

CRANFIELD UNIVERSITY

SOUFIANE EL FASSI

ASSUMPTION MANAGEMENT IN MODEL-BASED
SYSTEMS ENGINEERING: AN AIRCRAFT DESIGN
PERSPECTIVE

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING

PhD

Academic Year: 2021–2022

Supervisors: Dr Atif Riaz, Prof. Marin D. Guenov

4th December 2021

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING

PhD

Academic Year: 2021–2022

SOUFIANE EL FASSI

Assumption Management in Model-Based Systems
Engineering: An Aircraft Design Perspective

Supervisors: Dr Atif Riaz, Prof. Marin D. Guenov
4th December 2021

Submitted in partial fulfilment of the requirement for the
degree of Doctor of Philosophy.

© Cranfield University, 2021. All rights reserved. No part of
this publication may be reproduced without the written
permission of the copyright holder.

Abstract

Early design of complex systems is characterised by significant uncertainty due to lack of knowledge, which can impede the design process. In order to proceed with the latter, assumptions are typically introduced to fill knowledge gaps. However, the uncertainty inherent in the assumptions constitutes a risk to be mitigated. In fact, assumptions can negatively impact the system if they turn out to be invalid, such as causing system failure, violation of requirements, or budget and schedule overruns.

Within this context, the aim of this research was to develop a computational approach to support assumption management in model-based systems engineering, with an explicit consideration of the uncertainty in assumptions. To achieve the research aim, the objectives were to: (1) devise methods to enable assumption management in a model-based design environment; and (2) devise methods to manage risk of change due to invalid assumptions, with an explicit consideration of both assumptions and margins. The scope was limited to the early stages of aircraft design.

To evaluate this research, a demonstration was performed based on two use cases to assess whether the methods work as intended. The developed methods were demonstrated to industry experts in order to obtain feedback on expected usefulness in practice, thus assessing the impact of this research. The experts concluded that the proposed methods are innovative, useful and relevant to industry, where these methods can lead to: (i) fewer undesired iterations, due to earlier identification and management of risks associated with assumptions; and (ii) a better margin balance, due to timely and interactive margin revision. Future work includes further industrial evaluation, extending the research scope and studying the scalability and associated costs of the proposed methods.

Keywords

Assumption; Belief Revision; Epistemic Uncertainty; Knowledge Maturity; Margin; Model-Based Systems Engineering (MBSE); Technical Risk Management.

Acknowledgements

This has been such a challenging, yet rewarding journey to pursue a PhD. Nearing the end of this journey, I need to express my sincere gratitude to all the people who have helped me.

A mes très chers parents Amal et Rachid, et ma très chère soeur Yasmina, merci infiniment pour votre amour et soutien qui m'ont donné la force de poursuivre ce doctorat. Je suis tellement chanceux que vous fassiez partie de ma vie. C'est grâce à vous et vos sacrifices que je suis devenu la personne que je suis aujourd'hui. Vous avez toujours été dans mes pensées malgré la distance qui nous sépare. Je vous aime très fort!

I would like to thank my supervisors Professor Marin D. Guenov and Dr Atif Riaz for their guidance and feedback throughout this research, and for the opportunity to be part of the Advanced Engineering Design Group. I would also like to extend my gratitude to Dr Arturo Molina-Cristóbal for his supervisory role during the first half of my PhD. To the past and present colleagues from the Advanced Engineering Design Group, thank you for your friendship and support which have made this journey more enjoyable. Thank you Xin Chen, Stevan van Heerden, Yogesh Bile, Sergio Jimeno Altelarra, Takashi Shibui, Gabriele Mura and Mario Carta.

I could not have completed this work without the help of so many other people. I would like to thank Teresa Townsend who has been very supportive during my most difficult times. During my time at Cranfield, I had the privilege to meet some wonderful people. Thank you Alban, Amel, Jesús, Lina, Mounia, Neda, Netasha, and Sophia for your friendship.

I would also like to thank the experts from Airbus for participating in my research evaluation.

Contents

Abstract	iii
Acknowledgements	v
Contents	vi
List of Figures	ix
List of Tables	xii
List of Algorithms	xiii
List of Abbreviations	xiv
1 Introduction	1
1.1 Context and Motivation	1
1.2 Aim and Objectives	5
1.3 Scope	6
1.4 Research Methodology	6
1.4.1 Research Clarification	7
1.4.2 Descriptive Study I	8
1.4.3 Prescriptive Study	8
1.4.4 Descriptive Study II	8
1.5 Thesis Structure	9
2 Background	11
2.1 Introduction	11
2.2 Systems Engineering	11
2.2.1 Definitions and Life Cycle	11
2.2.2 Model-Based Systems Engineering	13
2.3 Aircraft Design	20
2.4 Technical Risk Management	22
2.4.1 Overview	22
2.4.2 Uncertainty Quantification & Management	23
3 Literature Review	32
3.1 Introduction	32
3.2 Assumption Management	32
3.2.1 Assumption: Definition and Lifecycle	33

3.2.2	Industrial Perspective on Assumption Management	34
3.2.3	Academic Perspective on Assumption Management	40
3.3	Belief Revision and Consistency Maintenance	46
3.3.1	Introduction	46
3.3.2	Extra-Logical Factors in Belief Revision	49
3.3.3	Belief Revision Theories	49
3.3.4	Limitations	52
3.4	Knowledge Maturity	53
3.5	Margin Allocation and Management	56
3.5.1	Overview	56
3.5.2	Margin as a Change Absorber	58
3.5.3	Existing Approaches to Margin Allocation and Management	60
3.6	Conclusions	63
4	Assumption Management: Process Description and Supporting Methods	67
4.1	Introduction	67
4.2	Assumption Management: Process Description	68
4.2.1	ISO/IEC/IEEE 24774:2021	69
4.2.2	Process Description in Accordance with ISO/IEC/IEEE 24774:2021	70
4.2.3	Discussion	75
4.3	Design Belief Network	76
4.3.1	Graph-Theoretical Structure	76
4.3.2	Assumption Dependencies	79
4.3.3	Level of Confidence	80
4.4	Constraint Violation and Conflict Detection	83
4.4.1	Algorithm for Conflict Detection	84
4.4.2	Discussion	88
4.5	Conclusions	88
5	Managing Risks Associated with Assumptions	91
5.1	Introduction	91
5.2	Knowledge Maturity Assessment	92
5.2.1	Variables Selection	93
5.2.2	Normalisation	94
5.2.3	Weighting	96
5.2.4	Aggregation	99
5.2.5	Link to other Maturity Assessments	100
5.2.6	Interpretation and Visualisation	101
5.3	Assumption Matrix	104
5.4	Margin Allocation	108
5.5	Change Absorber Localisation	111
5.6	Margin Revision	115
5.7	Summary and Conclusions	119
6	Evaluation	121
6.1	Introduction	121
6.2	Software Prototype	122

6.3	Demonstration Use Cases	128
6.3.1	Use Case 1: Design of a Lightweight Fighter Aircraft	128
6.3.2	Use Case 2: Conflict between Collaborating Teams	132
6.4	Demonstration of the Developed Methods	135
6.4.1	Design Belief Network	135
6.4.2	Managing Risks Associated With Assumptions	143
6.4.3	Detection of Conflicting Assumptions	162
6.5	Industry Feedback	166
6.5.1	Purpose	166
6.5.2	Approach	166
6.5.3	Results	167
6.6	Summary and Conclusions	174
7	Summary and Conclusions	178
7.1	Introduction	178
7.2	Research Summary	178
7.3	Research Contributions	180
7.4	Limitations and Future Work	184
	References	188
	A Evaluation: Software Implementation	211
	B Evaluation: Computational Workflow in Use Case 2	218
	C Evaluation: Uncertainty Analysis	237
	D Evaluation: Questionnaire	240
	E Publications by the Author	249

List of Figures

1.1	Design process paradigm shift (adapted from [8])	2
1.2	Stages of the research methodology (adapted from [20])	7
1.3	Thesis Structure	9
2.1	Stage-Gate process overview ($G = Gate, S = Stage$) (adapted from [25]) .	13
2.2	RFLP model augmented with a computational domain (adapted from [37])	16
2.3	Example of a computational workflow	16
2.4	(a) Unordered pair of vertices, (b) Ordered pair of vertices	17
2.5	Example of a graph	18
2.6	Example of a UML class diagram (adapted from [40])	19
2.7	Types of dependencies in UML class diagrams	19
2.8	Risk management process	23
2.9	Uncertainty definition (adapted from [46])	24
2.10	Evolution of uncertainty (adapted from [47])	24
2.11	Uncertainty-based design problems (adapted from [49])	25
2.12	Robustness/Reliability in terms of probability density (adapted from [48])	25
2.13	Generic UQ&M methodology (adapted from [60])	28
2.14	McManus and Hastings framework for uncertainty mitigation and exploit- ation [50]	30
3.1	Example of a metagraph (adapted from [99])	45
3.2	Problem solver-TMS interfacing (adapted from [123])	51
3.3	Margin balance (adapted from [137])	56
4.1	Assumption management within systems engineering	68
4.2	IDEF0 representation of the assumption management process. IDEF is a family of Modelling Languages, and IDEF0 is the method for Function Modelling	74
4.3	Maturity gates at Airbus (adapted from [163])	75
4.4	UML class diagram of the <i>Design Belief Network</i>	77
4.5	Correspondence between the proposed data structure and the R-F-L-C do- mains	78
5.1	Notional progress of KMI over time ($G = Gate, S = Stage$)	102
5.2	Notional comparison of KMI (current project shown in blue) ($G = Gate,$ $S = Stage$)	103
5.3	Visualisation of the individual constituents of Knowledge Maturity	104
5.4	Concept of the <i>Assumption Matrix</i>	105

5.5	Margin Redundancy Example (Computational model from [41])	108
5.6	Sankey diagram to visualise margin-assumption associations	111
5.7	Example of a propagation path	112
5.8	Concept of Margin Space	116
6.1	Schematic of the software implementation	122
6.2	Capturing an assumption in AirCADia Architect	124
6.3	Streamlit user interface	125
6.4	UML class diagram of an observer pattern (adapted from [183])	127
6.5	Computational Workflow for Aircraft Sizing	130
6.6	Interface between high-lift devices and structural layout (adapted from [41], [184])	132
6.7	Simplified, high-level representation of the computational workflow	133
6.8	R-F-L domains (AirCADia Architect)	136
6.9	Computational workflow for sizing and performance assessment (AirCADia Explorer)	137
6.10	User interface to record assumptions (AirCADia Architect)	138
6.11	User interface to access captured assumptions (AirCADia Architect)	139
6.12	User interface to link with existing assumptions (AirCADia Architect)	140
6.13	Recording the identified conflict between assumptions α_{10} and α_{11} (AirCADia Architect)	141
6.14	<i>DBN</i> corresponding to Use Case 1 (Software prototype tool)	142
6.15	Knowledge maturity assessment corresponding to Use Case 1 (Software prototype tool)	144
6.16	Confidence assessment in Use Case 1 (Software prototype tool)	145
6.17	Illustration of KMI progress over time	147
6.18	<i>Assumption Matrix</i> corresponding to Use Case 1 (Software prototype tool)	148
6.19	System model colouring to match assumption prioritisation in the <i>Assumption Matrix</i> (AirCADia Architect)	149
6.20	Sankey diagram of margin-assumption dependencies corresponding to Use Case 1 (Software prototype tool)	150
6.21	Non-mitigated assumptions corresponding to Use Case 1 (Software prototype tool)	151
6.22	Updated Sankey diagram of margin-assumption dependencies corresponding to Use Case 1 (Software prototype tool)	152
6.23	Instance of margin redundancy detection (Software prototype tool)	152
6.24	List of assumptions, which represent potential change initiators (Software prototype tool)	153
6.25	Demonstration of <i>Change Absorber Localisation</i> with α_{10} (Software prototype tool)	154
6.26	Assigning a margin to the fuselage volume (AirCADia Architect)	155
6.27	Demonstration of <i>Change Absorber Localisation</i> following margin assignment (Software prototype tool)	156
6.28	Demonstration of component freezing (AirCADia Architect)	156
6.29	Demonstration of <i>Change Absorber Localisation</i> following component freezing (Software prototype tool)	157

6.30	Suggested margin revisions in the presence of change (Software prototype tool)	158
6.31	Margin Space (AirCADia Vision)	159
6.32	Constraint violation (AirCADia Vision)	160
6.33	Constraint satisfaction (AirCADia Vision)	161
6.34	Detected set of conflicting assumptions in Use Case 2 (Prototype Tool) . .	163
6.35	Parallel coordinates plot showing feasible combinations (AirCADia Vision)	163
6.36	Contour plot of c_s/c vs. x_{FS} (AirCADia Vision)	165
6.37	Contour plot with an additional constraint (AirCADia Vision)	165
6.38	Answers to Questions 6.1, 6.2, 6.3 and 6.4	168
6.39	Answers to Questions 7.1 and 7.2	169
6.40	Answers to Questions 8.1, 8.2 and 8.3	170
6.41	Answers to Questions 9.1, 9.2 and 9.3	171
6.42	Answers to Questions 10.1, 10.2 and 10.3	172
B.1	Computational workflow corresponding to Use Case 2 (AirCADia Explorer)	219
B.2	Interface between the front spar and retracted slat	230
B.3	Illustration of the flapped wing area (from [41])	231
B.4	Execution of the computational workflow - Input values (AirCADia Explorer)	235
B.5	Execution of the computational workflow - Output values (AirCADia Explorer)	236

List of Tables

2.1	Multiplicity values in UML Class Diagrams (adapted from [40])	20
3.1	Types of assumption dependencies according to Yang <i>et al.</i> [95]	44
3.2	Knowledge Maturity Scale [128]	55
5.1	Scale to select the intensity of importance [172]	97
6.1	List of assumptions corresponding to Use Case 1	131
6.2	Description of key parameters in Use Case 2	134
6.3	List of assumptions corresponding to Use Case 2	134
6.4	Results of the uncertainty analysis	146
6.5	Full-factorial DoE setup	163
6.6	Participants information	167
6.7	Likert-type questions related to DBN	167
6.8	Likert-type questions related to Knowledge Maturity Assessment	169
6.9	Likert-type questions related to Assumption Matrix and Change Absorber Localisation	170
6.10	Likert-type questions related to Margin Allocation and Revision	171
6.11	Likert-type questions related to the overall approach	172
6.12	Open-ended questions	174
B.1	Description of the parameters in Use Case 2	220
B.2	Description of the computational models in Use Case 2	224

List of Algorithms

1	<i>Confidence</i> Assessment	82
2	Detection of Conflicting Assumptions	86
3	DFS(G,u)	87
4	Assumption Matrix Generation	107
5	Margin Allocation Analysis	110
6	Change Absorber Localisation	114
7	Margin Revision	118

List of Abbreviations

AADF	Architectural Assumption Documentation Framework
AR	Aspect Ratio
ATMS	Assumption-based Truth Maintenance System
BFS	Breadth-First Search
BFT	Bladder Fuel Tank
CBE	Current Best Estimate
CPM	Change Prediction Method
DBN	Design Belief Network
DFS	Depth-First Search
DoD	Department of Defense
DoE	Design of Experiment
DRM	Design Research Methodology
DSM	Design Structure Matrix
ICAM	Integrated Computer Aided Manufacturing
IDEF	ICAM DEFinition
INCOSE	International Council on Systems Engineering
JTMS	Justification-based Truth Maintenance System
KMI	Knowledge Maturity Index
KMMM	Knowledge Management Maturity Model
LoC	Level of Confidence
MBSE	Model-Based Systems Engineering

NTSB	National Transportation Safety Board
PDF	Probability Density Function
PMF	Probability Mass Function
R-F-L-C	Requirements - Functional - Logical - Computational
RFLP	Requirements Functional Logical Physical
TMS	Truth Maintenance System
TRL	Technology Readiness Level
SFC	Specific Fuel Consumption
SRL	System Readiness Level
SysML	Systems Modeling Language
UML	Unified Modeling Language
UQ	Uncertainty Quantification
UQ&M	Uncertainty Quantification & Management
VDVT	Variable Dihedral Vertical Tail
WCE	Worst-Case Estimate

Chapter 1

Introduction

1.1 Context and Motivation

In 1903, the Wright brothers invented mankind's first practical airplane for sustained, powered and controlled flight [1]. Since then, aircraft design has tremendously developed, especially during the period 1945-1960 where the first jetliners were introduced [2]. However, aircraft design has been conservative for many decades as we are still adopting the classic *tube and wing* configuration today. Conservatism stems from the need to mitigate the risk of expensive redesigns and performance penalty, which is associated with uncertainty as an inherent characteristic of complex systems design. Aerospace vehicle design is a complex, multidisciplinary endeavour which requires the simultaneous design of its systems (e.g. wing), subsystems (e.g. environmental control system) and components (e.g. actuators). The performance of the resulting product not only depends on each subsystem independently, but also on the inherent inter-dependencies [3], which introduce further uncertainty.

Uncertainty due to lack of knowledge, known as epistemic uncertainty [4], can impede the design process. In order to proceed, assumptions are typically introduced to fill knowledge gaps. However, the uncertainty inherent in the assumptions constitutes a risk to be mitigated, especially at early-stage design where about 70% of the budget is committed

[5]. Moreover, since architectural decisions are the most impactful design decisions, it is of the utmost importance to examine an architecture's vulnerability to its assumptions [6], in addition to managing these assumptions.

Such interplay between uncertainty and the importance of decisions made throughout product development is at the heart of a paradigm shift in complex systems design. This paradigm shift has been underway, as suggested by the *1996 NSF Strategic Planning Workshop* [7], by which downstream knowledge and *system performance - cost* trade-offs were brought earlier in the design process. This is illustrated in Figure 1.1.

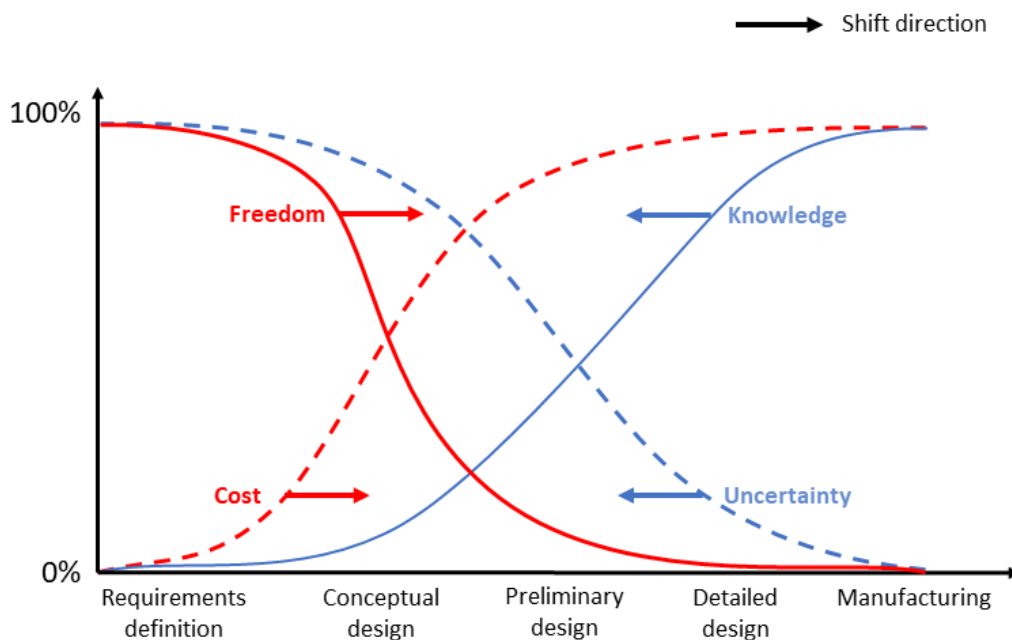


Figure 1.1: Design process paradigm shift (adapted from [8])

Design freedom represents the flexibility in the design [8], which is necessary to accommodate changes as issues are encountered. Figure 1.1 shows how freedom significantly decreases at the conceptual stage, due to freezing the concept, which limits flexibility in subsequent stages. At the same time, committed costs significantly increase at conceptual design, which are due to the aforementioned highly impactful decisions. Therefore, the trend has been towards shifting to the right both the *Freedom* curve (i.e. increasing flexibility for subsequent stages), and the *Cost* curve (i.e. delaying highly impactful decisions). For instance, one way to delay decisions and maintain some flexibility

is through set-based design [9], as opposed to the traditional point-based design. Design margins are also used to proactively address the rapid decline in flexibility [8], where the excess introduced by margins can be used to facilitate future changes. Additionally, available knowledge is at its lowest at early-stage design, which led to attempts at bringing more knowledge earlier in the process [7] (e.g. bringing physics-based modelling earlier in the process). This trend would result in a faster decrease in uncertainty, as illustrated in Figure 1.1 where both the *Knowledge* and *Uncertainty* curves are shifting to the left.

Therefore, it is of the utmost importance to start managing uncertainty as early in the design as possible, especially that assumptions can negatively impact the system if they turn out to be invalid. In fact, assumptions about the system's environment underlie many safety requirements [10]. Some potential consequences are system failure, incomplete requirements, violation of requirements, inconsistency issues, and budget and schedule overruns [11]. Implicit assumptions are also a leading cause of the aforementioned consequences [11]. In the context of software development, Lehman states that “*invalid assumptions constitute the primary source of project and system misbehaviour or, in the extreme, failure*” [12].

Examples of invalid assumptions causing major losses include the following:

- The recent fatal accidents involving the Boeing 737 Max 8 (Lion Air flight 610 on October 29, 2018 and Ethiopian Airlines flight 302 on March 10, 2019), which were caused by assumptions on pilot recognition made during the design process. The report produced by the U.S. National Transportation Safety Board (NTSB) [13] points out that “*the accident pilot responses to the unintended MCAS¹ operation were not consistent with the underlying assumptions about pilot recognition and response that Boeing used, based on FAA guidance, for flight control system functional hazard assessments, including for MCAS, as part of the 737 MAX design.*” For instance, one of the aforementioned assumptions was that “*uncommanded system inputs are readily recognizable and can be counteracted by overriding the failure by movement*

¹MCAS refers to Maneuvering Characteristics Augmentation System

of the flight controls 'in the normal sense' by the flight crew and do not require specific procedures” [13]. One of the resulting recommendations from the NTSB to Boeing was to “incorporate design enhancements (including flight deck alerts and indications), pilot procedures, and/or training requirements, where needed, to minimize the potential for and safety impact of pilot actions that are inconsistent with manufacturer assumptions” [13]. Another recommendation from the NTSB was to “develop robust tools and methods, with the input of industry and human factors experts, for use in validating assumptions about pilot recognition and response to safety-significant failure conditions as part of the design certification process” [13].

- The failure of the Ariane 5 maiden flight on June 4, 1996, which was caused by assumptions made in the software of the inertial reference system. Such assumptions made by designers included [14]: (a) *“the alignment function would have no effect after launch. No effect had been seen for Ariane 4.”*; (b) *“it was not necessary to protect against an excessive value of the inertial reference system variable related to the horizontal velocity. No protection had been needed for Ariane 4”*; (c) *“arithmetic overflow protection was unnecessary. An overflow had never occurred for Ariane 4”*; and (d) *“two identical components made a dual redundant system”*. The inquiry board investigating the incident recommended in its report [15] to *“identify all implicit assumptions made by the code and its justification documents on the values of quantities provided by the equipment. Check these assumptions against the restrictions on use of the equipment.”*

The aforementioned examples illustrate the necessity to manage assumptions as a way of mitigating the risk of negative impact. The current systems engineering processes (e.g. ISO/IEC/IEEE 15288 [16] and NASA SP-2016-6105 [17]) already support working with assumptions by explicitly documenting and reviewing them as the design progresses. Nevertheless, this traditional document-centric approach has shown its limitations in modern complex systems design where the significant amount of assumptions and their dependencies make assumption management cognitively heavy. In fact, a study

that surveyed practitioners showed that simply documenting assumptions has little benefit [18].

Another paradigm shift has been taking place from traditional (i.e. document-based) systems engineering to model-based systems engineering (MBSE). This was encouraged by the increasing complexity of systems, where MBSE is seen as an approach for “*managing complexity, maintaining consistency, and assuring traceability during system development*” [19]. Systems engineering activities, such as assumption management, suffer from some limitations imposed by a document-based approach. Such limitations include recording the generated information in multiple text documents, which makes it difficult to maintain and synchronise such information [5]. In fact, a manual update of documents is necessary in case of changes, which can be highly costly and error-prone in large-scale projects. Moreover, natural language in documents can lead to ambiguity and implementation errors. Therefore, assumption management, in addition to the other systems engineering activities, would benefit from the aforementioned paradigm shift towards MBSE.

1.2 Aim and Objectives

Based on the motivation to this research and analysis of the literature, the aim is formulated as follows:

Aim

Develop a computational approach to support assumption management in model-based systems engineering, with an explicit consideration of the uncertainty in assumptions.

The objectives that shall be addressed to achieve the overall aim are formulated as follows:

1. Devise methods to enable assumption management in a model-based design environment.

2. Devise methods to manage risk of change due to invalid assumptions, with an explicit consideration of both assumptions and margins.

1.3 Scope

The scope of this research was limited to system design and technical risk management in a systems engineering process. This research particularly focused on the early design stages of aircraft, where the decisions made have the highest impact on product development. Thus, developing support for early design is expected to have the largest positive effect on the overall value of aircraft programmes, especially that invalid assumptions can cause costly redesigns or catastrophic failures if not managed early. Furthermore, although this research was approached from an aircraft perspective due to familiarity and access to experts in aircraft design, the proposed methods are generic and should be applicable to other complex engineering systems.

Organisational aspects were considered outside the scope as the focus was on computational support. Such organisational aspects include for instance how knowledge management is conducted within a company, or which role within a company should be responsible for carrying out a given task. Furthermore, such aspects would differ from one organisation to another, thus limiting generalisation. The methods developed in this research are expected to support existing processes within organisations.

1.4 Research Methodology

The research methodology adopted for the present work is the Design Research Methodology (DRM), which was proposed by Blessing and Chakrabarti [20] as a generic and systematic methodology to conduct design research. The DRM is particularly suited for prescriptive engineering design research, where the research described in this thesis would correspond to a *Type 3* study. The core of a *Type 3* study is the development of design support, where there is sufficient literature to describe the phenomenon under study, but

either support does not exist, or existing support is insufficient. However, the literature must indicate or demonstrate the need for developing new support, or improving existing support, in order to improve the existing situation. As shall be demonstrated in Chapter 3, there is sufficient literature to describe the phenomenon of design assumptions and how they affect the design of complex engineered systems. Furthermore, the literature indicates limitations in current practice in managing assumptions. Therefore, the focus of the presented research in this thesis is the development of support for assumption management. The followed steps of the research methodology are shown in Figure 1.2 and described in the subsequent sections. Note that a *Type 3* study consists of a preliminary evaluation only due to the available timeframe, and the four stages are not executed linearly but there rather are iterations within and between the aforementioned stages.

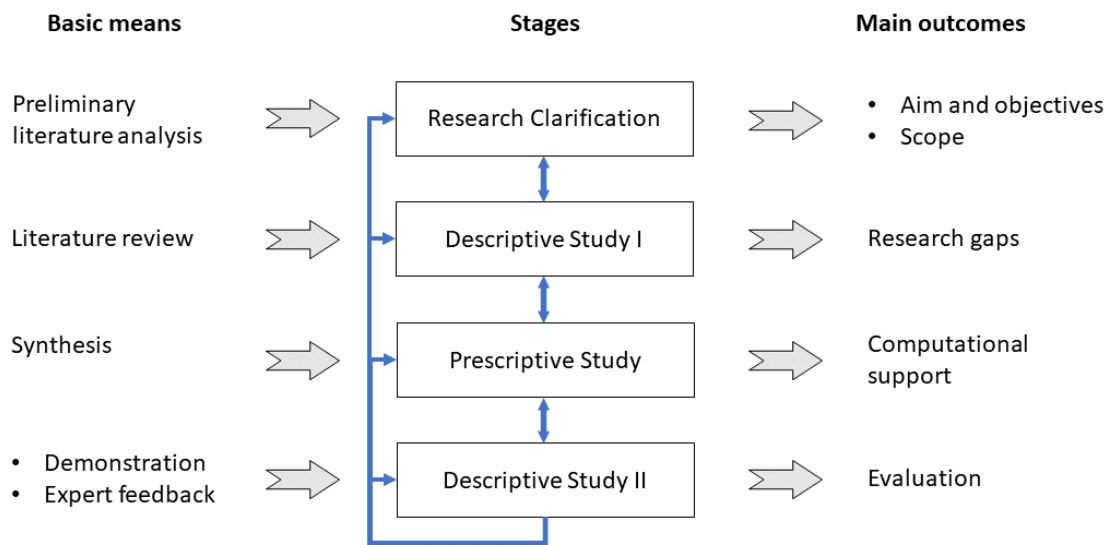


Figure 1.2: Stages of the research methodology (adapted from [20])

1.4.1 Research Clarification

This stage consisted of clarifying the overall research aim, developing a research plan, and framing the subsequent stages [20]. The literature was analysed for an initial understanding of the context in which this research is taking place, with a focus on assumptions. *Research Clarification* led to formulating the aim, objectives and scope of this research.

1.4.2 Descriptive Study I

This stage consisted of reviewing the literature to improve the understanding of assumptions in the early design of complex systems, in addition to state-of-the-art academic research and current industrial practice in managing assumptions. *Descriptive Study I* led to identifying research gaps, as well as relevant concepts and methods to inform the *Prescriptive Study*. Recall that there are iterations between the DRM stages. The improved understanding from *Descriptive Study I* allowed to iteratively refine the aim and objectives throughout this PhD.

