definition shall be minimized.

Based on the constraints outlined above, a proposed approach for the modelling of executable functional architecture is described in *chapter 3.1,* while the execution and simulation implementation of this model is explained in *chapter 3.2.* Subsequently, in *chapter 3.3,* three different approaches for conducting a Functional FMEA study within a defined scope in LISA mission are proposed, including:

- traditional document-based,
- static model-supported,
- executable model based FMEA approach.

3.1 FUNCTIONAL ARCHITECTURE MODELING

This chapter explains the process pursued to model the executable functional architecture inside the MBSE environment using Cameo Systems Modeler.

As a basis for a functional FMEA study, the principle functions of the systems of interest shall be analysed, to be able to identify failures that traverses a functional chain which can finally be traced back to technical components. In order to identify the functional chain, the functions, their behavioural

nature, and their relationship with each other shall be defined. This definition must be traceable along a hierarchical decomposition, involve traceable parameters exchanged in between elements, and allow execution to simulate and record the overall system behaviour. In order to satisfy these requirements, the following modelling concepts are performed for a model-based FMEA analysis of the space segment of the LISA mission.

- Model the functions and the system components that are realizing them. These elements will be the root cause of failures in the real implemented system, therefore the primary artefacts for the FMEA study. This process is described in the *section 3.1.1.*
- Model functional interfaces and the functional flows occurring inside them. The interfaces and the flows allow the traceability of failed parameters along the functional chain in the architecture. This process is described in the *section 3.1.2.*
- Model functional behaviours defining how functions convert inputs into outputs, functional modes that separates and controls the different behaviours of a function, and functional constraints that defines the (in)valid conditions for the behaviours. This is described in the *section 3.1.3.*

The modelling approach proposed in this chapter is an addition into the default MOFLT methodology, in terms of functional behaviour, constraint and interface modelling. The outcome of the modelling approach is then used to conduct simulations and FMEA studies in the following chapters.

3.1.1 Modelling of Technical Components and Functions

The functions are the key artefacts to be analysed in a functional FMEA, whereas technical components are the artefacts to be affected as an outcome of a failure. As a reference example to conduct the FMEA studies, a set of functions that defines the laser acquisition sequence operational scenario has been identified from the functional requirements and functional breakdown (see *Table 1.2*). These functions correspond to the lowest level (leaf) functions in the breakdown structure that can be allocated to a technical system of interest and establish the baseline architecture. The decision of only showing leaf functions and technical components in the architecture view instead of logical components and high level functions is that the technical definitions (components are interfaces) are mature enough to reflect during phase B1 of LISA development lifecycle.

In the model, the functions are implemented as <<Function>> (SysML: Block) elements, and the technical components of interests are modelled as <<Technical Component>> (SysML: Block). The model elements are located inside the packages in their respective MOFLT viewpoints (F and T) in order to conveniently navigate inside the model as can be seen in *Figure 3.1.*

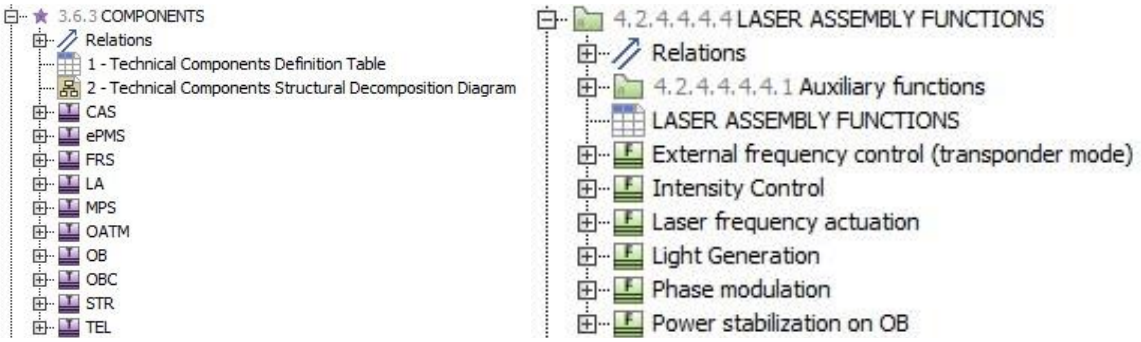

Figure 3.1: Example of technical components (left) and functions (right) in the model containment tree view.

The next step is to establish the traceability links in between created model elements. In order to establish the breakdown structure and hierarchical traceability, directed composition links are used in between child and parent functions. After the generation of the functions, they are linked to the <<SOIRequirement>> (SysML: Requirement) elements in the model using the <<SatisfiedBy>> relation. The allocation of functions to the technical components is done via creating part properties inside the technical components in an internal block diagram (see *Figure 3.2*). These part properties are stereotyped as <<Functional Part>>, which are an instantiation of the function type it represents. This operation creates a directed composition link between the functions and the technical component.

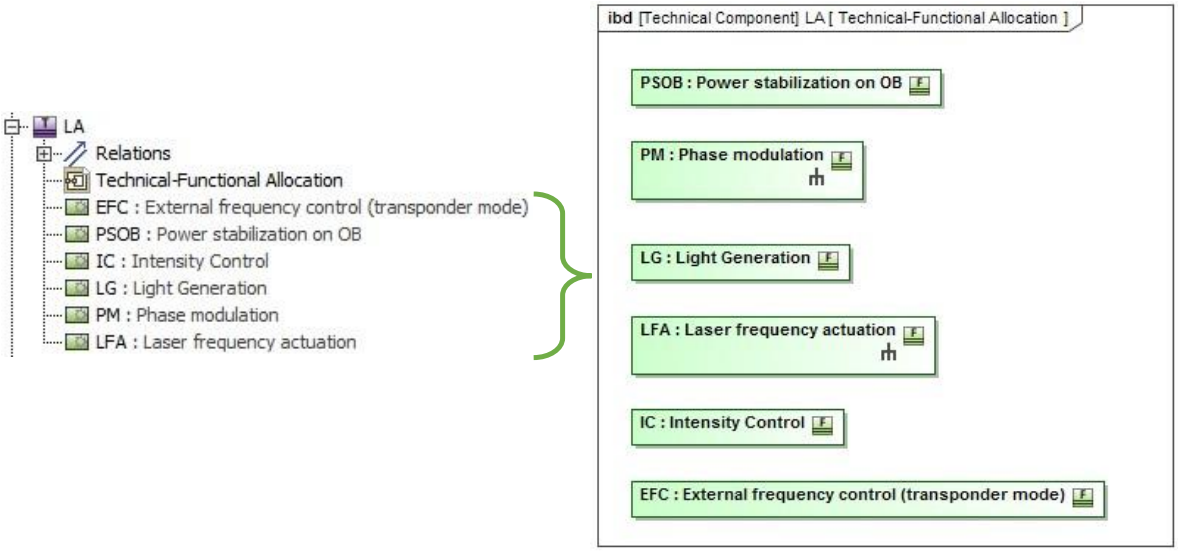

Figure 3.2: Allocation of functions to technical components. Left: The functional parts are contained inside the technical component LA in the containment tree view. Right: The functional parts are depicted as green rectangles inside the ibd of the technical component. Each functional part has an instance name followed by a function type, divided by a semicolon in between (e.g. “LG : Light Generation”).

Finally, all the defined technical components are instantiated and shown on an ibd inside a Spacecraft-Technical block, which is stereotyped by <<Technical Component>> and <<System of Interest>>, creating the baseline of the spacecraft architecture to be analysed (see *Figure 3.3*). The described modelling steps in functional-technical domain establish the functional breakdown, function to technical component allocation, and function to requirement traceability. This comes with the benefit of having formal links established in between the model elements (requirement, function, and technical component) ensuring model coherence and robustness and to be used in the FMEA study. The summary of traceability links established can be seen in the data model sketched in the profile diagram in *Figure 3.4.*

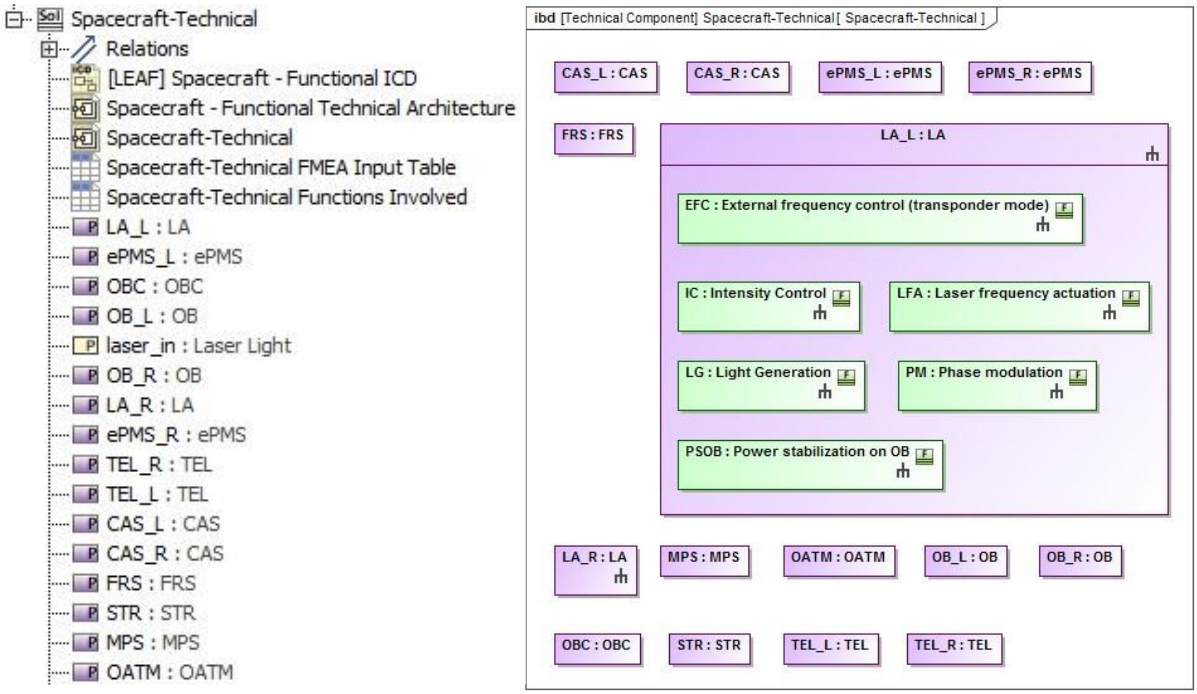

Figure 3.3: Collecting the technical components inside the Spacecraft-Technical block generates the definition of the spacecraft type with the components and functions inside.

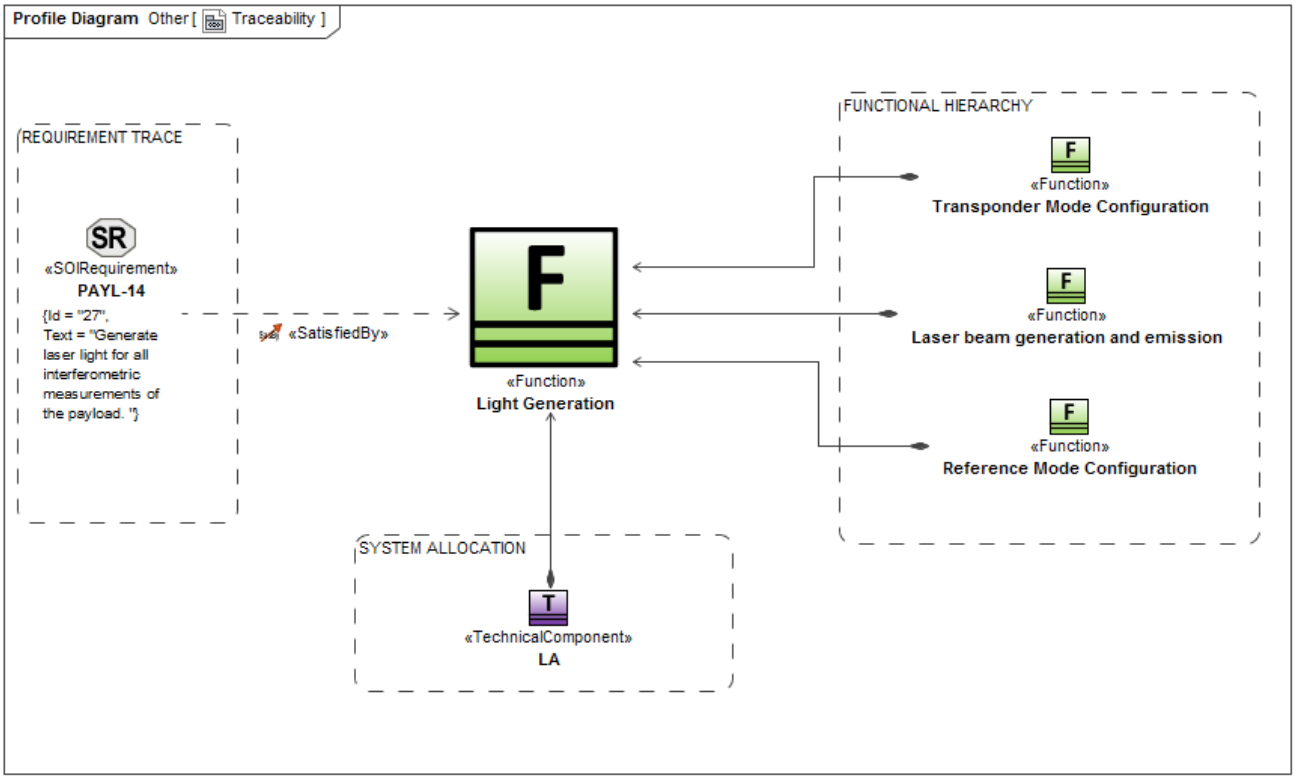

Figure 3.4: Profile diagram showing relationships in between functions, systems and requirements.

3.1.2 Modelling of Functional Interfaces and Flows

Functional inputs and outputs are the artefacts establishing the paths that have been used to exchange parameters. These exchanged parameters affect the behaviour of each function and would be the key elements to trace potential failures in the systems. The standard MOFLT process only supports the definition of interfaces and flows, however it does not implement parameters, values, or exchange mechanisms for them. This does not allow to reflect nor execute the behavioural aspects of the functional architecture. Therefore, a customized implementation has been pursued on top of the standard MOFLT modelling approach.

Following the MOFLT approach, the functional interfaces are modelled with <<Functional Interface>> (SysML: Proxy Port) for all the functions in the scope of the architecture. Functional interfaces are defined for each input-output parameters of the functions (see *Figure 3.5*). These interfaces are then typed by <<Functional Interface Type>> (SysML: Interface Block) elements. The items that are exchanged between the functional interfaces are modelled as <<Functional Flow>> (SysML: Flow Property) inside the functional interface types (see *Figure 3.6*).

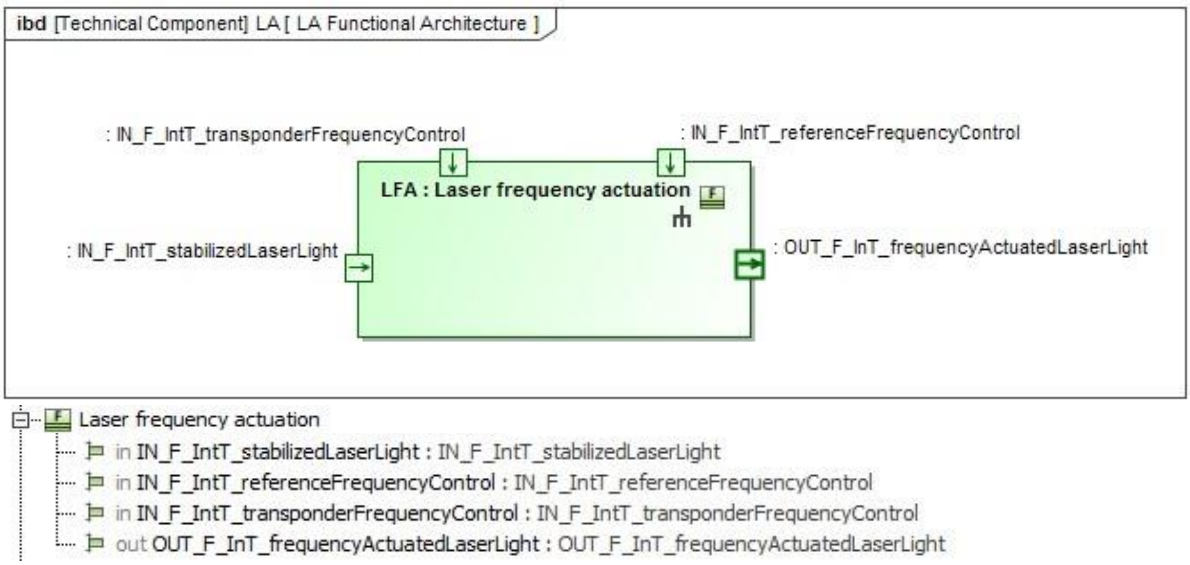

Figure 3.5: Containment tree and ibd view of functional interfaces of “Laser frequency actuation” function.

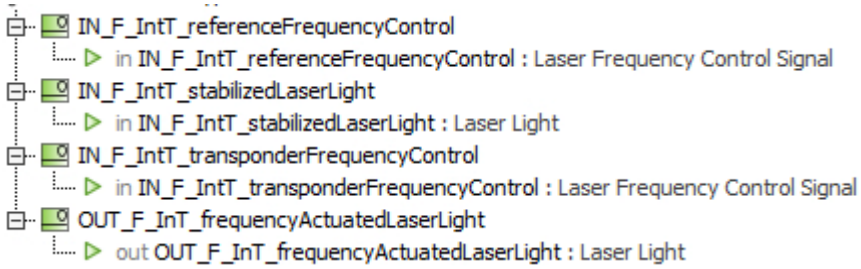

Figure 3.6: Containment tree view showing the functional interface types. Inside, functional flows are defined which are typed by functional flow types.

For the executable functional architecture, it is utmost important to define the functional flows and their values, as they will be exchanged in between functions during execution. From this point on, the MOFLT process is extended by adding more details to the modelled functional flows. According to its nature, the functional flows are typed by a block or a <<Functional Flow Type>> (SysML: Value Type).

The block type allows storing multiple parameters with values (as SysML: Value Property) in a structured way, whereas functional flow type can only store one parameter with a value assigned. In order to trace the parameter exchanges, the flows are given string values which shows the state of the flow during execution. As an example to illustrate the adopted concepts, the laser light flow is used. Majority of the functions in laser assembly and optical bench transfer laser light while changing its distinct attributes. In order to capture these effects, laser light is modelled as a block with four value properties as frequency, intensity, phase and state (see *Figure 3.7*). All the functional flows that convey laser light are then typed by the laser light block. This enables the functional flows to inherit the properties stated in *Table 3.1* and change accordingly, enabling the traceability of changes along the functional path (see *Figure 3.8*). Other functional flows that deemed not to be much detailed are typed by functional flow type hosting string values that shows the state of the functional flow, for example as "–" (invalid) or "on" (valid).

Figure 3.7: Containment tree view of Laser Light flow type

Table 3.1: Attributes of Laser Light flow type and their values

Value Property	Type	Value
frequency	String	–, initial, controlled, stabilizing, stabilized
intensity	String	–, initial, set, stabilized
phase	String	–, initial, PRN_modulated, sideband_modulated, PRN_sideband_modulated
state	String	–, on

Once all the functions in the scope have been equipped with functional interfaces, they are connected using a Functional Connection (SysML: Connector). This link enables the traceability of the functional flows as well as flow exchange during execution. The connection only allows the flow to change its properties in an input interface if the flow inside the connected output interface is changed. When all of the functional interfaces are connected inside each technical components (see *Figure 3.9*) and then inside the Spacecraft-Technical ibd, functional-technical architecture is established. The next step would be to define behaviours for each function in the study scope.

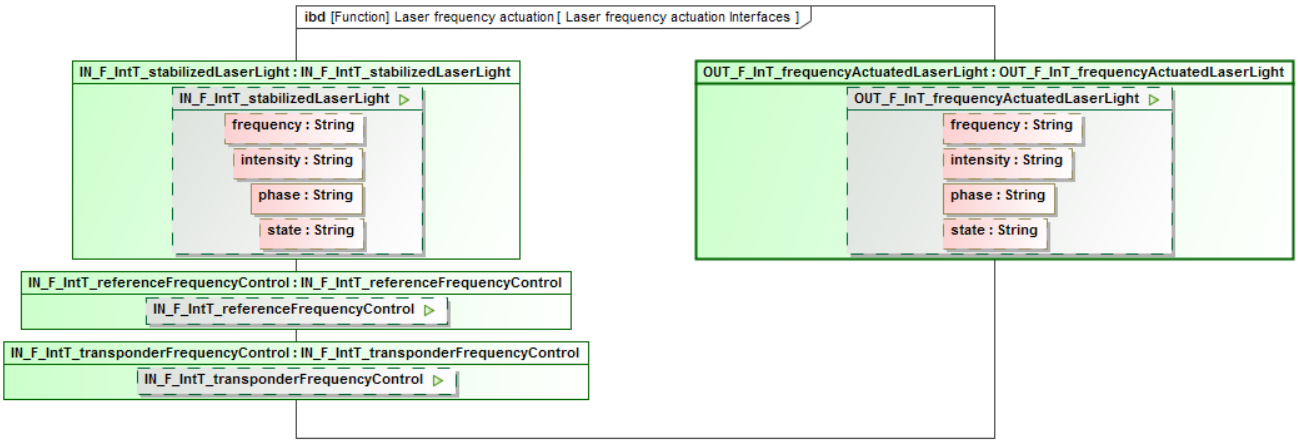

Figure 3.8: ibd showing the functional interfaces and functional flows of the Laser frequency actuation function.