1.4.3 Prescriptive Study

This stage consisted of developing a computational support to achieve the research aim. The synthesis, informed by the acquired understanding from *Descriptive Study I*, involved adapting and extending concepts and methods from different fields such as engineering design, systems engineering, mathematics and computer science. This led to proposing novel methods that support assumption management in Model-Based Systems Engineering.

1.4.4 Descriptive Study II

This stage aimed at evaluating the applicability and usefulness of the developed methods in addressing the identified industrial problem. The evaluation consisted of two parts: a demonstration (Section 6.4) and feedback from experts in the aerospace industry (Section 6.5). Evaluating the developed methods required an implementation into a prototype software tool, as described in Section 6.2. Furthermore, an initial evaluation was performed in accordance with a *Type 3* study [20], in readiness for scaling to a future industrially-inspired case study.

The *demonstration* consisted of applying the developed methods to: (1) the hypothetical design of a fighter aircraft and (2) conflicting assumptions in collaborative design, in or-

der to assess whether the methods work as intended.

The *industry feedback* session consisted of demonstrating the developed methods to industry experts to obtain feedback on expected usefulness in practice, thus assessing the impact of this research.

1.5 Thesis Structure

The thesis structure is shown in Figure 1.3. Each chapter is related to the corresponding stage of the research methodology.

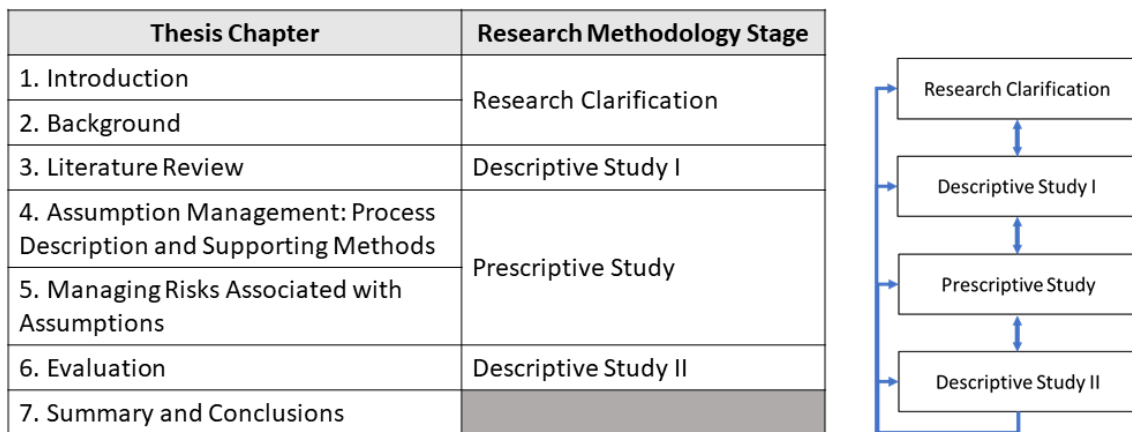


Figure 1.3: Thesis Structure

Chapter 2 introduces definitions and concepts underlying this research. In Chapter 3, limitations of existing approaches, as well as opportunities for improvement, are identified. The developed methods are presented in Chapters 4 and 5.

In Chapter 4, the assumption management process is described in accordance with ISO/IEC/IEEE 24774:2021 [21] (Section 4.2), in addition to presenting a graph-theoretical structure (*Design Belief Network*) to capture assumptions, their uncertainty and dependencies (Section 4.3). Furthermore, an algorithm to detect conflicting assumptions that lead to constraint violation is proposed in Section 4.4.

In Chapter 5, a set of methods is presented: (a) a composite indicator, *Knowledge Maturity Index (KMI)*, to assess the overall risk of change due to lack of knowledge (Section 5.2); (b) a method (*Assumption Matrix*) to prioritise individual assumptions in terms of

their risk to initiate change (Section 5.3); (c) an algorithm to provide the status of margin allocation, with an explicit consideration of assumptions (Section 5.4); (d) an algorithm to detect the closest margin to an assumption that can absorb change (Section 5.5); and (e) an algorithm to suggest margin revisions following changes in assumptions (Section 5.6).

Chapter 6 reports the evaluation of the developed support by introducing a prototype software tool (Section 6.2), applying the support to two demonstration use cases (Sections 6.3 and 6.4), and discussing the feedback from aircraft design experts regarding the usefulness of the support (Section 6.5).

Finally, Chapter 7 presents the conclusions regarding the academic and practical contributions, in addition to avenues for future work. The thesis chapters are supported by Appendices A-E.

Chapter 2

Background

2.1 Introduction

This chapter provides an overview of topics and concepts relevant to this research. Section 2.2 introduces the systems engineering process and the life cycle stages (Section 2.2.1), then model-based systems engineering is presented along with relevant topics (i.e. *RFLP Paradigm*, *Graph Theory* and *UML Class Diagram*) (Section 2.2.2). Section 2.3 gives a brief overview of the aircraft design process and its three major phases. Finally, Section 2.4 introduces technical risk management and describes the process of uncertainty quantification and management.

2.2 Systems Engineering

2.2.1 Definitions and Life Cycle

The International Council on Systems Engineering (INCOSE) defines a *system* as “*an integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements*” [5]. This interaction between elements leads to the notion of *system architecture*, which is defined

as “the embodiment of concept, the allocation of physical/informational function to the elements of form, and the definition of relationships among the elements and with the surrounding context” [6]. Finally, *systems engineering* is defined by the INCOSE as “a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods” [22].

According to ISO/IEC/IEEE 15288 [16]: “A system progresses through its life cycle as the result of actions, performed and managed by people in organizations, using processes for execution of these actions.” The ISO/IEC/IEEE 24748-1 describes a generic set of life cycle stages [23]:

1. **Concept:** This stage involves identifying stakeholders’ needs, exploring concepts and proposing viable solutions.
2. **Development:** This stage involves refining system requirements, creating a solution description, building the system, and verifying and validating the system.
3. **Production:** This stage involves producing the system and inspecting/testing it.
4. **Utilization:** This stage involves operating the system to satisfy the users’ needs.
5. **Support:** This stage involves providing sustained system capability.
6. **Retirement:** This stage involves storing, archiving or disposing of the system.

Note that there are crosscutting technical management activities that bridge between the technical teams (e.g. system design or product integration) and project management (e.g. scheduling or cost estimation) [17]. Such crosscutting technical management activities include technical planning, requirement management, interface management, technical risk management, configuration management, technical data management, technical assessment, and decision analysis [17].

To proceed from one stage to the next, *decision gates* are used in project management. Decision gates (also called *control gates*, *milestones* or *reviews*) are approval events

during a project, which ensure that scheduled activities are first satisfactorily completed before pursuing subsequent activities [5]. One model of such an approach to project management is the *Stage-Gate Process*, as proposed by Cooper [24].

The *Stage-Gate Process* consists of five main stages and five gates, as illustrated in Figure 2.1. The fundamental idea is that in each stage, activities are carried out to collect knowledge, which is then aggregated and analysed to inform decision-making at the gates [25]. A gate acts as a decision point, where the progress made up to the gate is reviewed [26]. A decision can then be made to either proceed to the next stage, do some rework before moving to the next stage, or terminate the project.

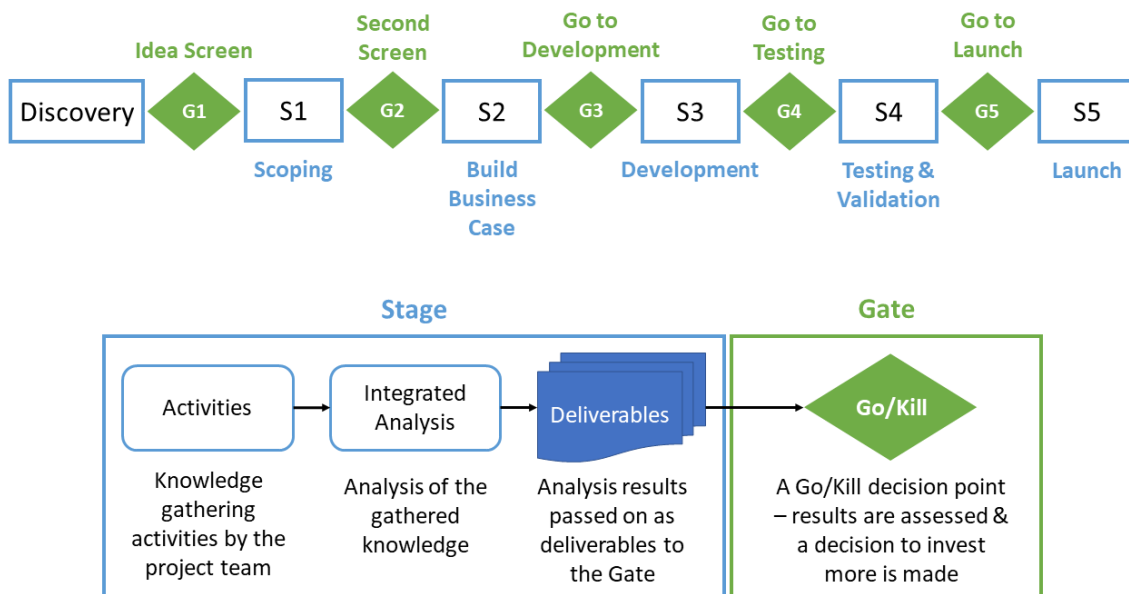


Figure 2.1: Stage-Gate process overview ($G = Gate$, $S = Stage$) (adapted from [25])

2.2.2 Model-Based Systems Engineering

As industry is currently facing increasing complexity of systems, a shift to Model-Based Systems Engineering (MBSE) is currently underway to manage complexity, maintain consistency, and assure traceability during system development [19]. INCOSE defines MBSE as the “*formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases*” [5].

The traditional document-centric approach to systems engineering involves capturing all the generated information about systems in multiple text documents, which makes it difficult to maintain and synchronise such information [5]. In contrast, the generated information about systems is stored within a system model in the MBSE approach. A system model, which represents a central repository composed of multiple complementary perspectives (or views), acts as the “sole source of truth” about the system under development [19]. Therefore, the system model can be used to propagate changes made to all relevant elements in the model, instead of necessitating a manual update of documents [27]. Furthermore, since natural language can lead to ambiguity and implementation errors, formal modelling languages such as SysML¹ have been developed to support MBSE. Thus, a *system* is represented by a *model*, which in turn is described by a *modelling language* [19].

In the context of system design, the system model provides the current design’s description, the mission description and requirements formulated in a data structure adequate for design space exploration [27]. Additionally, architectural decisions must also be modelled accordingly, which can be done via the RFLP paradigm. The latter is introduced in the following subsection.

RFLP Paradigm

Functional reasoning, i.e. defining the functions to be performed by the system, plays a central role in system architecting [28]. One approach to functional reasoning is the RFLP paradigm [29] (illustrated in Figure 2.2), which is based on the VDI 2206 standard *Design methodology for mechatronic systems* [30]. RFLP considers that system architecting (via functional reasoning) is distributed over four notional domains: *Requirements*, *Functional*, *Logical* and *Physical*, which can be defined as follows:

- Requirements domain (R): contains the requirements which can be hierarchically decomposed. The requirements are mapped to the functions that fulfil them in the

¹Systems Modeling Language: <https://sysml.org/> (Accessed: 26/11/2021)

Functional domain.

- Functional domain (F): contains the functions that the system must perform, which can also be hierarchically decomposed. The functions are mapped to the components that realise them.
- Logical domain (L): contains the components (or elements of form) realising the system functions, in addition to their connectivity via ports to represent exchange of material, energy or information.
- Physical domain (P): contains a 3D CAD model which essentially captures the spatial/topological relationships amongst the components.

Bile *et al.* [31] then proposed to augment the RFLP model with a *Computational* domain (C) for automated systems sizing and performance assessment (as illustrated in Figure 2.2), followed by a graph-theoretical structure that captures the dependencies between the R-F-L-C domains [32]. This graph-theoretical structure is considered by Guenov *et al.* [32] as a means of capturing rationale during system architecting. Design rationale is “*an explicit expression of the reasons behind decision making when designing an artifact*” [33]. Design rationale can include, for instance, justification for decisions, considered alternatives and trade-off studies [34]. Note that assumptions are an important part of design rationale since they represent starting points for reasoning, and thus directly influence the decisions made [35].

The Computational Domain consists of a computational workflow, which can be defined as “*an ordered set of computational models*” [36]. Such models are associated with logical components to assess their behaviour, and the computational workflow sets the execution order of the models. An example of a computational workflow is shown in Figure 2.3, where boxes refer to computational models, blue circles refer to independent input variables and orange circles refer to output variables.

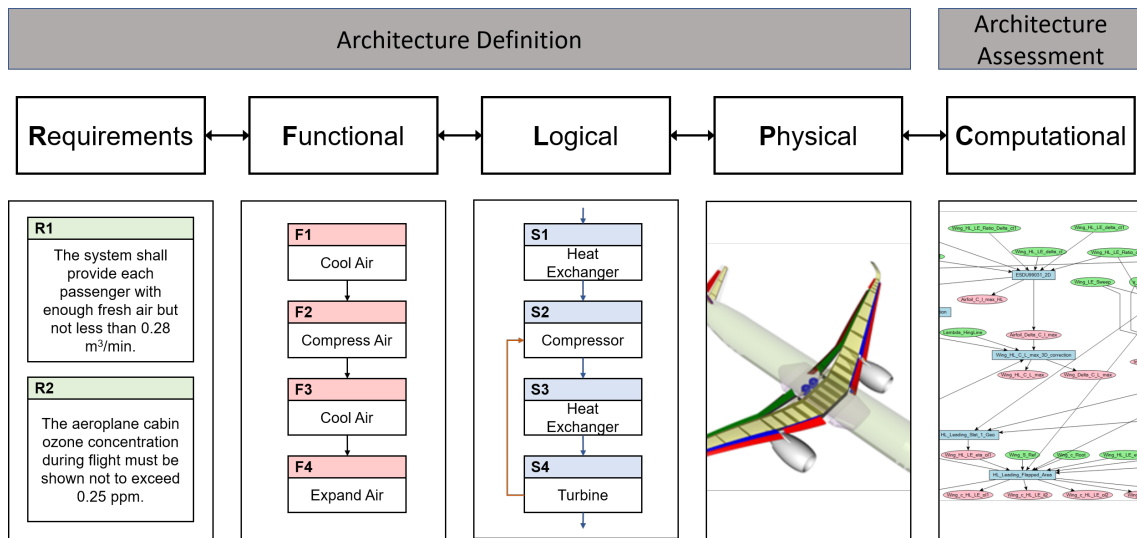


Figure 2.2: RFLP model augmented with a computational domain (adapted from [37])

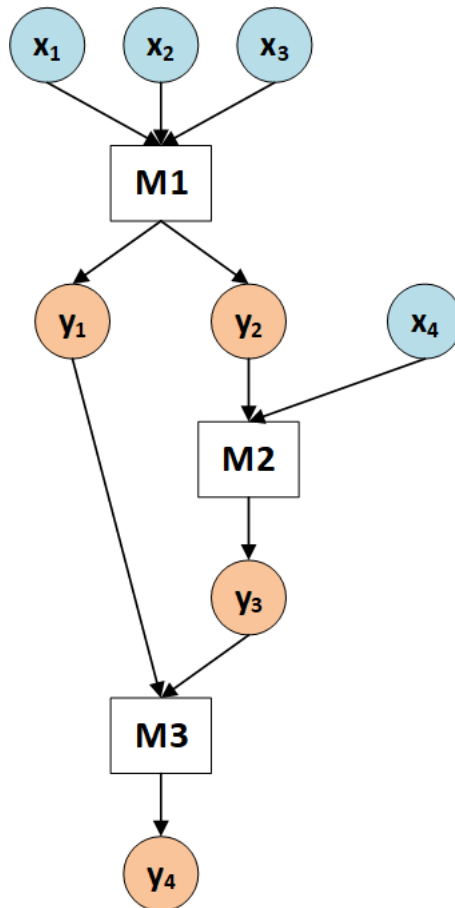


Figure 2.3: Example of a computational workflow

Graph Theory

Different mathematical tools have been used to formulate the theoretical foundations of MBSE, and one of such tools is graph theory. As shall be seen in this thesis, graph theory

source vertex A (and from the left edges) would visit the vertices in the following order: A, F, B, C, G, D, E . Vertices F, B and C are considered to be at distance 1 from A , whereas vertices G, D and E are considered to be at distance 2 from A .

- **Depth-First Search:** Considering a graph $G = (V, E)$ and a source vertex s , DFS traverses all unexplored edges leaving the most recently discovered vertex u . Then, once all edges from u have been explored, DFS backtracks to explore all edges leaving u 's predecessor (i.e. vertex from which u was discovered), and the search continues until all vertices from s have been discovered [39]. In Figure 2.5, a DFS starting from source vertex A (and from the left edges) would visit the vertices in the following order: A, F, G, D, B, C, E .

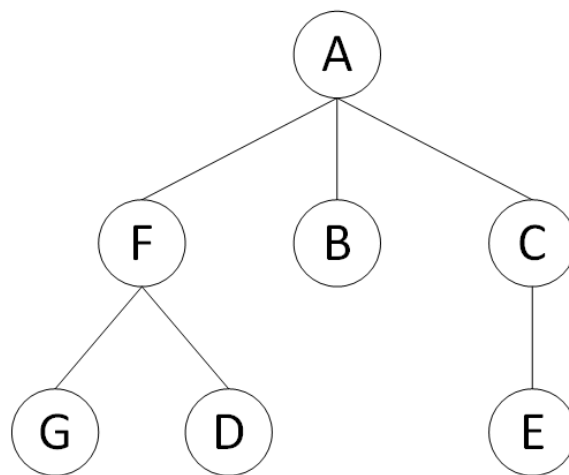


Figure 2.5: Example of a graph

UML Class Diagram

An edge connecting two vertices can actually represent logical connections between these two elements. To describe such logical dependencies, UML² class diagrams are typically used in object-oriented modelling. As shown in Figure 2.6, a class diagram consists of the class' name, attributes and operations [40]. For example in Figure 2.6, the *Flight* class has an attribute with the name *flightNumber*, and the type of that attribute is an

²Unified Modeling Language: <https://www.uml.org/> (Accessed: 26/11/2021)

Integer. The *Flight* class has also an operation *delayFlight*, which transforms an input (*numberOfMinutes* of type *Minutes*) to an output of type *Date* (i.e. the new arrival date in this example).

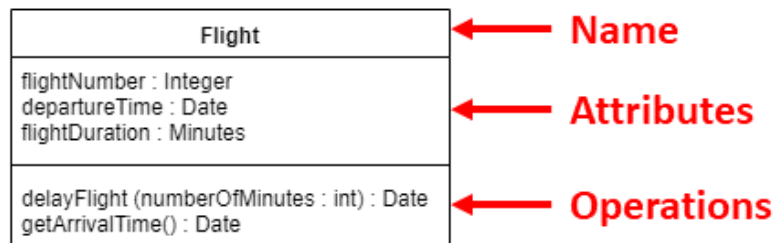


Figure 2.6: Example of a UML class diagram (adapted from [40])

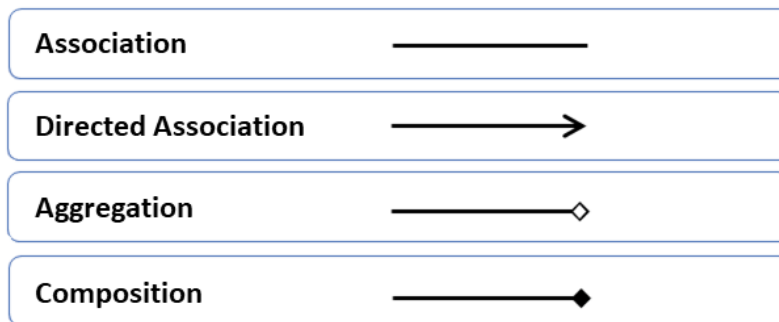


Figure 2.7: Types of dependencies in UML class diagrams

Some of the logical connections that are represented in UML class diagrams are as follows (cf. Figure 2.7) [40]:

- **Association:** This refers to a bi-directional relationship between two classes. For example, a *Flight* can have an assigned *Plane*, and the *Plane* is in turn said to have an assigned *Flight*. This means there is a (bi-directional) association between the *Flight* and *Plane* classes.
- **Directed Association:** This refers to a uni-directional relationship between two classes. For example, this can represent the fact that changes in a component may cause changes in the related parameters/models, whereas the opposite is not true.
- **Aggregation:** This refers to a “whole to its parts” relationship. For example, a parameter is *part of* a computational model.

Indicator	Meaning
0..1	Zero or one
1	One only
0..	Zero or more
	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

Table 2.1: Multiplicity values in UML Class Diagrams (adapted from [40])

- **Composition:** This is a special type of Aggregation, where destroying the whole implies that the parts are also destroyed. For example, a system is *composed of* components, so destroying the system implies destroying its components as well.
- **Reflexive Association:** This refers to a class being associated with itself, meaning that an instance of the class is related to another instance of the same class. For example, a top-level requirement can be associated with a lower-level requirement, where both are instances of the same class *Requirement*.

Multiplicity values can be added at the ends of the UML logical connections to indicate how many instances of some class an object can be related to. Table 2.1 describes some multiplicity values.

2.3 Aircraft Design

Aircraft design is an iterative process that typically starts from requirements definition. In civil aviation, requirements are defined by the aircraft manufacturer based on, for instance, input from customers, market analysis, certification specifications and previous experience [41]. According to Raymer [41], the aircraft design process consists of three major phases:

1. **Conceptual Design:** During this phase, a range of aircraft configuration concepts is considered, where a single best design and well-balanced set of requirements

are selected following trade studies. If no design can be found, requirements may need to be revised. To enable a computationally efficient exploration of the design space, low-fidelity models are used for rapid analysis of various concepts, where such models are usually empirical rather than physics-based. Note that, during conceptual design, design points are represented by top-level parameters, which are usually the *thrust-to-weight ratio* (T/W) (i.e. sea level static thrust over take-off gross weight) and the *wing loading* (W/S) (i.e. take-off gross weight over wing area). The "Master Equation" from Mattingly *et al.* [42] can be used to enable an energy-based constraint analysis, which allows then to identify design points that satisfy the design constraints. Once a design point (represented by T/W and W/S) is selected, the aircraft needs to be initially sized. Sizing is about estimating the take-off gross weight based on empirical models for empty weight, and mission analysis for the required fuel weight.

2. **Preliminary Design:** During this phase, the selected concept is refined and analysed in increasing detail (higher-fidelity models are used, such as computational fluid dynamics), to the point where the company has sufficient information to *freeze* the design. A key activity in this phase is lofting, which is the "*mathematical modeling of the outside skin of the aircraft with sufficient accuracy to ensure proper fit between its different parts*" [41]. The top-level parameters from conceptual design (e.g. take-off gross weight and sea level static thrust) are passed down to preliminary design as constraints, where different disciplinary teams collaborate to meet such constraints. Preliminary design concludes with a *full-scale development proposal*, which is passed on to the detail design phase.
3. **Detailed Design:** During this phase, the actual parts are designed for fabrication and assembly, including the small pieces such as brackets and structural clips. Every single part must be designed with its corresponding drawing/CAD file. Furthermore, testing efforts increase during this phase.

Note that, in this thesis, *early design* refers to the conceptual and preliminary design stages.

2.4 Technical Risk Management

2.4.1 Overview

According to the ISO/IEC/IEEE 15288 standard [16], “*the purpose of the Risk Management process is to identify, analyze, treat and monitor the risks continually*”. The Society for Risk Analysis defines a risk as “*the potential for realization of unwanted, negative consequences of an event*”, and is commonly described as the combination of the probability of occurrence (likelihood) and the severity of consequences (impact) [43].

According to INCOSE [5], NASA [17] and the U.S. DoD [44], the risk management process consists of four main activities:

1. **Risk Identification:** Identification and documentation of potential risks.
2. **Risk Assessment:** Evaluation of the likelihood and impact of the risks. Risks are then prioritised.
3. **Risk Mitigation:** Planning and implementation of strategies for reducing risks to an acceptable level.
4. **Risk Monitoring:** Tracking changes in risks and providing feedback to the aforementioned process activities (*e.g. a new risk is identified, or an existing risk must be re-assessed*).

The risk management process is illustrated in Figure 2.8.

Technical risk management provides crucial input to decision-making during product development. Aristodemou *et al.* [45] found, through a scoping review, that both *technical risk* and *technical uncertainty* are among the most important technology selection criteria in early-stage projects. *Technical risk* is determined by analysing uncertainty, the



Figure 2.8: Risk management process

critical assumptions, and the technology, whereas *technical uncertainty* is defined as lack of knowledge critical to decision-making [45].

Uncertainty thus plays a central role in risk management, and shall be the focus of the following section.

2.4.2 Uncertainty Quantification & Management

Nikolaidis defines certainty, in the context of decision theory, as “*the condition in which a decision maker knows everything needed in order to select the action with the most desirable outcome*” [46]. Thus, uncertainty can be seen as the gap between the present state of knowledge and certainty, where uncertainty can be broken down into reducible and irreducible uncertainty [46]. Figure 2.9 illustrates this definition.

Many instances of uncertainty exist in design: Model fidelity, experimental data, future requirements and so forth. Another important aspect is that uncertainty evolves during the engineering design process. As we progress in the design, more knowledge becomes available, thus reducing some instances of uncertainty. Figure 2.10 illustrates such

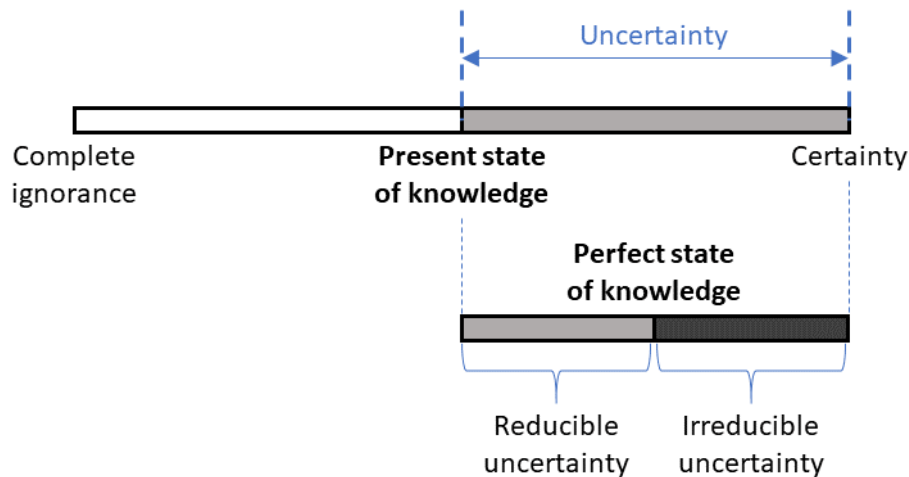


Figure 2.9: Uncertainty definition (adapted from [46])

an evolution. Furthermore, requirements change with time, which results in relaxing or tightening constraints as a response to uncertainty [3].

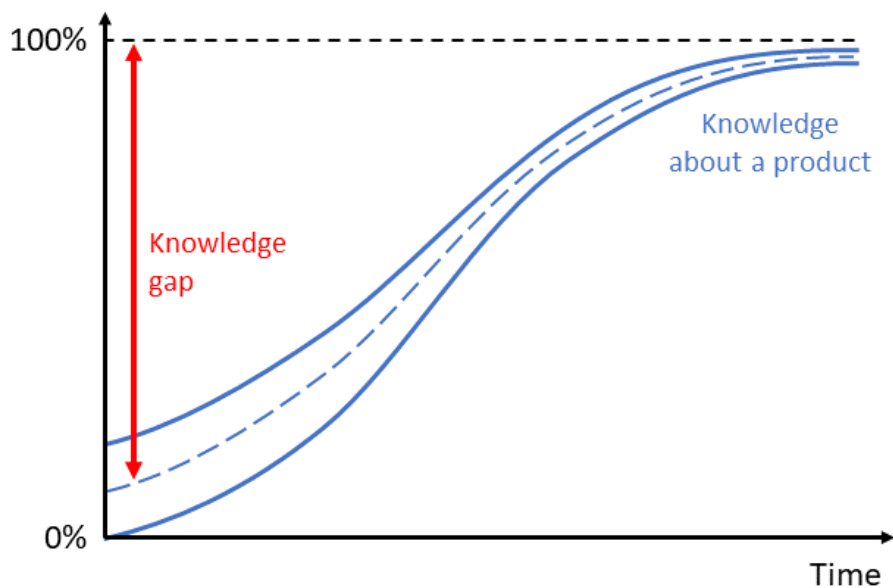


Figure 2.10: Evolution of uncertainty (adapted from [47])

Two major classes of uncertainty-based design problems exist: robust and reliability-based design [48]. Robust design seeks a design that is negligibly sensitive to small fluctuations in the uncertain quantities, whereas reliability-based design seeks a design whose probability of failure is within some acceptable threshold [48]. Figure 2.11 illustrates the classification of uncertainty-based design problems in terms of design risk, which is a

combination of the likelihood of an event and its consequences.