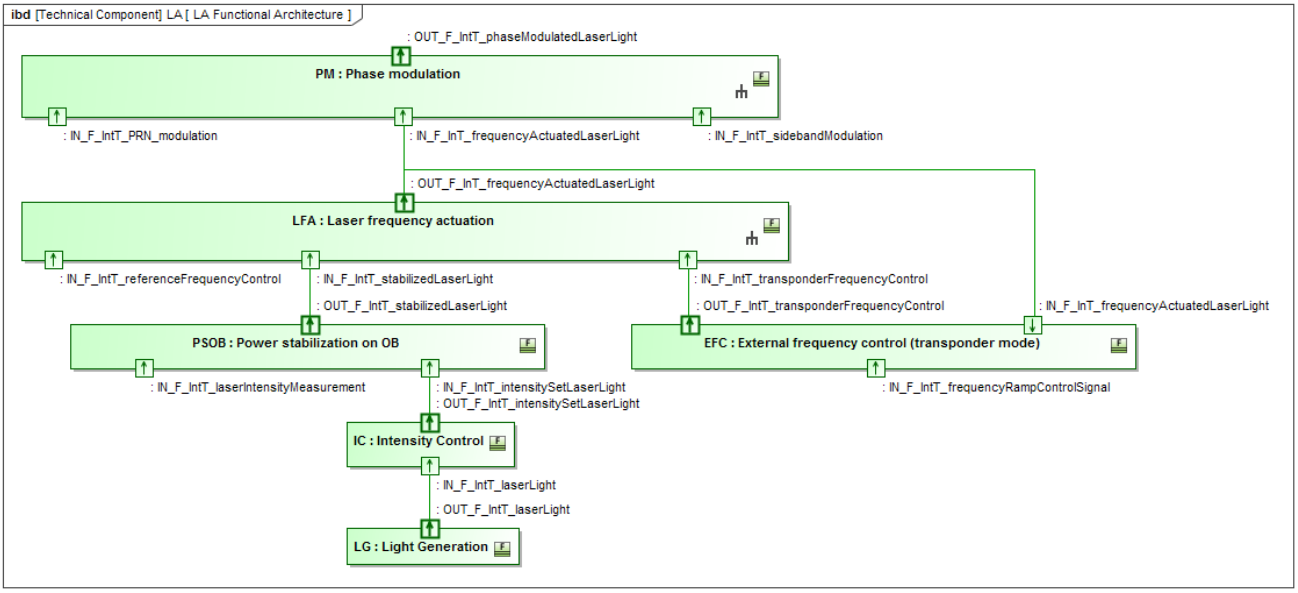

Figure 3.9: Example of a possible functional architecture of a Laser Assembly.

3.1.3 Modelling of Functional Modes, Behaviours and Failure Constraints

The previous steps define the structural aspects of the functional architecture. Even though it reflects valuable information about function dependencies and interconnections, by itself it is not enough to reflect what is actually happening in the architecture. It fails to answer the questions:

- Which functions are used or active?
- Which states are the functional flows in?
- Is the system working as desired?

Therefore, the next step in executable functional architecture modelling is establishing the behavioural aspects of the architecture. Functional behaviours are collection of actions that are performed by the function, showing how a function emerges to accomplish a desired need. Depending on different modes and constraints, a function can behave differently. It is in the interest of the systems engineers

to define what kind of behaviours a function possesses, at which modes they emerge, what causes a change in modes, and during which constraints they work properly. In the following sections, the approach to model the modes, behaviour and constraints of the functions are explained.

3.1.3.1 Functional Modes

Functional modes separate behavioural expectations from a function and control the behaviour that it performs. In the scope of the LISA model, majority of the functions have two modes defined as active and inactive. For example, in its inactive mode, it is not expected for a “Light generation” function to generate laser light, and vice versa, it is expected to generate light when active. The idea of this separation is to allow controlling what the function is expected to do by switching the modes, but also to allow capturing when function behaviour does not meet expectations, defined as failures.

The functional modes are modelled in a state machine diagram as <<Functional Mode>> (SysML: State) as can be seen in *Figure 3.10*. The state machine diagrams are defined inside each function and only for the function itself. Each mode has an opaque behaviour defined as a “do activity”. This opaque behaviour updates a string value property named as “function_state” that is defined inside the function, to the name of the mode the function is currently in. This enables any other entity that accesses the function_state value property to know, real time, what mode the function is in. This is a very important aspect of this modelling approach as the functional behaviours and constraints defined for the function will be referencing this function_state value property to perform their activities (see next *sections 3.1.3.2 and 3.1.3.3*).

Each mode has a transition defined as a signal trigger. The signals are named independent of the mode names (as integers), so that they can also be used for any other mode transitions throughout the model. The transitions from inactive mode are assigned a guard named “activation”. Activation is a Boolean value property defined inside the function, which has a default value of “true”. Therefore, as default, the guard will take the value of “true” allowing transition from inactive to active mode (or any other defined mode). If set to false, even if the signal trigger is detected, the transition will not happen. This is the main failure mode implemented to the functions for the functional FMEA study, i.e., “No Function”.

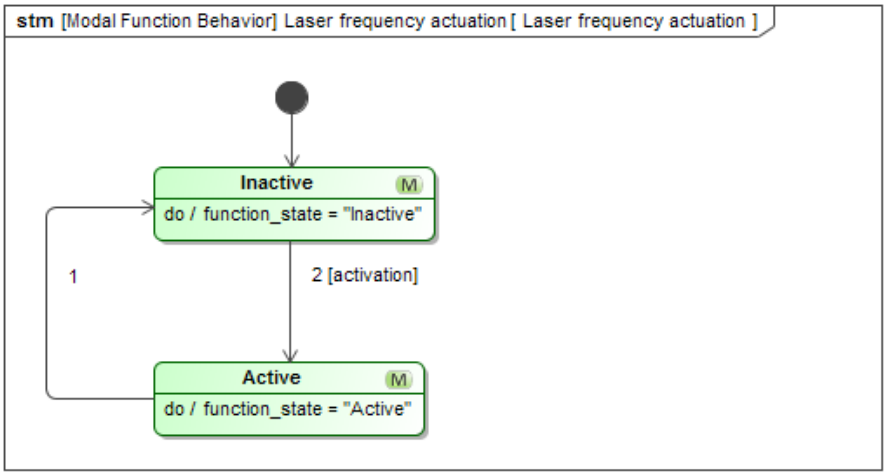

Figure 3.10: State machine diagram of Laser frequency actuation function. Functional modes “Inactive” and “Active” are depicted as the green rectangles with an “M” symbol. Inside the modes, opaque behaviours are defined with the “do/” statements. The mode transitions are depicted as arrows with the signal triggers “1” and “2”. The “activation” guard applied to the transition from inactive mode is shown inside the square brackets.

3.1.3.2 Functional Behaviours

After the definition of functional modes, the behaviours, i.e., what the function does with input/output (I/O) parameters in each mode has to be defined. This definition establishes the I/O value exchange mechanism that enables the execution to traverse through functions in the architecture. The collective work of these modelled behaviours emerge the overall functional behaviour of the system, also in the case of failures.

It is a common practice in literature [20, 21, 24, 27-29] as well as in MOFLT methodology, to model functional behaviours and exchanges with activities and object flows inside activity diagrams. These model elements and diagrams are clear to read and understand due to their symbolic nature. However, when executed, the values exchanged in between activities in an activity diagram do not traverse through the interfaces inside the architecture established in the ibd. This is an important problem because it is utmost desired that the consistency in between the behavioural and structural aspect of the system must be ensured. Therefore, the execution must traverse through both aspects of the system and serve as a validation tool to the functional architecture and behaviour. For this reason, as a common practice, signal exchanges via Send Signal Action and Receive Signal Actions, or SysML: Sequence Diagram are used to exchange information through the architecture [20, 30]. The problem in this methodology is that these exchange methods only handle discrete events and do not support continuous data exchange during execution (as in Cameo Systems Modeler v19SP4). In addition, these diagrams only execute once when called, therefore must be looped to ensure data exchange continuously, and even if done so, all data exchanges need to be synchronized in between behaviours to work properly. In addition, it was observed that the high number of loops and signal exchanges drastically increases the amount of memory usage causing the software to crash. In the frame of this thesis different methods know from literature [27, 31, 32] have been analysed and traded. Finally, it has been concluded that the parametric diagram is best suitable for this task, as it:

- allows direct visual access and exchange through interfaces in the architecture
- is continuously active, therefore, does not need memory expensive loops and synchronization
- offers direct and visual access to function parameters (e.g., function_state)
- does not need the effort of modelling explicit signal exchange actions.

Therefore, to model functional behaviours, the parametric diagram is used with SysML: Constraint Block and SysML: Constraint Property (see *Figure 3.11*). The behaviours are scripted inside the constraint properties using if-else conditions using the Groovy scripting language [33]. Behaviours are kept intentionally as basic and straightforward, i.e., if inputs are valid, outputs become valid, on the contrary if inputs are invalid, outputs also become invalid. The reason is to keep the complexity manageable, as the number of functions, systems and interactions involved are relatively high in the architecture scope. Then, all the parameters that are manipulated by the behaviours are connected to the functional flows inside the functional interfaces using SysML: Binding Connector. This connection ensures that the parameters at each end to assume the same value, transferring any change in the functional interface to the behaviour for the functional inputs and vice versa for the functional outputs.

As an example, the behaviour modelled for the “Laser frequency actuation” function belonging to LA can be given (see *Figure 3.11*). This function is responsible for changing the incoming laser frequency according to control inputs received from either “External frequency control” or “Internal frequency stabilization” functions. This function is necessary in order to lock the frequency of the laser onto either the laser light of a remote spacecraft or the neighbouring LA on the same spacecraft, which is eventually required for the science measurement.

The behaviour of the function is scripted inside the constraint property named “behaviour”. Since the function only manipulates the frequency of the laser light, only the “frequency” parameter coming from the laser input functional flow and going to the laser output functional flow is connected to the

constraint property. The rest of the parameters are directly relayed using the binding connectors. Inside the script, the outermost if-else if conditions specify the mode that the function is in, with the help of the bounded function_state value property defined inside the function. First, the behaviour for the “inactive” mode is defined:

```
>> out_frequency = frequency;
```

This corresponds to the action that when the function is “inactive”, it does not impose a change in the frequency parameter of the laser light. Then, for the “active” mode, valid conditions for each input functional flow values are listed, and for each condition an output value is assigned to the output functional flow value:

```
>> if (state == “on”)
```

Meaning that the laser light must be “on” first, for every other condition to be even considered.

```
>> if (IN_F_InT_referenceFrequencyControl == “on”) {out_frequency = “stabilized”;}
```

Meaning that if the reference frequency control signal is on, it directly stabilizes the laser light frequency.

```
>> if (IN_F_InT_transponderFrequencyControl != “-”) {out_frequency = IN_F_InT_transponderFrequencyControl;}
```

Meaning that if the transponder frequency control signal is not invalid, the frequency will be actuated to its this control signal value. And finally, if the conditions are not satisfied, the input value is relayed without change.

```
>> else {out_frequency = frequency;}
```

This logic can be applied to all the functions in the architecture and customized depending on their behavioural nature. An important advantage is that the methodology offers a modular approach, so only the function of interest, its properties and its interfaces are being considered. Which means that the functions can be relocated freely inside the architecture, connected to any desired function through its ports, as long as the functional flow is compatible.

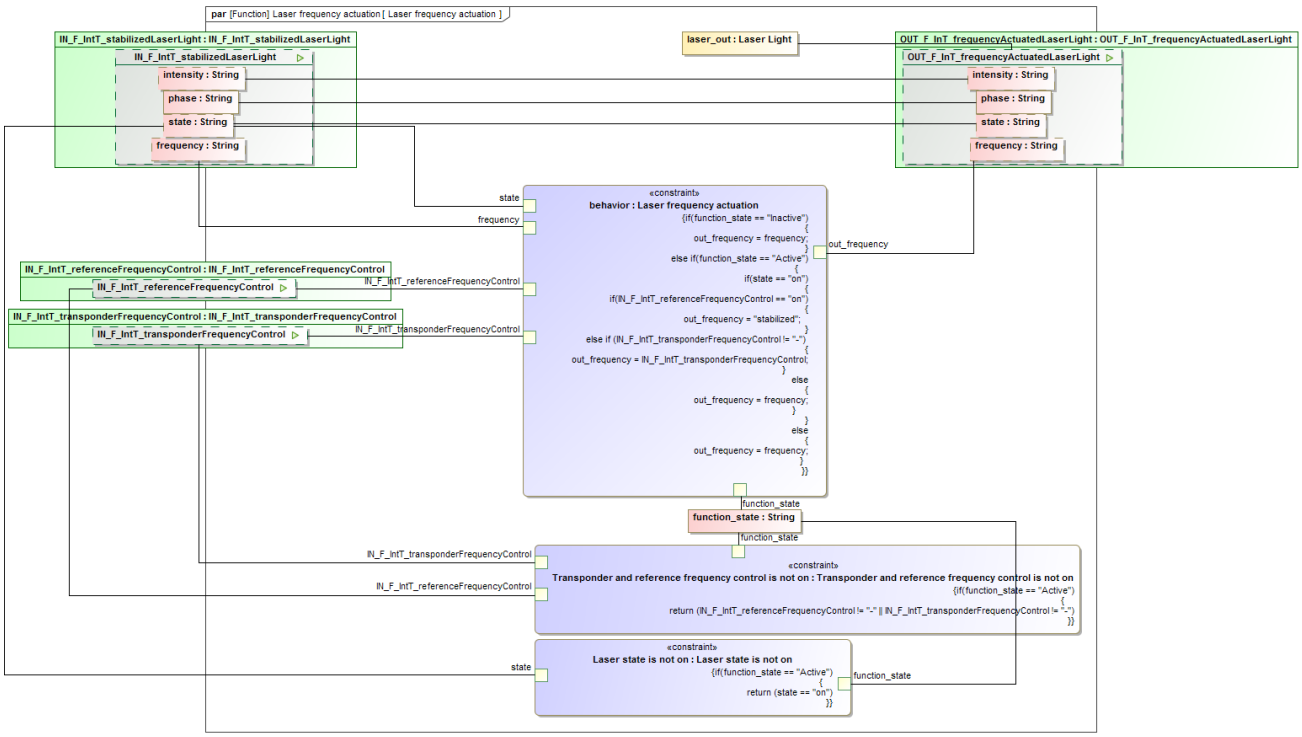

Figure 3.11: Parametric diagram showing the modelled behaviour and constraints for the “Laser frequency actuation” function belonging to LA.

3.1.3.3 Functional Constraints

Within the behaviour modelling, the functions behave nominally when all of its input conditions are satisfied, and non-nominally when not, which is reflected in its output. However, explicitly for the FMEA study, there needs to be a way to capture and record failures inside the function within the defined behaviour. This would enable the observation of the failure propagation along the functional chain and creation of the FMEA table as a result.

The failures in the modelled functions are defined in terms of “valid input conditions are not satisfied when the function is active”. Each of the failures is then modelled as a constraint property inside the parametric diagram of the function (see *Figure 3.11*). When the constraint is not satisfied and returns “false” due to failure condition, the constraint property gets a verdictKind value as “fail” (verdictKind is a Boolean-like value type defined in Cameo, taking the value of either “pass” or “fail”). By naming the constraint property as the failure condition itself, one can observe the cause of the failure directly in a verbal sense.

As an example, the “Laser frequency actuation” function needs at least one of the control inputs to work, either the signal “IN_F_InT_referenceFrequencyControl”, or the signal “IN_F_InT_transponderFrequencyControl”. On top of that, obviously, it needs the laser light to be “on” so that it can change its frequency. To reflect these conditions, two explicit constraint properties are defined, shown in *Figure 3.11.* During model execution, when the “Laser frequency actuation” function is in its “active” mode:

- Failure Case 1: If “IN_F_InT_referenceFrequencyControl” signal switches to “-” while “IN_F_InT_transponderFrequencyControl” signal already has the value “-”, the dedicated constraint “Transponder and reference frequency control is not on” will get the value “fail”.

- Failure Case 2: If “IN_F_InT_transponderFrequencyControl” signal switches to “-” while “IN_F_InT_referenceFrequencyControl” signal already has the value “-”, the dedicated constraint “Transponder and reference frequency control is not on” will get the value “fail”.
- Failure Case 3: If the laser light state switches from “on” to “-”, the dedicated constraint “Laser light state is not on” will get the value “fail”.

Therefore, once the constraints for a function are explicitly defined, they can also be explicitly identified when they fail, which gives the awareness to the user what exactly went wrong within the functional chain execution. This information can be used to fill the failure cause and effects entries of the FMEA table (described in *section 3.3.3.2*).

3.2 EXECUTABLE MODEL SIMULATION SETUP

In this chapter the implementation of the simulation in the model for the laser acquisition sequence scenario is explained. The reason to implement an operations-based simulation is to correctly capture the behaviour of the system according to its real mission usage. This way, it could be possible to find the failure cases that are not straightforward to find while using the static architecture views. The model execution for FMEA analysis has the two main pillars: modelling and setup of the execution context explained in *section 3.2.1* and modelling of the operation of interest explained in *section 3.2.2*. The behaviour and the generated artefacts resulting from the simulation using the Cameo Simulation Toolkit is explained in *section 3.2.3.*

3.2.1 Execution Context

The execution context, “Constellation_FMEA” block, is the main artefact that contains all the elements of interest reflecting the system definition, as well as the diagrams and scripts to enable the configuration of the analysis. The context involves three instances of the Spacecraft-Technical (see *Figure 3.3*), modelled as part properties for each of the spacecrafts SC 1-2-3. Inside the ibd, external functional interfaces of each spacecraft representing the laser links are connected using a functional connection, establishing the constellation architecture (see *Figure 3.12*).

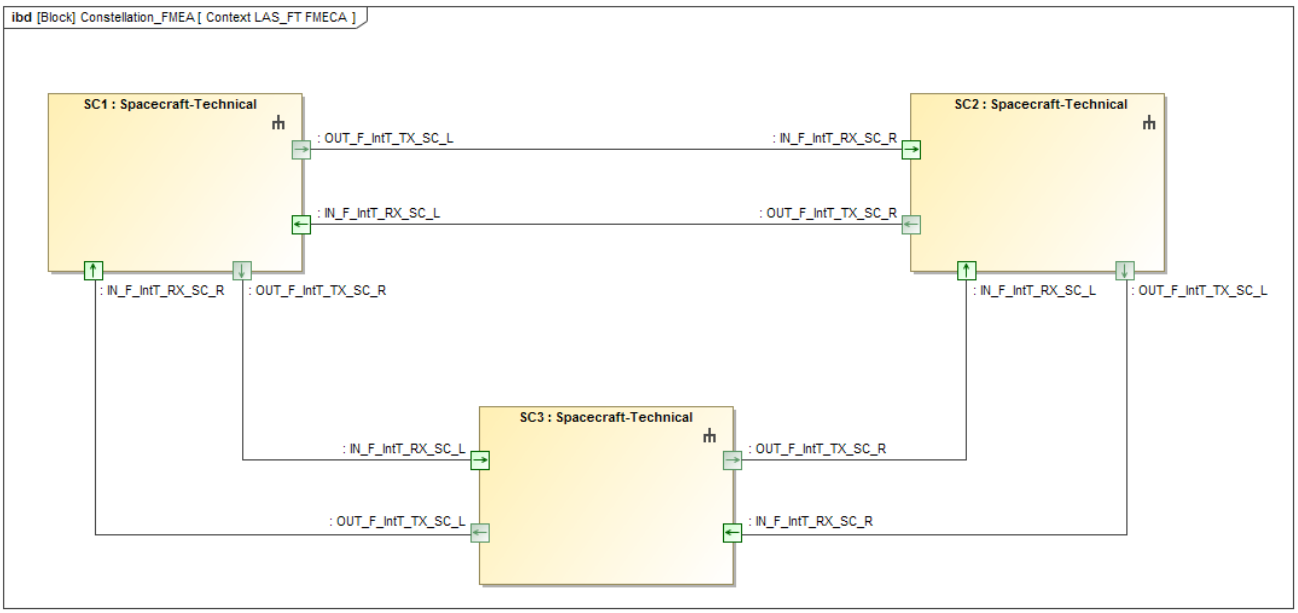

Figure 3.12: Ibd showing the execution context as the instantiation of the Spacecraft-Technical Type three times with laser links established as interface connections.

Inside the context (see *Figure 3.13*), three reference properties named as “Master”, “Slave_1” and “Slave_2” are defined and typed by the Spacecraft-Technical block. These reference properties represent the roles that the spacecrafts assume during laser acquisition sequence operations, as described in *section 1.2.3*. Three value properties defined inside the context, named as “master_sc_config”, “slave1_config” and “slave_2_config” take integer values representing which spacecraft is assigned to which role. The roles can be assigned before the simulation is started by modifying these configuration value properties. The context features a state machine diagram as a classifier behaviour to orchestrate the execution. The first state called “CONFIGURE” contains an activity called “Configure Roles” as a “do/ behaviour”. With the help of ALH API [34] scripts, each reference property representing the roles is assigned the value of individual spacecrafts SC 1-2-3 according to the configuration value properties. This enables to track and refer to the spacecrafts with both their roles and identification numbers. After the configuration steps, the execution transitions to the second state called “START”, which contains the laser acquisition operational phase behaviour as a “do/ behaviour”.

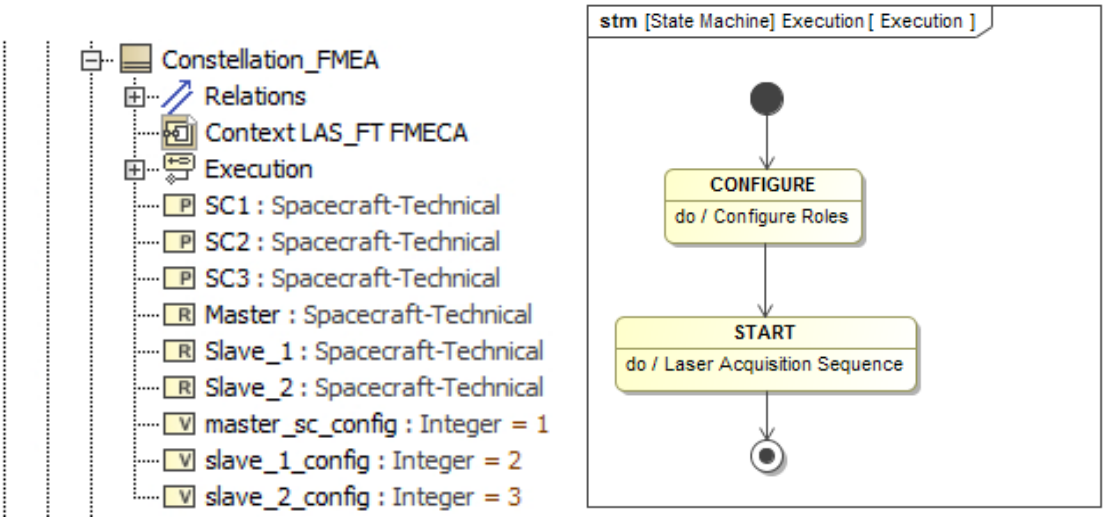

Figure 3.13: Left: Containment tree view of the execution context “Constellation_FMEA”. The part properties depicted with the symbol “P”, represents the three spacecrafts (SC1-2-3) in the constellation. The reference properties depicted with the symbol “R”, represents the three roles (master/reference, slave/transponder 1, slave/transponder 2) defined for the laser acquisition sequence. The value properties depicted with the symbol “V”, is used to assign a role to a specific spacecraft. Right: Classifier behaviour of the execution context as a state machine.

3.2.2 Laser Acquisition Operational Phase Behaviour

The operational phase behaviour elaborates on what each spacecraft in the constellation does to realize the laser acquisition sequence (refer to *section 1.2.3*). The activity diagram is used to model this scenario as defined by the MOFLT process. According to this scenario, the three spacecrafts, which are identical, are given separate roles as master, slave 1 and slave 2. These roles are depicted inside the activity diagram as swimlanes assigned to reference properties Master, Slave_1 and Slave_2. And according to these roles, individual operational tasks are performed.

Each operational task corresponds to a set of system functions that required to be in their so-called active mode (see *Figure 3.14*). It is important to note that it differs from switching a system on. A system can be on anytime, but it does not guarantee that it will provide the desired output. Whereas the logic in the function activation employed in the operational tasks is that, the desired output is

wanted exactly when the function is activated. An example can be given for the “Long-Arm interferometer” function. One can switch on the components responsible for long arm interferometry without the RX and local TX laser inputs are present. It will not provide a desired output until both laser inputs are present, however still, this action is operationally possible to do so and does not pose a failure. But on the functional view, the desired output of Long-Arm interferometer function can only be achieved when two laser inputs are present or else its constraints will fail, which means that the function is not performing correctly when needed i.e., activated.

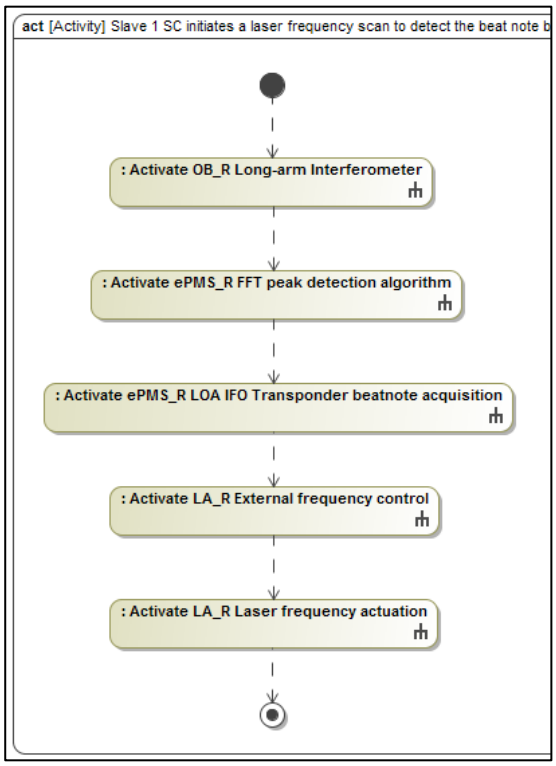

Figure 3.14: Functions to be activated during laser frequency locking operational task.

The activation is done by sending state transition signals to the respective functions using ALH API scripts (see *Figure 3.15*). The swimlanes in the operational phase behaviour diagram allow the execution to set its context into the respective reference property, which directs the context to one of the three spacecrafts. Ideally these commands should have been run inside the OBC of each spacecraft, and the signals should follow the functional connections. However, since the scope for this study is on the Payload functions, to keep the study simple this type of commanding is selected. After the command signal is sent, the execution goes into a loop node where it waits for the function to become active, and once done, it proceeds with the control flow. Notice that there is a decision node before this check loop which bypasses it, in case the “No Function” failure mode is set for the function.

It is important to point out that, with the adopted activation scheme, there is a check loop implemented with a direct feedback from the function mode status. It could be the case that the real system also follows a pre-defined timeline as an open loop commanding. This would open up new failure cases depending on the timing of the commands and functional response; however this is not included in the scope of this study.

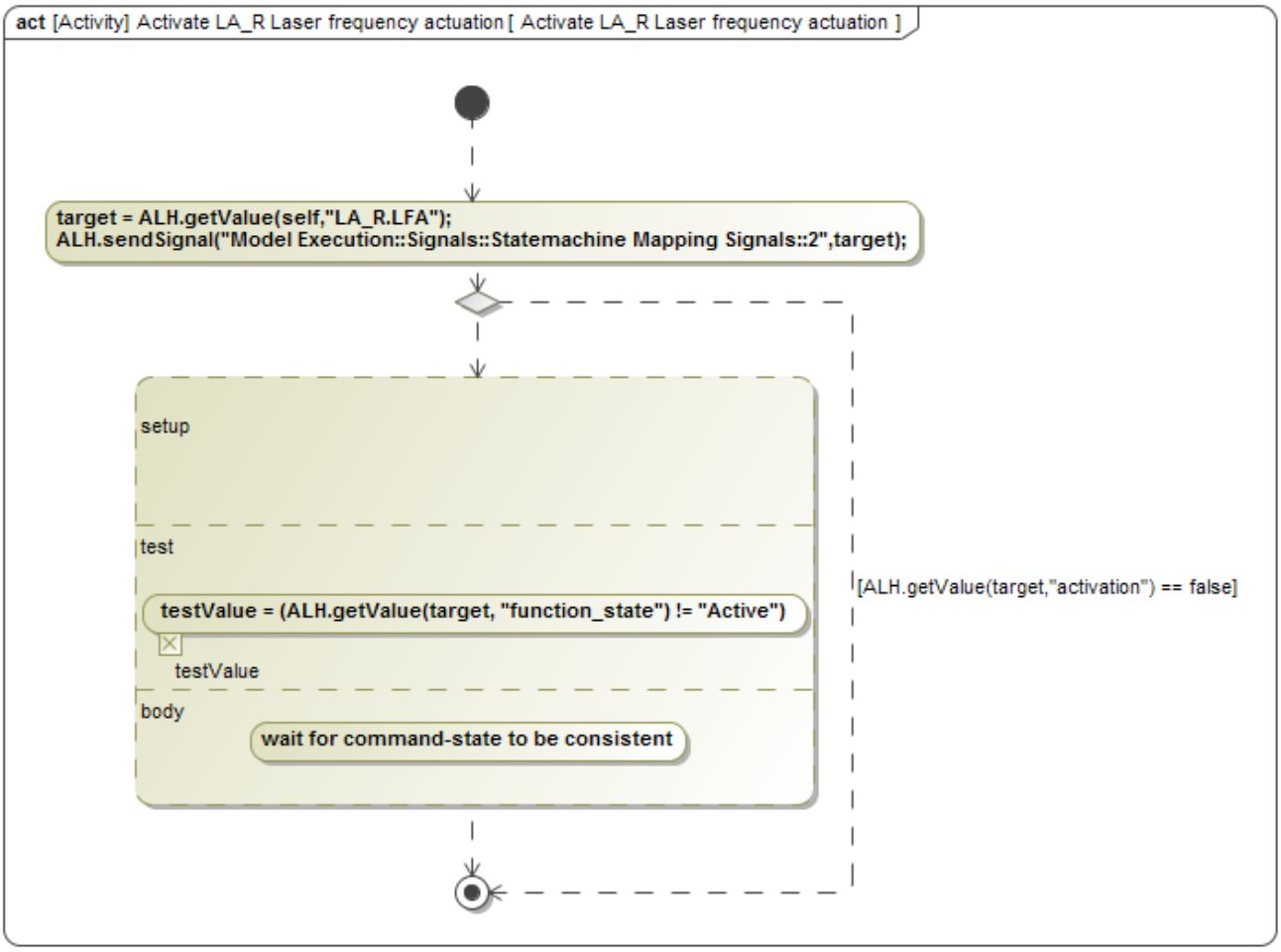

Figure 3.15: Activation of a target function implemented as an activity diagram. The first opaque action on top sends the signal that changes the mode of the desired function. The while loop node at the bottom checks and waits until the mode change has happened. The decision node in between is implemented to by-pass the check loop if a failure is deliberately injected to the function.

3.2.3 Execution Behaviour and Results

The execution is run using the simulation configuration named “LISA_FMEA”. After Cameo simulation toolkit finishes debugging the execution instance, the simulation waits for a starting command by the user. At this time, the user sets the activation property of the desired function inside the desired system into “false” to inject a failure (see *Figure 3.16*). After that, the start button is clicked in the simulation pane and the simulation is run. As a feature of the Cameo simulation toolkit, when a constraint is failed, it is highlighted in red including all the values that are present on the failure constraint through entire simulation context. This enables direct pinpointing of the failure location and propagation during execution. The execution runs over the laser acquisition sequence scenario, which takes around 2 minutes in a moderate business laptop (Intel i5-1135G7 processor, 8GB RAM). After the execution is finished, the results are saved as an instance into the pre-designated “Results” package.

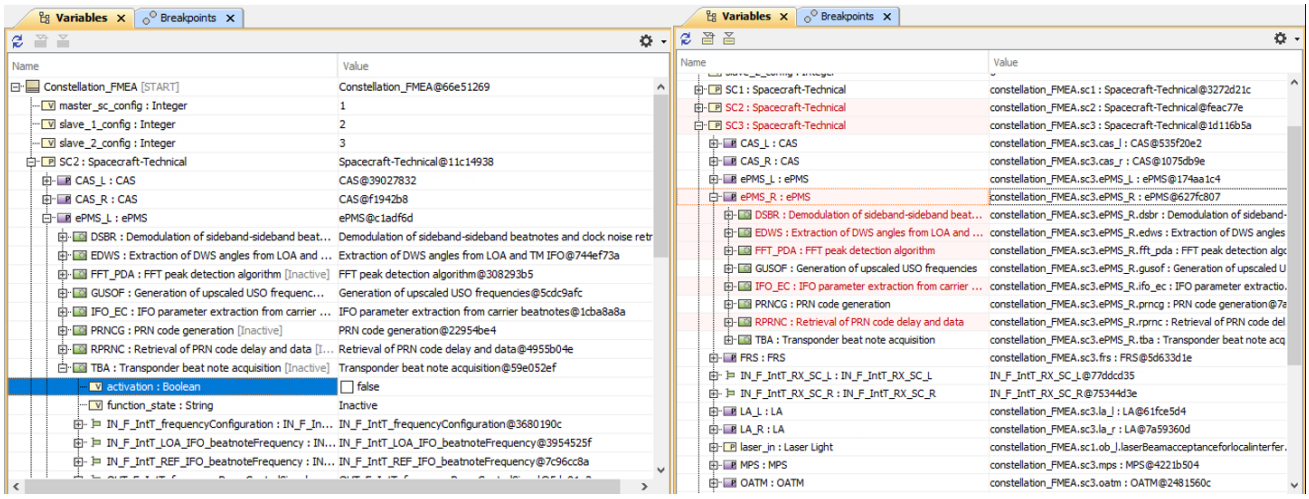

Figure 3.16: Left: Failure is injected to the SC2, left ePMS, Transponder beat note acquisition function by turning its activation value property from “true” to “false”. Right: During execution, model elements with failed parameters are highlighted in red.

In order to analyse the results, the instance is carried into another package called “Result Analysis”, with two tables inside called “Failure Mode”, and “Failure Effects”. The failure mode table searches for a slot instance that has a value “false” (see *Figure 3.17*). This slot corresponds to the activation value property that has been set to false beforehand to inject failure. The table columns are then configured to reflect which function, component, requirement, and system it belongs to, showing the comprehensive awareness of where the failure is injected. The failure effects table on the other hand, searches for a slot instance that has a value “fail” (see *Figure 3.19*). These slots correspond to the constraint properties that are violated during the simulation. The table columns are then configured to reflect which function, component, requirement, and system it belongs to, showing the comprehensive awareness of where the injected failure has propagated. In addition, the relation map “Bottom-up functional breakdown analysis” can be used to display the parent functions effected in the higher layers of abstraction for comprehensive analysis (see *Figure 3.18*). The analysis can be re-done for other simulation results by replacing the old simulation instance with the new one inside the “Result Analysis” package. The entries in the tables are then automatically updated. This feature is particularly useful to update the FMEA tables if the model is changed in the course of the project.

#	Component	Function	Requirement	Owner	Defining Feature	Value
70	ePMS	Transponder beat note acquisition	35 PAYL-22	constellation_FMEA.sc2.ePMS_L.tba : Tr…	activation : Boolean = true	false

Figure 3.17: The Failure Mode table shows detailed info on the injected failure.

Figure 3.18: The “Bottom up functional breakdown analysis” relation map is useful to observe the higher level effects of a failure happening inside a function.

#	Component	Function	Requirement	Owner	Defining Feature	Value
381	ePMS	Demodulation of sideband-sideband beatnotes and clock noise retrieval	SR 16 PAYL-3	constellation_FMEA.sc2.ePMS_L.dsbr : Demodulation ...	LOA_IFO not frequency stabilized : LO...	fail
383	ePMS	Demodulation of sideband-sideband beatnotes and clock noise retrieval	SR 16 PAYL-3	constellation_FMEA.sc2.ePMS_L.dsbr : Demodulation ...	REF_IFO not frequency stabilized : RE...	fail
388	ePMS	Extraction of DWS angles from LOA and TM IFO	SR 18 PAYL-5	constellation_FMEA.sc2.ePMS_L.edws : Extraction of ...	LOA_IFO not frequency stabilized : LO...	fail
398	ePMS	IFO parameter extraction from carrier beatnotes	SR 21 PAYL-8	constellation_FMEA.sc2.ePMS_L.ifo_ec : IFO paramet...	LOA_IFO not frequency stabilized : LO...	fail
399	ePMS	IFO parameter extraction from carrier beatnotes	SR 21 PAYL-8	constellation_FMEA.sc2.ePMS_L.ifo_ec : IFO paramet...	REF_IFO not frequency stabilized : RE...	fail
406	ePMS	Retrieval of PRN code delay and data	SR 34 PAYL-21	constellation_FMEA.sc2.ePMS_L.rprnc : Retrieval of P...	LOA_IFO not frequency stabilized : LO...	fail
426	ePMS	Demodulation of sideband-sideband beatnotes and clock noise retrieval	SR 16 PAYL-3	constellation_FMEA.sc2.ePMS_R.dsbr : Demodulation ...	REF_IFO not frequency stabilized : RE...	fail
442	ePMS	IFO parameter extraction from carrier beatnotes	SR 21 PAYL-8	constellation_FMEA.sc2.ePMS_R.ifo_ec : IFO parame...	REF_IFO not frequency stabilized : RE...	fail
475	LA	External frequency control (transponder mode)	SR 17 PAYL-4	constellation_FMEA.sc2.la_l.efc : External frequency ...	Frequency control signal is not on : Fr...	fail
493	LA	Laser frequency actuation	SR 25 PAYL-12	constellation_FMEA.sc2.la_l.lfa : Laser frequency act...	Transponder and reference frequency...	fail
761	ePMS	Demodulation of sideband-sideband beatnotes and clock noise retrieval	SR 16 PAYL-3	constellation_FMEA.sc3.ePMS_R.dsbr : Demodulation ...	LOA_IFO not frequency stabilized : LO...	fail
768	ePMS	Extraction of DWS angles from LOA and TM IFO	SR 18 PAYL-5	constellation_FMEA.sc3.ePMS_R.edws : Extraction of...	LOA_IFO not frequency stabilized : LO...	fail
778	ePMS	IFO parameter extraction from carrier beatnotes	SR 21 PAYL-8	constellation_FMEA.sc3.ePMS_R.ifo_ec : IFO parame...	LOA_IFO not frequency stabilized : LO...	fail
786	ePMS	Retrieval of PRN code delay and data	SR 34 PAYL-21	constellation_FMEA.sc3.ePMS_R.rprnc : Retrieval of ...	LOA_IFO not frequency stabilized : LO...	fail

Figure 3.19: The Failure Effects table shows the propagation of the failure along the constellation. The causes of failure can be seen by looking at the names of the failed constraint properties in the “defining feature” column.

3.3 FUNCTIONAL FMEA APPROACHES FOR THE LISA MISSION

This chapter provides details on three proposed approaches for conducting a Functional FMEA study within a defined scope in the LISA mission. These approaches are as follows:

- Traditional document-based FMEA approach, described in *section 3.3.1*.
- Static model-supported FMEA approach, described in *section 3.3.2*.
- Executable model-based FMEA approach, described in *section 3.3.3*.

Each of these methods utilizes either the system architecture model or documents in different ways to conduct an FFMEA. The introduction of three different approaches allows for comparison and evaluation of the benefits and weaknesses of establishing a model-based process for FMEA activities (see *chapter 4.1*).

3.3.1 FMEA Study: Traditional Document Based Analysis

This section elaborates on the proposed approach for a traditional document-based functional FMEA study for the LISA mission. The approach has been implemented by the combination of the process followed when performing FFMEA in ADS at early project phases and ECSS-Q-ST-30-02C standard. An overall summary can be found in *section 3.3.1.1*. Information on the preparation phase that involves activities required to be done before establishing the FMEA table is described in *section 3.3.1.2.* Explanation on how the FMEA table entries are deduced and filled is given in *section 3.3.1.3.* Example case studies using the methodology are explained in *chapter 4.1.*

3.3.1.1 Traditional Document Based Analysis Approach Summary

Functional FMEA study in the LISA mission is categorized as an early phase study and involves various documents and stakeholders from different departments, that need to interact and harmonize on the final deliverable. The study involves two main roles as the RAMS Engineering and Systems Engineering with three main artefacts per each system of interest: Design Description Document, Functional Analysis Document and Requirement Document (see *Figure 3.20*).

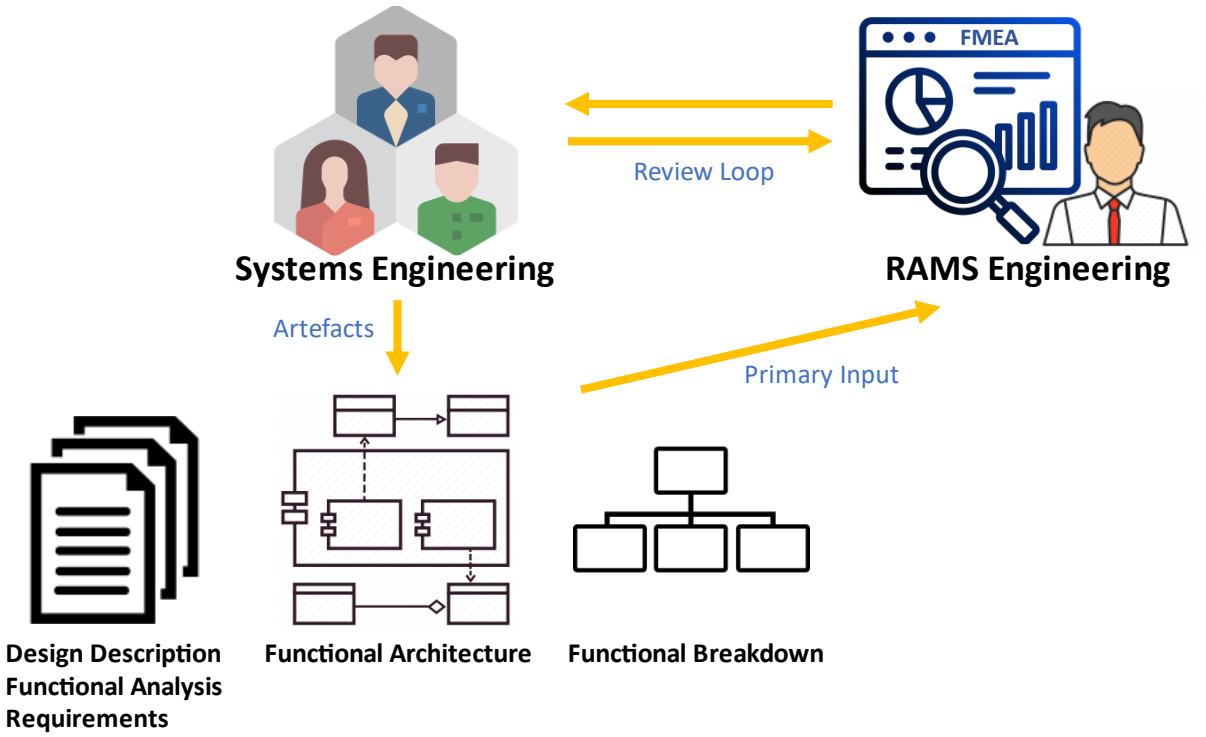

Figure 3.20: Context diagram of the traditional document-based FFMEA studies in early phases.