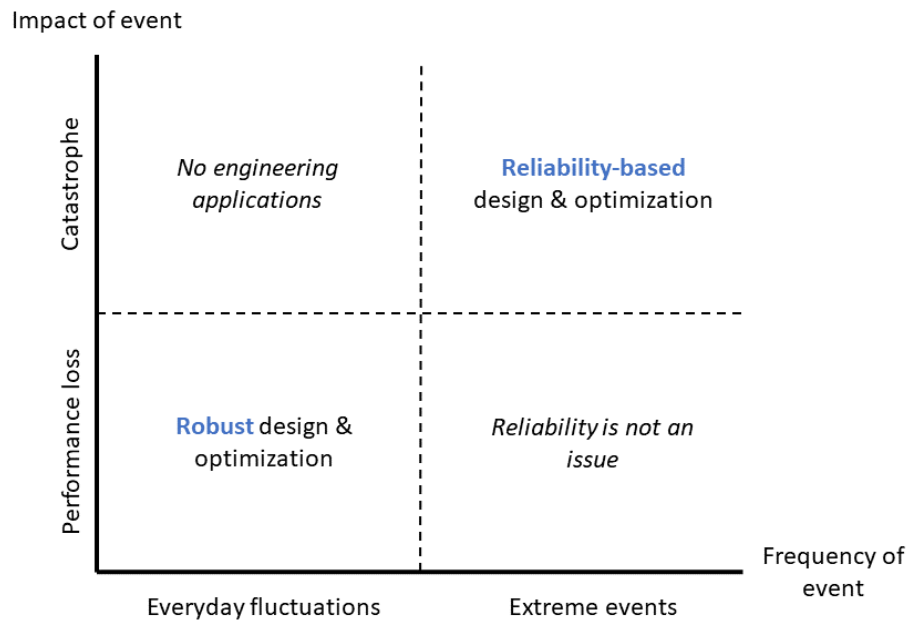


Figure 2.11: Uncertainty-based design problems (adapted from [49])

Another way of illustrating this classification is via probability density, as shown in Figure 2.12, where robust design is related to the event distribution in vicinity of the probability density function (PDF)'s mean, whereas reliability-based design is related to the event distribution at the PDF's tails.

Some of the main benefits of resorting to uncertainty-based design include increased confidence in analysis, reduced design cycle time, cost and risk, increased system/product performance, and increased robustness [48].

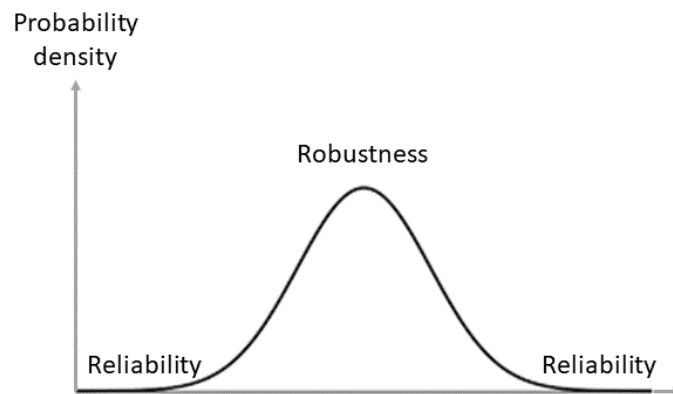


Figure 2.12: Robustness/Reliability in terms of probability density (adapted from [48])

In order to use uncertainty-based design methods, the design problem uncertainties must be characterised and managed, thus introducing the field of Uncertainty Quantification & Management (UQ&M). In the conservative and deterministic design approach, the traditional way of accommodating uncertainty is through using safety factors (also known as margins). It is a risk mitigation strategy which aims to design systems that are “*more capable*” and “*last longer than necessary*” [50]. Because of the resulting performance penalty and increased cost, Zang et al. [48] stated the critical need to develop new UQ&M methods and tools for applications to aerospace vehicle design. The other sectors where UQ&M is increasingly predominant are the automotive industry, the energy sector (nuclear, thermal power, or oil) and civil structures [51], [52].

Uncertainty Quantification (UQ) is not a new field, it actually dates back to the creation of the probability and statistics disciplines [53]. Smith [54] defines UQ as the “*synergy between statistics, applied mathematics and domain sciences required to quantify uncertainties in inputs and QoI [Quantities of Interest] when models are too computationally complex to permit sole reliance on sampling-based methods*”. Then, UQ is used to manage uncertainty by adjusting the design variables in order to fulfil decision-making criteria such as robustness and reliability. Typical uncertainty management methodologies include robust design optimisation, reliability-based design optimisation, and sensitivity analysis [3].

Industrial practice revealed, through ten cross-industry and cross-discipline case studies, that the objectives of any UQ&M treatment can be categorised into [51]:

- Understand: Understand and assess the influence of uncertainties in order to direct additional modelling or experiment.
- Accredit: Give credit to a given model or measurement method to ensure an acceptable level of fidelity, which could be achieved through calibration, model simplification, and validation.
- Select: Compare relative performance and optimise design, operation, and main-

tenance decisions.

- Comply: Demonstrate the system complies with requirements and regulatory threshold.

Therefore, the identification of the most important objective(s) of undergoing an uncertainty assessment, and of the related quantities of interest, is crucial in choosing the most relevant methodologies [51].

Sandia National Laboratories have been using a five-step methodology for UQ&M [55], which consists of the following:

1. Characterisation of uncertainty
2. Generation of sample
3. Propagation of sample through the analysis
4. Presentation of uncertainty analysis results
5. Determination of sensitivity analysis results

However, there was no generally accepted approach for UQ&M until the *European Safety, Reliability and Data Association (ESReDA)* Uncertainty Project Group proposed a generic UQ&M methodological framework derived from ten cross-industry and cross-discipline case studies [51], as represented in Figure 2.13. Since then, it has been increasingly adopted [3], [56]–[60].

Step 1: Problem specification

The first step is to specify the inputs, the model, the outputs, and the quantities of interest. The pre-existing model can be considered conceptually as a numerical function $G(u,d)$ linking uncertain inputs u and fixed inputs d to output variables of interest y , then decision-criteria are set based on the latter. In industrial practice, the classification of inputs as uncertain or fixed is not a matter of theory, but rather depends on the stage of the study as well as the decision-making process. Besides, the quantities of interest on the outputs depend on the objective of uncertainty assessment [51].

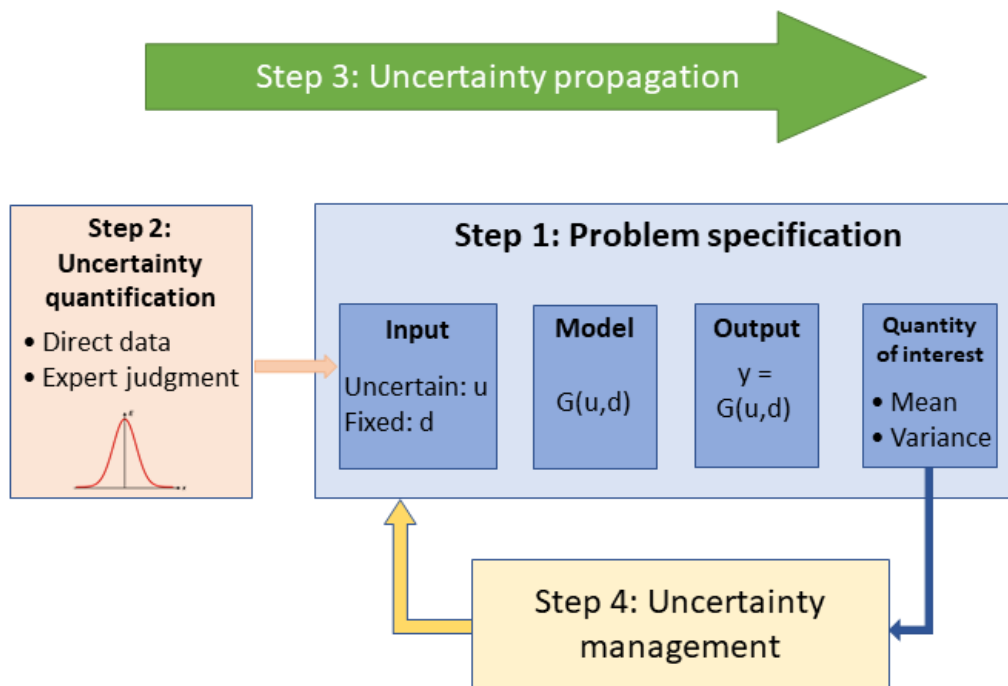


Figure 2.13: Generic UQ&M methodology (adapted from [60])

Step 2: Uncertainty quantification

Once the instances of uncertainty have been specified, these must be characterised (as reducible or irreducible) and quantified. Probability theory is the most commonly used technique to quantify uncertainty due to its relatively easy implementation [61]. Basically, a probability mass function (PMF – for discrete random variables) or a probability density function (PDF – for continuous random variables) is assigned to an uncertain variable to attribute a probability (likelihood) to each value the said uncertain variable can take. Practically, such a PDF (or PMF) can be fitted if sufficient data is available, and its model (e.g. Gaussian) can be selected depending on the characteristics of the uncertainty. However, little knowledge is available at the conceptual design stage, which impedes the selection of the right PDF model [61].

Step 3: Uncertainty propagation

The uncertainties quantified in Step 2 are then propagated through the analytical models to determine the uncertainties in the quantities of interest. This step is known as *uncertainty*

propagation (or *uncertainty analysis*). A variety of propagation methods exist, such as: Monte Carlo sampling methods [62], Polynomial Chaos Expansion [63], and Stochastic collocation [64].

Step 4: Uncertainty management

The design variables are refined to manage uncertainties, thus achieving the objective for uncertainty assessment. Some typical methodologies used in this step are *Robust Design Optimisation*, *Reliability-Based Design Optimisation*, and *Sensitivity Analysis* (which assesses the influence of input uncertainties on output variability). As illustrated in Figure 2.13, the generic UQ&M methodology is iterative in order to achieve the desired objective. For instance, sensitivity analysis is used to characterise inputs as uncertain or fixed. McManus and Hastings [50] developed a framework for uncertainty mitigation and exploitation in complex systems as an effort to support handling uncertainties in complex engineering systems. This was in response to an increasing need for “*robust*”, “*flexible*”, or “*evolutionary*” designs. According to McManus and Hastings, in an ideal world, “*methods would exist to collect knowledge of all the uncertainties facing a potential system, calculate all risks and opportunities implicit in these uncertainties, model the effects of all mitigation and exploitation strategies, and achieve all of the desirable system attributes*”. Thus, they proposed to break down the global problem into four, conceptually different categories: **Uncertainties**, which lead to **Risks (Opportunities)**, which are handled by **Mitigation (Exploitation)** strategies, thus leading to desired **Outcomes**. This is illustrated in Figure 2.14. Risks are defined as pathologies created by uncertainties and are often quantified as $(probability\ of\ uncertain\ event) \times (severity\ of\ consequences)$, whereas opportunities can arise from uncertainty (denoted as ‘+’ in Figure 2.14). Mitigations are techniques to minimise risk, whereas exploitations enhance opportunities. Outcomes are attributes which the decision-maker wants to achieve.



Figure 2.14: McManus and Hastings framework for uncertainty mitigation and exploitation [50]

Classification of Uncertainty

Classification of uncertainty is commonly used in the literature as a first step in modelling uncertainty, and the most common dualistic taxonomy is aleatory and epistemic uncertainties [4]. Aleatory uncertainty refers to intrinsic randomness that is considered to be irreducible, whereas epistemic uncertainty is due to lack of knowledge, thus making it reducible by acquiring more knowledge. Aleatory uncertainty could also be referred to as irreducible, objective, stochastic or primary, and epistemic uncertainty as reducible, subjective, cognitive or secondary. However, aleatory and epistemic uncertainty may not be completely disjoint since complete ignorance about an instance of uncertainty (i.e. epistemic uncertainty) could be reduced and reveal an aleatory aspect [46], whereas some may even argue that uncertainty is purely epistemological [65]. Nonetheless, this taxonomy remains useful to select the right modelling approach (e.g. probability theory for aleatory uncertainty). Various classifications of uncertainty in different fields exist in the literature [66]–[70], which reflect a lack of consensus among the different communities. These classifications are generally based on either *nature*, *level*, or *source* of uncertainty. *Nature* refers to the question: *Is uncertainty reducible by acquiring more knowledge, or is the outcome of the event random?*, whereas *Level* refers to the question: *To what extent is uncertainty quantifiable?* [71]. However, It must be noted that classification by *source*

has proved to be ineffective since enumerating all possible sources of uncertainty is bound to fail [72].

Another way of looking at uncertainty is from the perspective of design knowledge. From this perspective, Padulo and Guenov [73] proposed to classify uncertainty as either *about the problem* or *within the problem*. Uncertainty *about the problem* relates to the design problem formulation, where uncertainty arises when making assumptions to facilitate problem solving, defining the problem's boundaries or considering external factors affecting the design process. Uncertainty *within the problem* relates to problem-solving, where uncertainty manifests in the data and computational models used, as well as expert opinion. So far, the focus of researchers has been primarily on quantifying and managing computational uncertainty in problem solving (i.e. uncertainty *within the problem*). In contrast, much less attention has been devoted to managing uncertainty *about the problem*. Thus, this research focuses on uncertainty *about the problem*, and in particular assumptions.

Chapter 3

Literature Review

3.1 Introduction

As part of *Descriptive Study-I* of the research methodology, the purpose of this literature review is to improve the understanding of assumptions in the early design of complex systems, and highlight the limitations (and associated opportunities for improvement) of existing approaches to epistemic uncertainty management.

The review starts with assumption management (Section 3.2), where the definition of assumptions and their lifecycle, as well as assumption management in both industry and academia, are discussed. In Section 3.3, the Belief Revision literature is reviewed to discuss how formal philosophy has addressed assumption management. In Section 3.4, the *Knowledge Maturity* literature is reviewed to discuss how assessing knowledge in a project accounts for assumptions. Section 3.5 reviews margin allocation and management as it is the most common risk mitigation strategy in the context of epistemic uncertainty management. Finally, conclusions from the literature review are drawn in Section 3.6.

3.2 Assumption Management

3.2.1 Assumption: Definition and Lifecycle

According to Berner [74], there exist many definitions of the notion *assumption*. Furthermore, the distinction between closely related notions such as *axiom*, *premise*, or *presupposition* varies throughout the literature [74]. An *assumption* is commonly defined as “*a thing that is accepted as true or as certain to happen, without proof*” (*Oxford English Dictionary*), or “*something that you accept as true without question or proof*” (*Cambridge Dictionary*). A similar definition was proposed in the field of artificial intelligence, where an assumption is “*something which is accepted in the absence of evidence to the contrary*” [75].

According to the *Guidelines for Development of Civil Aircraft and Systems* (SAE ARP4754 A), assumptions are defined as “*statements, principles, and/or premises offered without proof*”, and “*assumptions may be used early in the development process as a substitute for more explicit knowledge that will be available later*” [76].

However, such commonly used definitions do not capture some essential characteristics of assumptions. Some of these characteristics were defined by Yang et al. [77] in the context of software development, where assumptions are:

1. subjective, i.e. can be seen as assumptions by some stakeholders, or design decisions by others;
2. related to other software artefacts, such as requirements or components;
3. dynamic, i.e. evolve with time; and
4. context-dependent, i.e. could be valid in one project, and invalid in another. As Brown [35] argued: “*Design reuse can violate assumptions, as conditions that were true, or were assumed to be true originally may no longer be the case in the new design context.*”

Another essential characteristic is that assumptions are inherently uncertain [74], [78], and the degree of confidence in making them varies based on the strength of background

knowledge. Therefore, the notion of *assumption* requires a definition that reflects its many characteristics. The following definition is thus proposed [79]:

Definition

An assumption is a context-dependent belief, with a varying degree of confidence, that requires validation to become knowledge.

An assumption bridges the gap between available knowledge and knowledge required to proceed with the design process.

The fact that assumptions evolve with time suggests the idea of a lifecycle. Ostacchini [80] proposed a simple assumption lifecycle model that is composed of three stages: An assumption is made at first, which then goes through changes, and ultimately may or may not fail (i.e. being invalidated). Note that Ostacchini's intention was to keep the lifecycle model simple, thus intentionally leaving out the case where the decision to invalidate an assumption is reversed. One observation that can be made is that, according to Ostacchini, invalidity is the only end state of an assumption. However, assumptions can also be validated and become knowledge. Another issue with the aforementioned model is that there lacks an in-depth discussion about the stage where assumptions change. Thus, some important questions such as *how changes during an assumption's lifecycle affect other elements of system design* remain to be answered.

The following sections are dedicated to the approaches to assumption management from the industrial (Section 3.2.2) and academic (Section 3.2.3) perspectives.

3.2.2 Industrial Perspective on Assumption Management

Representatives from industry expressed the importance of managing assumptions, as well as how challenging it can be for decision-makers to assess assumptions at the different product development stage-gates [26]. Matthiesen and Nelius [81] claimed in a recent study involving design practitioners that "*verifying assumptions on a system's function and behavior enhances the completeness and correctness of the analysis*". Another case

study [18] showed that documenting assumptions can reduce the number of design iterations, although the documentation process requires additional effort. In addition, a recent systematic mapping study on assumptions management in software development [11] listed numerous benefits of managing assumptions in the design and development of complex software systems, such as facilitating the verification of software systems and supporting the detection of requirements and design decisions mismatches. Furthermore, an exploratory study on assumptions management in software industry [82] revealed that the traceability of assumptions would be important in reviewing design decisions, and a formal (model-based) assumption management approach would facilitate maintenance and handover within projects as well as reduce costs due to the resulting reduced risks.

Systems engineering processes already support working with assumptions, albeit to a certain extent, by explicitly documenting and reviewing them during design [83]. According to Fieggen [84], effective assumption management in the context of Project Management Planning consists of:

- *Documenting assumptions;*
- *Testing assumptions;*
- *Attaching assumptions to tasks* (i.e. capturing assumption dependencies);
- *Highlighting high risk assumptions;* and
- *Managing assumptions,* through contingency planning.

In what follows, standards for systems engineering are reviewed, in a chronological order of publication, from an assumption management point of view.

EIA-632

According to the standard *EIA-632: Processes for Engineering a System* [85], technical evaluation for engineering a system consists of four processes: *Systems Analysis, Requirements Validation, System Verification,* and *End Products Validation*. In order to complete

Systems Analysis, design assumptions should be ensured to be valid and reasonable during the *Requirements Validation* process. The recommended practice is to “*record rationale for decisions and assumptions made*”, and then to “*analyze assumptions made with respect to defining system technical requirements to ensure that they are consistent with the system being engineered*”.

ISO/IEC 26702

Although the *ISO/IEC 26702: Standard for Systems Engineering* [86] recommends the validation of requirements, there is no explicit mention of the word “*assumption*”. However, recommended practice is expressed as follows:

- “*The enterprise shall capture pertinent design data in a repository for the evolving integrated data package and to provide a shared resource for the exchange and reuse of technical information.*”
- “*Design architecture definitions should be documented in the integrated repository, along with trade-off analysis results, design rationale, and key decisions to provide traceability of requirements up and down the architecture.*”

Thus, the standard recommends recording design rationale (which is supposed to include assumptions) within an integrated repository.

SAE ARP4754A

During system architecting, the standard recommends paying careful attention to the functional allocation process (i.e. allocation of aircraft functions to systems) and its underlying assumptions, where “*assumptions that are made in the course of this process become a vital part of the overall system requirements package and are subject to the same validation activity as are other requirements*” [76]. This implies that assumption management is restricted to the Requirements Domain only, whereas other standards (e.g. ISO/IEC/IEEE 15288:2015 and NASA SP-2016-6105 Rev2) recommend considering the decisions made

and their analysis as well. Such standards are discussed further in this section.

Another important consideration is when reusing existing certificated systems and components in new or derivative aircraft. Although such systems and components are mature, they do not necessarily meet the new system's requirements (recall that assumptions are *context-dependent*). Therefore, “*any derived requirements, assumptions, compatibility of the interfaces and the operational environment should be validated as well*” [76]. Once requirements are captured, alongside their underlying assumptions, the standard recommends for requirements validation that “*an independent reviewer should challenge the assumptions and interpretations of captured requirements with the requirement originator, ideally as they are being captured, in order to ensure that these requirements have the same meaning for the requirement originators and recipients*” [76]. It is also recommended for assumptions to be validated at each level of the requirements definition hierarchy. The process of validating assumptions ensures that these are (i) *explicitly stated*, (ii) *appropriately disseminated*, and (iii) *justified by supporting data* [76]. Such process may include reviews, analyses and tests, where the selected approach to manage assumptions during the design process is recommended to be defined within the validation plan.

INCOSE-TP-2003-002-04

The *INCOSE Systems Engineering Handbook* (INCOSE-TP-2003-002-04) recommends, as part of Requirements Management, to “*maintain throughout the system life cycle the set of system requirements together with the associated rationale, decisions, and assumptions*” [5]. Additionally, the assumptions are to be communicated through a *risk report*. However, this standard considers the Requirements Domain only, as opposed to the standards reviewed next.

ISO/IEC/IEEE 15288:2015

Similar to the INCOSE-TP-2003-002-04, the *Systems and Software Engineering – System Life Cycle Processes* standard (ISO/IEC/IEEE 15288:2015) also recommends recording

assumptions underlying the Requirements Domain, where “*the system requirements are recorded in a form suitable for requirements management through the life cycle. These records establish the system requirements baseline, and include the associated rationale, decisions and assumptions*” [16]. However, contrary to the INCOSE-TP-2003-002-04, ISO/IEC/IEEE 15288:2015 stipulates that the implementation of the *Decision Management* process is considered successful when the assumptions are identified. Thus, recommended practice regarding assumptions recording extends to the architectural decisions (i.e. Functional and Logical Domains).

NASA SP-2016-6105 Rev2

The *NASA Systems Engineering Handbook* (NASA SP-2016-6105 Rev2) recommends capturing assumptions as part of *Technical Requirements Definition*, *Logical Decomposition*, and *Design Solution Definition* processes [17]. Thus, both requirements and architectural decisions are concerned by recording underlying assumptions. Another important consideration is *Decision Analysis*, where the output of this process supports decision-making regarding selection amongst competing alternatives, under epistemic uncertainty. Thus, it is “*critical to understand and document the assumptions and limitation of any tool or methodology and integrate them with other factors when deciding among viable options*” [17]. Since computational models are used to analyse competing alternatives, and thus influence decision-making to select amongst these alternatives, recommended practice regarding assumptions recording extends to the Computational Domain as well.

Overarching Properties

There is an increasing need to reduce the time and cost of current certification processes without compromising safety [87]. Thus, the Federal Aviation Administration (FAA) launched initiatives to this end, which led to an approach called *Overarching Properties* [88].

Overarching Properties can be defined as a “*sufficient set of properties for making ap-*

proval decisions”, such that when approval is sought for an entity to be used on an aircraft, and that entity is shown to possess the so-called Overarching Properties, approval can then be granted [88].

The three Overarching Properties can be summarised as follows [87]:

1. **Intent:** “*Are we sure that the development team has understood what they were supposed to develop?*”
2. **Correctness:** “*Are we sure that the development team has properly implemented what they understood?*”
3. **Acceptability:** “*Are we sure that implementation choices made during the development process do not invalidate the original safety assessment?*”

A European research project called *Re-Engineering and Streamlining the Standards for Avionics Certification (RESSAC)* [87] conducted case studies in using Overarching Properties. According to the RESSAC working group, one of the evaluation criteria for the *Intent* property is the “*description of how any assumptions concerning the desired behaviour are shown to be relevant, complete, and accounted for in the DIB*” [87], where *DIB* refers to *Defined Intended Behaviour*. However, no method has been published to support such evaluation with respect to assumptions.

Discussion

Some of the reviewed standards (i.e. EIA-632, SAE ARP4754A and INCOSE-TP-2003-002-04) were found to restrict assumption management to the Requirements Domain only, thus making it a sub-activity of requirements management. However, explicit recommendations were identified from ISO/IEC/IEEE 15288:2015 and NASA SP-2016-6105 Rev2 to capture and validate assumptions underlying requirements, architectural decisions and decision analysis (i.e. R-F-L-C Domains). It is argued in this thesis that assumption management should be carried out beyond the Requirements Domain, where it would interact with the Functional, Logical and Computational Domains as well. One

reason to consider assumption management outside the Requirements Domain is the fact that assumptions and requirements are actually two different concepts [77]. An assumption is made to deal with uncertainty by tentatively filling a knowledge gap, whereas a requirement is derived from a stakeholder's need. Due to this fundamental difference, even *validation* has a different meaning in both assumption management and requirement management. Validating an assumption means it is proven to be true (recall that an assumption is believed to be true without proof), whereas validating a requirement means that the stakeholder's need is satisfied [16].

Performing assumption management activities in industry is still a major challenge. This is mainly due to the required effort, the lack of formal methods and tools to support such activities, or even the practitioners not being aware of making assumptions [11], [89]. In fact, an exploratory study on assumptions management in software industry [82] revealed that architects use generic tools such as *Microsoft Office* (mainly *PowerPoint*, *Word*, *Visio* and *Project*) or the *IBM Rational Software Architect Designer* to manage assumptions. Another important factor is the increasingly high number of assumptions, especially in innovative design. Consequently, a trade-off between the benefits of assumption management and dealing with the related difficulties becomes necessary [11].

The next section reviews the academic literature with respect to approaches to satisfy the industrial need for formalised (model-based) assumption management.

3.2.3 Academic Perspective on Assumption Management

The industrial perspective on assumption management revolves around the idea of documenting explicit assumptions in textual format. However, a recent study that surveyed practitioners has shown that simply documenting explicit assumptions has little benefit [18]. In the same study, the authors argued there is a practical need for formal methods and tools to manage assumptions.

Although model-based assumption management would require some additional effort early on, it would still benefit the entire project. According to Anzengruber *et al.* [89],

assumptions are necessary anyway to progress in the project due to the fact that acquiring information to fill the knowledge gap could be either too expensive, too time consuming, or not even possible early in the process. Additionally, more formal approaches would make assumptions traceable and enable the identification of entry points of iterations that result from changes in assumptions [89]. Therefore, a trade-off is required between additional effort early in the design process for formal assumption management, and the risk of (expensive) changes later in the process.

In software development, where assumptions are the main source of uncertainty as highlighted by the *Principle of Software Uncertainty* [90], assumption management is relatively more developed and has evolved faster than in other fields. There has been a recurrent issue of assumptions being unrecorded and part of the implicit rationale behind software design and development [91]. In fact, the *Principle of Software Uncertainty* justifies the need for effective rationale management, including assumption capturing and management [90]. Lehman and Fernández-Ramil [90] argue that “*practical approaches are needed for assumption management*”, and that there is room for extended research on methods and processes to capture assumptions as part of the design rationale. One of the earliest attempts in this context is the *Assumption Management System* proposed by Lewis *et al.* [92], which extracts assumptions from Java code (where assumptions are written using XML) and records them into a repository. This method has the benefit of enabling browsing and searching for assumptions [93]. However, it is limited to the implementation level, i.e. after the architecture is defined. Tirumala proposed an assumption management framework [94], which is based on formal assumption capturing. Such formalism allows to encode assumptions as part of the software architectural components, thus allowing to automatically check for mismatch between assumptions and their associated guarantees [93]. Furthermore, policies can be set on assumption selection and validation, meaning that “*a relevant subset of assumptions can be validated or flagged as invalid automatically as the system evolves*” [94]. Such functionality has practical value as it could lead to reducing the cost of managing assumptions. However, this framework

considers assumptions underlying software components only, whereas other architectural domains are also important to consider as discussed in Section 3.2.2. According to Yang *et al.* [95], software architects are usually unaware of which assumptions were made, their state (valid or invalid) and their dependencies. Furthermore, they argue that the previously proposed approaches to assumption management may not be suitable for architectural assumptions. This would be due to the fact that, in the context of software design, “*architects and designers are the major stakeholders in assumptions management*” [95], and their needs could differ from other stakeholders at different stages of software development (for instance project managers or requirements engineers). To this end, Yang *et al.* [77] proposed an iterative process for architectural assumption management in software development, which consists of four core activities:

1. **Architectural Assumptions Making:** refers to making new assumptions, and identifying implicit assumptions. The inputs to this activity are all *software artifacts*, including requirements and architectural design decisions. The outputs of this activity are *undocumented assumptions*.
2. **Architectural Assumptions Description:** refers to describing and recording assumptions, where the core elements of the description are the assumption’s ID, name, state, rationale, and related assumptions and other software artifacts. The inputs to this activity are *undocumented assumptions*. The outputs of this activity are *documented assumptions*.
3. **Architectural Assumptions Evaluation:** refers to assessing the validity and consistency of assumptions. The evaluation of an assumption can result in either: (a) no problem being identified; (b) the assumption being invalid; (c) the assumption being valid; or (d) it is not possible to evaluate the assumption due to, for instance, lack of information. The inputs to this activity are *documented assumptions*. The outputs of this activity are *evaluated assumptions* (e.g. which assumptions were found to be invalid).

4. **Architectural Assumptions Maintenance:** refers to adapting assumptions to fit the new context of software development, which includes eliminating outdated/invalid assumptions, and modifying conflicting assumptions. The issues identified in evaluation (e.g. an assumption being evaluated as invalid) are addressed in maintenance (e.g. the invalid assumption is modified to fit the new context). Then, the description is updated to capture changes made during maintenance. The inputs to this activity are *evaluated assumptions*. The outputs of this activity are *maintained assumptions* (e.g. invalid assumptions that have been modified).