The main role that is responsible for the study is the RAMS engineering, which is stationed outside the project and becomes involved during the study. This role has to gather necessary information from the project, get familiar to it, compile it in the desired format and perform the analysis. The support role is being undertaken by the systems engineering, that is responsible to provide all required information to the RAMS engineering regarding the system definition and review the FMEA analysis in the context of the mission. The process (see *Figure 3.21*) is by nature iterative due to the complexity of the system and requires close collaboration in between the roles. The less clarity in documents, the more need for one-to-one review sessions and the time needed scales strongly with mission functional complexity.

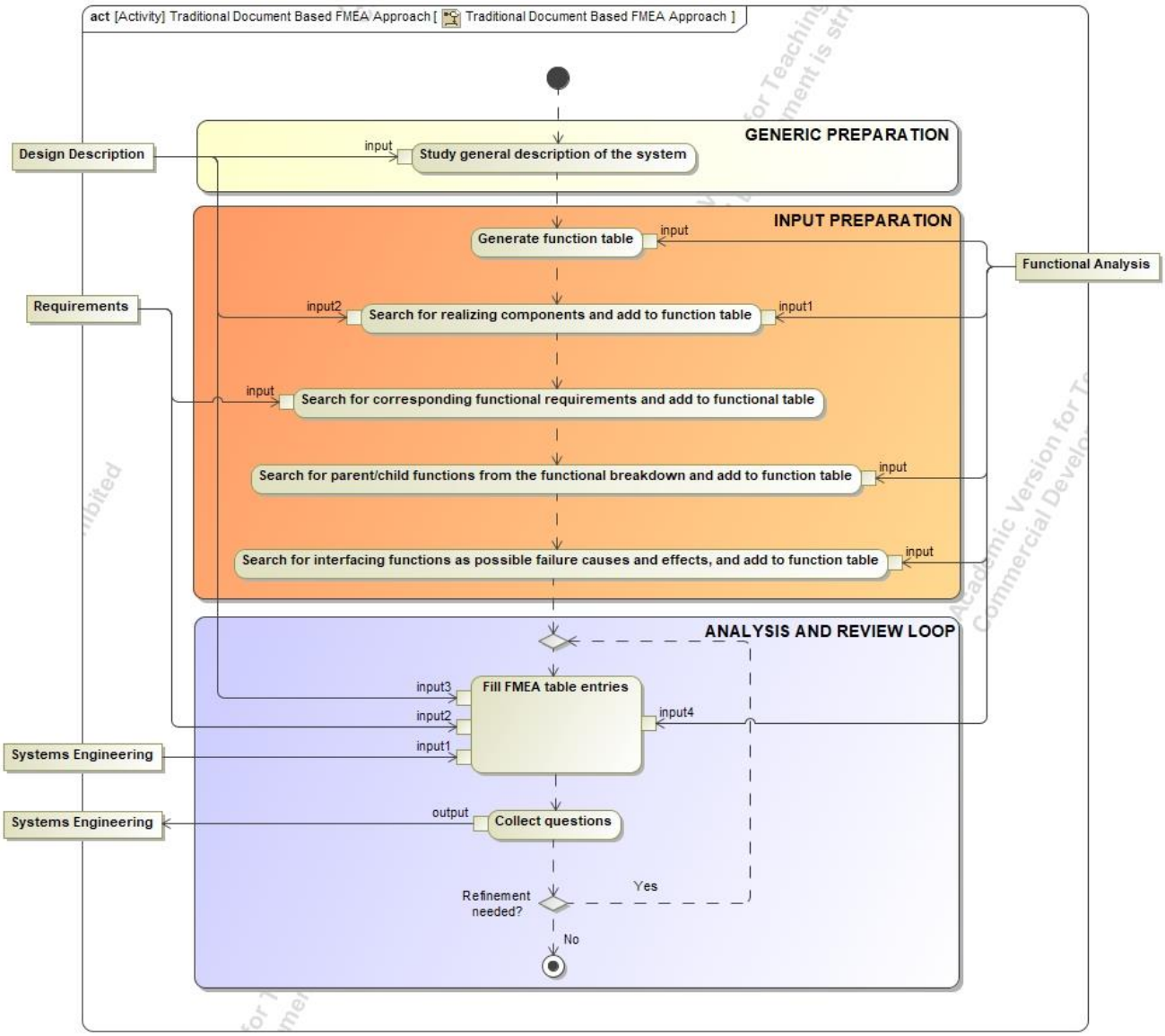

Figure 3.21: Process diagram for the traditional document based FMEA

3.3.1.2 Preparation

The document-based approach starts with the input from systems engineering as a flat list of functions and a functional breakdown. There are various options exists regarding the format these artefacts are supplied, and it is possible that different projects use different approaches. In the LISA mission FMEA study, the list is implemented as an excel file with the breakdown structure and the breakdown diagram is supplied as a picture or an html file. A functional architecture diagram is attached as a picture that shows the relationship between functions and systems of interests.

The inputs are supplemented by the aforementioned three documents. Descriptions of the functions can be found inside the functional analysis document and requirement document. The descriptions include what the function needs to achieve, which system it belongs to and the parameters it exchanges. The description of the system and system interfaces are found in the design description document, which gives a more physical understanding on the implementation.

When the inputs are supplied by the systems engineering, the compilation process on the RAMS engineering starts. RAMS engineering has to study the system design description first, to be able to familiarize with the system of interest and understand its generic working principle. Then the components and subsystems are added and matched to the list of functions. It follows with the requirements, and interfacing functions. When all this information is compiled (see *Table 3.2*), RAMS engineering can generate and start filling up the FMEA table.

Table 3.2: The desired format of information of RAMS engineering for the starting of FMEA study is a table that is the list of functions with the entries below.

Function	Name of the function
System	The system/component function belongs to
Description	Description of the function
Requirement	Requirement that the function is linked to
Function Input/Outputs	List of functional input/output parameters
Upstream Functions	List of functions that supply inputs to the function of interest
Downstream Functions	List of functions that receive outputs of the function of interest
Parent Functions	List of function that the function of interest is decomposed from
Child Functions	List of functions that the function of interest decompose into

3.3.1.3 Conducting FMEA

Using the compiled information, RAMS engineering starts to fill the first draft of the ECSS-Q-ST-30-02C FMEA table, first with the lowest level “leaf” functions that have been reflected in the functional architecture diagram. Following the functions and the connections inside the diagram, information for the FMEA table entries is deduced supplemented by the other input artefacts.

- Failure mode: In the scope of this thesis, only “No Function” failure mode is considered.
- Failure cause: The failure cause could be internal or due to the inputs from other functions. For the internal cause, if the function of interest has child functions, the failure mode of these functions are listed (no functionality of the child function) [16]. The functional breakdown structure is used to follow this traceability. In addition, all the inputs to the function of interest are listed inside this column with the upstream functions they come from. This is an assumption made, as done by Mhenni et.al. [24], for the “first draft” of FMEA results before being detailed or corrected during the review loop with the systems engineering.
- Mission phase / Operational mode: For the study conducted in the frame of this work, Mission phase / Operational mode is set as “Laser Acquisition Sequence”.
- Local failure effects: As a first entry, the “loss of function” is listed. Another effect would be the lack of outputs of the function of interest. This can also cause the failure of the downstream functions, for example, because no inputs are provided to these downstream functions. Therefore, in this column, all the outputs and downstream functions are listed. This is an assumption made, as done by Mhenni et.al. [24], for the “first draft” of FMEA results before being detailed or corrected during the review loop with the systems engineering.
- End failure effects: The end effect is defined as the failure mode of the parent function, which is “no functionality of the parent function”. The functional breakdown structure is used to follow this traceability.

The rest of the table entries are filled out within the interpretation of the RAMS engineering according to system nature, heritage information and engineering judgement.

From this point on, the first draft is finished, questions are collected and the review sessions with the systems engineering starts. In particular, the failures and local effects on the other functions must be agreed on within the context of the system definition. Some of the failures might pose no threat to the downstream functions, whereas some of them might propagate further even through other spacecrafts in the context of LISA constellation. This fact requires the expertise of the systems engineering to review the correctness and the reach of the failure cases. Through sessions in between roles and iterations, the final study is agreed on and delivered.

3.3.2 FMEA Study: Static Model-Supported Analysis

In the project, the “static model-supported FMEA approach” is defined as the process consisting of:

- generating inputs from the model for FMEA study directly, instead of supplying related documentation as the master input,
- and storing the output of the FMEA study (the filled out FMEA table) inside the model using Cameo SRAP profile.

The terminology “static” refers to the fact that there are no model execution and simulations involved in the process.

This section describes the process followed in this approach with an overall summary in *section 3.3.2.1*, information on the preparation phase in *section 3.3.2.2* and conducting the FMEA in *section 3.3.2.3*. Example case studies using the methodology are explained in *chapter 4.1*.

3.3.2.1 Static Model-Supported Analysis Approach Summary

Static model-supported approach (see *Figure 3.23*) shifts the information compilation process into the system architecture model instead of the responsibility of the RAMS engineering. In this approach, systems engineering generates the input table (see *Table 3.2*) directly in the model using the traceability links defined during default MOFLT systems engineering process. Therefore, the input table and all the associated model elements in its entries are available for the RAMS engineering discipline inside the model, to support conducting the FMEA study. It important to note that it is still possible to export the input table from the model, e.g., as an Excel file, if desired by the project team. The documents are still supplied, but as a supplement for RAMS engineering to refer to, for further details on the systems and functions that are not captured in the model (see *Figure 3.22*). RAMS engineering can benefit from the analysis maps and diagrams in the model to enhance their awareness on the system and the presentability of their study. The result of the study is stored inside the “Reliability and Safety” viewpoint of the model using the model elements from Cameo SRAP profile [19], with an FMEA table formatted in-line with the ECSS-Q-ST-30-02C standard (see *section 2.3.3*). This approach does not necessarily force RAMS engineering to fill up the table inside Cameo, since export and import from Excel is also possible, therefore offers a flexible working environment in between disciplines. Additionally, it allows a smooth transition towards a fully implemented MBSE work environment.

This approach assumes that the functional architecture modelling is already done as a part of the systems engineering process and stored in the model in a formalized way. In the LISA project, this had been achieved with the SysML model implemented in Cameo using MOFTL methodology, as described in *chapter 2.3*. If there are still documents that are detailing on the functions and the systems that are not involved in modelling scope, they are linked to the model (e.g. requirements database in DOORS).

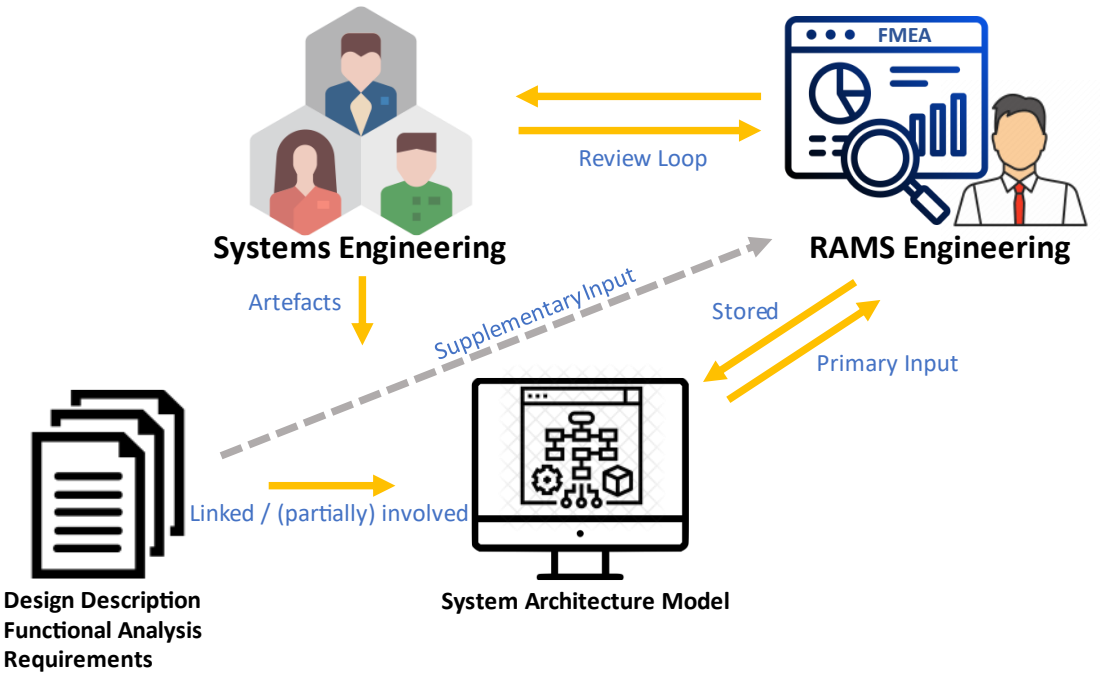

Figure 3.22: Context diagram for static model-supported FMEA

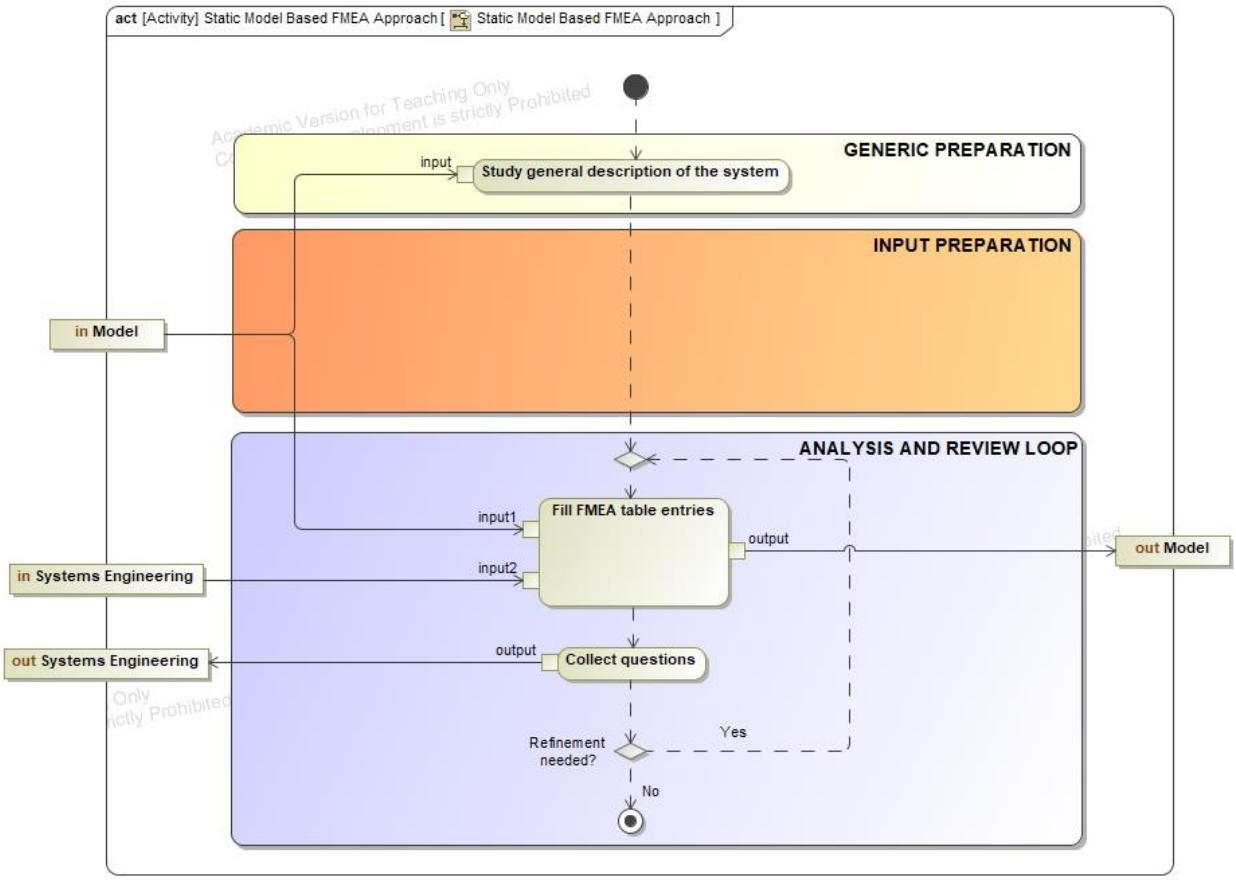

Figure 3.23: Process diagram for the static model-supported FMEA. Notice that the input preparation activities are eliminated since they are done by the model automatically.

3.3.2.2 Preparation

The preparation phase for the static model-supported approach is to establish the input to the RAMS engineering in the desired table format (see *Table 3.2*). Inside the model, several queries can be defined to compile this information.

First, using a generic table and metachain navigation (a querying method implemented in Cameo [35]), the functions that have been reflected in the functional architecture diagram are compiled. Then using the function-technical component link (“directed composition”) the system responsible for the performing of the function is shown. Function description and requirement are traced using the “satisfiedBy” link in between the function and the requirement elements. Functional input/outputs and the upstream downstream functions are traced using the functional connections in between functional parts inside the functional-technical architecture. And finally, the hierarchical parent/child functions are traced using the “directed decomposition” link in between the functions inside the functional breakdown structure (refer to traceability links descriptions in *section 3.1.1* and *Figure 3.4*).

3.3.2.3 Conducting FMEA

In the static model-supported approach, deducing the content of the FMEA table entities is exactly the same as the traditional document-based approach described in *section 3.3.1.3*. The main difference in this approach is that the way deduced information is stored and filled. Instead of a document or a dedicated software, the FMEA is conducted inside Cameo using the system definition in the model and dedicated model elements from Cameo SRAP FMEA profile.

As a first step, for the failure mode, which is identified as “No Function”, a failure mode element is created. This failure mode is then traced to the functions in the scope using the “failure mode” property. After this step, a macro called “create FMEA items” is run for each of the functions, which creates a “FMEA Item” model element. This is the main artefact that stores the main column entries of the FMEA, described in detail in *section 2.3.3*. For the rest of the column entries such as cause of failure, local effect of failure and final effect of failure, respective model elements are created and linked to the FMEA Item accordingly using the FMEA table inside the model (see *Figure 3.24*).

When the model links and elements for the functions are established, the FMEA table shows the functions and its entries dynamically, e.g., in case of a function change, or the system responsible for realizing it, the table will be automatically updated. The table is also interactive, allowing to navigate through model elements, or show their specifications when clicked on, which is used extensively for the reviewing process afterwards.

#	Name	Component	Item	Failure Mode	Cause Of Failure	Local Effect Of Failure	Final Effect Of Failure
1	Phase modulation Failure1	LA LA	Phase modulation	No Function	No laser light input from LA : Laser frequency actuation Internal No PRN modulation input from ePMS : PRN code generation No sideband modulation input from ePMS : Generation of upscaled USO freque	Failure in OB : Reference interferometer Loss of Function No laser light output Failure in OB : Long-arm interferometer Failure in OB : Laser beam acceptance for local interferometry Failure in OB : Laser beam acceptance for emission to remote SC	Failure in OMS : Optical transfer of local clock Failure in OMS : Ranging Failure in OMS : Optical Data Transfer
2	Reference interferometer Failure1	Optical_Bench OB	Reference interferometer	No Function	Internal No TX laser input from LA : Phase modulation No LO laser input from OB : Laser beam acceptance for local interferometry	No REF IFO optical beatnote output Loss of Function Failure in OB : Conversion of Photocurrent to Voltage	Failure in OMS : Transponder Mode Configuration Failure in OMS : Phase measurement between lasers
3	Laser frequency actuation Failure1	LA LA	Laser frequency actuation	No Function	Internal No stabilized laser input from LA : Power stabilization on OB No reference frequency control input from FRS : Internal frequency stabilisatio No transponder frequency control input from LA : External frequency control	Failure in LA : Phase modulation Failure in LA : External frequency control (transponder mode) Failure in FRS : Internal frequency stabilization Loss of Function No frequency actuated laser light output	Failure in OMS : Transponder Mode Configuration
4	Transponder beat note acquisition Failure1	ePMS ePMS	Transponder beat note acquisition	No Function	No LOA IFO beatnote frequency input from ePMS : FFT peak detection algorith No frequency configuration input from OBC : Command extended phasemeter No REF IFO beatnote frequency input from ePMS : FFT peak detection algorith Internal	Failure in LA : External frequency control (transponder mode) Loss of Function No frequency ramp control signal output	Failure in OMS : Transponder Mode Configuration
5	Conversion of Photocurrent to Voltage Failure1	OB Optical_Bench	Conversion of Photocurrent to Voltage	No Function	No REF IFO optical beat signal input from OB : Reference interferometer No LOA IFO optical beat signal input from OB : Long-arm interferometer Internal	Failure in ePMS : Extraction of DWS angles from LOA and TM IFO Failure in ePMS : Retrieval of PRN code delay and data Failure in ePMS : FFT peak detection algorithm No REF IFO electrical signal output No LOA IFO electrical signal output Failure in ePMS : IFO parameter extraction from carrier beatnotes Loss of Function Failure in ePMS : Demodulation of sideband-sideband beatnotes and c	Failure in OMS : Transponder Mode Configuration
6	Internal frequency stabilisation (reference mode) Failure1	FRS FRS	Internal frequency stabilisation (reference mode)	No Function	Internal No master locking scan parameters input from OBC : Command FRS	Loss of Function Failure in LA : Laser frequency actuation No reference frequency control output	Failure in OMS : Laser Mode Configuration Failure in OMS : Reference Mode Configuration

Figure 3.24: Excerpt from example FMEA table filled in using Cameo SRAP profile inside the model.

3.3.3 FMEA Study: Executable Model-Based Analysis

Executable model-based FMEA approach is defined as using the executable model simulations to identify the failure causes and effects in the functional architecture. This section describes the process followed in this approach with an overall summary in *section 3.3.3.1*, and information on conducting the FMEA in *section 3.3.3.2*. Example case studies using the methodology are explained in *chapter 4.1*.

3.3.3.1 Executable Model-Based Analysis Approach Summary

Executable model-based approach supports RAMS engineering during the deduction of failure causes and effects for the system functions along the functional chain by using the results of the failure case simulation of the executable model (as described in *section 3.2.3*). As a pre-requisite for this approach, the baseline MOFLT MBSE modelling activities with the prescribed extensions (as described in *chapter 3.1)* must be performed by Systems Engineering. The interactions between the stakeholders in this approach can be seen in *Figure 3.25.*

In terms of process (see *Figure 3.26*), this approach differs significantly from the previous ones, as it is more of a review task than deduction. The deduction of failure cause and effects is not performed manually by either RAMS discipline or by system engineering in this approach. This task is solely performed by the simulation of the system architecture model. Systems engineering is responsible for the correctness of the behaviour model as well as functional constraints, whereas RAMS engineering is responsible to operate the simulation and collect the results in a proper FMEA table format. Once done, the review session is initiated with Systems Engineering to either further investigate the simulation results, or edits in the FMEA table. This review session is not as extensive and cyclic as the review loops of the previous approaches. The reason is that the model provides extensive information in a clear format on the most complex/knowledge-intensive part of the FMEA study, which is the deduction of failure causes and effects. This approach also uses the same process as in static model-supported approach to compile and store desired information inside the model.

In terms of contents, the most significant difference is that this study is connected to functional behaviours and specific operational scenarios, meaning the failures can be directly traced to a non-nominal behaviour and its cause during a specific operation. This link can be used to pinpoint:

- the most critical causes of functional misbehaviours
- the most critical functions per each operation
- the most critical operations in the mission

This allows for necessary efforts to be directed towards strengthening these identified elements.

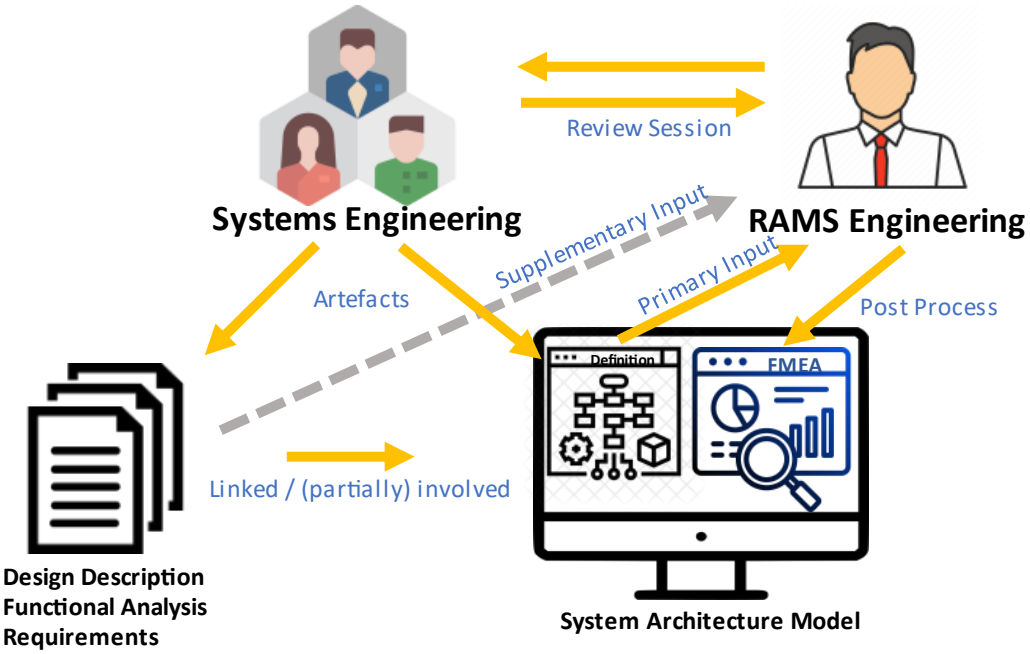

Figure 3.25: Context diagram for executable model based FMEA.

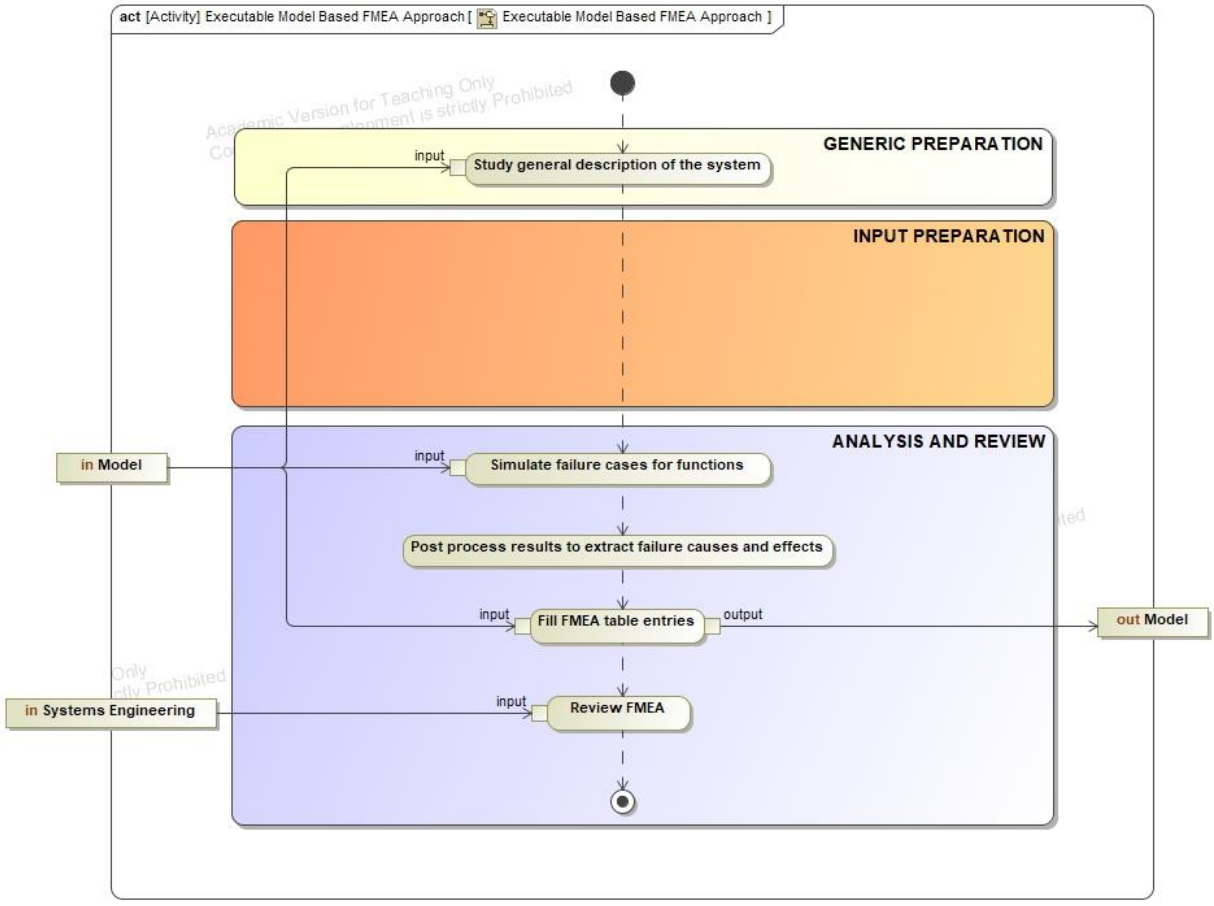

Figure 3.26: Process diagram for the executable model based FMEA. Notice that the review loop is eliminated as the model is generating the failure cause and effects information through simulations. A generic review with Systems Engineering might be needed at the end of the study.

3.3.3.2 Conducting FMEA

In the executable model-based approach, the execution context is run for each of the functions in the architecture, one by one, with the “No Function” failure mode applied as described in *section 3.2.3*. The failures that result in the “Failure Effects” table (*see Figure 3.19)* corresponds to the Failure Effects entry of the FMEA table. On the other hand, the constraints of the function (described in *section 3.1.3.3*) corresponds to the Failure Cause entry of the FMEA table.

An important point here is that there are two possible ways to conduct FMEA using this approach, with different failure effect results that answer the following questions:

- Generic FMEA: Which functions could be affected when “Function X” fails?
- Detailed FMEA: Which functions, in which systems are affected when “Function X” fails in “System Y”?

The generic FMEA includes all the possible failure effects independent of the system it happens in. In order to collect all the functional failure effects that can possibly happen, independent of the system it is used in, “No Function” failure mode is applied to all the instances of a specific function. As an example, a LA function has 2 instances per spacecraft, and along the constellation it has 6 instances. It could be that failure of each 6 instances could have different failure effects due to different operations of the systems. In order to capture every possible failure effect, all 6 instances have to be failed for the analysis of one function. After execution, the model generates the list of all failed function instances for entire constellation, which would include same functions failed inside different spacecrafts or systems. These repetitions need to be filtered out, as in FMEA table, entries are listed only once. Therefore, a simple post processing step is needed. In this study, the Failure Effects table in Cameo is exported to Excel and using the “Remove Duplicates” tool on the function name column, the list is reduced to the unique list of functions as failure effects. This list can then be added to the FMEA table as described in the previous step. For the failure causes there is no need to run the simulations again and they can be deduced from the list of function constraints.

On the other hand, the detailed FMEA is a system dependent way of reporting failures. It analyses the failure mode for each instance of a particular function, realized in different systems along the constellation (e.g., 6 instances of the “Laser frequency actuation” function in 6 LAs along constellation). Therefore, it includes unique failure effects that are identified along the constellation within the individual systems, in the context of their respective operational tasks. For this way, the process is to have the “Failure Mode” and “Failure Effects” table (*see Figure 3.17 and Figure 3.19)* for each instance of the functions in the architecture and list these in individual FMEA table rows. Although there is no additional modelling effort involved, it should be noted that the number of simulations needed will be equivalent to the number of function instances (e.g., 6 individual simulations have to be run for the 6 instances of the “Laser frequency actuation” function). The Failure Cause entry of the FMEA table still remains as the list of function constraints. Note that, post processing is not needed for this way of conducting FMEA. If desired, this way of reporting failures in individual systems and spacecraft can be adopted as a baseline. This would identify in which system of which spacecraft the functional failure is the most critical one and create a major situational awareness for the development team.

It is important to note that both methods are still dependent on the operational scenario that is being simulated. If during the operations, the failed function of interest is not used, it will not be captured in the failure effects results of the simulation. The different results of the two methods can be observed in *Table 4.6* and *Table 4.7.* If possible, it is recommended to include both FMEA approaches in the failure assessment because,

- Generic FMEA shows the worst-case combination by collecting all the functional failures that can happen during an operational scenario,

- Detailed FMEA shows the exact propagation of a specific failure along the constellation during an operational scenario.

This combination of both information is valuable for understanding the behaviour and impact of failures in the LISA constellation.

4 RESULTS AND DISCUSSION

This chapter presents the results of the work performed to compare the three FMEA results and methodology (traditional document-based, static model-supported, and executable model-based) in *chapter 4.1* and the contributions of the thesis work for the investigated problem in *chapter 4.2.*

4.1 FMEA COMPARISON

This chapter compares and discusses on the methodology and results of the three FMEA studies performed within this thesis. The comparison of the quality of results obtained from three different approaches is described in *section 4.1.1*. The comparison of the methodologies according to selected evaluation criteria (e.g., effort for preparation, information access and traceability etc.) is described in *section 4.1.2*. Discussion and conclusion on the comparison can be found in *section 4.1.3.*

4.1.1 Comparison of Examples from Investigated FMEA Results

This section investigates example FMEAs with 3 case studies:

- Correctness in results,
- Impact of architecture changes on the results
- Impact of operational scenario on the results

It's important to note that the results presented in this section correspond to the "first draft" of the FMEA, which was conducted solely by the RAMS engineering discipline, with or without the support of the model. This initial analysis does not involve review sessions with systems engineering, nor a detailed study and understanding of functional behaviours and descriptions by the individual conducting the analysis. This assumption is essential to ensure a meaningful comparison of the results, since, in terms of content, all three FMEA methods would result in the same manner at the end, if there were enough knowledge about the system, and enough time and effort was spent.

4.1.1.1 Case 1: Correctness in Results

The first case is the difference in the correctness of the results obtained. As an example, the “Laser frequency actuation” function belonging to LA is given. This function is responsible for changing the incoming laser frequency according to control inputs received from either “External frequency control” or “Internal frequency stabilization” functions. This function is necessary in order to lock the frequency of the laser onto either the laser light of a remote spacecraft or the neighbouring LA, which is eventually required for the science measurement. The frequency actuated laser light output of the function is then used by the phase modulation function where its phase is modulated with information for inter-spacecraft communication.

For the traditional document-based approach, as described in *section 3.3.1.2*, from the documents the input table is compiled for this function (see *Table 4.1*). On the contrary, for the static model-supported approach, the input table is compiled automatically from the model as described in *section 3.3.2.2*. Using the information from input table and the functional architecture view in *Figure 6.1*, FMEA table row is filled out as described in *section 3.3.1.3 and 3.3.2.3*. On the executable model-based FMEA side, simulation is run by failing the function for all LAs in the constellation, and the generic results are collected following the process described in *section 3.3.3.2***.**

Investigating the “first draft” results obtained from traditional document based and also from the static model-supported approaches in *Table 4.2*, it can be seen that there are some incorrect deductions. The assumptions that correlate upstream-downstream functions to the failure cause and failure effects

(refer to *section 3.3.1.3*) turn out to be incorrect. For example, in reality, the laser light frequency can still be actuated without the need for intensity stabilization. Therefore, a non-stabilized intensity input is not a cause of failure for the function of interest. In addition, the failure of the “Laser frequency actuation” function does not impose a local effect of failure in the “Phase modulation” function since the laser light phase can still be modulated even if the frequency is not actuated. This entry needs to be corrected during the review loop with the systems engineering. On the contrary, when looking at the results of the executable FMEA approach, one can see that the “Phase modulation” function is not affected as the system behaviour dictates, showing the correct failure effects even 5 levels downstream that comes after the “Phase modulation” function.

Table 4.1: Input table for the “Laser frequency actuation” function. Compiled either manually using document-based approach or automatically using the model-based approach.

Function	Laser frequency actuation
System	LA
Description	Actuate the laser frequency
Requirement	PAYL-12
Function Inputs/Outputs	in IN_F_IntT_stabilizedLaserLight in IN_F_IntT_referenceFrequencyControl in IN_F_IntT_transponderFrequencyControl out OUT_F_InT_frequencyActuatedLaserLight
Upstream Functions	Internal frequency stabilisation (reference mode) Power stabilization on OB External frequency control (transponder mode)
Downstream Functions	Internal frequency stabilisation (reference mode) Phase modulation External frequency control (transponder mode)
Parent Functions	Transponder Mode Configuration

Table 4.2: FMEA table for “Laser frequency actuation” failure. First row filled using the static approaches, whereas second row filled using the generic executable approach. Incorrect assumptions of static approaches are given in red colour, whereas additional entries from executable approach are highlighted in blue.

Item / block	Function	Failure Mode	Failure Cause	Mission Phase / Op. mode	Failure Effects a. Local Effect Of Failure	Failure Effects b. Final Effect Of Failure
LA	Laser frequency actuation	No Function	Internal No stabilized laser input from LA : Power stabilization on OB No reference frequency control input from FRS : Internal frequency stabilisation No transponder frequency control input from LA : External frequency control	Laser Acquisition Sequence	Loss of Function No frequency actuated laser light output Failure in LA : External frequency control Failure in LA : Phase modulation Failure in FRS : Internal frequency stabilization	Failure in OMS : Transponder Mode Configuration
LA	Laser frequency actuation	No Function	Internal No reference and transponder frequency control input No laser light input	Laser Acquisition Sequence	Loss of Function No frequency actuated laser light output Failure in ePMS : Extraction of DWS angles from LOA and TM IFO Failure in ePMS : IFO parameter extraction from carrier beatnotes Failure in ePMS : Retrieval of PRN code delay and data	Failure in OMS : Transponder Mode Configuration

					Failure in ePMS : Demodulation of sideband-sideband beatnotes and clock noise retrieval	

4.1.1.2 Case 2: Impact of Architecture Changes on the Results

The second case is to compare the impact of architecture changes on the results. For this purpose, a small change in LA architecture is done by switching the places of “Laser frequency actuation” and “Phase modulation” functions. This small change causes a relatively significant impact on the interfaces of the “Laser frequency actuation” function, adding 3 more functions to its downstream (see *Table 4.1* and *Table 4.3*). For the document based and static model-supported approaches, this change of architecture effects the result of the first draft significantly (see first rows of *Table 4.2* and *Table 4.4*). As the assumption has been made that all the downstream functions are affected by the failure in the function of interest, the new list of downstream functions is included in the FMEA table entry.

However, looking at the system behaviour, there should be no change in the FMEA results of both architectures. The change in the sequence of the functional chain actually have no effect on the system behaviour as “Laser frequency actuation” and “Phase modulation” functions are independent of each other. The executable model-based approach can successfully capture this phenomena (see second rows of *Table 4.2* and *Table 4.4* being exactly the same), since it only considers the functional behaviours and interface exchanges instead of the “architectural picture”. On the other hand, not only static approaches resulted in an unnecessary change in the table entries but also, they have to be checked and corrected referring to the functional behaviour descriptions.

This example shows the big difference between the flexibility and robustness of executable and static methods. With the static methods, each architectural change has to be analysed all over again, whether there should be a modification in failure effects and causes. On the other hand, the executable method only needs a simulation run (as described in *section 3.2.3*) to correctly show what the new effects are, independent of the scale of the change in the architecture view.

Table 4.3: Input table for the alternative architecture “Laser frequency actuation” function. Compiled either manually using document-based approach or automatically using the model-based approach.

Function	Laser frequency actuation
System	LA
Description	Actuate the laser frequency
Requirement	PAYL-12
Function Inputs/Outputs	in IN_F_IntT_phaseModulatedLaserLight in IN_F_IntT_referenceFrequencyControl in IN_F_IntT_transponderFrequencyControl out OUT_F_InT_frequencyActuatedLaserLight
Upstream Functions	Internal frequency stabilisation (reference mode) Phase modulation External frequency control (transponder mode)
Downstream Functions	Internal frequency stabilisation (reference mode) External frequency control (transponder mode) Reference Interferometer Long-Arm Interferometer Laser beam acceptance for local interferometry Laser beam acceptance for emission to remote SC

Parent Functions	Transponder Mode Configuration

Table 4.4: FMEA table for alternative architecture “Laser frequency actuation” failure. First row filled using the static approaches, whereas second row filled using the generic executable approach. Incorrect assumptions of static approaches are given in red colour, whereas additional entries from executable approach are highlighted in blue.

Item / block	Function	Failure Mode	Failure Cause	Mission Phase / Op. mode	Failure Effects a. Local Effect Of Failure	Failure Effects b. Final Effect Of Failure
LA	Laser frequency actuation	No Function	Internal No phase modulated laser input from LA : Phase modulation No reference frequency control input from FRS : Internal frequency stabilisation No transponder frequency control input from LA : External frequency control	Laser Acquisition Sequence	Loss of Function No frequency actuated laser light output Failure in LA : External frequency control Failure in FRS : Internal frequency stabilization Failure in OB : Reference Interferometer Failure in OB : Long-Arm Interferometer Failure in OB : Laser beam acceptance for local interferometry Failure in OB : Laser beam acceptance for emission to remote SC	Failure in OMS : Transponder Mode Configuration
LA	Laser frequency actuation	No Function	Internal No reference and transponder frequency control input No laser light input	Laser Acquisition Sequence	Loss of Function No frequency actuated laser light output Failure in ePMS : Extraction of DWS angles from LOA and TM IFO Failure in ePMS : IFO parameter extraction from carrier beatnotes Failure in ePMS : Retrieval of PRN code delay and data Failure in ePMS : Demodulation of sideband-sideband beatnotes and clock noise retrieval	Failure in OMS : Transponder Mode Configuration

4.1.1.3 Case 3: Impact of Operational Scenario on the Results

The third case is to compare the impact of operational scenario on the results. As the static approaches consider all the functions are active all the time, and the executable approach considers the activated functions according to the operational scenario, there are significant differences in the results. As an example, “Internal Frequency Stabilization” function belonging to FRS is given. This function is responsible for stabilizing the laser frequency by locking it onto the optical cavity inside the FRS. This function is necessary in order to stabilize the laser frequencies of the whole constellation, which is eventually required for the science measurement and inter-spacecraft communication.

Table 4.5: Input table for the Internal frequency stabilisation function. Compiled either manually using document-based approach or automatically using the model-based approach.

Function	Internal frequency stabilisation (reference mode)
System	FRS
Description	Ensure that at least one of both laser assemblies can be stabilized in frequency to an optical frequency reference (master mode).
Requirement	PAYL-10
Function Inputs/Outputs	in IN_F_IntT_masterLockingScanParameters in IN_F_InT_frequencyActuatedLaserLight out OUT_F_IntT_referenceFrequencyControl
Upstream Functions	Command FRS

	Laser frequency actuation
Downstream Functions	Laser frequency actuation
Parent Functions	Reference Mode Configuration Laser Mode Configuration

Based on the information from *Table 4.5* and the functional architecture, one can follow the static approach to deduce the failure effects to conduct FMEA, resulting in the first row of *Table 4.6*. There are two fundamental problems in this result:

- Tracing the local effects to only neighbouring (one upstream & downstream) functions does not capture the full effect of the failure of this function. The function failure creates a drastic propagation of errors and failures in the payload, which affects the whole constellation. To handle this correctly in the sense of a realistic failure effects analysis, RAMS engineering has to trace this failure along the constellation architecture. All the effected functions have to be manually identified one by one with consultation to functional descriptions and systems engineering during a review loop.
- Operationally, this function is active in only one of the spacecrafts (master/reference SC) and only for one of the LAs. Therefore, the failure in different components shows different effects. If the failure occurs in the function stabilizing the master/reference LA, the whole constellation is impacted as an effect. On the contrary, a failure in the “Internal frequency stabilization” function that is responsible for stabilizing the rest of the 5 LAs will not have any effect on the constellation operations, because it is not used.