However, the process proposed by Yang *et al.* [77] suffers from some limitations: (a) there is no mention of how the uncertainty in assumptions is assessed, and how such uncertainty evolves throughout software development; (b) there is no mention of how assumptions are evaluated; (c) there is no description of what happens to other related assumptions/artifacts when there are changes in an assumption; and (d) the process is not described according to a standard (e.g. ISO/IEC/IEEE 24774:2021 [21]).

Regarding *Architectural Assumptions Description*, Yang *et al.* [95] proposed the *Architectural Assumption Documentation Framework (AADF)*. The particularity of the AADF is that it captures the dependencies both amongst assumptions (what they call *Relationship Viewpoint*), and between assumptions and other artifacts (i.e. requirements, architectural design decisions, software components, and risks) (what they call *Tracing Viewpoint*). Such capability addresses the importance of considering both dependencies between assumptions and the design elements [35], [96], and dependencies amongst assumptions [97]. Table 3.1 describes the types of dependencies in the AADF, where an *artifact* could be either a requirement, an architectural design decision, a software component, or a risk.

A closer look at the types of dependencies in Table 3.1 led to the observation that these types are not mutually exclusive, which may lead to unnecessary effort and confusion. *Strengthening* is actually a characteristic of *Constraint* and *Causality* rather than a distinct type because, when A constrains/causes B, a higher likelihood of A being valid

Type	Description
Conflict	When an assumption conflicts with another assumption or artifact, they are considered as mutually exclusive
Con- straint	If an assumption or artifact A constrains assumption B, and A changes, then B would need revision
Causality	Artifact or assumption A causes the making of assumption B
Strengthening	Artifact or assumption A increases the likelihood of assumption B being valid
Weakening	Artifact or assumption A decreases the likelihood of assumption B being valid
Alternative	Assumption A can substitute assumption B

Table 3.1: Types of assumption dependencies according to Yang *et al.* [95]

would mean an increase in the likelihood of B being valid. Similarly, *Weakening* is a characteristic of *Conflict* in the sense that, if A conflicts with B, a higher likelihood of A being valid would mean a decrease in the likelihood of B being valid. This is due to the fact that, when two elements are conflicting, only one of the elements (at most) can be valid. Furthermore, *Constraint* can be seen as a characteristic of *Causality* in the sense that, if A caused B, and A changes, then B may need revision. Another issue is with the *Alternative* type, which actually refers to an instance of assumption duplication. Rather than keeping it as a dependency, this should be resolved by maintaining a unique instance of the assumption in order to maintain consistent assumption records. Therefore, it is argued here that *Strengthening*, *Weakening*, *Constraint* and *Alternative* should not be considered as types since they would unnecessarily increase the effort to capture dependencies, and also potentially confuse practitioners.

In contrast with purely software systems, there is a need to consider assumptions underlying computational models, which are needed to assess physical architectures. In fact, the identification and management of modelling assumptions has been recognised as a “*serious practical barrier and bottleneck that restricts simulation approaches*”, where there is a need to “*track the reliance on assumptions throughout the modeling, simulation, and result-reporting process*” [98]. To this end, Eriksson *et al.* [98] proposed to explicitly link assumptions to elements of an epidemiological simulation (e.g. disease and inter-

vention models), in order to support the interpretation of simulation results. However, it is unclear how assumptions themselves are modelled, where the authors only mentioned that assumptions are textually represented. This could be interpreted as simply attaching notes, written in natural language, to the corresponding models. Sadlauer *et al.* [18] proposed to use a graph that links assumption nodes with design parameter nodes. However, their approach is restricted to assumed values of design parameters. Furthermore, since the computational models are not captured, parameters dependency remains implicit, and assumptions underlying the computational models are not captured. Another approach was proposed by Basu and Blanning [99] where models are represented as metagraphs. Metagraphs are graph-theoretic constructs that “combine the properties of digraphs and hypergraphs in that edges may contain more than two elements and specify the direction of relationships.” [99]. For example, edge e_3 in Figure 3.1 connects a set of two nodes (b and c) to a single node (d).

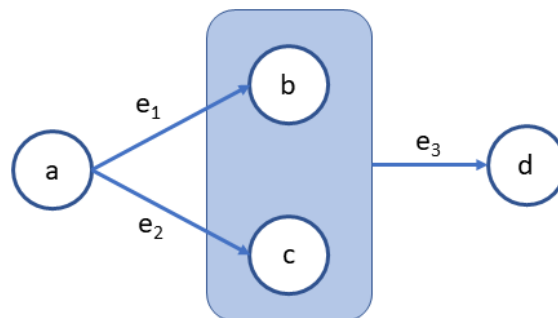


Figure 3.1: Example of a metagraph (adapted from [99])

In the method proposed by Basu and Blanning [99], inputs and outputs are represented as nodes, which are then connected by edges representing the corresponding models. Representing models as edges makes them implicit which means that, similar to [18], there is no explicit information about how parameters are related and what assumptions underlie the models. Modelling assumptions are then captured as propositional statements, which is only applicable to assumptions that can be expressed in parametric form. Furthermore, assumptions related to the decisions made (for instance architectural decisions) are not considered. Therefore, there is a need for an approach that would consider both decision-

making (i.e. architecture definition) and decision analysis (i.e. architecture assessment). Another important issue is that, although assumptions have varying degrees of confidence, none of the aforementioned approaches to assumption management included an assessment of the uncertainty inherent to both quantitative and non-quantitative assumptions. Sadlauer *et al.* [18] proposed a simplified, early concept (i.e. their concept was not validated) for confidence assessment. However, their concept is restricted to assumed values of design parameters, and thus excludes assumptions associated with requirements, functions, components and computational models. In the field of risk analysis, the results of quantitative risk assessments depend on the assumptions made, which requires communicating such uncertainty in the risk assessment [97]. Berner [74] argues that not all uncertainties can be quantitatively described, for instance the assumptions made or the models used can be wrong. Thus, Berner and Flage [97] proposed a semi-quantitative approach to assess assumption uncertainty, which uses the strength of background knowledge as an indication.

As defined in Section 3.2.1, an assumption is essentially a belief which needs revision as new knowledge is acquired. It is thus necessary to review the Belief Revision literature in order to examine existing approaches and identify their limitations.

3.3 Belief Revision and Consistency Maintenance

3.3.1 Introduction

As discussed in Section 3.2.1, assumptions are necessarily made at early-stage design to deal with epistemic uncertainty. In fact, such assumption-based reasoning is a typical human reasoning pattern where “*conclusions are based on knowledge of what is typical, or usual, when there is not information to conclude otherwise*” [100]. However, according to Mason [100], “*the difficulty in implementing assumption-based reasoning is that default assumptions and their consequents are tentative. Facts that are added might later be thrown out*”. Thus, the beliefs of the reasoning agent are subject to revision.

Belief Revision can be defined as a “*research area in formal philosophy that makes use of logic to produce models of how human and artificial agents change their beliefs in response to new information*” [101]. Such new information could either fill knowledge gaps, or correct an incorrectly held belief by the agent, but in either case the new information must be added to the agent’s set of beliefs [102]. However, this process of incorporating new beliefs is neither arbitrary nor trivial, as described by Delgrande *et al.* [102] in the following:

“Assume that the new information is given by a formula ϕ . Then, if the goal of revision is to incorporate this information, following the process of revision, ϕ should appear among the agent’s beliefs. One possibility would be to simply add ϕ to the agent’s beliefs; in such a case, ϕ would indeed be in the resulting belief set. However, ϕ might conflict with the agent’s prior beliefs, and if this was the case, the agent would fall into inconsistency. So, another reasonable principle is that an agent’s beliefs should be consistent after revision by a formula ϕ (unless ϕ itself is inconsistent). This in turn requires that an agent may also have to remove some beliefs in a revision. One possibility in this case would be to remove all of the agent’s prior beliefs. However, this is clearly far too drastic, and so one would want to stipulate that in some fashion the agent retains as many of its old beliefs as consistently possible.”

The research area of Belief Revision arose in the 1970s, where multiple approaches have been proposed [101]. This field is usually recognised as being initiated by Jon Doyle [103], who proposed the first domain-independent belief revision system called the *Truth Maintenance System* (TMS) [104]. The TMS was then followed by several belief revision systems, including the *Multiple Belief Reasoner* [105], the *Reasoning Utility Package* [106], and the *Assumption-based Truth Maintenance System* (ATMS) [107].

Belief revision systems are useful when there is a need to choose among alternatives [103], as is the case with engineering design. A designer, as a human agent, may have to

make assumptions in order to fill the knowledge gap. The designer may then infer other architectural artifacts (e.g. derived functions or solutions) based on these assumptions, and still further decisions from the aforementioned inferred artifacts. At some point, the accumulated architectural beliefs may become inconsistent due to some contradictions. It then becomes necessary to revise/retract some assumptions or architectural decisions. Thus, a belief revision system can be used to support designers in making such changes, especially when there is not only a significant number of architectural artifacts and assumptions, but also the numerous dependencies within a system architecture.

It is thus crucial to not only capture assumptions in an explicit way, but also check that these assumptions are consistent with the former beliefs. In fact, according to Graves and Bijan [108], “*many of the really expensive mistakes occur early in the analysis and design process and are the result of not capturing assumptions and checking consistency as the design process develops*”. Therefore, consistency maintenance can potentially reduce rework and shorten design cycles [108]. However, the challenge in this context is the integration of formal methods with industrial processes and tools so that they can be applied by practitioners [108].

Inconsistency can be categorised as either *syntactic* or *semantic*. Syntactic inconsistency is related to the *structure* of the rationale (which entails looking for missing information), whereas semantic inconsistency is related to the *meaning* (which entails looking into the content of the rationale) [109]. It is important to note that it is not always possible to resolve an inconsistency as soon as it is identified. According to Burge *et al.* [110], “*tolerating inconsistency may be necessary if inconsistencies are too expensive to repair, if the information required to resolve the inconsistency is not known at the current stage of the development, or if it is too early in the process to make the design decisions required for resolution*”. Thus, there are situations in practice where the resolution of inconsistency has to be deferred to a later time during product development.

3.3.2 Extra-Logical Factors in Belief Revision

Logic alone is not sufficient to determine which beliefs to give up as a way to restore consistency [111]. Thus, some extra-logical factors are needed. One of such factors is the fact that some beliefs are more important than others, meaning that the least important ones should be given up first if necessary [111]. As an example, safety requirements would be considered as more important than other types of requirements in safety-critical systems. This principle of ranking beliefs is known as *epistemic entrenchment* [112]. Beliefs can be ordered, for instance, according to their probability [113], their possibility [114], or how reliable their sources are [115].

Another extra-logical factor is the minimisation of information loss when giving up beliefs [111]. According to the notion of *informational economy*, the preference is towards retaining as many of the prior beliefs as possible during belief revision [102]. However, since information loss cannot be easily quantified, a practical solution remains to give up beliefs with the lowest degree of entrenchment (i.e. the least important) [111].

3.3.3 Belief Revision Theories

Two Approaches to Belief Revision

Two approaches to belief revision emerged as the research field was maturing [116]. The first approach, called *foundations*, emerged as a way to update databases, at a time where Artificial Intelligence was developing [116]. In the *foundations* approach, “*a belief holds as long as the system finds a justification in its support*” [117]. A pioneer of this approach is Jon Doyle with his TMS [104].

In the second approach, called *coherence*, “*a belief holds as long as it is coherent with the remaining beliefs of the system*” [117]. In this approach, one does not keep track of justifications [111], as opposed to *foundations*. The most influential work in relation with this approach is the *AGM* model, named after its three authors Carlos Alchourrón, Peter Gärdenfors, and David Makinson [118]. The *AGM* model refers to a set of postulates that

belief revision operators should satisfy [116].

The difference in whether justifications are considered would lead to believing that the two approaches are mutually exclusive, when in reality it is not necessarily the case. del Val [119] actually believes that both approaches are equivalent, whereas Doyle [120] argues that the TMS already incorporates the foundations of *coherence*, while offering a practical means of *mechanising* Belief Revision. Furthermore, there has been an attempt at combining the two approaches, resulting in Galliers' theory of *Autonomous Belief Revision* [115]. According to Gärdenfors [111], Galliers' theory of *Autonomous Belief Revision* "adds a foundational aspect to the belief revision model by working with assumptions of various kinds and justifications for the assumptions", where he gives as an example to clarify: "the endorsement of an assumption depends on whether it is communicated by a reliable source or a spurious source. In this way, her model of autonomous belief revision is a mixture of coherence and foundationalism".

Choosing a Belief Revision approach in the context of engineering design should prioritise practicality. Therefore, in what follows, the focus will be on the TMS as a practical method to enable Belief Revision, and also due to the fact that justifications are necessary in design rationale and in providing traceability. Another reason is that, while the AGM model represents a useful abstraction of the process of Belief Revision, its implementation for practical use is challenging [121]. Furthermore, the lack of justification in the *coherence* approach remains an issue nowadays, as Haas [122] identified shortcomings of the AGM model, which can all be attributed to its neglect of justification. Haas concluded then that these shortcomings can be resolved if justification is included.

Truth Maintenance Systems

A TMS can be defined as a "collection of procedures and data structures (a class of algorithms) used for accomplishing belief revision" [100], with the objective of maintaining the set of believed inferences during the reasoning process. The role of the TMS is to address the problem solver's need to update beliefs while maintaining consistency, so that

the problem solver can assume a “*contradiction-free database of beliefs*” [100].

In terms of implementation, the TMS is considered as a separate component, which is interfaced with the problem-solver, as illustrated in Figure 3.2. Essentially, the problem-solver draws conclusions based on its set of beliefs, whereas the TMS maintains the consistency of the belief set [100].

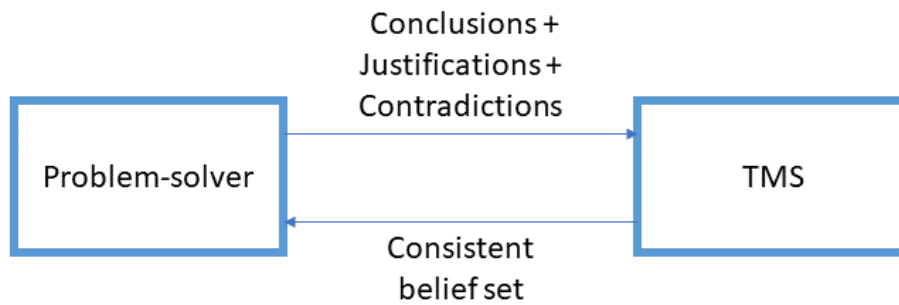


Figure 3.2: Problem solver-TMS interfacing (adapted from [123])

The problem-solver provides the TMS with the following information: (a) conclusions drawn, (b) their justifications, and (c) the encountered contradictions during the reasoning process. The reason for the latter is that, according to Doyle [120], “*since TMS has no knowledge of the meanings of nodes, the reasoner must tell it which nodes represent contradictions*”. Conversely, the TMS uses that information to label each element of the belief set as either *believed* or *disbelieved*, so that the problem-solver can query the TMS regarding which sentences to *believe* [100].

There are two broad categories of TMS-based approaches [100]:

1. **JTMS**: Justification-based (or single-context) TMS
2. **ATMS**: Assumption-based (or multiple-context) TMS

The ATMS, which was developed by de Kleer [107], contrasts with the JTMS in that it captures information about the default assumptions underlying the conclusions. Each believed sentence is explicitly linked to its assumptions. Thus, such difference in recording enables the ATMS to maintain multiple contexts (or environments) of belief during the reasoning process, whereas the JTMS can maintain a single context only [100].

An ATMS consists of two types of nodes: *assumptions* and *deduced nodes*. The *deduced nodes* represent the derived facts, which are then linked by the ATMS to their underlying *assumptions* [123]. *Premises* are by default valid in all contexts, and labelled by the ATMS with an empty assumption set $\{\}$. Assumptions underlying a derived fact are added to its label. If that label has an empty assumption set, this is interpreted as the derived fact not having any justification, thus relabelling it as *disbelieved* [100].

The contradictions, which are detected by the problem-solver and communicated to the TMS, are syntactic in nature. For a contradiction to be detected, both some fact A and its negation (i.e. A and $\neg A$) must explicitly belong to the belief set. This is interpreted as believing that A is both true and false, hence the contradiction. When the contradiction is discovered by the problem-solver and communicated to the TMS, the assumptions associated with the contradictory facts are gathered to form a new set called the '*NOGOOD*' assumption set. The '*NOGOOD*' set is then cached so that no further conclusions can be drawn from the inconsistencies [100].

3.3.4 Limitations

One of the identified limitations in Section 3.2.3 was the lack of assessing the uncertainty inherent in assumptions. It turns out that the same limitation applies to the ATMS, where the assumptions are considered as either true or false [124]. Thus, an ATMS cannot address varying degrees of confidence in assumptions.

A second limitation is that TMSs must be used in conjunction with an artificial problem-solver [123], which limits their applicability to support non-repetitive and creative activities undergone by human agents (such as engineering design). Furthermore, TMSs maintain logical consistency only [120], which only prevents belief sets from containing both a fact and its negation. This means that they look at the structure of the rationale only rather than its meaning as well. According to Doyle [120], the founder of TMS, "*many discussions of truth maintenance misrepresent the truth by claiming that TMS maintains consistency of beliefs*". This issue is not unique to TMS but rather characterises computer

programs in general, since they simply follow a set of instructions (i.e. syntactic entities) without reasoning about the meaning. In contrast, semantic inconsistencies would require human agents to identify them. Jakob *et al.* [125] give as an example “*a knife that is considered to be sharp and dull at the same time. Although this contradiction is obvious for humans, robots often lack the knowledge that the predicates hasProperty(knife, sharp) and hasProperty(knife, dull) contradict themselves*”. Thus, the detection of such inconsistencies cannot be fully automated. Therefore, an interactive approach involving the practitioner (as a human reasoner) would be more suitable in the context of engineering design.

A third limitation is that TMSs do not capture dependencies between assumptions. In a JTMS, an assumption is dependent on its negation only [100], i.e. for an assumption to be *believed*, its negation must be explicitly *disbelieved*. Whereas in an ATMS, an assumption cannot have a dependent node [126]. Therefore, a conflict relationship between assumptions, as suggested by Yang *et al.* [95], cannot be captured.

A fourth limitation is that an ATMS does not tell which assumption is problematic, but rather captures the inconsistent set of all assumptions that led to the contradiction, then searches the label of each node to check whether it contains that inconsistent set of assumptions. Extra-logical factors, as discussed in Section 3.3.2, are thus needed to prioritise assumptions and inform which ones should be revised or removed.

3.4 Knowledge Maturity

As previously stated, assumptions are made to fill knowledge gaps, thus enabling to progress in the design. The status of such knowledge gaps must be assessed as part of gate reviews (see Section 2.2.1 for an overview of the Stage-Gate process). Knowledge Maturity can be defined as a state that refers to how close knowledge at a decision gate is to knowledge needed for progressing in the development [26]. According to Johansson *et al.* [127], Knowledge Maturity is about, on one hand, “*providing design teams with*

insights about which areas they have sufficient knowledge and information in", and on the other hand, *"highlighting the areas where more knowledge is needed"*. Action can then be taken to revise areas of insufficient or poor quality knowledge. Thus, the goal of Knowledge Maturity is *"being confident in the fact that the decisions are made from a solid knowledge base"* [127].

Johansson *et al.* [128] developed a *Knowledge Maturity Scale* (Table 3.2) through workshops with the participation of collaborators from the aerospace industry as part of the VIVACE project [129]. These workshops revealed that practitioners consider the quality of the knowledge used at any decision point to consist of three dimensions: *input*, *method* and *expertise*. The *input* dimension refers to the reliability and trustworthiness of information entering the design activity. The *method* dimension refers to the confidence in the means used to process the input. In fact, such confidence varies from one method to another. For example, *"a full-scale prototype test might give the decision makers more confidence than a hand-based calculation of an idealized model, which is a lot cheaper"* [128]. Thus, it is important to select methods that provide sufficient confidence at any given decision point. The *expertise* dimension indicates the involvement of experts, which can be crucial especially at early design stages where epistemic uncertainty increases the reliance on intuition. Johansson *et al.* [128] argue that the aforementioned three dimensions offer a practical way to assess knowledge assets, rather than analysing their face value which would require a significant degree of abstraction.

In addition to the quality of knowledge, which is characterised by its aforementioned three dimensions, Johansson *et al.* [128] and their industrial collaborators also deemed important to assess the maturity of the process by which the knowledge is managed within the organisation. Knowledge management can be defined as *"the process of applying a systematic approach to the capture, structuring, management, and dissemination of knowledge throughout an organization to work faster, reuse best practices, and reduce costly rework from project to project"* [130]. The level of maturity of the knowledge management process differs across organisations. Such level can be assessed through *Knowledge*

5	Excellent	The content and rationale are tested and proven. They reflect a known confidence regarding, for instance, risks. The procedure to produce the content and rationale reflects an approach where verified methods are used and where workers continually reflect and improve. Lessons learned are recorded.
4	Good	<i>Intermediate</i>
3	Acceptable	The content and rationale are more standardised and defined (i.e. documented and formalised). There is a greater extent of detailing and definition (compared to previous level). The procedure to produce the content and rationale is more stable (compared to previous levels) with an element of standardisation and repeatability.
2	Dubious	<i>Intermediate</i>
1	Inferior	The content and rationale are characterised by instability (e.g. poor/no understanding of knowledge base). The procedure to produce the content and rationale is dependent on individuals, and formalised methods are nonexistent.

Table 3.2: Knowledge Maturity Scale [128]

Management Maturity Models (KMMM), which describe “*steps of growth that can be expected by the organization to reach its knowledge management development*” [131]. A typical KMMM consists of maturation levels, where each level indicates the existing capabilities in enabling knowledge management within the organisation [131]. Some of the most widely used KMMM are the *Infosys Model* [132], APQC’s *Knowledge Management Capability Assessment Tool* [133], and the *Siemens AG KMMM* [134].

During a Stage-Gate process, assessing assumptions constitutes a challenge for decision-makers at the gates, where assumptions could be mistaken for proven knowledge if not properly managed [26]. According to Johansson [26], assumption management is one of the key aspects of knowledge maturity, where “*a focus on knowledge maturity forces assumptions to be assessed regarding the origin and what is the confidence level in them*”. Although assumption management plays a crucial role in Knowledge Maturity, no method has been proposed to assess Knowledge Maturity with an explicit consideration of assumptions.

Once risk is analysed with support from Knowledge Maturity assessment, the risk can

then be managed by using mitigation strategies such as design margins.

3.5 Margin Allocation and Management

3.5.1 Overview

Traditionally, margins have been used as a risk mitigation strategy to deal with epistemic uncertainty inherent in engineering design. In fact, margins play a crucial role in managing engineering change and design iteration because, as margins are used up, changes can propagate through the design which in turn can lead to undesired iterations [135].

Different terms and definitions for the notion of *margin* exist. Eckert *et al.* [136] defined a margin as the extent to which the value of a parameter exceeds what is necessary for requirements fulfilment. A margin can in effect both account for the uncertainties and provide flexibility to accommodate future changes [135]. Thus, the system becomes more capable and lasts longer than necessary [50].

However, there is a necessary trade-off between risk mitigation and performance. As illustrated in Figure 3.3, if the assigned margin is too low, there is an increased risk of major re-design, which would in turn impact the cost and schedule. In contrast, a margin that is too high would lead to unnecessary weight, therefore reducing performance and marketability.



Figure 3.3: Margin balance (adapted from [137])

The most common approach to quantify margins is to use a deterministic representation, based on industrial standards, historical data or designers' intuition and experience [138]–[140]. Such formulation can be expressed in terms of worst-case estimates (*WCE*) and current best estimate (*CBE*) [139], as formulated in Equation 3.1.

$$\%margin = \frac{WCE - CBE}{CBE} \times 100 \quad (3.1)$$

However, such an approach has proved to be conservative and leading to over-designed products, thus resulting in performance penalty and increased cost. Furthermore, margins are added by different stakeholders without a unified way to allocate them and assess their impact on the design [141]. This is an ongoing issue [135], where a need has been identified to develop a tool that tracks margins along with the rationale underlying their change, in addition to the need to develop mathematical models for margin management. A mathematical model has been attempted by Touboul *et al.* [142], which considers the margin as a distance measure from the system state to some state with non-acceptable risk, while translating stakeholders requirements and constraints into sets of equations and inequalities. Although formalisation is important to maintain consistency in margin setting, and perhaps enable margin optimisation, the industrial applicability and practical usefulness of the model proposed by Touboul *et al.* still remain to be validated.

Another issue is margin redundancy, which has been discussed by Gale [143] in the context of ship design. According to Gale, margin redundancy is one of the main challenges in practice, which can occur due to lack of communication as different collaborators assign margins individually. This situation can then lead to unnecessary over-design. This can also be explained by the fact that within the same organisation, collaborators use different terms for (or even have different definitions of) the notion of *margin* [135], [144].

3.5.2 Margin as a Change Absorber

As mentioned earlier, margins play an important role in the management of engineering change, and that is because margins are capable of absorbing change. Engineering change can occur, for instance, as a response to new or changing requirements, unexpected interactions, integration issues, or simply design mistakes. Note that change is only required when no sufficient margin is left to absorb it [135].

The main challenge is when change propagates to other systems, which could then lead to costly redesigns. The complexity of the system itself affects the extent to which change can propagate [145], where a single change can set off a cascade of subsequent changes depending on the parts connectivity. For example, there was an upgrade programme of the F/A-18 for the Swiss Air Force [146], where a change of material from aluminium to titanium (to increase fatigue life) has propagated by causing changes to the fuselage, shifting the centre of gravity, increasing take-off gross weight, changing the flight control software and so forth. Because some of these changes were not anticipated, the cost of this project significantly increased. Furthermore, although the F/A-18 was designed with changeability in mind by including features like modular interfaces, the lack of margins still resulted in costly redesigns [147]. Lack of knowledge about margins in a design is also a challenge, as it has been recognised as a “*key issue in predicting change propagation within complex engineering systems*” [148].

One of the first approaches to predict and manage changes in complex engineered systems was the *Change Prediction Method* (CPM) [145]. The CPM makes use of *Design Structure Matrices* (DSM) to capture component dependencies. Each link in the DSM represents a risk measure, which is the product of the likelihood and impact of change propagation from a component to an adjacent one. It is important to note that values of likelihood and impact derive from previous projects or expert opinion, which can be highly effort-intensive (or even unfeasible) in the context of large complex systems. In fact, Clarkson *et al.* [145] do not recommend the use of CPM on product models with more than 50 components (*a component here can also refer to a sub-system, thus reducing*

the level of detail). Change propagation from an initiating component to a target component is simulated via a breadth-first search. However, a limit was introduced by Clarkson *et al.* where the change propagation is considered to stop after three or four steps. The absence of explicit change absorbers in the product model could explain the need to put such a practical limit, which actually assumes that all changes would not propagate beyond four steps.

Cheng and Chu [149] proposed a graph-theoretical approach (where nodes correspond to components) to assess change impact, which uses the notion of *centrality* allowing to find the most important nodes. Hamraz *et al.* [150] reported that Cheng and Chu's approach considers only propagation paths with the highest likelihood, as opposed to CPM which considers all paths. Although this can reduce computational cost, the same limitations of CPM still apply, notably the subjective setting of impact values.

Koh *et al.* [151] extended the CPM by proposing normalised indices to prioritise components in terms of change propagation risk. However, their method still suffers from the aforementioned limitations of CPM as it uses the same values from the generated DSMs and considers that change can propagate up to four steps.

Long and Ferguson [152] pointed out that design margins are implicitly considered when eliciting likelihood and impact values through expert judgement, which prevents the revision of change propagation prediction as margins evolve. For instance, when a margin is used up, the likelihood of change propagation should increase due to less flexibility. Thus, Long and Ferguson [152] proposed to extend the CPM by accounting for the effect of decreasing margins, which is limited to independent modifications, while keeping margins implicit. It is important to note that when a change is initiated, Long and Ferguson [152] assume that margin is used up, which in turn increases change propagation probabilities by some value informed by the scale of anticipated change. Moreover, change propagation is simulated by drawing from a uniform distribution to determine whether a component is part of the propagation path. This implies that different runs would randomly predict different propagation paths, which may not all exist in reality. Another

limitation is that margin increase is not accounted for, in which case Long and Ferguson [152] suggest to manually reduce likelihood probabilities as it is outside the scope of their work.

Although assumptions are known to cause change propagation in engineering design [153], no approach has been proposed to explicitly relate assumptions to change propagation analysis. In this context, Brahma and Wynn [153] suggest that “*approaches to track assumptions made during the design process could help to more effectively predict the impact of changes during that process*”, and that design change considerations could be integrated with margin management.

3.5.3 Existing Approaches to Margin Allocation and Management

To address the limitations of using conservative margins, some probabilistic approaches have been developed. Thunnissen [139] proposed a six-step method for the probabilistic assignment of margins at the conceptual stage, arguing that the traditional (deterministic) process in industry is often reactive rather than proactive, and lack of rigour makes it unlikely optimal. The six steps consist of (1) identifying tradable parameters (i.e. parameters that are important to satisfy the requirements, such as mass and power); (2) generating analysis models to compute the identified tradable parameters; (3) classifying parameters as either constants, design variables or requirements; (4) modelling uncertain variables as PDFs; (5) propagating uncertainty via a Monte Carlo simulation; and (6) analysing the resulting uncertainty in tradable parameters, where three percentiles (95, 99, and 99.9) are derived to provide a low, medium, and high confidence estimate (respectively) in the probability that such tradable parameters will not be exceeded. The difference between the percentile and the deterministic value is used to provide a margin to be allocated at the current stage of the design.

Zang *et al.* [154] proposed a new strategy for probabilistic margin allocation from an aircraft conceptual design perspective, arguing that the use of margins in conceptual design is far less apparent than in preliminary and detailed stages of aircraft design. Lack

of probabilistic margin allocation may be related to probabilistic uncertainty assessment being more intrusive to an already established design process, and also more computationally expensive [154]. Thus, Zang *et al.* claim that their strategy non-intrusively allocates discipline-level performance margins (assigned to inputs of aircraft sizing and performance models) in a rigorous way, thus increasing the confidence in conceptual design. The approach consists of sampling uncertain design variables, which then provides a set of random outputs from which probabilities of constraint satisfaction are calculated. This strategy is similar in principle to Thunnissen's [139], except there is here an optimisation loop to satisfy the probability of success while a figure of merit is optimised. For instance, margins can be allocated to meet a 95% probability of success while *Take-Off Gross Weight* is minimised.

A similar strategy was proposed by Yuan *et al.* [140], where a sizing loop for range performance analysis is used. Once it converges to the target aircraft range, there are three steps to reach optimised margins: (1) a Monte Carlo simulation is used to propagate uncertainty based on assumed probability distributions for the uncertain parameters, thus obtaining a PDF of aircraft range; (2) in case the (assumed) 90% confidence level is not achieved with initial (nominal) parameters, margins would then be assigned to the uncertain parameters to improve the confidence level, which requires the aircraft to be re-sized, and then the failure probability is re-estimated for the chosen margins; and finally (3) an external optimisation loop is applied to identify more efficient margins.

Cooke *et al.* [155] proposed a method for probabilistic design space exploration and margin allocation called "*Sculpting*", which is based on a Bayesian Belief Network that represents the design space, and accounts for dependencies between design variables, uncertain variables and margins. This network can then serve to generate parallel coordinates plots that assist decision-making regarding margin allocation, for instance by showing the different combinations that yield compliant solutions.

Hall *et al.* [156] proposed a probabilistic approach that uses surrogate models to conduct multi-disciplinary analyses, which in turn allow to obtain a PDF corresponding to

a *figure of merit* through uncertainty propagation. Analysing the uncertainty in figures of merit would then inform margin allocation. Besides, Hall *et al.* make use of the *MoSSEC* standard [157] to declare and share uncertainties on the design parameters, which should make uncertainties and margins more explicit amongst collaborators. Note that Hall *et al.* explicitly expressed their intention to identify and quantify aleatory uncertainty in particular.

Other probabilistic approaches have been developed in different fields, such as bridge design [158], ship design [159] and so forth.

Margins can also be introduced when a design involves component reuse. In such context, Brahma and Wynn [160] proposed the *Margin Value Method* (MVM), which focuses on margins introduced by off-the-shelf components and reused components from product platforms or previous generations. The example given by the authors is that, if the product being designed requires a motor of 3.72 kW, but the supplier only provides 3.5 kW and 4 kW models, selecting the larger model introduces a residual margin of 280 W. MVM can be used in the context of an existing design, where incremental improvement is desired. According to the authors, such incremental improvement can be achieved by locating components with residual margins, and prioritising them for redesign. MVM necessitates knowledge about design parameters and their dependencies, which means that a computational workflow is required. Furthermore, Brahma and Wynn proposed metrics to support margin analysis. Their metric regarding *impact on performance* is analogous to a simple *One-at-a-Time* sensitivity analysis, meaning that only one margin is changed at a time, while all the others are kept the same. Thus, there is an underlying assumption that all margins are treated as independent. Moreover, the weights for the metrics aggregation have to be assumed, which means that the perceived value of a margin is highly subjective. Thus, it could lead to potentially very different recommendations for design improvement depending on how the weights are set.

In terms of interactive margin management, Guenov *et al.* [161] proposed an approach that allows to explore the effects of margins on: other margins, performance and prob-

abilities of constraint satisfaction. Their approach is based on a concept called *Margin Space*, which is a hypercube consisting of the ranges of all assigned margins, and is bidirectionally linked to the design space. It is important to note that margins are assumed to be independent, which means that margin redundancy is not considered. Each point in the *Margin Space* is considered as a margin combination, which feasibility depends on whether the resulting performance meets the constraints. One relevant limitation, which was pointed out by the authors [161], is that margin evolution due to changing uncertainty is not accounted for. For instance, validating assumptions would reduce epistemic uncertainty, which in turn should prompt a decrease in the mitigating margins.

Probability theory proved to be useful in strategies where margin allocation needs to be informed by the characterisation of aleatory uncertainty. Such aleatory uncertainty in aircraft design includes for instance manufacturing errors or environmental conditions (e.g. air density at cruise altitude). However, a question still remains regarding how assumptions (and their changes during their lifecycle) explicitly affect margin allocation and revision.

In fact, and in the context of this research, margins are assigned to mitigate risks associated with lack of knowledge, whereas assumptions are made to fill knowledge gaps. Therefore, it seems only logical that there should be explicit relationships between margins and assumptions (for which margins are used for risk mitigation). However, and to the best of the author's knowledge, no such relationships have been discussed in the published literature.

3.6 Conclusions

The purpose of this literature review was to highlight the limitations (and associated opportunities for improvement) of existing approaches to epistemic uncertainty management. The conclusions of this literature review are the following:

- Epistemic uncertainty exists primarily in the form of assumptions early in the design

process. An industrial need has been identified for formalised (model-based) assumption management, as simply documenting assumptions proved to have little benefit.

- Current definitions of the notion of *assumption* were found to be simplistic, as they do not reflect the different characteristics of an assumption. A new definition was thus attempted by the author. Furthermore, a simplistic description of the assumption's lifecycle has been observed in the literature, which does not address important issues such as the implications of varying confidence in assumptions, or the impact of changing assumptions on other elements of system design. Therefore, an opportunity has been identified to propose a description of changes relating to assumptions and their implications.
- Although some industrial standards consider assumptions only in the context of requirements management, it is argued in this thesis that assumption management should not be limited to the requirements domain, but rather consider as well assumptions related to the functional, logical and computational domains.
- The field of software engineering has developed the most advanced approaches to (semi-)formal assumption management. An opportunity has been identified to simplify the assumption dependency types that have been proposed by researchers in the aforementioned field. These types were found to be non-mutually exclusive, thus inducing a form of redundancy. Such redundancy could result in unnecessarily increasing the effort to capture dependencies, and also potentially confusing practitioners.
- Although assumptions have varying degrees of confidence, none of the reviewed approaches to assumption management included an assessment of the uncertainty in assumptions associated to all R-F-L-C domains (i.e. not just assumed values of design parameters, as in [18]). An opportunity to use the strength of background

knowledge for uncertainty assessment in assumption management has been identified.

- Limitations of the reviewed methods for belief revision have been identified, including (a) a lack of assessing the uncertainty in assumptions; (b) the fact that TMS must be used in conjunction with an artificial problem-solver; (c) the fact that TMS considers logical contradiction only (i.e. belief set contains both a fact and its negation); (d) dependencies amongst assumptions are not considered; and (e) lack of extra-logical factors to prioritise assumptions for revision.
- Although assumption management plays a crucial role in Knowledge Maturity, no method has been proposed to assess Knowledge Maturity with an explicit consideration of assumptions.
- A lack of explicitly considering margins in existing approaches for change propagation prediction has been identified. Thus, there is an opportunity to explicitly account for margins in the context of change propagation analysis. Furthermore, although assumptions are known to cause change propagation in engineering design, no approach has been proposed to explicitly relate assumptions to change propagation analysis.
- The explicit relationships between assumptions and margins have not been treated in the published literature. Moreover, a lack of tools that track margins along with the rationale underlying their change has been reported in the literature. Thus, there is an opportunity to provide justification for margin revision following changes in assumptions.

In the next two chapters, the aforementioned limitations and opportunities are addressed. In Chapter 4, the process of assumption management is described in accordance with the ISO/IEC/IEEE 24774:2021 standard, and methods are proposed to support a model-based approach to assumption management. In Chapter 5, a set of methods is

proposed to support the assessment, mitigation and monitoring of risks associated with assumptions, with an explicit consideration of margins.

Chapter 4

Assumption Management: Process Description and Supporting Methods

4.1 Introduction

In the motivation to this thesis (Section 1.1), the need for a model-based approach to assumption management was identified in order to replace the traditional document-centric approach. The work presented in this chapter is in pursuit of *Objective 1*, i.e. to devise methods to enable assumption management in a model-based design environment.

The chapter is structured as follows: first, a description of the assumption management process in accordance with ISO/IEC/IEEE 24774:2021 is presented in Section 4.2. This is followed by the *Design Belief Network* method (Section 4.3), which shall serve as the data structure capturing assumptions, their dependencies (Section 4.3.2) and uncertainty (Section 4.3.3). In Section 4.4, an algorithm to detect conflicting assumptions that lead to constraint violation is presented. Finally, conclusions are drawn in Section 4.5.

4.2 Assumption Management: Process Description

It was concluded from the literature review (Section 3.6) that assumption management should not be limited to the requirements domain, but rather consider as well assumptions related to the functional, logical and computational domains. Thus, assumption management is considered as a separate process in this thesis. Assumption management can actually be seen as bridging between *System Design & Analysis* and *Technical Risk Management* (Figure 4.1). The rationale captured during system design allows to describe the assumptions made, which are then communicated to the risk management process. This in turn results in assigning margins for risk mitigation, which then affect further design decisions. Additionally, when there are changes in assumptions, the assumption management process provides justification to revise margins. Furthermore, since the description of assumptions includes their association with elements of the system model, such elements can be traced back to their underlying assumptions. Such interactions of the assumption management process with *System Design & Analysis* and *Technical Risk Management* are discussed in more detail throughout Chapters 4 and 5.

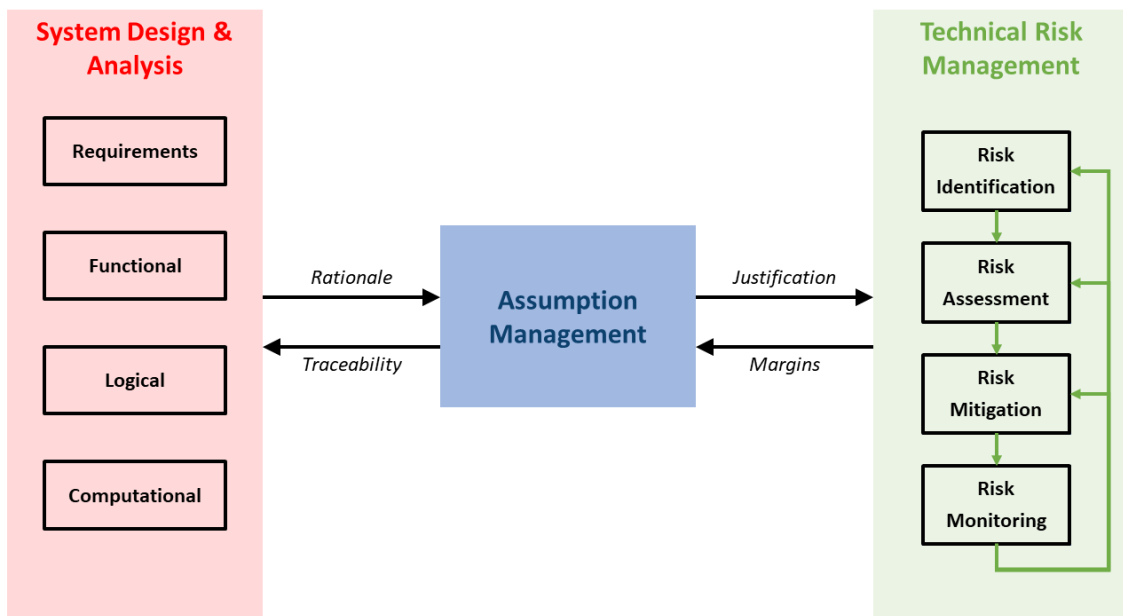


Figure 4.1: Assumption management within systems engineering

As discussed in Section 3.2.3, Yang *et al.* [77] identified the core activities of assump-

tion management from a systematic mapping study on assumptions and their management in software development. These activities, which are *assumption making, description, evaluation and maintenance* (as defined in Section 3.2.3), shall serve as a starting point to the process description proposed in this thesis (Section 4.2.2). The following section introduces the standard used for process description, the ISO/IEC/IEEE 24774:2021 [21].

4.2.1 ISO/IEC/IEEE 24774:2021

The ISO/IEC/IEEE 24774:2021 standard (Systems and software engineering — Lifecycle management — Specification for process description) “*presents requirements for the description of processes in terms of their format, content and level of prescription*” [21].

According to ISO/IEC/IEEE 12207:2017 [162], a process is a “*set of interrelated or interacting activities that transforms inputs into outputs*”. An activity is a “*set of cohesive tasks of a process*”. A task is defined as a “*required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process*”.

Elements constituting a process description can be summarised as follows [21]:

- **Name:** Formulated as a short noun phrase ending with the word *process*.
- **Purpose:** The purpose of the process is formulated as one or more high-level objectives of carrying out the process.
- **Outcomes:** Results achieved by the process. Note that outcomes are distinguished from outputs as they do not refer to documents or any other information items.
- **Activities:** These are considered as constructs regrouping associated tasks. Any timing or sequencing requirements on activities must be explicitly stated.
- **Tasks:** Any timing or sequencing requirements on tasks must be explicitly stated.
- **Inputs:** Items that are transformed by the process to outputs.
- **Outputs:** Can be of two main types: artefacts and information items. Artefacts include “*prototypes, models, system components and elements, as well as finished*

products and services". Information items include for instance plans, records, reports and specifications.

- **Controls and Constraints:** Controls relate to "*regulatory authorities, organizational policy, adherence to voluntary standards, and agreements with suppliers and customers*". Constraints relate to "*external environmental or business factors*".
- **Notes:** "*Describe the intent or mechanics of a process or process element*".

According to ISO/IEC/IEEE 24774:2021, the minimum required elements of a process description are the name, purpose, and outcomes. A detailed description of the specification for process description is presented in [21].

4.2.2 Process Description in Accordance with ISO/IEC/IEEE 24774:2021

Name

Assumption Management Process.

Purpose

The purpose of the assumption management process is to describe, evaluate and maintain assumptions throughout their lifecycle.

Outcomes

As a result of the successful implementation of the assumption management process:

- a) Assumptions are described and recorded.
- b) Assumptions are evaluated.
- c) Changes in assumptions are identified and evaluated, and necessary action(s) invoked.
- d) Consistency amongst assumptions, and between assumptions and elements of the system model is maintained.

- e) Traceability between elements of the system model and their underlying assumptions is established.

Activities and tasks

The project shall implement the following activities and tasks:

A. **Make and describe assumptions.** This activity consists of the following tasks:

A.1 Make new assumptions to fill knowledge gaps or replace invalid assumptions.

NOTE 1 A knowledge gap refers to an instance of uncertainty due to lack of knowledge. For example, initial aircraft sizing requires having a value for the maximum lift coefficient (C_{Lmax}), which is not known at that stage (hence the knowledge gap). Therefore, an assumed value is assigned to C_{Lmax} , which is considered to tentatively fill the aforementioned knowledge gap.

NOTE 2 Assumptions evaluated as invalid from previous executions of the process can be decided to be replaced by new assumptions, which thus triggers a new iteration of the assumption management process.

A.2 Describe and record assumptions.

NOTE Describing new assumptions entails: (i) assigning them an ID, a name, a textual description, a status and any other information deemed relevant (e.g. owner's name, manager's name, ...); (ii) assessing the level of confidence in making them; and (iii) capturing their dependencies with respect to elements of the system model. In this thesis, elements of a system model refer to requirements, functions, solutions, parameters and computational models.

A.3 Define an evaluation plan for each assumption.

NOTE Evaluation plans consist of reviews, analyses and tests (as recommended by SAE ARP4754A [76]). Additionally, constraints that may hinder evaluation must be identified. Such constraints include technical feasibility, time, cost and availability of resources. Thus, evaluation methods are selected

with respect to the aforementioned constraints.

B. Evaluate assumptions. This activity consists of the following tasks:

B.1. Execute evaluation plans.

B.2. Review evaluation results.

NOTE 1 Depending on the evaluation results, assumptions will be either validated/invalidated (i.e. the collected evidence is sufficient to prove their truth/falsehood), or their level of confidence is reassessed (i.e. the collected evidence is not sufficient to prove their truth/falsehood, but allows to become more or less confident in an assumption).

NOTE 2 If conflicts between assumptions have been identified in Task B.1, these conflicts are reviewed in B.2 to determine whether the information and resources required to resolve the conflicts are available at the current stage of product development.

NOTE 3 The outcomes of reviewing evaluation results are recorded in an evaluation report. The evaluation report includes collected evidence, updated confidence assessments, validated/invalidated assumptions, identified conflicts, and new associations between assumptions and elements of the system model.

C. Maintain assumptions. This activity consists of the following tasks:

C.1. Maintain consistency.

C.2. Record maintenance results.

NOTE Maintenance results are recorded in a maintenance report, which communicates the changes made in assumptions for maintenance purposes, in addition to invalidated assumptions that have been flagged for replacement.

NOTE Maintaining consistency in Task C.1 involves the following:

- Update the confidence in, the status (valid/invalid) and dependency of evaluated assumptions according to the evaluation report.

- If an assumption A_1 is validated, and some other assumption A_2 conflicts with A_1 , A_2 must be invalidated. This is because when two assumptions are conflicting, one of the assumptions (at most) can be valid.
- If an assumption A_1 is invalidated, decide whether it must be replaced by a new one in Activity A.
- Review system model elements associated with invalidated assumptions.
- If the level of confidence in an assumption A_1 increases, and some other assumption A_2 conflicts with A_1 , the level of confidence in A_2 must decrease.
- If the level of confidence in an assumption A_1 increases, decrease the risk prioritisation of A_1 . Conversely, if the level of confidence in an assumption A_1 decreases, increase the risk prioritisation of A_1 . This can be supported by the *Assumption Matrix* method (Section 5.3).
- Resolve conflicting assumptions (if possible at the current stage of product development). Although not all conflicts between assumptions can be automatically detected (thus requiring the involvement of human agents by conducting peer-reviews), conflicting assumptions leading to a constraint violation could be automatically detected via a computational workflow. This is discussed in Section 4.4.
- Revise margins associated with evaluated assumptions for risk mitigation. Since assumptions can be evaluated as invalid, and thus potentially cause costly redesigns, the risks associated with assumptions should be managed. To this end, margins are typically used as a risk mitigation strategy. Managing risks associated with assumptions will be discussed in more detail in Chapter 5.

NOTE The process activities are to be conducted iteratively throughout product development. Activities A, B and C are to be conducted sequentially (i.e. $A \rightarrow B \rightarrow C$), and the tasks are performed according to the process model in Figure 4.2.

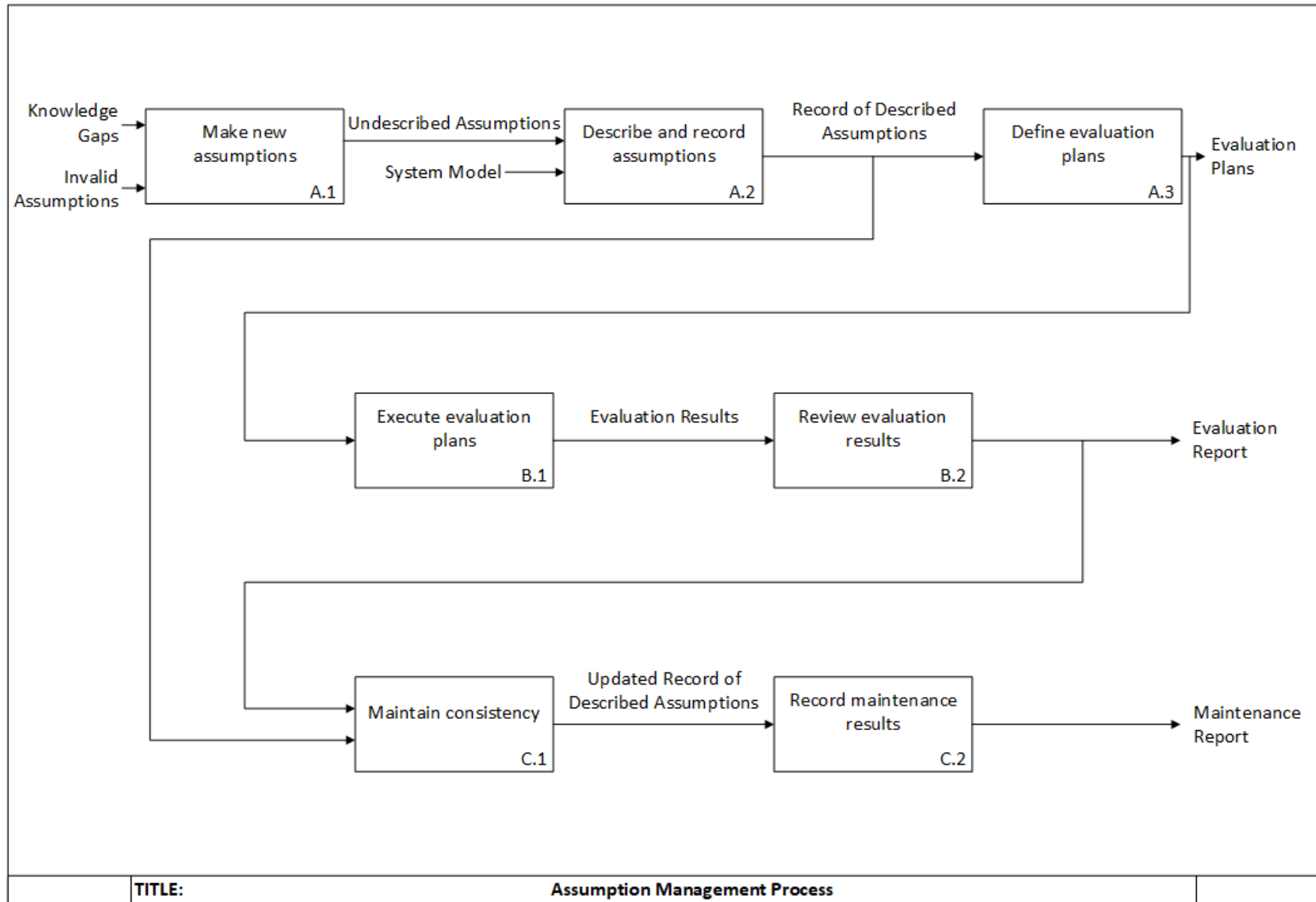


Figure 4.2: IDEF0 representation of the assumption management process. IDEF is a family of Modelling Languages, and IDEF0 is the method for Function Modelling

4.2.3 Discussion

The evaluation plan (Task A.3) shall contain both the information needed to prove an assumption is true (i.e. valid) and when it is to be conducted. The latter depends on when the information needed is expected to become available. This can be related to the systems engineering stages. For example, a particular value of the maximum lift coefficient is assumed to be achievable with the selected high-lift device during the design process. To validate this assumption, the information needed will become available at system integration, where data can be collected on the aerodynamic performance of the high-lift device and any interference caused by its deployment. An organisation could also use its own stage-gate process to plan for evaluation. For example, to validate an assumption on pilot behaviour made during the design of flight control systems, evaluation is carried out at Maturity Gate 11 (cf. Figure 4.3) by collecting data on pilot behaviour during the flight test.

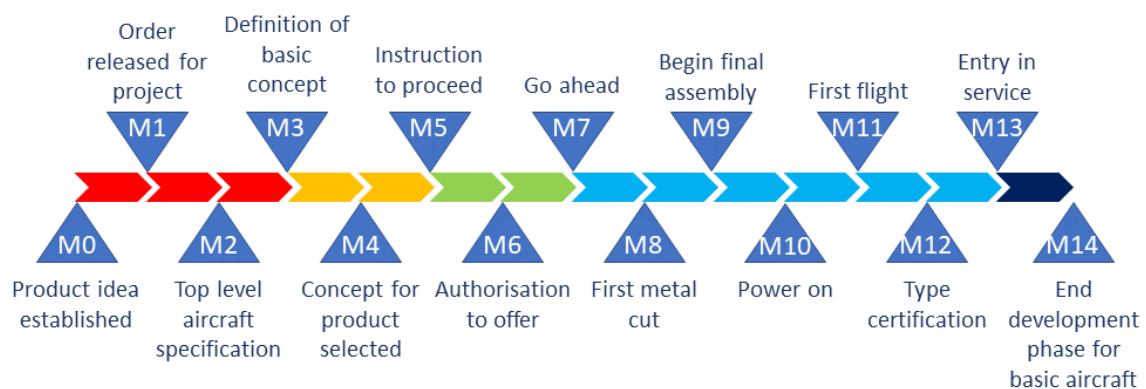


Figure 4.3: Maturity gates at Airbus (adapted from [163])

Another important point about assumption evaluation is the required effort. If there is a significant number of assumptions to be evaluated, a trade-off is necessary between how comprehensive the evaluation of each assumption must be, and the resulting benefit. For instance, it may not be practical to monitor and reassess the level of confidence in all assumptions as more knowledge becomes available throughout product development. In this regard, a prioritisation of assumptions may become necessary. Additionally, identifying conflicts between assumptions through peer-review may also be challenging with a

significant number of assumptions to be evaluated. Therefore, computational means may become necessary to support such a task.

The proposed description of what consistency maintenance involves (pp. 72-73) extends the process description provided by Yang *et al.* [77], which was found in Section 3.2.3 to lack a description of what happens to other related assumptions/elements of the system model when there are changes in an assumption. Another limitation identified in Section 3.2.3 was the lack of a standardised process description, which has been addressed in this research via the ISO/IEC/IEEE 24774:2021 [21].

4.3 Design Belief Network

To enable assumption management in MBSE, a data structure representing the system model is needed. To this end, a graph-theoretical structure (*Design Belief Network*, or *DBN* for short) is presented in this section. The DBN allows to capture assumptions, their uncertainty and dependencies during system architecting. This network can in turn provide traceability to support systems engineering activities, for instance *Requirements Management* where requirements can be traced back to their underlying assumptions, or *Configuration Management* where architectural decisions can be traced back to their underlying assumptions.

4.3.1 Graph-Theoretical Structure

The RFLP model is used to represent the notional domains of system architecting, and it has been augmented with a *Computational* domain (C) for automated sizing and performance assessment by Bile *et al.* [31]. As discussed in Section 3.3.4, there is a need for extra-logical factors. The *Computational* domain can be one such factor as it can enable predicting the impact of some assumptions on system performance (e.g. predicting take-off performance when considering different assumed values for C_{Lmax}). This in turn can allow to identify which beliefs (i.e. design decisions and assumptions) violate perform-

ance constraints.

Guenov *et al.* [32] proposed a graph-theoretical structure that captures the dependencies between the R-F-L-C domains, which is part of a novel framework for interactive systems architecting in early design. This graph-theoretical structure proposed by Guenov *et al.* shall serve as the foundational data structure for the DBN.

Figure 4.4 illustrates the proposed data structure for the DBN, where the existing data structure from Guenov *et al.* [32] was extended by adding the *Assumption* and *Margin* classes. Instances of the classes in Figure 4.4 constitute the nodes of the DBN, and the depicted dependencies constitute the edges of the DBN. Note that, as illustrated in Figure 4.5, the *Component* class refers to the elements of the *Logical* domain, whereas the *Model* and *Parameter* classes refer to elements of the *Computational* domain. Also note that the *Physical* domain has not been included as it is considered outside the scope of this research, given that a detailed physical view definition is not available early in the design process. However, it could be possible to use simple shapes (e.g. cuboids and cylinders) to represent the spatial and topological layouts in early design. Such avenue could be explored as part of future work.

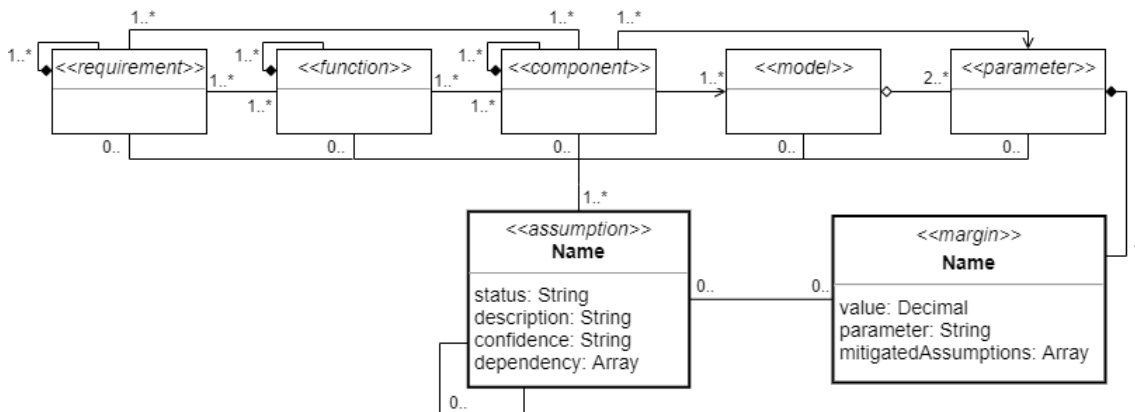


Figure 4.4: UML class diagram of the *Design Belief Network*

In the proposed data structure of the DBN (Figure 4.4), the attributes of the *Assumption* class are defined as follows:

- *status*: a string representing the status of the assumption, which can take the value of either '*Awaiting Evaluation*', '*Valid*' or '*Invalid*'.