On the other hand, as shown in *Table 4.7*, a detailed model-based FMEA (as described in *section 3.3.3.2*) is conducted on the 6 different instances of “Internal Frequency Stabilization” function. The analysis reveals that only the function inside the FRS of Spacecraft 1, that is responsible for stabilizing the Left LA (master/reference LA), ends up in a drastic functional failure effect propagating along the constellation. This shows that the static approaches fail to reflect the listed aspects of the functional failure, whereas the executable model-based approach captures the full effect of the failure in all the components it is allocated to, according to the operational use.

This example, even though obvious for the payload experts, is especially chosen to be easy to grasp the concept. However, the same difference in results goes for all other functions in the architecture. These functions may not be so intuitive to investigate, as their failure might propagate differently depending on which component and during which operation it happens. This is a key limitation in the early phase functional FMEA that uses static approaches. Because, for a non-expert on the system functions (in this case RAMS engineering), there is limited expertise, information or intuition on how much the operational aspects affect the behaviour and the level of propagation of a specific functional failure. This results in a limited idea on how much effort that has to be spent on to analyse a specific function, which eventually might end up in an FMEA not reflecting the full coverage of a functional failure.

It is important to note that, according to the preference of the RAMS and systems engineering, the number of failure effects entries can be reduced. Or a generic failure case independent of the components (generic FMEA approach) can be investigated by only listing the affected functions once (see second row of *Table 4.6*). This is achieved by failing the function of interest in all the components along constellation and removing the duplicates of the functions as described in *section 3.3.3.2*.

Table 4.6: FMEA table for Internal frequency stabilisation (reference mode) failure. First row filled using the static approaches, whereas second row filled using the generic executable approach. Entries from executable approach are highlighted in blue.

Item / block	Function	Failure Mode	Failure Cause	Mission Phase / Op. mode	Failure Effects a. Local Effect Of Failure	Failure Effects b. Final Effect Of Failure
FRS	Internal frequency stabilisation (reference mode)	No Function	Internal No master locking scan parameters input from OBC : Command FRS No frequency actuated laser light input from LA : Laser frequency actuation	Laser Acquisition Sequence	Loss of Function No reference frequency control output Failure in LA : Laser frequency actuation	Failure in OMS : Reference Mode Configuration Failure in OMS : Laser Mode Configuration
FRS	Internal frequency stabilisation (reference mode)	No Function	Internal No master locking scan parameters input No laser light input	Laser Acquisition Sequence	Loss of Function No reference frequency control output Failure in LA : Laser frequency actuation Failure in ePMS : Extraction of DWS angles from LOA and TM IFO Failure in ePMS : IFO parameter extraction from carrier beatnotes Failure in ePMS : Retrieval of PRN code delay and data Failure in ePMS : Demodulation of sideband-sideband beatnotes and clock noise retrieval	Failure in OMS : Reference Mode Configuration Failure in OMS : Laser Mode Configuration

Table 4.7: Part of an FMEA table for “Internal frequency stabilisation (reference mode)” failure, filled in using the detailed executable model-based approach. All the 6 instances of the function and where they belong in the constellation are shown in first two columns. The failure effects are captured with their exact location in the constellation (e.g., constellation.sc1.ePMS_L). The cause of failure for the affected functions are also listed next to their respective entries. As anticipated, the functions responsible for stabilizing slave/transponder LAs does not have failure effects as they are not used operationally.

Item / block	Function	Failure Effects (with Failure Cause)
SC1.FRS	Internal frequency stabilisation (reference mode) – LA_L	constellation.sc1.ePMS_L: Demodulation of sideband-sideband beatnotes and clock noise retrieval - LOA_IFO not frequency stabilized constellation.sc1.ePMS_L: Demodulation of sideband-sideband beatnotes and clock noise retrieval - REF_IFO not frequency stabilized constellation.sc1.ePMS_L: Extraction of DWS angles from LOA and TM IFO - LOA_IFO not frequency stabilized constellation.sc1.ePMS_L: IFO parameter extraction from carrier beatnotes - LOA_IFO not frequency stabilized constellation.sc1.ePMS_L.: IFO parameter extraction from carrier beatnotes - REF_IFO not frequency stabilized constellation.sc1.ePMS_L: Retrieval of PRN code delay and data - LOA_IFO not frequency stabilized constellation.sc1.ePMS_R: Demodulation of sideband-sideband beatnotes and clock noise retrieval - LOA_IFO not frequency stabilized constellation.sc1.ePMS_R: Demodulation of sideband-sideband beatnotes and clock noise retrieval - REF_IFO not frequency stabilized constellation.sc1.ePMS_R: Extraction of DWS angles from LOA and TM IFO - LOA_IFO not frequency stabilized constellation.sc1.ePMS_R: IFO parameter extraction from carrier beatnotes - LOA_IFO not frequency stabilized constellation.sc1.ePMS_R: IFO parameter extraction from carrier beatnotes - REF_IFO not frequency stabilized constellation.sc1.ePMS_R: Retrieval of PRN code delay and data - LOA_IFO not frequency stabilized constellation.sc1.LA_L: Laser frequency actuation - Transponder and reference frequency control is not on constellation.sc2.ePMS_L: Demodulation of sideband-sideband beatnotes and clock noise retrieval - LOA_IFO not frequency stabilized constellation.sc2.ePMS_L: Demodulation of sideband-sideband beatnotes and clock noise retrieval - REF_IFO not frequency stabilized constellation.sc2.ePMS_L: Extraction of DWS angles from LOA and TM IFO - LOA_IFO not frequency stabilized constellation.sc2.ePMS_L: IFO parameter extraction from carrier beatnotes - LOA_IFO not frequency stabilized constellation.sc2.ePMS_L: IFO parameter extraction from carrier beatnotes - REF_IFO not frequency stabilized constellation.sc2.ePMS_L: Retrieval of PRN code delay and data - LOA_IFO not frequency stabilized constellation.sc2.ePMS_R: Demodulation of sideband-sideband beatnotes and clock noise retrieval - LOA_IFO not frequency stabilized constellation.sc2.ePMS_R: Demodulation of sideband-sideband beatnotes and clock noise retrieval - REF_IFO not frequency stabilized constellation.sc2.ePMS_R: Extraction of DWS angles from LOA and TM IFO - LOA_IFO not frequency stabilized constellation.sc2.ePMS_R: IFO parameter extraction from carrier beatnotes - LOA_IFO not frequency stabilized constellation.sc2.ePMS_R: IFO parameter extraction from carrier beatnotes - REF_IFO not frequency stabilized constellation.sc2.ePMS_R: Retrieval of PRN code delay and data - LOA_IFO not frequency stabilized constellation.sc3.ePMS_L: Demodulation of sideband-sideband beatnotes and clock noise retrieval - LOA_IFO not frequency stabilized constellation.sc3.ePMS_L: Demodulation of sideband-sideband beatnotes and clock noise retrieval - REF_IFO not frequency stabilized constellation.sc3.ePMS_L: Extraction of DWS angles from LOA and TM IFO - LOA_IFO not frequency stabilized constellation.sc3.ePMS_L: IFO parameter extraction from carrier beatnotes - LOA_IFO not frequency stabilized constellation.sc3.ePMS_L: IFO parameter extraction from carrier beatnotes - REF_IFO not frequency stabilized constellation.sc3.ePMS_L: Retrieval of PRN code delay and data - LOA_IFO not frequency stabilized constellation.sc3.ePMS_R: Demodulation of sideband-sideband beatnotes and clock noise retrieval - LOA_IFO not frequency stabilized constellation.sc3.ePMS_R: Demodulation of sideband-sideband beatnotes and clock noise retrieval - REF_IFO not frequency stabilized constellation.sc3.ePMS_R: Extraction of DWS angles from LOA and TM IFO - LOA_IFO not frequency stabilized constellation.sc3.ePMS_R: IFO parameter extraction from carrier beatnotes - LOA_IFO not frequency stabilized constellation.sc3.ePMS_R: IFO parameter extraction from carrier beatnotes - REF_IFO not frequency stabilized

		constellation.sc3.ePMS_R: Retrieval of PRN code delay and data - LOA_IFO not frequency stabilized
SC1.FRS	Internal frequency stabilisation (reference mode) – LA_R	No Failure Effect
SC2.FRS	Internal frequency stabilisation (reference mode) – LA_L	No Failure Effect
SC2.FRS	Internal frequency stabilisation (reference mode) – LA_R	No Failure Effect
SC3.FRS	Internal frequency stabilisation (reference mode) – LA_L	No Failure Effect
SC3.FRS	Internal frequency stabilisation (reference mode) – LA_R	No Failure Effect

4.1.2 Discussion and Assessment of Investigated FMEA Methodologies for the LISA Mission

In this section, the methodologies are compared and assessed for the traditional document based, static model-supported and executable model based FMEA approaches according to the defined criteria in *Table 4.8*. The criteria are determined in the light of reflecting the aspects that are encountered during the project evolution and are considered important for an effective FMEA analysis by the project team.

Table 4.8: Criteria table showing the comparison criteria with descriptions used for evaluating the different FMEA methodologies.

Criteria	Description
Quality of Input	Correctness and level of detail in information of the input to the FMEA study.
Effort for Preparation	Effort needed for the preparation before starting to fill the FMEA table entries.
Information Access and Traceability	Convenience to access to desired information and trace in between functions-requirements-systems-failures.
Quality of Results	Correctness and level of detail in information of the FMEA entries of the “first draft” before systems engineering review (only Failure Cause and Effects are considered).
Effort to Obtain Results	Effort needed to obtain Failure Cause and Effects information.
Filling the FMEA Table	Convenience to fill the FMEA table entries.
Effort for Initial Setup	Effort needed to initially setup the environment used for FMEA study.
Maintenance	Effort needed to maintain the consistency along the system definition and the FMEA entries during project evolution.

By comparing these three different approaches, it is possible to gain insights into the strengths and weaknesses of each method. A point-based qualitative assessment is employed to evaluate each method against the defined criteria explained in *Table 4.8*. Through this analysis, the strengths and limitations of each approach are carefully considered, conclusions are made on whether the model-based methodologies are the most suitable for meeting the specific needs of the LISA project and recommendations are given to further improve on this conclusion for each defined criterion. The summary of the comparison is given in *Table 6.1* in APPENDIX B, whereas the overall conclusions are given in the next section.

Table 4.9: Description of points of the qualitative assessment.

Point	Description
1	Poor: The FMEA approach has significant weaknesses and limitations that hinder its effectiveness in the given criterion. There are significant issues that make the approach difficult to use or that impact obtaining satisfactory results.
2	Below Average: The FMEA approach has some strengths, but also significant limitations that impact its ability to perform optimally in the given criterion. The approach has some gaps in coverage or require additional effort to achieve satisfactory results.
3	Average: The FMEA approach has a number of strengths and weaknesses that balance each other out. It is generally effective in the given criterion, but requires additional support or modifications to work optimally in certain situations.
4	Above Average: The FMEA approach has notable strengths and is generally effective in the given criterion. It requires some adjustments to work optimally in certain situations, but overall it is a strong approach.
5	Excellent: The FMEA approach has exceptional strengths and is highly effective in the given criterion. It is versatile and can be applied in a variety of situations with little or no modification.

4.1.2.1 Quality of Input

The document-based approach has some disadvantages compared to the static model-supported approach and the executable model-based approach. Both approaches receive the same information of the system description, functional list, architecture, and breakdown, however in different formats. The document-based approach relies on excerpts from documents, which is not interactable and spread through various pages and chapters. This increases the chance of inconsistency in the information, unless specifically checked for, and may not provide a complete picture of the system's characteristics. Additionally, the use of worded information may make it harder to understand and extract the input. On the other hand, the static model-supported approach, and the executable model-based approach both offer more consistent and complete information, which is important for ensuring the accuracy and completeness of the FMEA analysis. The model enables to conveniently navigate through relations in between elements to find, collect and present desired information. It also provides a visual representation of the system's characteristics, which is easier to understand than worded information. Finally, these approaches offer the ability to dynamically update the input, which is crucial for ensuring the accuracy and completeness of the FMEA analysis input when changes occur over time.

Both static model-supported and executable model-based approaches offer exceptional strengths in ensuring high-quality and consistent input for the FMEA study, while the document-based approach suffers from significant weaknesses and limitations, hence resulting in the assessment shown in *Figure 4.1*. For the LISA project, it is recommended to pursue the model-based/supported approaches in terms of this criterion.

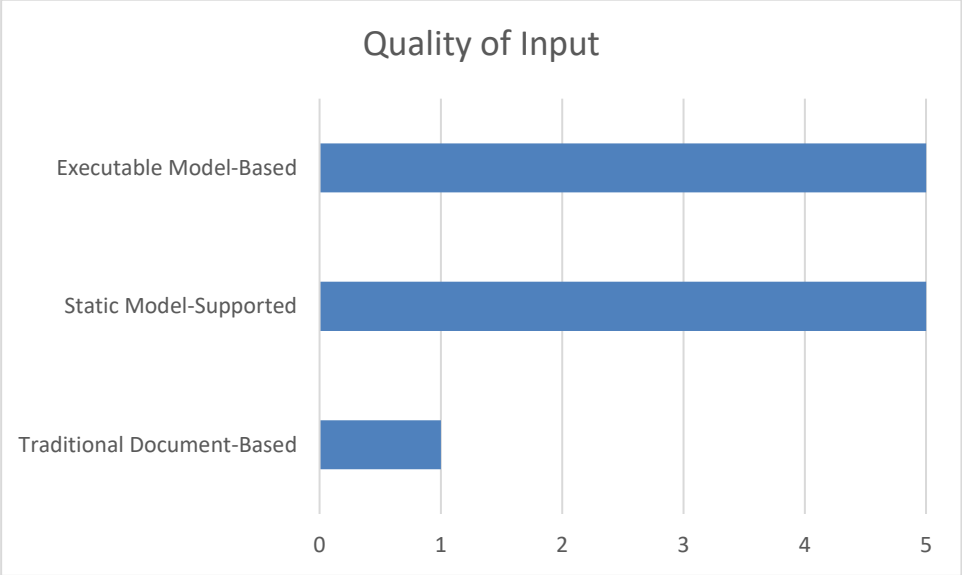

Figure 4.1: Point based qualitative assessment of approaches according to “Quality of Input” criterion.

4.1.2.2 Effort for Preparation

The document-based approach requires significant time and effort to compile information from multiple documents as described in *section 3.3.1.2*. The need to manually extract information from various sources for the preparation to the FMEA study makes this process time-consuming and more prone to errors, especially when the number of functions to be analysed is high. In contrast, both the static model-supported approach and the executable model-based approach offer significant advantages in terms of reducing the effort for this preparation. These approaches use the queries and relations in between the model elements to automatically compile the required information, which is then presented real-time in a table format, eliminating the need for manual operation. Once the initial setup is done in the model, the process becomes more efficient, streamlined, and less prone to errors, making the overall time and effort required for preparation significantly less.

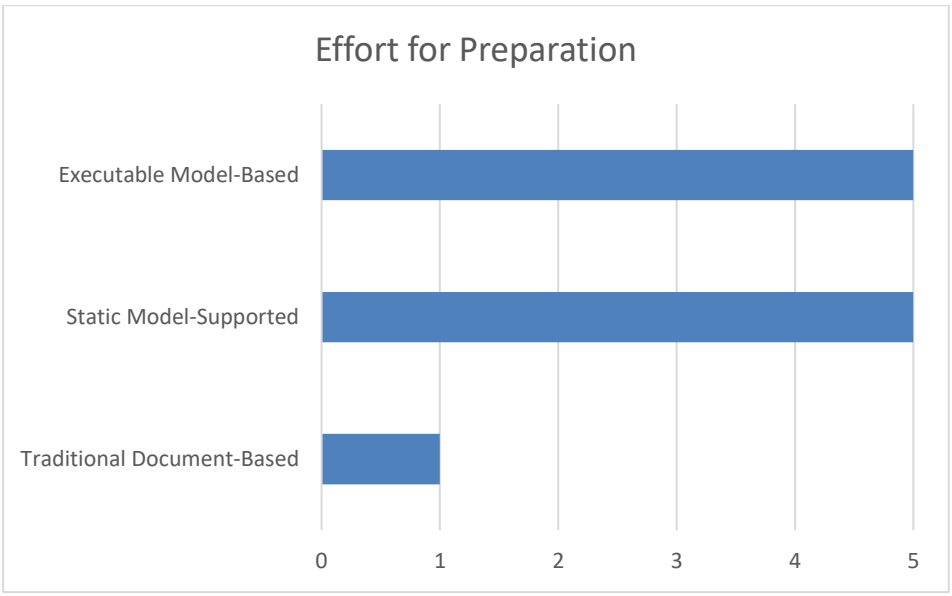

Figure 4.2: Point based qualitative assessment of approaches according to “Effort for Preparation” criterion.

In terms of the preparation effort, both static model-supported and executable model-based approaches demonstrate exceptional strengths by automating the information compilation effort, while the document-based approach presents significant weaknesses and limitations, hence resulting in the assessment shown in *Figure 4.2*. For the LISA project, it is recommended to pursue the model-based/supported approaches in terms of this criterion.

4.1.2.3 Information Access and Traceability

The document-based approach has some limitations in terms of this criterion. It requires explicit searching for the necessary information, which requires expertise, is time-consuming, and might result in missing some essential information. Moreover, there is no formal traceability between the contents of the documents, making it difficult to track the origin of the information. Navigation through the documents is either manual or through hyperlinks if implemented, which is inconvenient. On the other hand, both static model-supported and executable model-based approaches have significant advantages overcoming these problems. They use model diagrams and relations to represent the system's functions, requirements, systems, and failures, containing all necessary information together. Additionally, full traceability is established between the model elements (shown in *Figure 3.4)*, which makes it easier to track the origin of the information. Navigation is more convenient using the interactive diagrams, tables, and model elements.

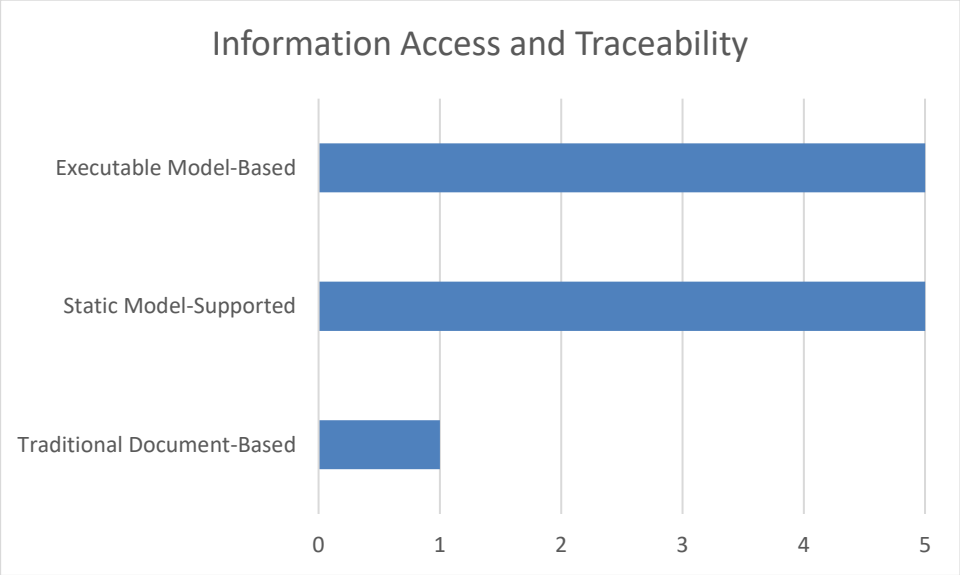

Figure 4.3: Point based qualitative assessment of approaches according to “Information Access and Traceability” criterion.

In terms of the "Information Access and Traceability" criterion, both static model-supported and executable model-based approaches are superior, with the exceptional strengths, to the document-based approach which has significant weaknesses and limitations, hence resulting in the assessment shown in *Figure 4.3*. For the LISA project, it is recommended to pursue the model-based/supported approaches in terms of this criterion.

4.1.2.4 Quality of Results

When considering the criterion of “Quality of Results”, both the document-based and static model-supported FMEA approaches have limitations, as highlighted in section 4.1.1. These approaches rely on static functional breakdown and architecture views, which leads to missing details in the analysis results or incorrect assumptions due to a lack of functional behavioural awareness. Additionally, operational aspects are not included by default, requiring manual tracing of failure propagation in operational context, if included in the study. Furthermore, the lack of an explicit cause of failure in the local failure effects entry hinders the ability to address underlying issues. In contrast, the executable model-based approach overcomes these limitations by utilizing functional behaviour, constraint, and system operation simulations, ensuring consistency with the system definition. It incorporates operational aspects, traces failure propagation along individual systems in the constellation, and explicitly identifies the cause of failure in the local failure effects entry. These features contribute to higher quality analysis results, as the executable model-based approach is more reliable and effective in identifying potential failures and addressing their underlying causes.

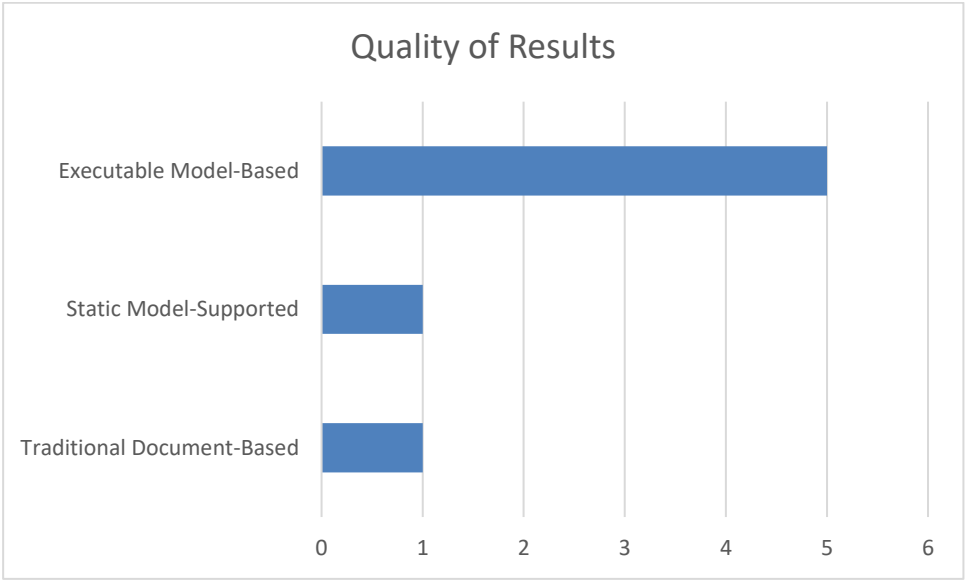

Figure 4.4: Point based qualitative assessment of approaches according to “Quality of Results” criterion.

In terms of the "Quality of Results" criterion, the executable model-based approach is superior with its exceptional strengths. Both the static model-supported approach and the document-based approach exhibit notable weaknesses and limitations, hence resulting in the assessment shown in *Figure 4.4*. For the LISA project, it is recommended to pursue the executable model-based approach in terms of this criterion, especially for payload systems, as its complex nature requires highly detailed and quality analysis results.

4.1.2.5 Effort to Obtain Results

The document-based approach in FMEA requires a manual deduction process for all functions in the architecture, which is highly work-intensive and time-consuming. It also relies on a significant amount of system knowledge and/or expert assistance, making it less accessible to individuals without specialized expertise on the system of interest. Additionally, review loops are necessary for result refinement, further increasing the overall effort required. In contrast, the static model-supported approach provides instant access to the requirement-mission-operation-function-system information

of the analysed failure, which saves time in the initial stages of analysis. However, it still requires a manual deduction process for all functions in the architecture and relies on significant system knowledge and/or expert assistance for the identification of the failures. Review loops are also needed for result refinement. On the other hand, the executable model-based approach offers a more efficient alternative. It involves a quick simulation run for each function, which provides a list of failure effects and propagation with minimal effort. This approach eliminates the required time and effort for the manual deduction and is independent of the expertise of the person conducting the study. It also eliminates the need for frequent review loops, as only one review session is necessary at the end of the study. Similar to the static model-supported approach, the executable model-based approach also offers instant access to the requirement-mission-operation-function-system information of the analysed failure.

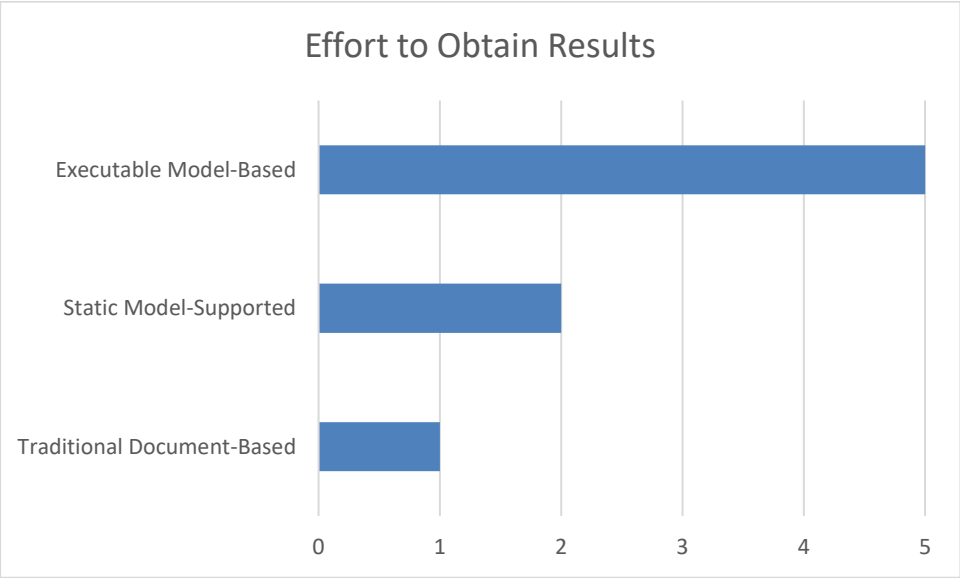

Figure 4.5: Point based qualitative assessment of approaches according to “Effort to Obtain Results” criterion.

When considering the "Effort to Obtain Results" criterion, the executable model-based approach stands out as superior, showcasing exceptional strengths. On the other hand, while the static model-supported approach does possess certain strengths, it shares significant weaknesses and limitations with the document-based approach, hence resulting in the assessment shown in *Figure 4.5*. For the LISA project, it is recommended to pursue the executable model-based approach in terms of this criterion, especially for payload systems, as its complex nature requires high effort and expertise to obtain results.

4.1.2.6 Filling the FMEA table

The document-based approach offers a straightforward and convenient method for filling the FMEA table in Excel, allowing for customized and flexible wording, easy editing, and the implementation of drop-down lists. However, it still lacks formal traceability of failures to the system definition, necessitating manual navigation from the FMEA table to the related documents. In contrast, the static model-supported and executable model-based approaches utilize the model and SRAP elements to populate the FMEA table. These approaches provide formal traceability of failures to the system definition and enable convenient navigation from the interactive FMEA table to the system definition. However, each unique FMEA table entry in the model-based approaches requires the generation of a new model element and its linkage to the corresponding function's FMEA item, which is highly

labour-intensive, especially when custom wording for the entries is desired. This process also increases the number of elements in the model, potentially impacting software performance. While drop-down lists are still possible in the model-based approaches, their implementation is less convenient, requiring profile customization. Additionally, further profile customizations are needed to assign model definition elements to table entries.

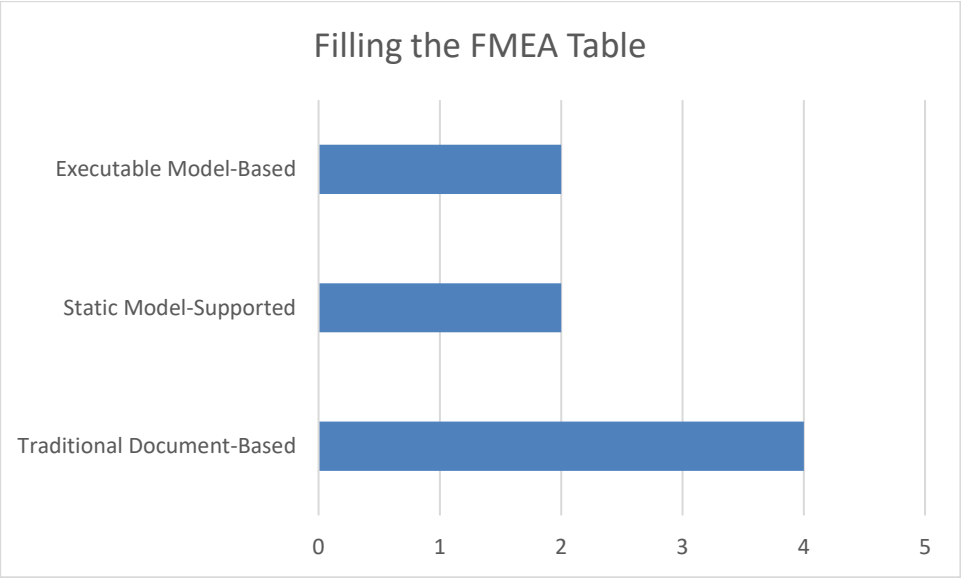

Figure 4.6: Point based qualitative assessment of approaches according to “Filling the FMEA table” criterion.

When evaluating the "Filling the FMEA table" criterion, it becomes evident that both the executable model-based approach and the static model-supported approach, despite their respective advantages, suffer from significant limitations that hinder their convenience and usability. In comparison, the document-based approach outperforms the other approaches due to its inherent simplicity and flexibility, hence resulting in the assessment shown in *Figure 4.6*. To pursue the model-based/supported approaches in terms of this criterion, it is crucial and strongly recommended to create a new customized profile for Airbus Space Systems-RAMS applications in Cameo System Modeler instead of using the Cameo SRAP profile. Until it is implemented, for the LISA project, it is recommended to use Excel to fill in the FMEA table and importing the filled in table into the model. This would result in a higher convenience for the RAMS engineering for filling in the FMEA table entries, while the later model import effort would still allow exploiting the advantage of model traceability and navigation.

4.1.2.7 Effort for Initial Setup

The document-based approach offers a straightforward setup process by implementing the template provided in the ECSS-Q-ST-30-02C standard in Excel, requiring low effort. Similarly, the static model-supported approach benefits from a low setup effort as the template can be conveniently implemented in the Cameo tool, on top of a pre-existing model following the default R-MOFLT process. In contrast, the executable model-based approach necessitates additional time and effort for the modelling process, involving the creation of functional failure modes, constraints, behaviours, and simulation setup within the model. This undertaking calls for specialized expertise and training in modelling. As a result, the document-based and static model-supported approaches exhibit lower initial setup effort, a significant strength, while the executable model-based approach requires a significant amount of

effort due to modelling and simulation setup requirements, one of the key shortcomings of this particular method, hence resulting in the assessment shown in *Figure 4.7*.

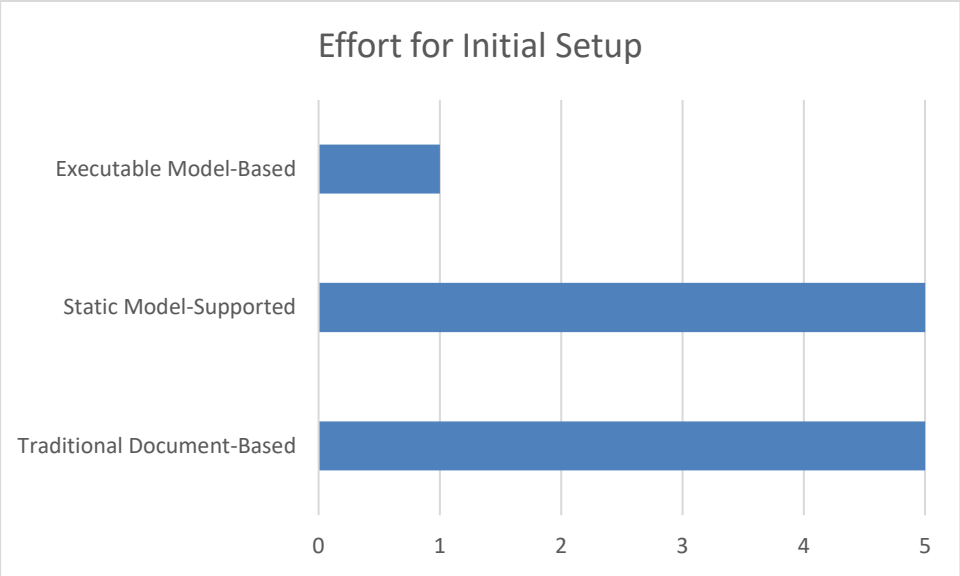

Figure 4.7: Point based qualitative assessment of approaches according to “Effort for Initial Setup” criterion.

To pursue the executable model-based approach based on this criterion, it is essential and highly recommended to leverage the additional modelling effort for other applications. In the context of the LISA project, these additional modelled artifacts serve purposes such as comprehending and validating functional behaviours, operational tasks, and phase behaviours. This effectively justifies the investment of effort by providing a robust rationale beyond the scope of FMEA alone.

4.1.2.8 Maintenance

In terms of “Maintenance” criterion, the document-based approach requires manual maintenance for the entire document chain, which means that any modifications made to a document is not automatically reflected in the FMEA table. As a result, inconsistencies are very likely to arise throughout the project evolution, requiring explicit checks to identify them. On the other hand, for the static model-supported approach, as well as the executable model-based approach, maintenance can be done continuously. The model has validation rules embedded, providing warnings during function-requirement changes, and automatically updating system allocation or functional hierarchy with no need for manual maintenance. However, when updating the Failure Effects and Failure Causes in the FMEA entries, both the document-based approach and the static model-supported approach require manual re-checking and deduction from scratch, as described in *section 4.1.1.2*, which is highly labour-intensive. In contrast, the executable model-based approach only requires re-running quick simulations to obtain the new list of failure propagation information, which is then used to update the corresponding entries. This highlights the substantial difference in flexibility and robustness of the executable model-based approach compared to the others, hence resulting in the assessment shown in *Figure 4.8*.

It is highly recommended to pursue the executable model-based approach in terms of this criterion. This approach is the most effective especially in early phases, like in LISA project, when the project evolution is highly dynamic and continuous updates in the system definition are present.

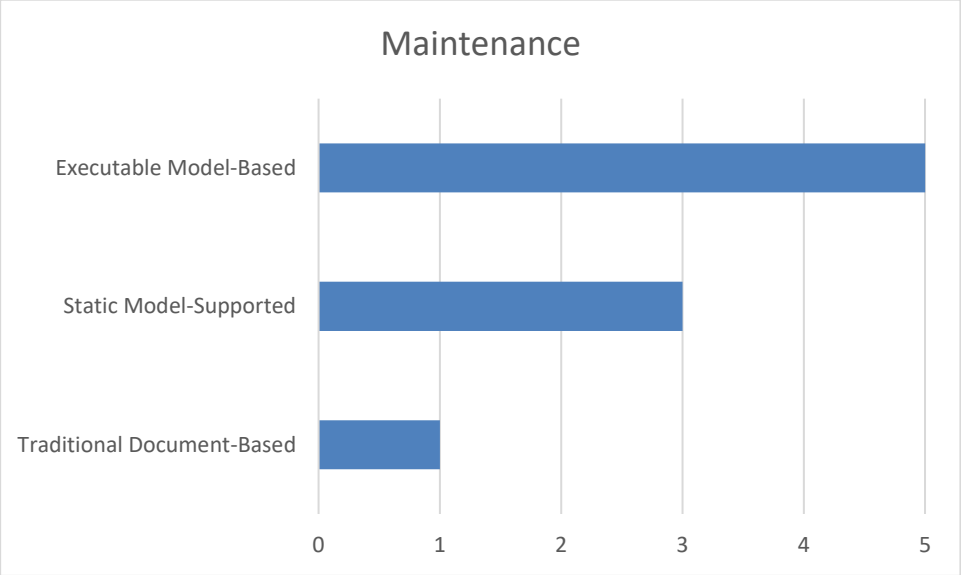

Figure 4.8: Point based qualitative assessment of approaches according to “Effort for Initial Setup” criterion.

4.1.3 Comparison Conclusion

Looking at the detailed comparison of the results in *section 4.1.1* and the assessment of methodologies in *section 4.1.2*, it is concluded that the offered executable model-based approach is the most suitable one to pursue for a complex space science mission like LISA. It shows significant advantages compared to the traditional document-based and static model-supported approaches in 6 out of the 8 defined criteria as shown in *Figure 4.9.*

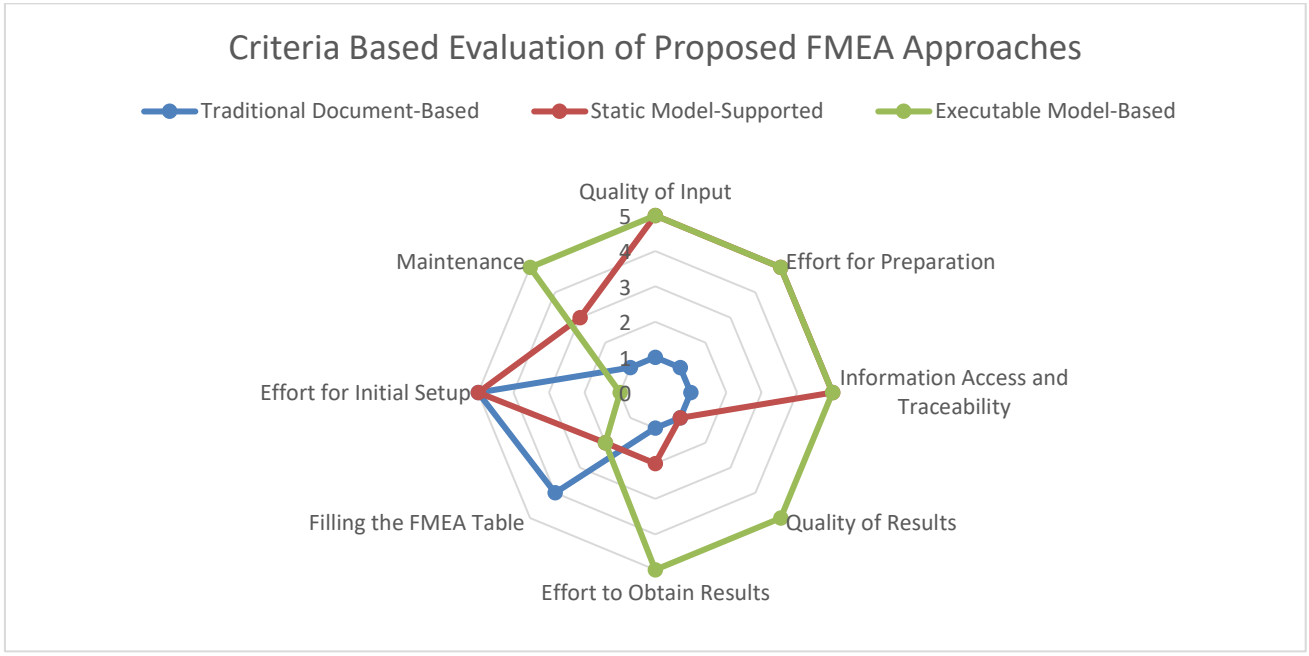

Figure 4.9: Point-based qualitative assessment of traditional document based, static model-supported and executable model-based approaches according to defined criteria.

In order to manage complexity, the executable model-based functional FMEA approach offers the very much needed details and information to help identifying the un-intuitive failure cases. The fact that accessing this information with a relatively quick simulation is very much valuable and provide an important situational awareness to the development team, since the whole failures-functions-systems-requirements-operations linking is visible and conveniently accessible. The time and effort savings in terms of conducting FMEA, updating FMEA during changes, and avoiding review loops to correct wrong/missing entries is significant, which justifies the implementation effort in the long run. It is still utmost essential to improve upon its limitations, therefore it is suggested to invest in specialized expertise and training for modelling, create a customized profile for RAMS applications, and leverage additional modelled artifacts also for other purposes than FMEA. These enhancements would further enhance the effectiveness and efficiency of the executable model-based approach in the LISA project, particularly for payload systems.

For missions or systems with low functional complexity, the investment in additional training and modelling for an executable functional architecture is not justified. Existing systems can be understood with minimal simulations to grasp their overall behaviour. In such projects, from a perspective of quick benefit and effort, it is highly logical to transition to a static model-supported FMEA approach as soon as possible. This approach is readily available with minimal effort if the development team is already engaged in MBSE or R-MOFLT practices. If a model does not yet exist, it is still recommended to undertake the effort in order to benefit from the many advantages of MBSE implementation. However, the modelling scope should be limited to employing functional hierarchy and architecture without delving deeply into the functional behaviours and constraints necessary for an executable model, in order to maintain a reasonable level of effort.

4.2 THESIS SUMMARY AND CONTRIBUTIONS

This thesis contributes to the field of space systems engineering by offering a novel MBSE-assisted FMEA methodology and highlighting the benefits of this model-based approach in improving traditional FMEA results and process. In detail the following points have been performed and achieved:

1- Define and implement a novel executable functional architecture modelling approach as an extension to Airbus R-MOFLT methodology for operational-functional-technical simulations that traverse through the functional architecture definition. Refer to *chapters 3.1 and 3.2*

2- Implement and test the static model-assisted FMEA method in an existing system architecture model and utilize it to support a functional FMEA study. Refer to *section 3.3.2*

3- Define, implement, and test the executable model-based FMEA method, which utilizes a failure propagation analysis for critical operational scenarios using the implemented executable functional architecture model. Refer to *section 3.3.3*

4- Perform a thorough investigation and comparison between traditional (document based) FMEA and model based FMEA approaches in terms of the listed criteria below. Analyse and find out why executable model-based FMEA is the most suitable method for a complex space science mission like LISA. Refer to *chapter 4.1*

   - Quality of Input
   - Effort for Preparation
   - Information Access and Traceability
   - Quality of Results
   - Effort to Obtain Results

- Filling the FMEA table
- Effort for Initial Setup
- Maintenance

At the culmination of this thesis, the research objectives (refer to *chapter 1.4*), and identified gaps in the literature (refer to *chapter 2.4*), namely the inclusion of operations in FMEA, the consideration of the effects of exchanged interface flows on system functions, the integration of functional behaviours and architecture in FMEA, and the provision of detailed modelling and analysis descriptions, have been successfully addressed and achieved.

The approaches and results in this thesis can be used to develop an MBSE adoption strategy for complex space missions, serve as a guide for system modelling in MBSE environment, and integrate RAMS engineering discipline into early phase model-based system development.

5 FUTURE WORK

This chapter presents potential areas for future work related to the thesis topic presented in *Table 5.1*. It includes a compilation of recommendations for the further development of this thesis, as well as for improving the modelling process and the integration of RAMS into MBSE methods and procedures.

Table 5.1: Recommendations for future work.