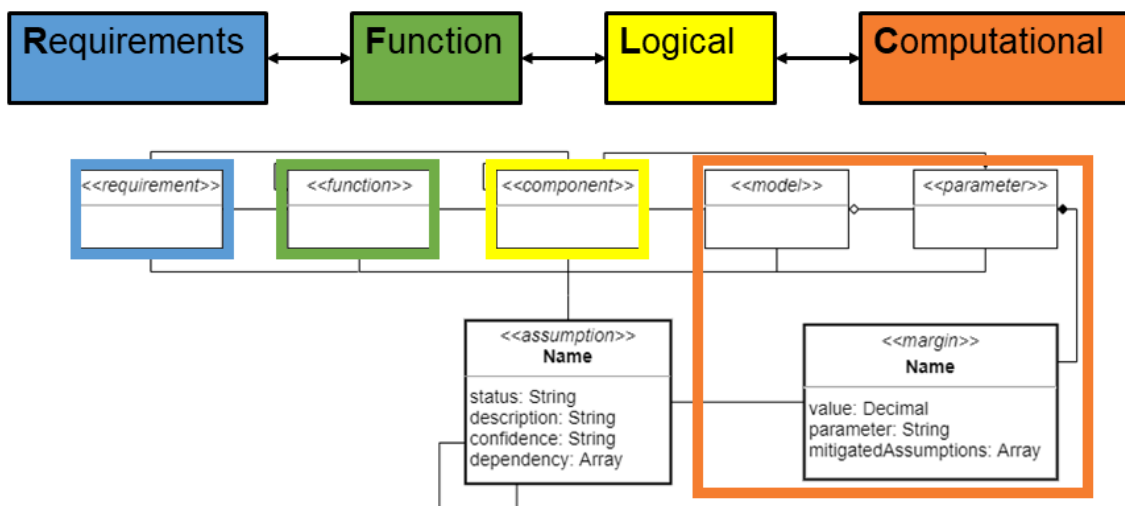


Figure 4.5: Correspondence between the proposed data structure and the R-F-L-C domains

- *description*: a string representing a textual description of the assumption, including the reasons behind it.
- *confidence*: a string representing the level of confidence in making the assumption, which is assessed based on the approach presented in Section 4.3.3.
- *dependency*: an array storing the assumption dependencies, which are described in Section 4.3.2.

These attributes extend the core elements in describing assumptions as identified by Yang *et al.* [77], by taking into consideration the uncertainty in assumptions (i.e. the *confidence* attribute). The proposed class definition can be extended to include other attributes such as the identity of the assumption's owner, date of capturing and so forth.

The attributes of the *Margin* class are defined as follows:

- *value*: a decimal representing the value that is assigned as a margin.
- *parameter*: a string referring to the parameter that the margin is assigned to.
- *mitigatedAssumptions*: an array storing the set of assumptions for which the margin is explicitly mitigating the risk.

4.3.2 Assumption Dependencies

We can distinguish between two types of assumption dependencies: *Inter-domain* and *Intra-domain*.

Inter-domain dependencies consist of the following:

- *Assumption* \longleftrightarrow *Requirement*: occurs when interpreting top-level requirements or deriving new requirements.
- *Assumption* \longleftrightarrow *Function*: occurs as part of the rationale behind defining the functions.
- *Assumption* \longleftrightarrow *Solution*: occurs as part of the rationale behind selecting the solutions.
- *Assumption* \longleftrightarrow *Parameter*: occurs when assigning a value to an uncertain parameter.
- *Assumption* \longleftrightarrow *Model*: refers to the assumptions underlying the computational models.
- *Assumption* \longleftrightarrow *Margin*: refers to the assumption for which the margin explicitly mitigates the risk.

Whereas intra-domain dependency consists of conflict relationships (or conflict edges) between assumptions. Recall that when two assumptions are conflicting, one of the assumptions (at most) can be valid.

The aforementioned inter-/intra-domain dependencies represent a simplification compared to the types of assumption dependencies proposed by Yang *et al.* [95], where it has been identified in Section 3.2.3 that the six types can be reduced to only two: *Conflict* and *Causality*. In fact, it can be argued that the inter-domain dependencies are a form of *Causality* relationships, since architectural elements (e.g. definition of requirements or selection of components) can 'cause' to make assumptions, which in turn could influence

(or 'cause') other architectural elements. Note that an assumption could indirectly influence another one through common architectural elements. For example, if assumptions A_1 and A_2 are both associated with solution C_1 , and A_1 causes change in C_1 , this situation may necessitate to revise A_2 .

Furthermore, a need has been identified in Section 3.2.3 to capture the dependencies between assumptions and the computational models for decision analysis (i.e. architecture assessment). To address the aforementioned need, dependencies between assumptions and elements of the *Computational* domain (i.e. parameters, models and margins) have been included.

4.3.3 Level of Confidence

As discussed in Section 3.2.3, although assumptions have varying degrees of confidence, none of the reviewed approaches to assumption management included an assessment of the uncertainty inherent to assumptions. To address this limitation, the following approach to assess the strength of background knowledge is presented.

The level of confidence in making an assumption is considered in this research to be based on the strength of background knowledge. The latter can be assessed using the guidelines proposed by Flage and Aven [164], which focus in particular on the data, models and expert opinion. Note that, in the original guidelines, Flage and Aven considered assumptions also as part of the background knowledge underlying risk analysis. Therefore, these guidelines had to be adapted for assessing the strength of the background knowledge that influences the assumptions themselves, which means that poor assumptions are consequences of weak background knowledge (rather than causing weak background knowledge as in the original guidelines). The guidelines can be summarised as follows:

1. The level of confidence in an assumption is considered as *high* if **all** of the following conditions are met:

- (a) *Data Reliability*: much reliable data are available;
 - (b) *Model Realism*: the phenomena involved are well understood, the models used are known to give predictions with the required accuracy;
 - (c) *Expert Agreement*: there is broad agreement among experts.
2. The level of confidence in an assumption is considered as *low* if **one or more** of the following conditions are met:
- (a) *Data Reliability*: data are not available or are unreliable;
 - (b) *Model Realism*: the phenomena involved are not well understood, models with a high degree of realism (e.g. physics-based) are not applicable / models are non-existent or believed to give poor predictions;
 - (c) *Expert Agreement*: there is a lack of agreement among experts.
3. The level of confidence in an assumption is considered as *Moderate* for intermediate states, for instance some reliable data are available, and the phenomena involved are well understood, but the models used are considered simple.

Thus, the practitioner is expected to assess the strength of background knowledge based on the three criteria: *Data Reliability*, *Model Realism* and *Expert Agreement*, where each criterion is assigned a value of either *High*, *Moderate* or *Low*. Then, the values of these assessment criteria are combined to generate a value for the *confidence* attribute of the *Assumption* class, as prescribed in Algorithm 1.

Algorithm 1: Confidence Assessment

```

input : Assumption object  $\alpha$ , and its corresponding values of DataReliability,
          ModelRealism and ExpertAgreement as assigned by the user.
output: Value of the confidence attribute of  $\alpha$ 

1 begin
2   if DataReliability == High and ModelRealism == High and
      ExpertAgreement == High then
3     |  $\alpha$ .Confidence  $\leftarrow$  High
4   end if
5   else if DataReliability == Low or ModelRealism == Low or
      ExpertAgreement == Low then
6     |  $\alpha$ .Confidence  $\leftarrow$  Low
7   end if
8   else if DataReliability ==  $\emptyset$  or ModelRealism ==  $\emptyset$  and
      ExpertAgreement  $\neq$   $\emptyset$  then
9     |  $\alpha$ .Confidence  $\leftarrow$  Value(ExpertAgreement)
10  end if
11  else
12    |  $\alpha$ .Confidence  $\leftarrow$  Moderate
13  end if
14  return  $\alpha$ .Confidence
15 end

```

In lines 2-4, the level of confidence is considered as *High* if all three criteria have been assessed as *High*. In lines 5-7, the level of confidence is considered as *Low* if any of the three criteria has been assessed as *Low*. In lines 8-10, the level of confidence takes the value that was assigned to *Expert Agreement* in case *Data Reliability* and *Model Realism* are not relevant for a particular assumption. Finally, in lines 11-13, the level of confidence is considered as *Moderate* for the remaining intermediate states.

From a *Belief Revision* perspective, the level of confidence can be used as a way to prioritise beliefs for changing/removal. According to the principle of *Epistemic Entrenchment* (Section 3.3.2), some beliefs are more important than others, meaning that the least important ones should be given up first if necessary [111]. Thus, assumptions can be ranked in terms of their level of confidence.

4.4 Constraint Violation and Conflict Detection

As discussed in Section 3.3.1, inconsistency can be categorised as either *syntactic* or *semantic*. Syntactic inconsistency is related to the *structure* of the rationale (which entails looking for missing information), whereas semantic inconsistency is related to the *meaning* (which entails looking into the content of the rationale) [109]. One form of semantic inconsistency in engineering design is conflicts in design collaboration. According to Wang *et al.* [165], there are five steps to conflict resolution: conflict detection, conflict identification, negotiation team formation, solution generation, and solution evaluation.

One of the limitations identified from the literature review (Chapter 3) is the fact that the contradictions, which are detected by the problem-solver and communicated to the TMS, are syntactic in nature. For a contradiction to be detected, both some fact A and its negation (i.e. A and $\neg A$) must explicitly belong to the belief set. This is interpreted as believing that A is both true and false, hence the contradiction. When the contradiction is discovered by the problem-solver and communicated to the TMS, the assumptions associated with the contradictory facts are gathered to form a new set called the '*NOGOOD*' assumption set. The '*NOGOOD*' set is then cached so that no further conclusions can be drawn from the inconsistencies [100]. Therefore, resolving semantic inconsistency, such as conflicts encountered in collaborative design, requires a novel approach.

To this end, constraint violation is used as an indicator of the presence of conflicting assumptions. The idea is that when an architecture is defined (including the assumptions) and then assessed via the Computational Domain, some parameter values can violate constraints. The assumptions leading to the constraint violation are gathered (via traversing the DBN) to form a new assumption set representing potentially conflicting assumptions. Such a set is then passed on to experts for conflict identification and resolution. The benefit of such approach is to reduce the entire set of assumptions to a subset of potentially conflicting assumptions. Therefore, this should prevent the impractical task of reviewing all assumptions in order to pinpoint the conflicting ones.

4.4.1 Algorithm for Conflict Detection

To support the detection of conflicting assumptions, Algorithm 2 is proposed. A depth-first search (DFS) is used as a subroutine for graph traversal, where Algorithm 2 traverses the computational workflow (i.e. the *Parameter* and *Model* nodes of the DBN) when searching for the conflicting assumptions. To implement a DFS, the algorithm proposed by Cormen *et al.* [39] is used. The proposed method follows three main steps:

- Step 1.** Execute the computational workflow to check for constraint violation. Both the computational workflow and constraints formulation are considered as inputs to Algorithm 2.
- Step 2.** Collect all assumptions that may have contributed to the constraint violation via a reversed traversal of the computational workflow. Such graph traversal shall start from the parameters directly included in the formulation of the violated constraint. This step returns an initial set of conflicting assumptions.
- Step 3.** For computational workflows containing multi-input/multi-output models, the parameters associated with the initial set of conflicting assumptions are varied. This allows to remove any assumptions that do not influence the constraint violation, and therefore reduce the number of assumptions to consider.

In lines 4-7 of Algorithm 2, each node (or vertex) u in the computational workflow G is initially 'coloured' as *WHITE* to indicate that u has not been discovered yet by the DFS. Additionally, the predecessor of each u ($u.\pi$) is set as *NIL* to indicate that no predecessor has been discovered since the DFS has not started yet. In line 8, the global time counter (*time*) is reset to 0. *time* is used for timestamping, where the DFS records when u is discovered (using the attribute $u.d$), and when it finishes u (using the attribute $u.f$) [39]. These timestamps allow then to change the 'colour' of visited vertices from *WHITE* (i.e. before time $u.d$), to *GRAY* (i.e. between time $u.d$ and time $u.f$) and finally to *BLACK* (i.e. after time $u.f$) [39]. Lines 4-8 allow to initialise the DFS as suggested by Cormen *et al.* [39], which is done for each parameter in P_i (line 3) corresponding to violated constraint

M_i (line 2). Following the aforementioned initialisation, the DFS is performed in line 9 to collect the conflicting assumptions. Note that DFS(G,v) in line 9 is presented separately in Algorithm 3 as it is called recursively.

Once an initial set of conflicting assumptions (CA_i) has been detected following the DFS (Step 2), the goal is then to vary each parameter associated with assumptions in CA_i in case of multi-input/multi-output models (Step 3). To this end, a Boolean variable *Impact* is defined in line 12 to store whether the parameter has an impact on the violated constraint, whereas lines 13-14 refer to sampling the aforementioned parameters over a range to simulate their variation. In line 15, G is executed in order to obtain the baseline values (i.e. before variation) for all the parameters in P_i for comparison afterwards. Then, in line 17, the value of parameter p is set to the sample value x in the current iteration of the For loop (line 16), and G is re-executed to obtain the updated values of all the parameters in P_i in line 18. Lines 19-24 check whether there has been variation in any of the parameters in P_i , so that if it is the case, the value of *Impact* is changed to *True*. Lines 25-27 serve to stop iterating over the sampled values once the parameter has been identified as impacting the violated constraint (i.e. *Impact = True*). In lines 29-31, if the parameter is not found to impact the violated constraint (i.e. *Impact = False*), this parameter and its associated assumption are removed from CA_i (i.e. the assumption is no longer considered as contributing to the constraint violation). Finally, line 34 returns the set of conflicting assumptions associated with each violated constraint.

Algorithm 2: Detection of Conflicting Assumptions

input : Computational workflow as a graph $G = (V, E)$ (G is a subgraph of the DBN, thus contains the associated assumptions); Set of violated constraints $M = \{M_1, M_2, \dots, M_i, \dots, M_m\}$ with corresponding set of constituting parameters $P_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,n}\}$ ($p_{i,1}, p_{i,2}, \dots, p_{i,n} \in G.V$)

output: Dictionary of conflicting assumptions CA_i corresponding to each violated constraint M_i

```

1 begin
2   for each constraint  $M_i$  in  $M$  do
3     for each  $v$  in  $P_i$  do
4       for each vertex  $u$  in  $G.V$  do
5          $u.colour = WHITE$ 
6          $u.\pi = NIL$ 
7       end for
8        $time = 0$ 
9       DFS( $G, v$ )
10    end for
11    for each  $p$  in  $CA_i.keys$  do
12       $Impact = False$ 
13       $\delta \leftarrow$  Assign percentage of variation
14       $P_{SV} \leftarrow (1 - \delta)p.value : stepsize : (1 + \delta)p.value$ 
15       $P_{i,0} \leftarrow$  Execute  $G$  to get baseline values  $\forall$  parameters in  $P_i$ 
16      for each  $x$  in  $P_{SV}$  do
17         $p.value = x$ 
18         $P_{i,1} \leftarrow$  Execute  $G$  to get updated values  $\forall$  parameters in  $P_i$ 
19        for  $k$  in  $[1, |P_i|]$  do
20          if  $P_{i,0}[k] \neq P_{i,1}[k]$  then
21             $Impact = True$ 
22            break
23          end if
24        end for
25        if  $Impact == True$  then
26          break
27        end if
28      end for
29      if  $Impact == False$  then
30         $CA_i.remove(p : CA_i[p])$ 
31      end if
32    end for
33  end for
34  return  $CA_i$  for every constraint  $M_i$  in  $M$ 
35 end

```

Algorithm 3: DFS(G, u)

input : Computational workflow as a graph $G = (V, E)$ (G is a subgraph of the DBN, thus contains the associated assumptions); Vertex u where the search starts from

```

1 begin
2    $\forall$  edges in  $G.E$ , if  $\exists$  assumption  $\alpha : (u, \alpha) \in G.E$  then
3     |  $CA_i.append(u : \alpha)$ 
4   end if
5    $time = time + 1$ 
6    $u.d = time$ 
7    $u.colour = GRAY$ 
8   for each  $v$  in  $G.Adj[u]$  do
9     | if  $v.colour == WHITE$  then
10    |    $\forall$  edges in  $G.E$ , if  $\exists$  assumption  $\alpha' : (v, \alpha') \in G.E$  then
11    |   |  $CA_i.append(v : \alpha')$ 
12    |   end if
13    |    $v.\pi = u$ 
14    |   DFS( $G, v$ )
15    | end if
16  end for
17   $u.colour = BLACK$ 
18   $time = time + 1$ 
19   $u.f = time$ 
20 end

```

Regarding Algorithm 3, which was adapted from [39], lines 2-4 check whether u is associated with an assumption α , so that the entry (key: u , value: α) can be added to the dictionary CA_i of conflicting assumptions. Then, in lines 5-7, the timestamp is incremented by 1 and the 'colour' of u is changed to *GRAY* to signify that u has been discovered in the depth-first search. Lines 8-16 explore each vertex v adjacent to u , where the DFS is recursively performed from v if the latter is *WHITE*. Lines 10-12 check whether v is associated with an assumption α' , so that the entry (key: v , value: α') can be added to the dictionary CA_i of conflicting assumptions. Finally, lines 17-19 change the 'colour' of u to *BLACK* and increment the timestamp to signify that all vertices adjacent to u have been explored.

4.4.2 Discussion

The presented approach to conflict detection is, to some extent, comparable to the one adopted in the TMS (i.e. the ‘NOGOOD’ set). However, the contribution made in this research is making use of an extra-logical factor (i.e. the Computational Domain) to support the detection of semantic inconsistencies, instead of focusing solely on logical contradictions (as in the TMS). Recall from Section 3.3.2 that in belief revision, when inconsistencies occur (especially semantic ones), logical considerations alone cannot tell us which particular beliefs should be changed/removed in order to restore consistency. Thus, some other means – known as extra-logical factors – are needed. However, extra effort may be required to define explicit constraints between domains (which could be significant for novel designs). A trade-off between the benefit from the proposed approach and the initial effort needed must then be considered.

Although the focus was on conflicting assumptions that lead to constraint violation, other inconsistencies exist and can be considered for future work. For instance, there is assumption duplication, where two assumption objects are created independently (e.g. by different teams), while they refer to the same assumption in meaning.

4.5 Conclusions

This chapter presented a process description and supporting methods that were developed to achieve Objective 1, i.e. to devise methods to enable assumption management in a model-based design environment.

A standardised description of the assumption management process, in accordance with ISO/IEC/IEEE 24774:2021 [21], was presented in Section 4.2.2. This description, which is based on a more comprehensive definition of assumptions proposed by the author, takes into consideration both the uncertainty in assumptions and the implications of changes in assumptions. This addresses limitations of existing descriptions of the assumption lifecycle as identified in Section 3.2, which include (i) the lack of accounting

for uncertainty in assumptions; (ii) the lack of describing how changes during an assumption's lifecycle affect elements of a system model; and (iii) the lack of a standardised description of the assumption management process.

A graph-theoretical structure, the *Design Belief Network* (Section 4.3), was proposed in this chapter to capture and codify assumptions and their dependencies in a model-based manner (consistent with the RFLP model). This is in response to an identified need in industry for more formalised (model-based) assumption management, as simply documenting assumptions proved to have little benefit. In this graph, a new class (*Assumption*) was proposed to enable the explicit capturing of assumptions, including the dependencies both amongst assumptions, and between assumptions and elements of the system model. Additionally, the assumption dependency types proposed in the literature have been simplified, since these types were found to be non-mutually exclusive, thus inducing a form of redundancy. Such redundancy could result in unnecessarily increasing the effort to capture dependencies, and also potentially confusing practitioners. Furthermore, although assumptions have varying degrees of confidence, a lack of assessment of the uncertainty inherent in assumptions was identified in Section 3.2.3. Thus, the proposed method includes an assessment of the level of confidence in assumptions, which is based on the strength of background knowledge.

The *Design Belief Network* contrasts with the document-centric approach, such as using an 'Assumption Log', which suffers from some limitations in the context of large complex systems. First, there is the lack of capturing dependencies, where the complexity of the design makes it very challenging to mentally keep track of all dependencies. This would also mean that traceability cannot be as easily and consistently provided. Due to this lack of dependency, there is a higher risk of assumption duplication when using a document. Furthermore, since there is no assessment of the confidence in the traditional approach, it can be challenging to prioritise assumptions in terms of their likelihood of being invalid.

An algorithm was presented in Section 4.4 to detect conflicting assumptions that lead

to constraint violation. The presented approach is, to some extent, comparable to the one adopted in the TMS (i.e. the 'NOGOOD' set). However, the contribution made in this research is making use of an extra-logical factor (i.e. the Computational Domain) to support the detection of semantic inconsistencies, instead of focusing solely on logical contradictions (as in the TMS). This is expected to reduce the time and cost of identifying conflicting assumptions, especially when it has to be done manually in large scale projects. Furthermore, the algorithm can be used in conjunction with existing mechanisms from collaborative design coordination (e.g. explicit constraints between different domains), thus supporting conflict resolution.

Chapter 5

Managing Risks Associated with Assumptions

5.1 Introduction

In the motivation to this thesis (Section 1.1), it was shown that invalid assumptions can cause highly expensive redesigns, or even catastrophic failures. Therefore, such risks need to be appropriately managed. The work presented in this chapter is in pursuit of *Objective 2*, i.e. to devise methods to manage risk of change due to invalid assumptions, with an explicit consideration of both assumptions and margins.

The chapter is structured as follows: first, knowledge maturity assessment is presented in Section 5.2, where a composite indicator called the *Knowledge Maturity Index* (KMI) to indicate the overall risk of change due to lack of knowledge is described. This is followed by the *Assumption Matrix* method (Section 5.3), which uses both the level of confidence and the dependencies of the captured assumptions as a way to prioritise the latter in terms of the risk to initiate change. In terms of risk mitigation, Section 5.4 presents an algorithm to provide the status of margin allocation with respect to assumptions, whereas Section 5.5 presents an algorithm which detects the closest margin that can absorb change initiated by an invalid assumption. In terms of risk monitoring, Section 5.6 presents an algorithm

that suggests margin revisions following changes in assumptions, while using the *Margin Space* concept to ensure constraint satisfaction. Finally, conclusions are drawn in Section 5.7.

5.2 Knowledge Maturity Assessment

Reviewing the literature on managing the risks associated with epistemic uncertainty led to finding that knowledge maturity is affected by assumption management (see Section 3.4). Knowledge maturity is a concept to support decision making in a gated process, and highlight the status of knowledge going into the gate [26]. According to Johansson [26], assessing assumptions and judging their validity is a challenge for decision makers at the gates because they can be mistaken for proven knowledge. Johansson then points to the importance of assumption management in maturity assessment. However, no formal approach has been proposed to assess knowledge maturity with an explicit consideration of assumptions. Therefore, a need has been identified to devise a method in order to assess knowledge maturity in a systematic way, with an explicit consideration of assumptions. Consequently, a composite indicator called the *Knowledge Maturity Index* (KMI) has been constructed by the author following a rigorous methodology jointly developed by the OECD and the European Commission [166]. According to the OECD [167], a composite indicator is “*formed when individual indicators are compiled into a single index, on the basis of an underlying model of the multi-dimensional concept that is being measured*”. Thus, composite indicators allow to summarise complex and multi-dimensional concepts (such as maturity) in order to support decision-makers, and assess progress over time [166].

The following sections present the different steps for constructing a composite indicator, according to OECD’s methodology [166].

5.2.1 Variables Selection

Recall that an assumption is a context-dependent belief, with a varying degree of confidence, that requires validation to become knowledge. An assumption bridges the gap between available knowledge and knowledge required to proceed with the design process. Furthermore, knowledge maturity can be defined as a state that refers to how close knowledge at a decision gate is to knowledge needed for progressing in the development [26]. Therefore, assumptions provide a direct and explicit means of assessing knowledge maturity, where validating assumptions would indicate progress towards full knowledge maturity.

As discussed in Section 3.4, the quality of knowledge going into a decision point consists of three dimensions: *Input*, *Method* and *Expertise*. Since the *Level of Confidence* attribute of the captured assumptions is an assessment of the available data and their reliability (i.e. in line with the *Input* dimension), model realism (i.e. in line with the *Method* dimension), and expert agreement (i.e. in line with the *Expertise* dimension), the selection of *Level of Confidence* as a variable would fit the dimensions proposed by Johansson *et al.* [128].

In addition to the confidence in individual assumptions, these should also be consistent as a whole (i.e. not conflicting amongst each other). Therefore, the amount of conflicts amongst assumptions can provide an indication of their consistency, and thus be used as a variable. Recall that assumption conflicts are captured by the *Conflict* edges of the *DBN*, as discussed in Section 4.3.2.

Therefore, the variables that will form the basis of the KMI are (a) the number of assumptions, (b) the *Level of Confidence*, and (c) the number of *Conflict* edges.

Recall from Section 3.4 that in addition to the quality of knowledge, Johansson *et al.* [128] and their industrial collaborators also deemed important to assess the maturity of the process by which the knowledge is managed within the organisation. Note that knowledge management is outside the scope of the current research and, as mentioned in Section 3.4, models already exist to assess it. Nonetheless, the organisation applying the proposed

methods in this thesis is expected to have suitable knowledge management capabilities allowing to create and share system models and *Design Belief Networks*. Thus, the focus shall remain on the quality of knowledge.

5.2.2 Normalisation

This step is about normalising the selected variables to obtain a common scale. To this end, the normalisation method *Distance to a Reference* is adopted. This method “*measures the relative position of a given indicator vis-à-vis a reference point. This could be a target to be reached in a given time frame*” [166]. From a mathematical view, the resulting normalised indicator is the ratio of the variable’s value at a given time to the reference value. Such method is suitable as the purpose is to track progress towards reaching full maturity (reference point).

Validation Indicator (I_1)

The first normalised indicator is I_1 , the *Validation Indicator*, which refers to the ratio of the number of validated assumptions to the total number of assumptions made. The number of assumptions serves as a proxy for estimating the extent of the knowledge gap, where validated assumptions indicate progress towards closing the gap. The reference point here is the case where all assumptions have been validated (i.e. *Number of validated assumptions = Total number of assumptions*). I_1 can be calculated using Equation 5.1.

$$I_1 = \frac{\#Validated\ Assumptions}{\#All\ Assumptions} \quad (5.1)$$

Confidence Indicator (I_2)

The second normalised indicator is I_2 , the *Confidence Indicator*, which refers to the average of the *Level of Confidence* in non-validated assumptions. I_2 would thus indicate the status of the remaining assumptions still awaiting evaluation. The reference point here is the case where all assumptions have been validated.

In order to obtain a quantitative assessment, an ordinal conversion is used where $LoC = Low \rightarrow 10\%$, $LoC = Moderate \rightarrow 50\%$ and $LoC = High \rightarrow 90\%$.

I_2 can be calculated using Equation 5.2.

$$I_2 = \begin{cases} \frac{1}{n} \times \sum_{i=1}^n LoC(\text{assumption}_i), & \text{if } n > 0 \\ 1, & \text{if } n = 0 \end{cases} \quad (5.2)$$

where n refers to the number of non-validated assumptions.

$I_2 = 1$ when the set of all assumptions has been narrowed down to a set of validated assumptions.

Consistency Indicator (I_3)

The third normalised indicator is I_3 , the *Consistency Indicator*, which is the complement of *Conflict Density* (Δ). Δ refers to the ratio of the number of *Conflict* edges with respect to the maximum possible number of conflicts. The latter corresponds to the worst-case scenario where each assumption conflicts with every other assumption. Therefore, I_3 provides an indication of how far we are from such a scenario (i.e. the reference point corresponds to: *number of Conflict edges = maximum possible number of conflicts*).

I_3 can be calculated using Equation 5.3.

$$I_3 = 1 - \Delta = \begin{cases} 1 - \frac{2|E|}{|V|(|V|-1)}, & \text{if } |V| > 1 \\ 1, & \text{otherwise} \end{cases} \quad (5.3)$$

where $|E|$ is the number of *Conflict* edges, $|V|$ is the number of non-validated *Assumption* nodes from the DBN, and $\frac{|V|(|V|-1)}{2}$ is the maximum possible number of conflicts (according to the *graph density* measure [168]).

$0 \leq I_3 \leq 1$, where $I_3 = 1$ would mean that all conflicts have been resolved (i.e. $|E| = 0$), and $I_3 = 0$ refers to the extreme case where each assumption conflicts with every other assumption (i.e. $|E| = \frac{|V|(|V|-1)}{2}$).

5.2.3 Weighting

In this step, individual indicators are assigned weights before aggregating them into a composite indicator. There are three main categories of weighting approaches [169]:

1. **Equal Weighting:** All the indicators are given the same weight.
2. **Statistical Approach:** According to the OECD [166], “*weights may also be chosen to reflect the statistical quality of the data*”. Thus, different statistical analysis methods (such as *Principal Components Analysis* and *Regression Analysis*) can be used to assign weights.
3. **Weighting Based on Judgement:** Weights are assigned based on expert opinion.

However, since the selected indicators are correlated (e.g. an increase in I_1 would result in decreasing the *number of non-validated assumptions* parameter in both I_2 and I_3), assigning the same weight to them could lead to a double-counting of their effect. Therefore, equal weighting is considered as not applicable to the KMI. Furthermore, due to the absence of relevant data on assumption management from industry, the statistical approach is not applicable to the KMI.

According to the OECD [166], the *Analytic Hierarchy Process* (AHP) is a commonly used technique in the context of multi-attribute decision-making, which enables decomposing a problem into a hierarchical structure facilitating evaluation through pairwise comparisons. Forman [170] argues that AHP allows decision-makers to derive weights instead of arbitrarily assigning them, via the logical application of data, experience and intuition. In their discussion on the weighting of composite indicators, Greco *et al.* [171] argue that AHP leads to weights that are less susceptible to errors of judgement. Thus, AHP (as a *Weighting Based on Judgement* approach) is adopted for the KMI.