Topic	Description
Airbus Space Systems-RAMS profile	A new customized profile for Airbus Space Systems-RAMS applications in Cameo System Modeler is needed. Cameo Safety and Reliability Analyser Plugin is limited in terms of addressing the RAMS analysis needs and ease in usability. The new profile shall out-of-the-box, • contain standards used in space systems (e.g., ECSS-Q-ST-30-02C), • does not require model elements (e.g., Cause of Failure, Final Effect of Failure, etc.) to be created per each FMEA entry, possibly by allowing linking failure properties to already existing model elements used for system definition, • allow creation of a failure database and list suggestions and possible entries in the FMEA table for the user, • contain easily customisable FMEA tables (e.g., allow custom naming for table entries that are traced to model elements used for system definition) • be integrated to R-MOFLT framework and make use of its dedicated artefacts (e.g., automatically fill in mission phase, operational mode entries in FMEA table using function-operational-mission traceability links).
Additional operational scenarios	Modelling of more operational scenarios enables to cover more/all LISA mission phases (this thesis work focused only on laser link acquisition).
Automated FMEA table generation inside Cameo from simulation results	Although automated tables are generated from simulations in the form of Failure Cause and Failure Effects tables, it needs manual work to put these results in the FMEA table format. It can be achieved using scripts in the model.
Monte Carlo Analysis	Implementation of scripts to inject random failures for specific/all operational scenarios to enable a Monte Carlo Analysis. This would enable identification of, • Sensitivity of system functions to failures for the entire space segment, • System failure rates/probability (i.e., for entire LISA constellation)
Severity and Criticality evaluation	It is possible to automatically evaluate the Severity and Criticality of the system functions using the model. Simulations can be used to identify end effects of the failures that is traced to mission level showing their Severity. Combined with the

	Probability data either found via simulations or using heritage information, Criticality metric can be calculated.
Architectural clustering analysis	Architectural clustering analysis can be done automatically in the model to identify: • Functions highly dependent on each other • Components highly dependent on each other • Logical dependency/clustering • Failures highly dependent on each other This data can be used to prioritize functions, components and logical systems or develop mitigating actions to break potential failure chains.
Open-loop functional commanding scheme	The operational-functional analysis in this thesis work followed a closed-loop commanding scheme (see *section 3.2.2*). It could be the case that the real system also follows a pre-defined timeline as an open loop commanding. This would open up new failure cases depending on the timing of the commands and functional response. Therefore, it is recommended to implement an option to perform a mission timeline based operational-functional analysis and simulation in the model.
Component state– Function integration	Integration of payload component states and their respective functions in the model execution is a key future activity to: • manage the complexity of the payload operations, • simulate and understand payload component behaviour also in the case of functional failures, • identify the most critical payload component states. This activity would enable directly referring to payload component states in the operational tasks instead of functions to be activated
Using AI language models to help with modelling	It is highly recommended to experiment with using AI language models to convert verbal behavioural descriptions into SysML diagrams and scripts for constraint properties to help with users and modelers. • Functional behaviours can be hard to put into a script format, that requires high effort in the case of complex behaviours. AI language models show promising potential to transfer verbal descriptions, which are more intuitive to human mind, into if-else conditions that the model can use during execution. • It is also worth trying whether a diagram (e.g., SysML Activity Diagram, etc.) can be generated from a verbal description of a function behaviour and vice versa using AI tools.
Data for FMEA effort	It is important to collect detailed data that reflects the time and effort of the modelling and FMEA activities from LISA and other projects throughout Airbus Space Systems for a more thorough comparison. This is important to identify: • The kind of projects that should employ an executable model-based, static model-supported or any other approach. • Aspects of the approaches that should be improved.

6 APPENDIX

6.1 APPENDIX A

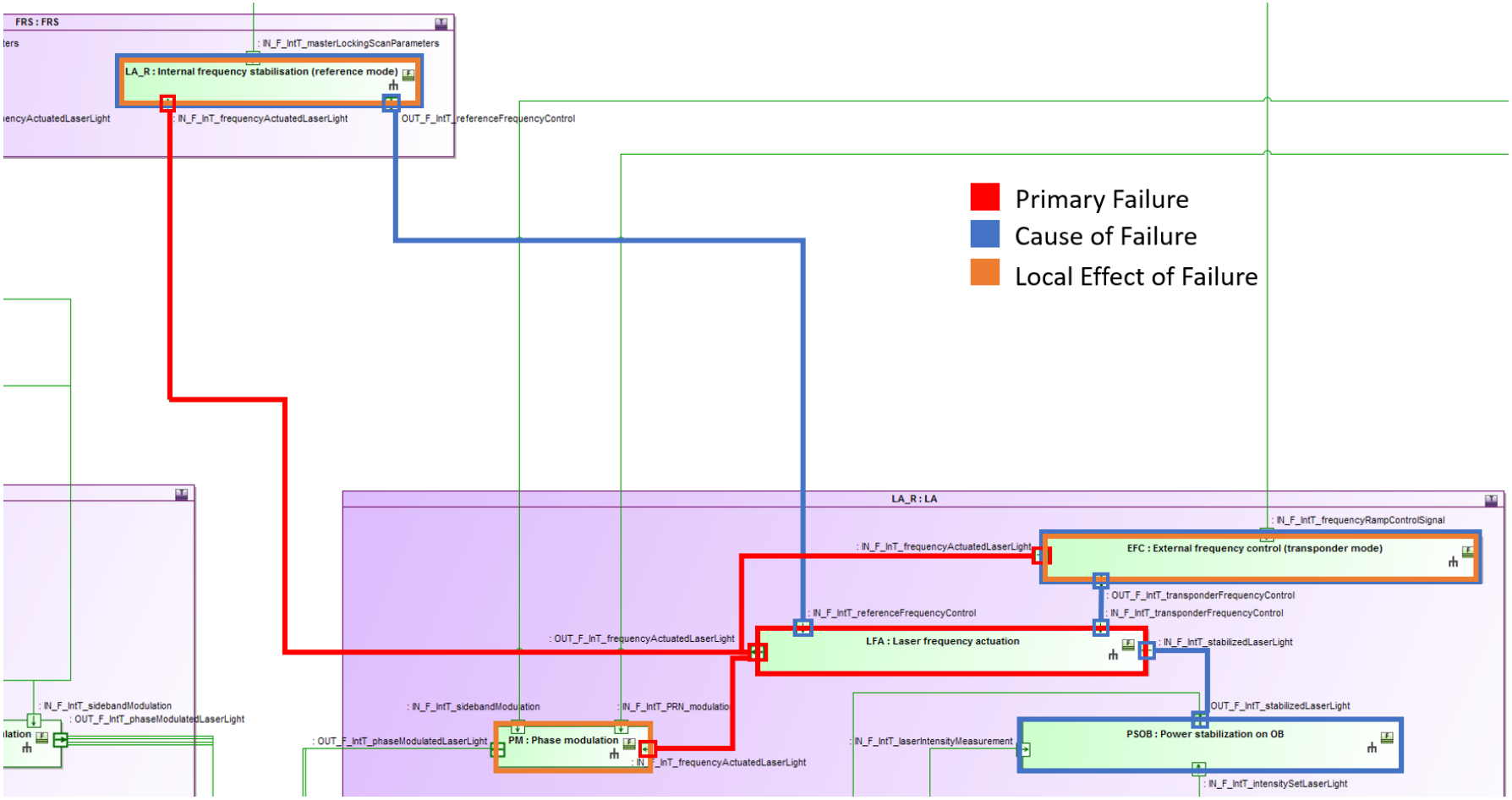

Figure 6.1: Deduction of failure causes and effects from the functional architecture using the static approaches. The proposed method assumes that the upstream functions are possible causes of the primary failure, whereas downstream functions are affected as a result of the primary failure.

6.2 APPENDIX B

Table 6.1: Comparison summary table comparing different methodologies of traditional document based, static model-supported and executable model based FMEA approaches with pros [+] and cons [-] indicated.

	Document Based	Static model-supported	Executable Model Based
Quality of Input	**Method:** Excerpts from documents [-] Higher chance of inconsistency in information. [-] Often not enough to explain overall system characteristics. [-] Worded information, relatively hard to understand.	**Method:** Model, dynamic tables [+] Consistent [+] Dynamically updated. [+] Overall system characteristics represented with requirement-mission-operation-function-system viewpoints all together. [+] Visual, easier to understand than worded information.	**Method:** Model, dynamic tables [+] Consistent [+] Dynamically updated. [+] Overall system characteristics represented with requirement-mission-operation-function-system viewpoints all together. [+] Visual, easier to understand than worded information.
Effort for Preparation	**Method:** Information compilation from different documents. [-] Extraction of information from multiple documents needed. [-] Need time, effort, and expertise to find desired information.	**Method:** Automatic compilation using queries. [+] Required information automatically shown in table format. [+] Queries do not need manual operation once setup.	**Method:** Automatic compilation using queries. [+] Required information automatically shown in table format. [+] Queries do not need manual operation once setup.
Information Access and Traceability	**Method:** Access through different documents. [-] Information needs to be explicitly searched for. [-] No formal traceability in between contents of the documents. [-] Navigation is manual or via hyperlinks (if implemented).	**Method:** Usage of model diagrams and relations. [+] Model contains necessary information altogether. [+] Full traceability established in between functions-requirements-systems-failures. [+] Convenient navigation using the interactive diagrams,	**Method:** Usage of model diagrams and relations. [+] Model contains necessary information altogether. [+] Full traceability established in between functions-requirements-systems-failures. [+] Convenient navigation using the interactive diagrams,

		tables, and model elements.	tables, and model elements
Quality of Results	**Method:** Deduction through static functional breakdown, architecture views. [-] Might include wrong assumptions due to lack of functional behavioural awareness. [-] Does not include operational aspects. [-] Failure propagation along constellation has to be traced manually, if included in the analysis. [-] Does not include explicit cause of failure in local failure effects entry.	**Method:** Deduction through static functional breakdown, architecture views. [-] Might include wrong assumptions due to lack of functional behavioural awareness. [-] Does not include operational aspects. [-] Failure propagation along constellation has to be traced manually, if included in the analysis. [-] Does not include explicit cause of failure in local failure effects entry.	**Method:** Usage of functional behaviour, constraint, and system operation simulations. [+] Consistent with system definition due to direct simulation of SE artefacts instead of deductions from contained information in the SE artefacts. [+] Include operational aspects and usage of functions. [+] Traces failure propagation along constellation. [+] Explicit cause of failure is identified in local failure effects entry.
Effort to Obtain Results	**Method:** Manual deduction, Review loops with Systems Engineering [-] Work-intensive deduction process for all functions in the architecture. [-] Requires significant amount of system knowledge and/or expert assistance. [-] Requires review loops for result refinement.	**Method:** Manual deduction, Review loops with Systems Engineering [+] Instant access to the requirement-mission-operation-function-system information of the identified failure [-] Work-intensive deduction process for all functions in the architecture. [-] Requires significant amount of system knowledge and/or expert assistance. [-] Requires review loops for result refinement.	**Method:** Executable model simulation. [+] Quick simulation run per each function to receive list of failure propagation. [+] Review session at the end of study instead of frequent review loops. [+] Instant access to the requirement-mission-operation-function-system information of the identified failure
Filling the FMEA table	**Method:** Excel worksheets [+] Customized and flexible wording possible.	**Method:** Model, Excel and/or SRAP elements [+] Formal traceability of failures to system definition.	**Method:** Model, Excel and/or SRAP elements [+] Formal traceability of failures to system definition.

	[+] Convenient and straightforward editing. [+] Easy to implement drop-down lists. [-] No formal traceability of failures to system definition. [-] Manual navigation from FMEA table to documents	[+] Convenient navigation to system definition directly from interactive FMEA table. [-] Every unique table entry needs a new model element. [-] Editing needs to be done on the model elements instead of directly on the table. [-] Drop-down lists require profile customization. [-] Need profile customization to allow assignment of model definition elements to table entries.	[+] Convenient navigation to system definition directly from interactive FMEA table. [-] Every unique table entry needs a new model element. [-] Editing needs to be done on the model elements instead of directly on the table. [-] Drop-down lists require profile customization. [-] Need profile customization to allow assignment of model definition elements to table entries.
Effort for Initial Setup	**Initial Setup:** Excel ECSS-Q-ST-30-02C FMEA table template. [+] Minimal effort to setup template	**Initial Setup:** Cameo ECSS-Q-ST-30-02C FMEA table template. [+] Minimal effort to setup template	**Initial Setup:** Model functional failure modes, constraints and behaviours. Model functional chain in operational tasks. Establish simulation setup in the model. [-] Extra time for the modelling process [-] Requires training and expertise for modelling.
Maintenance	**Method:** Regular manual document update, manual deduction, Review loops with Systems Engineering. [-] No automation in terms of updating FMEA entries. [-] No indication of inconsistencies. [-] Manual and work intensive process repetition in case of functional architecture changes for Failure Cause and Effects entries.	**Method:** Validation rules, dynamic tables, manual deduction, Review loops with Systems Engineering. [+] Dynamic changes in definition are reflected on FMEA table. [+] Validation rules give warning during inconsistencies. [-] Manual and work intensive process repetition in case of functional architecture	**Method:** Validation rules, dynamic tables, executable model simulation. [+] Dynamic changes in definition are reflected on FMEA table. [+] Validation rules give warning during inconsistencies. [+] Quick simulation run per function is needed to obtain the new failure propagation information.

		changes for Failure Cause and Effects entries.	

ACKNOWLEDGEMENT

I am truly grateful for the support and encouragement of all those who contributed to the realization of this work.

First and foremost, I would like to thank my supervisor Tobias Ziegler, for putting his trust in this work and me at the first place. Witnessing his leadership in the project, having a fair share of the challenges that came with it, and opportunities he had supplied to me is an invaluable experience and a big inspiration to an enthusiastic young engineer like me. It takes a lot to motivate people to employ new ways of working, and voluntarily bring disruptive ideas into a project as big and complex as LISA. I am sure his vision will pave the way for a way more efficient systems engineering process for the new projects to come and inspire the others to push the boundaries in their work.

I would like to express my sincere gratitude to Arno Dietrich, for supporting this initiative and making this thesis possible. His sincere mentorship and guidance, combined with all the opportunities he had offered, allowed me to interact thoroughly with the Airbus community and develop my skills. I really appreciate the time and interest he had allocated to the students in his department and treating us as an integral part of his team. He had shown and embraced the effort of learning from our experiences as students, as much as we had learned from the team.

On TU Munich side, I would like to thank my supervisor Alessandro Golkar for accepting me as his thesis student and providing guidance throughout the process. Many thanks to Jaspar Sindermann for his invaluable guidance, ideas, and feedback throughout the process of maturing this thesis. Their positive and enthusiastic attitude, as well as investment of time and interest to this work, have been greatly appreciated.

Huge thanks for all the members of the LISA project team, especially to Christian Greve for all his time and support on the realization of this work, as well as the lectures on the LISA mission and the payload systems, opening my mind about the scientific side of this project. I want to extend my deep thanks to Johannes Buerkle, Stephane Estable and Simon Eitelbuss for all their help and feedback on this thesis work.

Many thanks to Peter Mathias Koch, whose mentorship, guidance, and feedback were instrumental during my time in Airbus. I am equally grateful to Andre Lessow, who introduced me to the Airbus family with Peter, and provided me with the opportunity to work on this exciting topic at the first place.

I would like to convey my heartfelt appreciation to Jacopo Aurigi, a remarkable colleague, mentor, and friend who has been a constant source of inspiration and has taught me a great deal, both professionally and personally, during my time at Airbus. Collaborating with him to brainstorm innovative ideas to overcome the many challenges we encountered during this project was an invaluable experience. His positive attitude, intelligence, and willingness to teach and share his knowledge have been immensely valuable to me.

I would also like to acknowledge my dear friend group "Mahlzeit", who made my time at Airbus unforgettable. I am immensely grateful to my dear officemate, Mario Izquierdo Serra, for his constant support and companionship throughout this journey, and Niccolò Bardazzi and Alexandre Gol Mestre, whose blindingly bright minds and unique personalities made every moment worth cherishing. Lastly, I would like to extend a special thanks to all the students and interns of Airbus Defence and Space Friedrichshafen for the wonderful memories.

Finally, I would like to express my deepest appreciation to my dear family for their unwavering support and belief in me throughout my academic and professional journey. Their love and encouragement have been my source of strength and inspiration, and I am forever grateful for their presence in my life.

REFERENCES

[1] J. Würtenberger and H. Kloberdanz, "An approach to identifying the ideal time to perform an FMEA during the product development process," presented at the 1st International Symposium on Robust Design, København, DK, 2014.

[2] J. M. Stecklein, J. Dabney, B. Dick, B. Haskins, R. Lovell, and G. Moroney, "Error cost escalation through the project life cycle," in *14th Annual International Symposium*, 2004, no. JSC-CN-8435.

[3] Object Management Group. "What is sysML?" https://www.omgsysml.org/what-is-sysml.htm (accessed February, 2023).

[4] Dassault Systèmes. "Cameo Systems Modeler." https://www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/ (accessed February, 2023).

[5] N. Charpigny, "An Executable System Model for Behavioural Analyses of the LISA Mission," Vehicle Engineering, School of Engineering Sciences, KTH Royal Institute of Technology, 2019.

[6] J. Aurigi, "Behavioral Modelling and Simulations for Constellation Acquisition of the LISA Mission," 2020.

[7] N. Charpigny, G. Hechenblaikner, T. Ziegler, and G. Pisacane, "An Executable System Model for Behavioural Analyses of the LISA Mission," presented at the Tag des Systems Engineering 2019, München, 6. - 8. November 2019, 2019.

[8] P. Amaro-Seoane *et al.*, "Laser Interferometer Space Antenna," *LISA Consortium,* 2017. [Online]. Available: https://www.elisascience.org/files/publications/LISA_L3_20170120.pdf.

[9] B. P. Abbott *et al.*, "Observation of Gravitational Waves from a Binary Black Hole Merger," *Physical Review Letters,* vol. 116, no. 6, 2016, doi: 10.1103/physrevlett.116.061102.

[10] ESA. "LISA mission moves to final design phase." ESA. https://www.esa.int/Science_Exploration/Space_Science/LISA_mission_moves_to_final_design_phase (accessed February, 2023).

[11] S. Barke, *Inter-spacecraft frequency distribution for future gravitational wave observatories*. Hannover: Gottfried Wilhelm Leibniz Universität Hannover, 2015.

[12] G. Mueller, P. McNamara, I. Thorpe, and J. Camp, "Laser frequency stabilization for LISA," 2005.

[13] S. Friedenthal, R. Griego, and M. Sampson, "INCOSE model based systems engineering (MBSE) initiative," in *INCOSE 2007 symposium*, 2007, vol. 11: sn.

[14] L. Wang, M. Izygon, S. Okon, H. Wagner, and L. Garner, "Effort to accelerate MBSE adoption and usage at JSC," in *AIAA SPACE 2016*, 2016, p. 5542.

[15] IBM. "IBM Engineering Requirements Management DOORS Family." https://www.ibm.com/products/requirements-management (accessed February, 2023).

[16] *ECSS-Q-ST-30-02 C Space Product Assurance–Failure Modes, Effects (and Criticality) Analysis (FMEA/FMECA)*, ECSS, 2009.

[17] S. Burge, "The Systems Engineering Tool Box - Functional Failure Modes and Effects Analysis (FFMEA)." [Online]. Available: https://www.burgehugheswalsh.co.uk/Uploaded/1/Documents/FFMEA-Tool-v2.pdf

[18] G. Biggs, K. Post, A. Armonas, N. Yakymets, T. Juknevicius, and A. Berres, "OMG standard for integrating safety and reliability analysis into MBSE: Concepts and applications," in *INCOSE International Symposium*, 2019, vol. 29, no. 1: Wiley Online Library, pp. 159-173.

[19] No Magic. "Cameo Safety and Reliability Analyzer." https://docs.nomagic.com/display/CSRA190SP4/Cameo+Safety+and+Reliability+Analyzer (accessed February, 2023).

[20] B. Francesco, "A Model-Based RAMS Estimation Methodology for Innovative Aircraft on-board Systems developed in a MDO Environment," Politecnico di Torino, 2020.

[21] G. Girard *et al.*, "Model based safety analysis using SysML with automatic generation of FTA and FMEA artifacts," in *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference (Esrel 2020 PSAM 15), 1-5 November 2020, Venice, Italy*, 2020, no. CONFERENCE: 1-5 November 2020.

[22] F. Schummer and M. Hyba, "An Approach for System Analysis with MBSE and Graph Data Engineering," *arXiv preprint arXiv:2201.06363,* 2022.

[23] Neo4j Inc. "Neo4j Graph Database." https://neo4j.com/product/neo4j-graph-database/ (accessed February, 2023).

[24] F. Mhenni, N. Nguyen, and J.-Y. Choley, "SafeSysE: A safety analysis integration in systems engineering approach," *IEEE Systems Journal,* vol. 12, no. 1, pp. 161-172, 2016.

[25] M. Chami and J.-M. Bruel, "A survey on MBSE adoption challenges," 2018.

[26] J.-B. Bernaudin, "MBSE on MSR ERO: a use case," presented at the Model Based Space Systems and Software Engineering ~ MBSE2021, 2021. [Online]. Available: https://indico.esa.int/event/386/contributions/6227/attachments/4269/6378/1145%20-%20mbse%20on%20msr%20ero%20a%20use%20case.pdf.

[27] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[28] No Magic. "Modeling functional flows with Activities." https://docs.nomagic.com/display/SYSMLP190/Modeling+functional+flows+with+Activities (accessed February, 2023).

[29] R. Krishnan and S. V. Bhada, "An integrated system design and safety framework for model-based safety analysis," *IEEE Access,* vol. 8, pp. 146483-146497, 2020.

[30] MBSE Execution. "Information Exchange Between Blocks Simulation." https://www.youtube.com/watch?v=Cwz_tuKX_xs&ab_channel=MBSEExecution (accessed February, 2023).

[31] MBSE Execution. "System Interface Simulation in SysML | 4 Methods How to Pass Data Through Proxy Port." https://www.youtube.com/watch?v=di7oJYtp1T8&ab_channel=MBSEExecution (accessed February, 2023).

[32] MBSE Execution. "Pass Values Through Ports." https://www.youtube.com/watch?v=1PDKgfX-uhM&ab_channel=MBSEExecution (accessed February, 2023).

[33] Apache Groovy Project. "Groovy Programming Language." https://groovy-lang.org/ (accessed February, 2023).

[34] No Magic. "ALH APIs." https://docs.nomagic.com/display/CST190SP3/ALH+APIs (accessed February, 2023).

[35] No Magic. "Using Metachain Navigation." https://docs.nomagic.com/display/MD190SP4/Using+Metachain+Navigation (accessed February, 2023).