AHP is essentially an ordinal pairwise comparison of the individual indicators, by answering the following questions: (1) *Which of the two indicators is more important?* (2) *By how much is it more important?*

Intensity of importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
2	Weak	Experience and judgment slightly favor one activity over another
3	Moderate importance	
4	Moderate plus	
5	Strong importance	Experience and judgment strongly favor one activity over another
6	Strong plus	An activity is favored very strongly over another; its dominance demonstrated in practice
7	Very strong or demonstrated importance	
8	Very, very strong	The evidence favoring one activity over another is of the highest possible order of affirmation
9	Extreme importance	
Reciprocals of above	If activity i has one of the above nonzero numbers assigned to it when compared with activity j , then j has the reciprocal value when compared with i	

Table 5.1: Scale to select the intensity of importance [172]

To answer the first question: I_1 should be the dominating indicator since validating assumptions gives a direct and explicit indication of progress made in closing the knowledge gap. Thus, I_1 is more important than I_2 , and I_1 is more important than I_3 . However, the comparison of I_2 and I_3 is not as obvious. Therefore, I_2 and I_3 shall be considered as equally important at this point, where the potential variability introduced by this choice will be discussed in the evaluation (Section 6.4.2).

The answer to the second question is based on the fundamental scale of relative importance proposed by Saaty and Vargas [172], as summarised by Table 5.1.

Since there is currently no evidence from practice, it would not be possible to assign an intensity of importance higher than 5. Thus, in order to preserve the aforementioned pairwise importance comparison, the resulting comparison matrix is as follows:

$$\begin{array}{c}
 I_1 \quad I_2 \quad I_3 \\
 \begin{array}{c}
 I_1 \\
 I_2 \\
 I_3
 \end{array}
 \begin{bmatrix}
 1 & 4 & 4 \\
 \frac{1}{4} & 1 & 1 \\
 \frac{1}{4} & 1 & 1
 \end{bmatrix}
 \end{array}$$

$$\begin{array}{c}
 I_1 \quad I_2 \quad I_3 \quad | \quad \text{Mean} \\
 I_1 \left[\begin{array}{ccc|c} \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{array} \right] \\
 I_2 \\
 I_3
 \end{array}$$

Therefore, the weights corresponding to the individual factors are the following:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{6} \\ \frac{1}{6} \end{pmatrix}$$

where ω_1 , ω_2 and ω_3 correspond to I_1 , I_2 and I_3 , respectively. Note that the weights have been calculated based on the author's best understanding and intuition, and that future research involving a real industrial setting would be required for validation.

5.2.4 Aggregation

This step is about combining the individual indicators into a composite indicator through an aggregation function. One of two approaches is generally used: linear (or additive) aggregation and geometric (or multiplicative) aggregation. Although linear aggregation is the simplest and most widely used, it has some requirements and properties which are not applicable to all settings. One of the main properties is that linear aggregation implies high compensability (e.g. if two indicators I_1 and I_2 are summed, a high value of I_1 would compensate for a low value of I_2). In the case of the KMI, high compensability is not desired as it would not be appropriate to consider, for instance, that a high value of *Consistency* (I_3) would compensate for a low value of *Validation* (I_1). In contrast, compensation in geometric aggregation is limited and acceptable which, according to Gan *et al.* [169], is due to the fact that the *geometric-arithmetic means inequality*¹ “limits the ability of indicators with very low scores to be fully compensated for by indicators with high scores”.

¹The interested reader is referred to [174].

A commonly used geometric aggregation function is the *weighted geometric mean function*, which is defined by Equation 5.4:

$$CI = \prod_{k=1}^n (I_k)^{\omega_k} \quad (5.4)$$

where I_k is the k^{th} indicator, ω_k is the weight corresponding to I_k , and n is the number of indicators. Zhou and Ang [175] compared different aggregation functions according to an information loss criterion called *Shannon-Spearman Measure*, and they found that the *weighted geometric mean function* is associated with the minimum information loss in the construction of composite indicators.

Therefore, KMI can be formulated as follows:

$$KMI = \prod_{k=1}^3 (I_k)^{\omega_k} = I_1^{\frac{2}{3}} \times I_2^{\frac{1}{6}} \times I_3^{\frac{1}{6}} \quad (5.5)$$

5.2.5 Link to other Maturity Assessments

This step from OECD's methodology is about how the KMI relates to the most common maturity assessments, i.e. (a) *design maturity* and (b) *technology maturity*.

Design Maturity

Design maturity is “based on the percentage of releasable design drawings” [176]. In fact, the U.S. Government Accountability Office (GAO) considers that design maturity is reached (*i.e. design maturity is assigned a value of 1*) when 90% of the design drawings have been released, which is recommended to occur by the Critical Design Review [177]. A similar notion, i.e. task completion, is employed by different organisations, such as BAE Systems which uses the completion of required design tasks to objectively quantify design maturity [178]. Furthermore, Harrison [178] argues that design maturity should consider not only task completion, but also the risk of change for a more accurate assessment.

If the design maturity is based only on task completion (as defined by GAO), then releasing design drawings or successfully completing design reviews imply that the underlying assumptions would have been peer-reviewed, and subsequently validated. Thus, both Knowledge Maturity and Design Maturity would increase, which makes them positively correlated. If the design maturity considers the risk of change as well, then the remaining assumptions and their dependencies can inform the risks of potential changes due to assumptions being invalid. In this case, the KMI (and its individual indicators) could complement design maturity assessment.

Technology Maturity

The maturity of individual technologies has been assessed since the 1980's using *Technology Readiness Levels* (TRL). The maturity of integrated technologies into a system has later been assessed using approaches like *System Readiness Levels* (SRL) [179]. On one hand, TRL (or SRL) are associated with the solutions in the Logical Domain of a system architecture. On the other hand, assumptions linked to the Logical Domain elements are captured within the DBN, thus influencing the KMI. Since the feasibility of some technologies and their readiness within some timeframe can be assumed early in the design process, increasing the TRL (or SRL) implies increasing the LoC of (or validating) assumptions underlying the aforementioned technologies. This in turn increases the KMI. Therefore, Knowledge Maturity (via the KMI) and Technology Maturity (via TRL or SRL) are positively correlated.

5.2.6 Interpretation and Visualisation

This final step is about discussing the interpretation of the constructed composite indicator and its visualisation. It is important to note that, as discussed previously, the KMI is meant to provide an indication rather than an exact measurement. That is because the KMI was partly constructed based on ordinal data (i.e. ordinal conversion in I_2), which means that it is not possible to attach a meaning to the resulting value. However, since ordinal data

allow ranking, it is possible to use the KMI to assess progress over time (e.g. $KMI_{t=t1} > KMI_{t=t0}$). Furthermore, the KMI can allow to compare the knowledge maturity of the current project to that of other projects in the portfolio or past projects.

To support decision-making, there is a need for visualising knowledge maturity. According to Johansson *et al.* [128], there is actually a need for visualising both the current level of maturity and alternatives comparison. Ultimately, “*a team needs to see the influence that its individual and collective actions have on the knowledge maturity and the project*” [128].

To illustrate changes of a composite indicator across time, line charts can be used [166]. Figure 5.1 illustrates progress of KMI over time for the current project, whereas Figure 5.2 illustrates a comparison of the current project with other projects (either from the current portfolio or past projects).

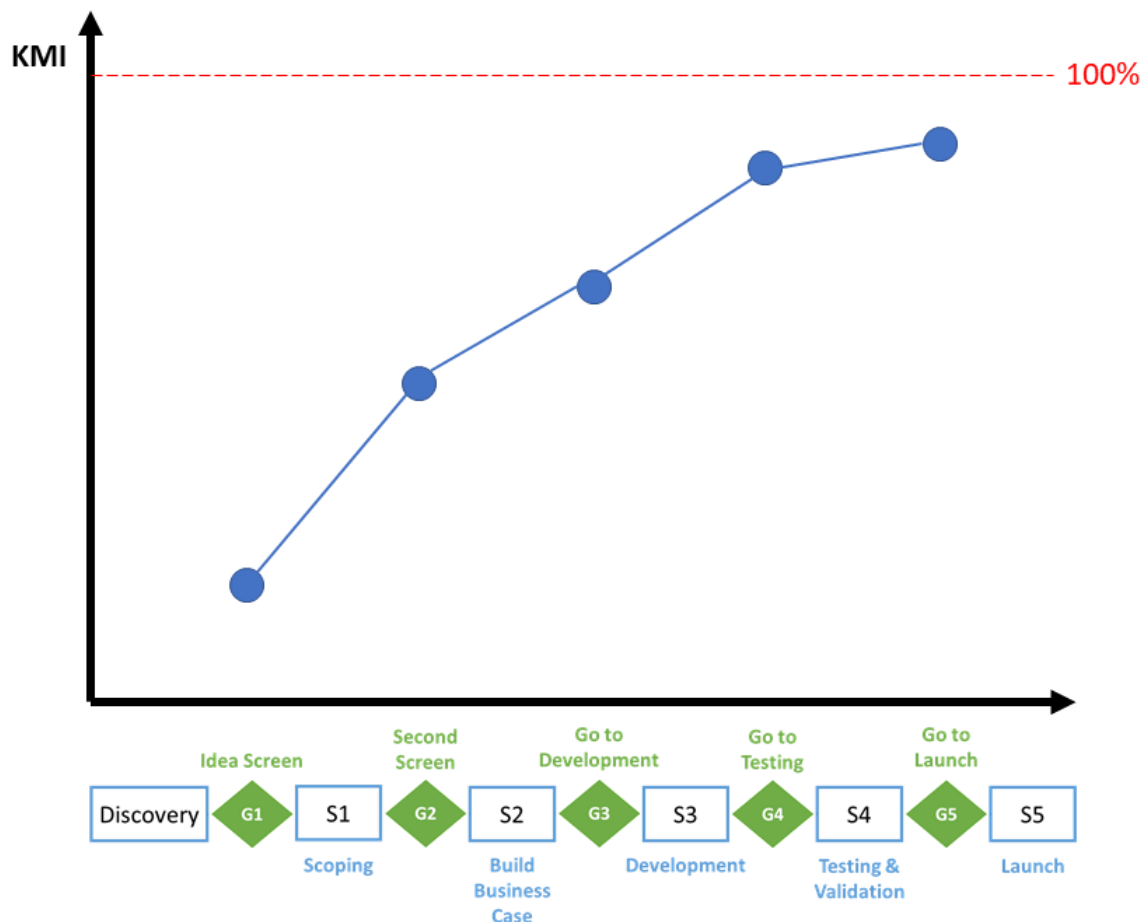


Figure 5.1: Notional progress of KMI over time ($G = Gate$, $S = Stage$)

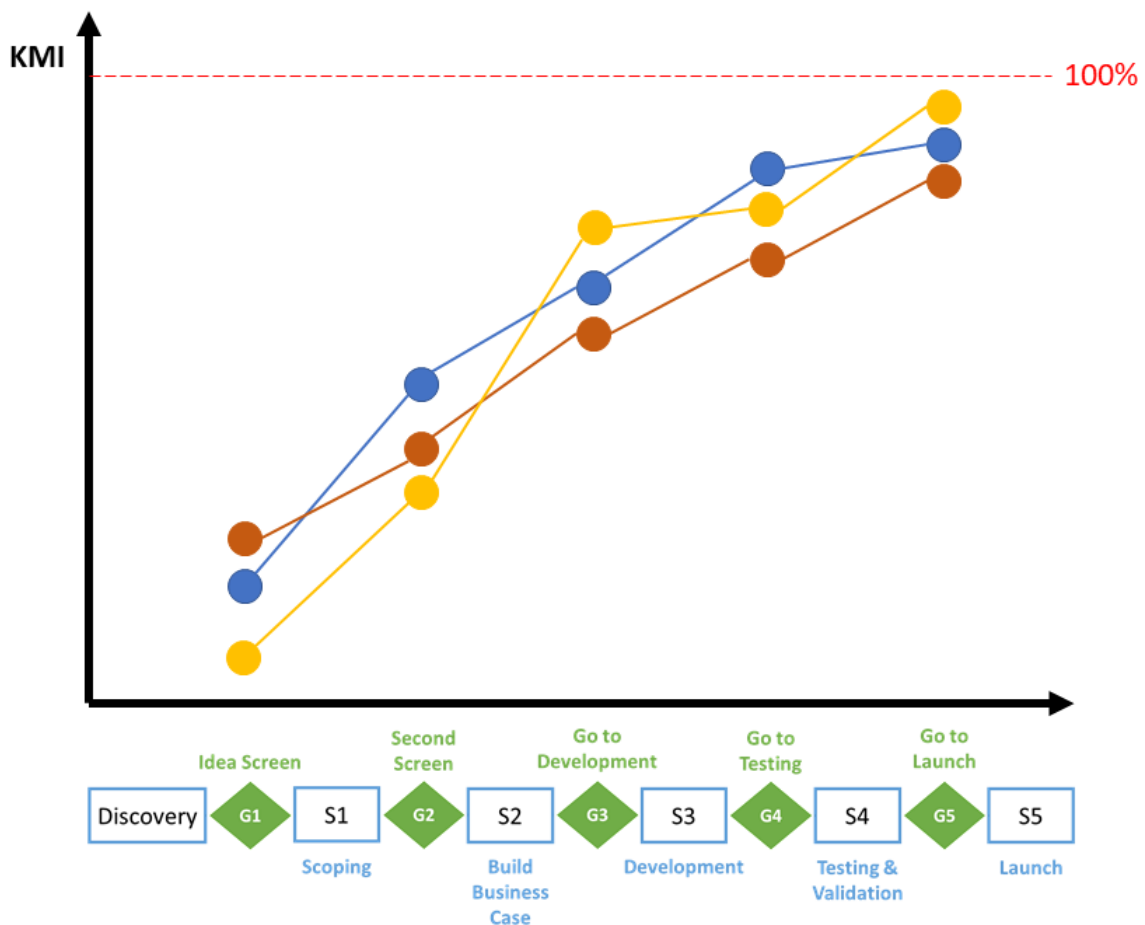


Figure 5.2: Notional comparison of KMI (current project shown in blue) ($G = Gate$, $S = Stage$)

Furthermore, decomposing the KMI into its influencing factors is also important for identifying which specific areas of knowledge quality need improvement in order to increase maturity. Thus, the use of gauges is proposed to visualise the individual indicators, whereas a radar chart is suggested to visualise how the *LoC* criteria (Section 4.3.3) have been assessed on average. This is believed to inform the decision about which aspect of the background knowledge should be strengthened. These visualisation methods are illustrated in Figure 5.3.

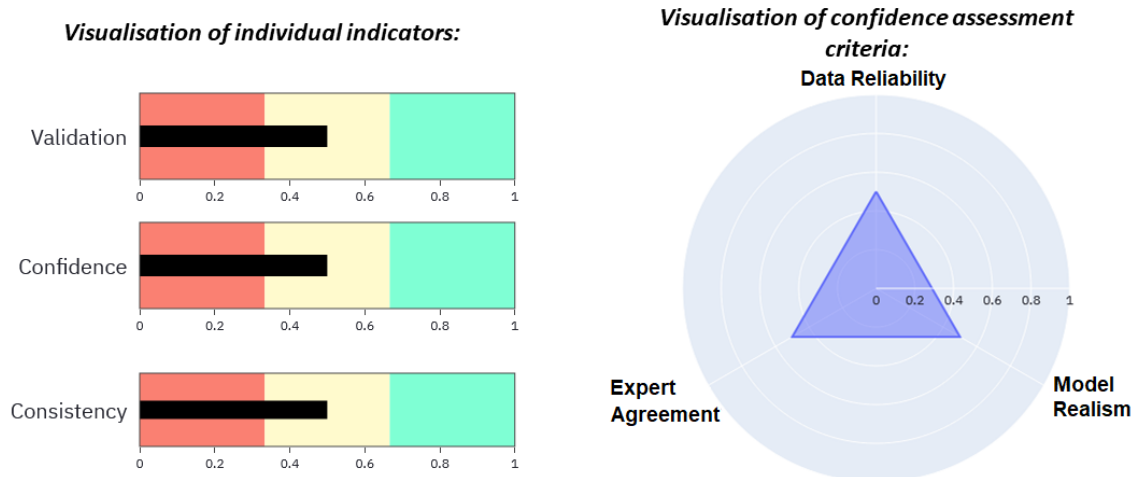


Figure 5.3: Visualisation of the individual constituents of Knowledge Maturity

5.3 Assumption Matrix

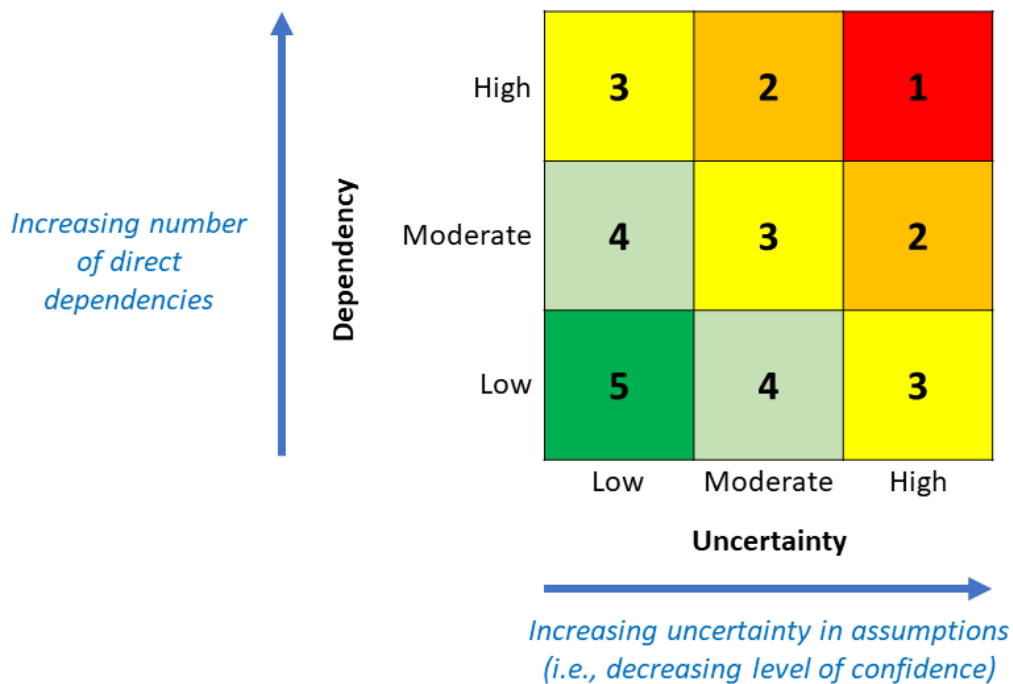
After obtaining an indication of the overall risk of change via knowledge maturity assessment, the goal is then to determine which assumptions represent the biggest threats. Thus, the *Assumption Matrix* is proposed to prioritise assumptions in terms of their risk to initiate change.

As shown in Figure 5.4, the two dimensions of the matrix consist of:

1. *Uncertainty*: refers to the lack of confidence in the assumption. Thus, the higher the uncertainty, the higher the likelihood of an assumption becoming invalid, and therefore initiating a change.
2. *Dependency*: refers to the nodes in the DBN that are directly linked to the assumption. Thus, a higher dependency means that more architectural elements can be directly affected by the invalid assumption.

The matrix allows therefore to categorise all captured assumptions into five risk priority categories, where *Category 1* refers to the highest risk.

Uncertainty is considered to be the complement of the *Level of Confidence*. For instance, an assumption with a **Low** *LoC* would be considered to have a **High** *Uncertainty*. *Dependency* can be measured by the number of edges connected to an assumption node.

Figure 5.4: Concept of the *Assumption Matrix*

An assumption node with the maximum number of edges in the DBN is considered to have a **High** *Dependency*, whereas an assumption node with the minimum number of edges in the DBN is considered to have a **Low** *Dependency*. Any assumption with an intermediate number of edges would be considered to have a **Moderate** *Dependency*. Note that assumptions in a *Conflict* relationship (cf. intra-domain dependency in Section 4.3.2) are by default assigned to the *Category 1* risk priority due to the certainty that at least one of the conflicting assumptions is invalid.

The procedure to generate an *Assumption Matrix* is outlined in Algorithm 4. *MinDependency* in line 2 refers to the minimum number of assumption dependencies in the DBN, whereas *MaxDependency* in line 3 refers to the maximum number of assumption dependencies.

In lines 4-7, *Category 1* risk priority corresponds to either assumptions that are in a *Conflict* relationship, or assumptions that have a *Low LoC* and the maximum number of dependencies.

In lines 8-10, *Category 2* risk priority corresponds to either assumptions that have a *Low*

LoC and an intermediate number of dependencies, or assumptions that have a *Moderate LoC* and the maximum number of dependencies.

In lines 11-13, *Category 3* risk priority corresponds to either assumptions that have a *Low LoC* and the minimum number of dependencies, or assumptions that have a *Moderate LoC* and an intermediate number of dependencies, or assumptions that have a *High LoC* and the maximum number of dependencies.

In lines 14-16, *Category 4* risk priority corresponds to either assumptions that have a *High LoC* and an intermediate number of dependencies, or assumptions that have a *Moderate LoC* and the minimum number of dependencies. Finally, in lines 17-19, the remaining assumptions are assigned the *Category 5* risk priority.

One practical implication of the *Assumption Matrix* is the ability to visualise the assumption prioritisation directly from the architectural model, through the R-F-L views. Due to the fact that inter-domain dependencies are captured within the DBN, it is possible to use some visual indication (e.g. change the colour of the architectural object to match the category of risk priority) within a model-based design tool. Such capability is demonstrated in Section 6.3, where for instance a red-coloured component would mean that it is linked to a *Category 1* assumption. Thus, the component has a high risk of experiencing change due to an invalid assumption.

The assumption prioritisation can also help with addressing the issue of scalability when the developed methods are applied in a real industrial setting. Thus, an organisation can decide to focus on high priority assumptions only in order to reduce to reasonable amounts the time and cost associated with the methods proposed in this thesis.

Algorithm 4: Assumption Matrix Generation

input : *DBN* as a graph G , where the nodes correspond to the assumptions and elements of the $R - F - L - C$ domains, and edges correspond to the dependencies.

output: *Assumption Matrix*, such that each assumption is assigned a category of risk priority.

```

1 begin
2    $MinDependency \leftarrow$  Minimum number of assumption dependencies in  $G$ 
3    $MaxDependency \leftarrow$  Maximum number of assumption dependencies in  $G$ 
4   for each Assumption  $\alpha$  in  $G$  do
5     if ( $\exists$  Conflict edge  $\in \alpha.dependency$ ) or ( $\alpha.confidence == Low$  and
6        $Size(\alpha.dependency) == MaxDependency$ ) then
7        $Category(\alpha) \leftarrow 1$ 
8     end if
9     else if ( $\alpha.confidence == Low$  and
10       $MinDependency < Size(\alpha.dependency) < MaxDependency$ ) or
11      ( $\alpha.confidence == Moderate$  and
12       $Size(\alpha.dependency) == MaxDependency$ ) then
13        $Category(\alpha) \leftarrow 2$ 
14     end if
15     else if ( $\alpha.confidence == Low$  and
16       $Size(\alpha.dependency) == MinDependency$ ) or
17      ( $\alpha.confidence == Moderate$  and
18       $MinDependency < Size(\alpha.dependency) < MaxDependency$ ) or
19      ( $\alpha.confidence == High$  and
20       $Size(\alpha.dependency) == MaxDependency$ ) then
21        $Category(\alpha) \leftarrow 3$ 
22     end if
23     else if ( $\alpha.confidence == High$  and
24       $MinDependency < Size(\alpha.dependency) < MaxDependency$ ) or
25      ( $\alpha.confidence == Moderate$  and
26       $Size(\alpha.dependency) == MinDependency$ ) then
27        $Category(\alpha) \leftarrow 4$ 
28     end if
29     else
30        $Category(\alpha) \leftarrow 5$ 
31     end if
32   end for
33 end

```

5.4 Margin Allocation

Before presenting the algorithm supporting margin allocation analysis (Algorithm 5), it is important to first clarify what is meant by margin redundancy in the present context. Figure 5.5 illustrates a simple computational model for initially estimating the aspect ratio (AR) as a function of maximum Mach (M_{Max}). The value of M_{Max} is set by a requirement, whereas AR results from a computation. What could happen is that a margin can be assigned to M_{Max} because of requirement uncertainty, whereas another margin can be assigned to AR due to uncertainty in both the model itself and the model input (i.e. M_{Max}). Therefore, the two margins would be double-accounting for the uncertainty in M_{Max} . Although this could seem trivial due to the simplicity of the example, the model would in practice be part of a large computational workflow, where manually identifying instances of margin redundancy would be much more challenging. A computational workflow comprising the model in Figure 5.5 is presented in the demonstration use case (Section 6.3.1).

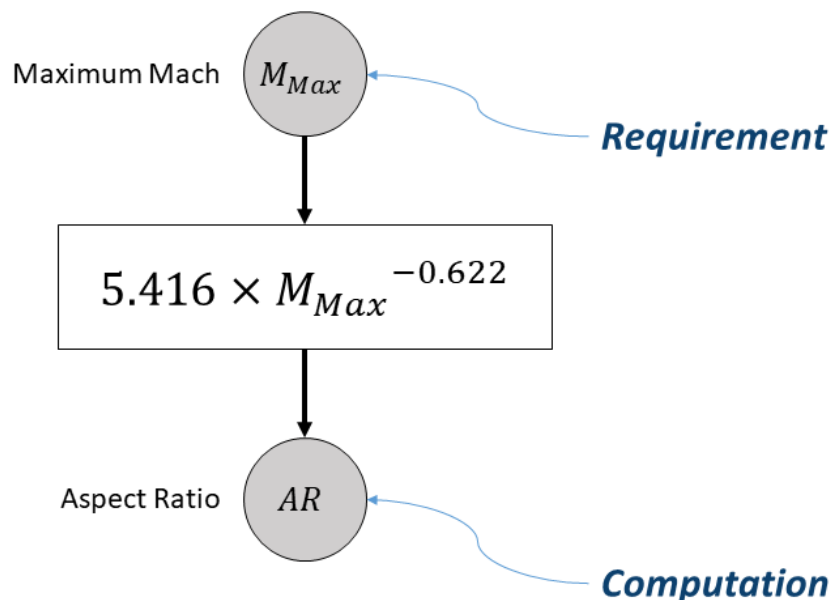


Figure 5.5: Margin Redundancy Example (Computational model from [41])

Algorithm 5 allows to provide the status of margin allocation by using information

contained within the *DBN*. This means that the user is notified about both the assumptions that have not been explicitly associated with margins (i.e. assumptions for which the risk is not explicitly mitigated), and instances of margin redundancy (as described earlier).

In lines 2-4, the *Assumption*, *Margin* and *Model* nodes are extracted from the *DBN* to be stored in separate lists. In line 5, an empty list is initialised to store all assumptions that have been associated with margins.

In line 6, a *For* loop over all margins is initiated, where assumptions that were explicitly associated with some margin *M* by the user are appended to the *mitigatedAssumptions* attribute of *M* (line 8) and the *Mitigated* list (line 9). Furthermore, if the parameter linked to *M* is detected as a root node (i.e. has no parent node) (line 11), then it is interpreted as an independent parameter which value is set by a requirement. Thus, the potential presence of margin redundancy as described earlier can be detected (lines 12-16), where the independent parameter would be an input to a computational model, and there is another margin on the latter's output.

In lines 19-23, a *For* loop over all assumptions is initiated, where the user is notified about any assumption that has not been explicitly associated with a margin. This is to inform future needs for margin allocation, especially with respect to high priority assumptions (refer to Section 5.3 for assumption prioritisation).

After margins have been allocated, the explicit association between margins and assumptions for which they are mitigating the risk can be visualised using a *Sankey Diagram*, as illustrated by Figure 5.6. This should provide practitioners with an overall view of how the different margins have been allocated to mitigate the risks of assumptions being invalid.

Algorithm 5: Margin Allocation Analysis

input : *DBN* as a graph G , where the nodes correspond to the assumptions and elements of the $R - F - L - C$ domains, and edges correspond to the dependencies.

output: Update the *Margin* nodes in G to include the new assumption associations. Notify user about unmitigated risks and detected instances of margin redundancy.

```

1 begin
2   Assumptions  $\leftarrow$  List of all Assumption nodes in  $G$ 
3   Margins  $\leftarrow$  List of all Margin nodes in  $G$ 
4   Models  $\leftarrow$  List of all Model nodes in  $G$ 
5   Mitigated  $\leftarrow$  Initialise list of all mitigated assumptions
6   for each  $M$  in Margins do
7     if an Assumption  $\alpha$  has been explicitly associated with  $M$  by user then
8       Append  $\alpha$  to  $M$ .mitigatedAssumptions;
9       Append  $\alpha$  to Mitigated
10    end if
11    if hasParentNode( $M$ .parameter) == False then
12      for each  $Mod$  in Models do
13        if  $Mod$ .Input ==  $M$ .parameter and
14           $\exists M' \in$  Margins :  $Mod$ .Output ==  $M'$ .parameter then
15            Notify user about potential redundancy among margins  $M$  and
16               $M'$ 
17            end if
18          end for
19        end if
20      end for
21    for each  $\alpha$  in Assumptions do
22      if  $\alpha \notin$  Mitigated then
23        Notify user about unmitigated risk relating to  $\alpha$ 
24      end if
25    end for
26  end

```



Figure 5.6: Sankey diagram to visualise margin-assumption associations

5.5 Change Absorber Localisation

Once margins are assigned as a risk mitigation strategy, the following method allows to detect the closest margin to an assumption that can absorb change. A potential change propagation path is suggested, which starts from an assumption (acting as a *Change Initiator*), and ends at a margin (acting as a *Change Absorber*). Figure 5.7 illustrates a simple example of a propagation path. Basically, change initiated by the assumption *BFT Sizing* could potentially propagate by first causing change in the bladder fuel tank, which in turn can affect the fuselage. Since there is a margin on the fuselage volume (so that the extra volume can accommodate future changes), this margin is detected as a change absorber through the dependencies between the Logical and Computational domains (i.e. the link between the fuselage component and the fuselage volume parameter). Note that a propagation path can also contain *Requirement* and *Function* objects.

This capability to detect a change absorber is made possible because both assumptions and margins are explicitly captured as part of the *DBN*. In fact, the *DBN* contains edges between conflicting assumptions, edges between assumptions and margins for which they are explicitly mitigating the risk, and edges between assumptions and elements of the

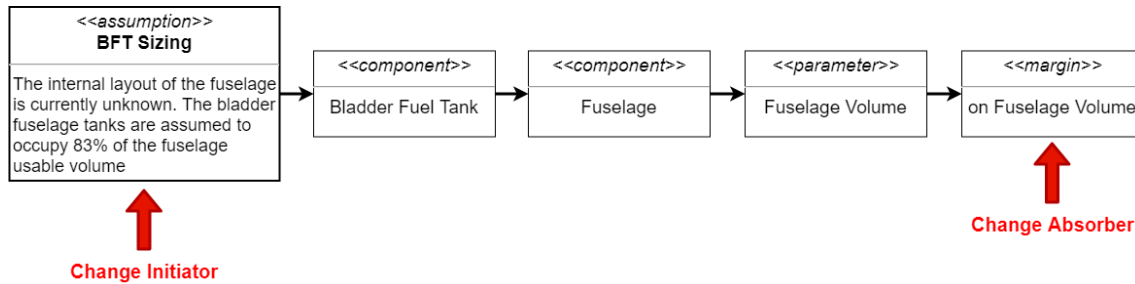


Figure 5.7: Example of a propagation path

$R - F - L - C$ domains. Furthermore, a *Status* attribute was added to the *Component* class which records information on whether the solution has been frozen. A component is frozen when its design is fixed, and can only be changed if the integrity of the product is threatened [180].

Therefore, the extent to which the *impact* of change can be assessed is reliant on the aforementioned dependencies, where it is possible to notify about:

- which assumptions are conflicting with the *change initiator*;
- which assumptions are associated with elements along the propagation path;
- which components along the propagation path are *frozen*; and
- which margins need to be revised, and whether they should be increased/reduced.

This is covered in Section 5.6 on margin revision.

As opposed to the existing change propagation methods that are restricted to the logical domain, the proposed approach traverses all domains. Therefore, whether an assumption is related to a requirement, a solution, or any other element, a change absorber can be found by traversing the entire network.

The proposed approach is described in Algorithm 6. First, Dijkstra's algorithm [181], a search algorithm for finding shortest paths between vertices in a graph, is used to find the shortest path between an *assumption* (i.e. *change initiator*) and a *margin* (i.e. *change absorber*) in the *DBN*. In Algorithm 6, the application of Dijkstra's method to find the shortest path from an *assumption* to a *margin* is referred to as *DijkstraPath(assumption,*

margin). For the interested reader, Dijkstra's algorithm and its description can be found in [38, p. 82].

In lines 2-3, empty lists *ListOfPaths* and *AffectedAssumptions* are initialised. *ListOfPaths* stores the shortest path (via Dijkstra's algorithm) from the *change initiator* assumption to each margin in the *DBN*, whereas *AffectedAssumptions* stores all the assumptions that can be affected along the propagation path.

In lines 4-12, the algorithm checks first if the *change initiator* assumption is explicitly associated with a margin, thus meaning that the latter could potentially absorb the change before it propagates. Otherwise, the shortest path from the *change initiator* assumption to each margin in the *DBN* ($DijkstraPath(\alpha_{source}, M)$) is determined and added to *ListOfPaths*.

In line 13, the overall shortest path (*ShortestPath*) is defined as the path within *ListOfPaths* that has the minimum number of nodes. Then, in line 14, all the nodes in *ShortestPath* that correspond to *components* are added to *ShortestPath_Components*.

In lines 16-25, the algorithm checks whether *ShortestPath* is actually valid since it could correspond to a margin that is completely unrelated to the *change initiator* assumption and cannot possibly act as a change absorber. For instance, the margin is on the internal missiles bay, whereas the *change initiator* assumption is associated with the aircraft tail. Thus, the validity of *ShortestPath* is evaluated by finding out whether the components in *ShortestPath_Components* are hierarchically related, i.e. use the recorded information on *Parent-Child* relationships from the *DBN*. If *ShortestPath* turns out to be valid, the parameter *PathValid* takes the value of *True*, therefore ending the *while* loop. Otherwise, the invalid *ShortestPath* is removed from *ListOfPaths*, and the commands in lines 13 and 14 are repeated.

In lines 26-34, if *ListOfPaths* is not empty, it would mean that a valid *ShortestPath* is found, and thus returned. Otherwise, all paths in *ListOfPaths* would be invalid, meaning that there is no margin acting as a change absorber. Therefore, change could potentially propagate to the top (Aircraft) level.

Algorithm 6: Change Absorber Localisation

input : *DBN* as a graph G , and assumption α_{source} representing the change initiator.

output: *ShortestPath* from α_{source} to the closest margin M_{CA} acting as a change absorber, and *AffectedAssumptions* containing affected assumptions.

```

1 begin
2   ListOfPaths  $\leftarrow$  Initialising empty list
3   AffectedAssumptions  $\leftarrow$  Initialising empty list
4   for each Margin M in  $G$  do
5     if  $\alpha_{\text{source}}$  in set of assumptions explicitly mitigated by margin  $M$  then
6       Notify user that  $\alpha_{\text{source}}$  is already mitigated by  $M$ ;
7       break
8     end if
9     else
10      ListOfPaths  $\leftarrow$  Append DijkstraPath( $\alpha_{\text{source}}, M$ )
11    end if
12  end for
13  ShortestPath  $\leftarrow$  DijkstraPath( $\alpha_{\text{source}}, M_{\text{CA}}$ ), such that for all paths in
    ListOfPaths: DijkstraPath( $\alpha_{\text{source}}, M_{\text{CA}}$ ) has the minimum number of
    nodes
14  ShortestPath.Components  $\leftarrow$  List of all nodes in ShortestPath that refer to
    Component nodes in the DBN
15  PathValid  $\leftarrow$  False
16  while PathValid == False and ListOfPaths  $\neq \emptyset$  do
17    if Component nodes in ShortestPath.Components are hierarchically
    related then
18      PathValid  $\leftarrow$  True
19    end if
20    else
21      ListOfPaths  $\leftarrow$  ListOfPaths – ShortestPath;
22      ShortestPath  $\leftarrow$  Next shortest path in ListOfPaths;
23      ShortestPath.Components  $\leftarrow$  List of all nodes in ShortestPath that
    refer to Component nodes in the DBN
24    end if
25  end while
26  if ListOfPaths  $\neq \emptyset$  then
27    for each node in ShortestPath do
28      AffectedAssumptions  $\leftarrow$  Append assumption(s) associated with
    node
29    end for
30    return (ShortestPath, AffectedAssumptions)
31  end if
32  else
33    Notify user that there is no margin acting as a change absorber. Change
    could potentially propagate to the Aircraft level
34  end if
35 end

```

Additionally, the assumption dependencies as recorded in the *DBN* are used to determine all assumptions that could be affected along the propagation path. This includes conflicting assumptions because if an assumption α_1 is conflicting with another assumption α_2 , and α_1 is invalidated, then the risk associated with α_2 should decrease.

Furthermore, the *Status* attribute of each component within the returned *ShortestPath* can be checked to determine whether that component has been frozen. Such information is also expected to support the assessment of change impact, in the sense that frozen components are more costly to change.

5.6 Margin Revision

After margins have been allocated to implement the risk mitigation strategy, the next step of the risk management process is to keep monitoring risks, and thus providing feedback to the previous risk management steps (i.e. risk identification, assessment and mitigation). To this end, an algorithm supporting margin revision is devised (Algorithm 7), which suggests margins for revision as the design progresses and more knowledge is acquired. Specifically, the algorithm monitors changes in the *Assumption* nodes of the *DBN* to detect margins for revision. For instance, if the level of confidence in an assumption increases, the algorithm would suggest to reduce the associated margin since there is a lower risk for the assumption to be invalid. However, the margins cannot be arbitrarily and independently changed without risking constraint violation. Thus, before presenting Algorithm 7, the *Margin Space* concept that is applied in the case of margin increase is introduced.

Guenov *et al.* [161] proposed an approach for interactive margin management, which allows to explore the effects of margins on: other margins, performance, and probabilities of constraint satisfaction. Their approach is based on a concept called *Margin Space*, which is a hypercube consisting of the ranges of all assigned margins, and is bidirectionally linked to the design space [161]. Once the interval of each margin is defined, the

Margin Space results from the Cartesian product of all margins. It is important to note that margins are assumed to be independent, which means that margin redundancy is not considered. Each point in the *Margin Space* is considered as a margin combination, which feasibility depends on whether the resulting performance meets the constraints. Figure 5.8 illustrates the concept of *Margin Space*, where each point refers to a margin combination, and the isocontour represents a constraint that divides the space into a feasible and an unfeasible part.

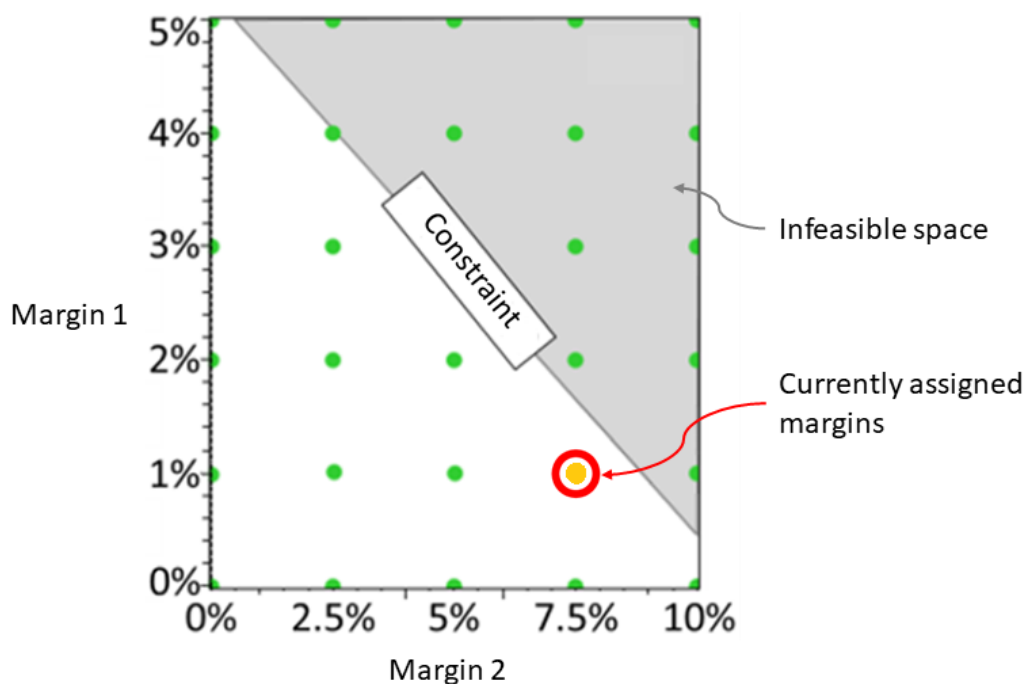


Figure 5.8: Concept of Margin Space

The following is a description of Algorithm 7, which applies when a change is detected in an assumption α , so that different rules are checked to determine the relevant suggestions for revising the associated margin M . In lines 2-3, the *Assumption* and *Margin* nodes are extracted from the *DBN* to be stored in separate lists.

If the level of confidence in α increases, or the latter is validated, the corresponding margin M is suggested to be reduced due to a lower risk for the assumption to be invalid (lines 5-7). Whereas if the level of confidence in α decreases, M is suggested to be increased due to a higher risk for the assumption to be invalid, and this margin increase can be in-

formed by the use of the *Margin Space* (lines 8-11). Moreover, if α is invalidated and replaced by another assumption α' , which is also mitigated by the same margin M , then the user is notified to revise M according to the new assumption (lines 12-17). In practice, such assumption replacement is captured by adding an edge between α' and M , while deleting the edge between α and M .

Regarding margin revision due to changes in assumption conflicts, if α conflicts with an assumption α'' and the level of confidence in α'' increases, then M is suggested to be increased (lines 20-23). This is due to the fact that when two assumptions conflict, only one of the assumptions (at most) can be valid. Thus, increasing the level of confidence in one assumption increases the risk of the other assumption being invalid. Again, the suggested margin increase is supported by the *Margin Space*. However, if the conflict is resolved, the corresponding margins can be reduced (lines 24-26).

In case the user associates a new assumption with an existing margin (i.e. the size of the *mitigatedAssumptions* attribute of the margin increases), the latter is suggested to be increased in order to mitigate an additional risk. Such decision is informed by the use of the *Margin Space* (lines 29-32).

Note that during margin trade-off, if a margin is reduced to 0, it should no longer act as a change absorber in *Change Absorber Localisation* (Section 5.5). Whereas if a margin is increased from 0, it becomes a potential change absorber. Thus, the suggested propagation path may be affected by margin revision. Furthermore, margins related to frozen components should not be traded-off as they are fixed.

Algorithm 7: Margin Revision

```

input : DBN as a graph  $G$ . Assumption  $\alpha$  where change has been detected.
         Margin  $M$  that is associated with  $\alpha$ .
output: Notify user suggesting the relevant revision of  $M$ .

1 begin
2   Assumptions  $\leftarrow$  List of all Assumption nodes in  $G$ 
3   Margins  $\leftarrow$  List of all Margin nodes in  $G$ 
4   // Margin revision due to changes in the level of confidence or status
5   if ( $\alpha$ .confidence increases) or ( $\alpha$ .status == "Valid") then
6     | Notify user to reduce  $M$ 
7   end if
8   else if  $\alpha$ .confidence decreases then
9     | Notify user to increase  $M$ ;
10    | Open Margin Space to inform margin increase
11  end if
12  else if ( $\alpha$ .status == "Invalid") and ( $\alpha$  is replaced by a new assumption  $\alpha'$ )
13    then
14    | Notify user to revise  $M$  according to  $\alpha'$ ;
15    | if  $M$  is to be increased then
16    | | Open Margin Space to inform margin increase
17    | end if
18  end if
19  // Margin revision due to changes in conflicting assumptions
20  for each  $\alpha''$  in Assumptions do
21    | if ( $\alpha''$  conflicts with  $\alpha$ ) and ( $\alpha''$ .confidence increases) then
22    | | Notify user to increase  $M$ ;
23    | | Open Margin Space to inform margin increase
24    | end if
25    | else if Conflict between  $\alpha$  and  $\alpha''$  is resolved then
26    | | Notify user to reduce  $M$ 
27    | end if
28  end for
29  // Margin revision to accommodate new assumptions
30  if Size( $M$ .mitigatedAssumptions) increases then
31    | Notify user to increase  $M$ ;
32    | Open Margin Space to inform margin increase
33  end if
end

```

5.7 Summary and Conclusions

This chapter presented a set of methods that were developed to achieve Objective 2, i.e. to devise methods to manage risk of change due to invalid assumptions, with an explicit consideration of both assumptions and margins.

The first method is about knowledge maturity assessment (Section 5.2), where a composite indicator called the *Knowledge Maturity Index* (KMI) was presented to indicate the overall risk of change due to lack of knowledge. The KMI makes use of assumptions as a proxy to estimate the extent of the knowledge gap, where validating assumptions implies progress towards closing the gap. In addition to the KMI, visualisation techniques were used to support decision-making, and the ability to assess progress of knowledge maturity over time was discussed. This is in an attempt to address a limitation identified in the literature review (cf. Section 3.6), where although assumption management plays a crucial role in knowledge maturity, no method has been previously proposed to assess knowledge maturity with an explicit consideration of assumptions.

The second method is called the *Assumption Matrix* (Section 5.3), which allows to prioritise assumptions in terms of risk to initiate change. Such prioritisation is enabled by both the level of confidence and the dependencies of the captured assumptions. One practical implication of this method is to visualise assumption prioritisation directly from a system model.

The third method, *Margin Allocation* (Section 5.4), provides the status of margin allocation with respect to assumptions. The status of margin allocation includes the following information: (i) captured associations between margins and assumptions; (ii) assumptions, prioritised according to the *Assumption Matrix*, for which the risk has not been explicitly mitigated by margins; and (iii) detected instances of margin redundancy. This tackles a limitation identified in the literature review (cf. Section 3.6), where the explicit relationship between assumptions and margins has not been explored in the published literature.

The fourth method enables *Change Absorber Localisation* (Section 5.5), where an al-

gorithm was devised to detect the closest margin to an assumption that can absorb change. A potential change propagation path is suggested, which starts from an assumption (acting as a *Change Initiator*), and ends at a margin (acting as a *Change Absorber*). This capability is made possible because both assumptions and margins are explicitly captured as part of the *DBN*. Furthermore, the proposed method benefits from the dependencies provided by the *DBN* between assumptions (as *Conflict* relationships), between assumptions and margins for which they are explicitly mitigating the risk, and between assumptions and elements of the $R - F - L - C$ domains (as opposed to the logical domain only in existing methods). Information on already frozen components can be provided due to the addition of a *Status* attribute to the *Component* class. This method attempts to address a limitation identified in the literature review (cf. Section 3.6), where (i) a lack of explicitly considering margins in existing approaches for change propagation prediction has been identified; and (ii) no approach has been proposed to explicitly relate assumptions to change propagation analysis (although assumptions are known to cause change propagation in engineering design).

The fifth method relates to *Margin Revision* (Section 5.6), where an algorithm was devised to detect and suggest margin revisions following changes in assumptions, while using the *Margin Space* concept to ensure constraint satisfaction. This allows to monitor risks associated with assumptions, while updating the risk mitigation strategy accordingly. This method can be seen as providing justification for margin revision as the design progresses and more knowledge is acquired, which addresses a limitation identified in the literature review (cf. Section 3.6), i.e. a lack of tools that track margins along with the rationale underlying their change.

Chapter 6

Evaluation

6.1 Introduction

This chapter reports the conducted evaluation of the proposed approach. As stated in Section 1.4, the research reported in this thesis corresponds to a *Type 3* study of the DRM [20], which consists of a preliminary evaluation since it is not possible to evaluate the proposed approach in a real industrial setting. This preliminary evaluation consists of two parts:

- A *demonstration*, which consisted of applying the developed methods to: (1) the hypothetical design of a fighter aircraft and (2) conflicting assumptions in collaborative design, in order to assess whether the developed methods work as intended.
- An *industry feedback* session, which consisted of demonstrating the developed methods to industry experts to obtain feedback on expected usefulness in practice, thus assessing the impact of this research.

To enable the evaluation of the proposed methods, the latter were implemented into a prototype software tool, which interacts with a software tool called AirCADia¹ [182] for model-based design and analysis.

¹AirCADia is not publicly available outside Cranfield University. The interested reader is invited to contact the Advanced Engineering Design Group at Cranfield University.

The rest of this chapter is organised as follows. Section 6.2 describes the prototype software tool. Section 6.3 describes the use cases, whereas Section 6.4 presents the demonstration results. The industrial feedback session and its results are presented in Section 6.5. Finally, conclusions are drawn in Section 6.6.

6.2 Software Prototype

The proposed methods required a software implementation for their evaluation. To this end, a software prototype tool has been developed. The prototype tool extends an existing tool called AirCADia [182] by implementing features related to assumption management. AirCADia is a software tool for interactive model-based design and analysis, which is used in this evaluation for its architectural definition/assessment and design space exploration capabilities. Figure 6.1 illustrates the software implementation of the prototype tool.

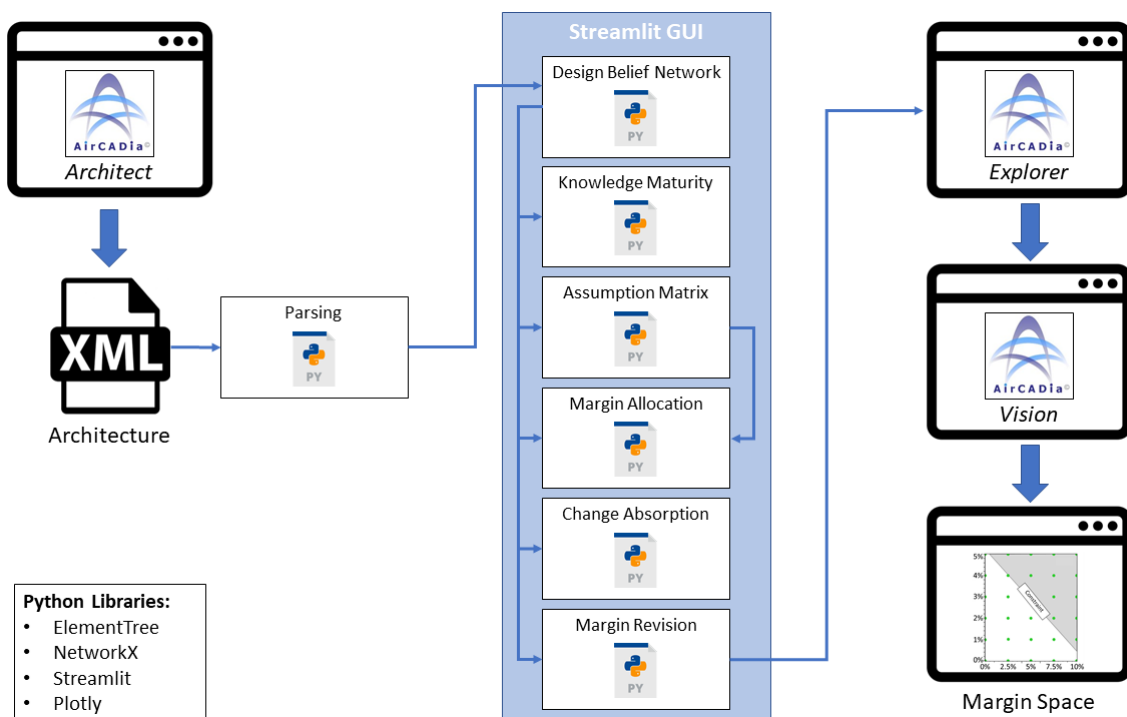


Figure 6.1: Schematic of the software implementation

AirCADia is divided into different modules, which include:

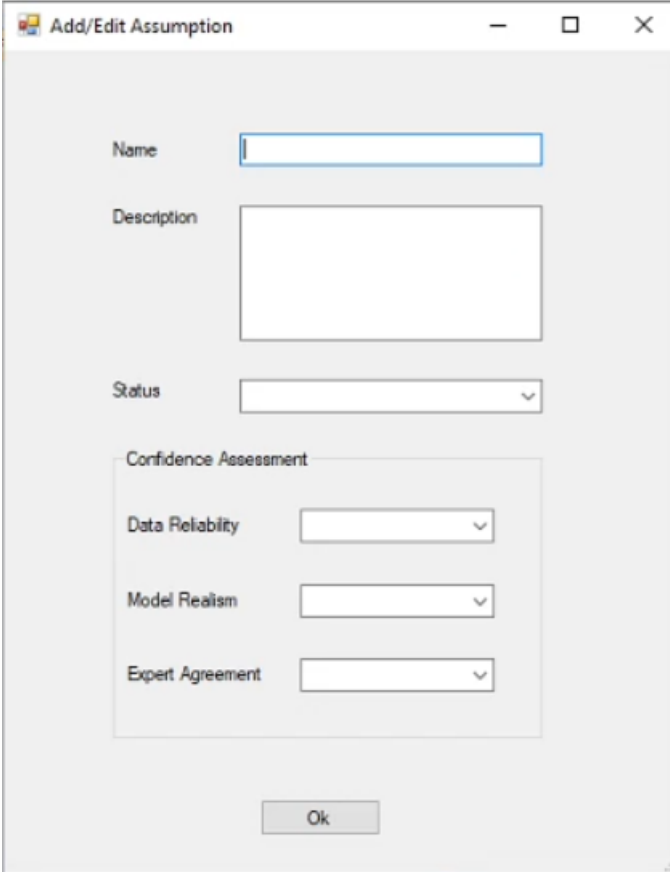
- AirCADia Architect: allows defining the system architecture according to the RFLP model.
- AirCADia Explorer: allows formulating and executing computational design studies.
- AirCADia Vision: allows interactive visualisation for the rapid exploration of potential design solutions.

To demonstrate my proposed methods, some new features related to assumption management have been added to AirCADia Architect:

- To capture a new assumption, the user can right-click on any element from the R-F-L-C domains and select the appropriate command. Figure 6.2 shows the form that captures the information relevant to the assumption data-object.
- The user can link between two assumptions identified as conflicting. The user can also right-click on any element from the the R-F-L-C domains to associate it with an already captured assumption.
- From the *Tools* menu, the user can display a list of all captured assumptions and margins, along with their attribute values. This shall be demonstrated in Section 6.4.
- From the *Tools* menu, the user can select a command that changes the colour of all elements in the requirements, functional and logical views to match the risk prioritisation from the Assumption Matrix method. This shall be demonstrated in Section 6.4.

The system model created in AirCADia Architect is saved as an XML file, which is then passed as an input to a Python script for parsing (*Parsing Script* in Figure 6.1). This script makes use of the *ElementTree XML API*² to extract from the XML file the require-

²<https://docs.python.org/3/library/xml.etree.elementtree.html> (Accessed: 26/11/2021)



The image shows a software dialog box titled "Add/Edit Assumption". It features a title bar with standard window controls (minimize, maximize, close). The main area contains several input fields: a "Name" text box, a "Description" text area, a "Status" dropdown menu, and a "Confidence Assessment" section. This section includes three sub-dropdowns labeled "Data Reliability", "Model Realism", and "Expert Agreement". An "Ok" button is positioned at the bottom center of the dialog.

Figure 6.2: Capturing an assumption in AirCADia Architect

ments, functions, solutions, parameters, computational models, margins and assumptions (along with their attributes), which are then returned as Python dictionaries.

To create the user interface, an open-source Python library called Streamlit³ was selected for its ease of use and integration with Python. Figure 6.3 shows the main page of the user interface.

To create a *DBN*, the prototype tool makes use of *NetworkX*⁴, which is an open source Python package to create and analyse complex networks. The *Design Belief Network* Python script takes as an input the dictionaries returned by the *Parsing Script*, which are then used to create nodes corresponding to requirements, functions, solutions, parameters, computational models, margins and assumptions. The nodes attributes are set as the values of the dictionaries' keys. The graph edges are created by using the dependency

³<https://streamlit.io/> (Accessed: 26/11/2021)

⁴<https://networkx.org/> (Accessed: 26/11/2021)

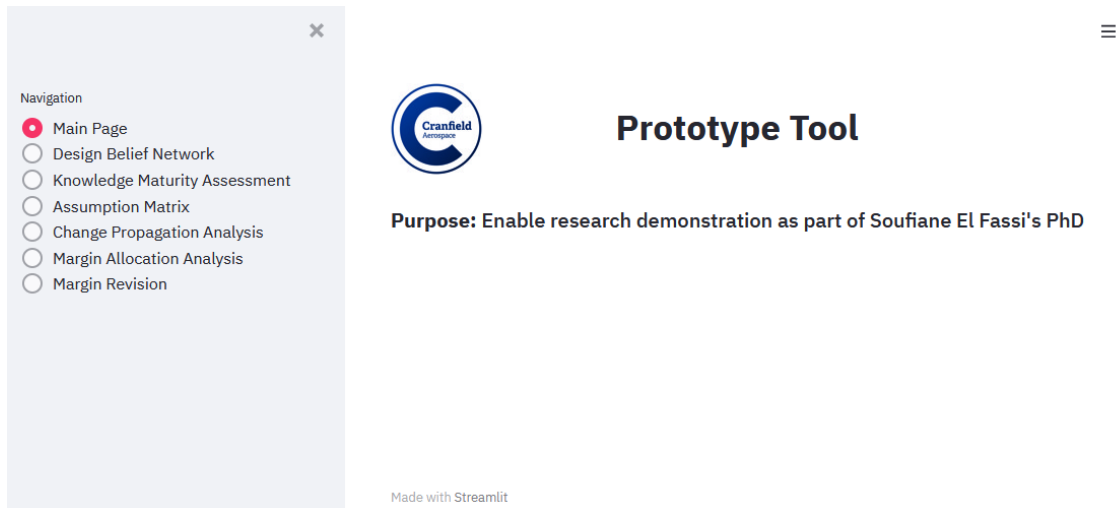


Figure 6.3: Streamlit user interface

values stored in the dictionaries. The resulting graph is then exported as a Python Pickle file, which can be used by subsequent methods. The *DBN* can be visualised using *Plotly*⁵, an open source Python library for interactive visualisation.

The *Knowledge Maturity* Python script implements Equation 5.5 (Section 5.2.4) to calculate the *KMI*. This is done by first reading the Pickle file from the *Design Belief Network* Python script to extract all the necessary information on the number of assumptions, their confidence assessment, their status, as well as the number of conflicts. In terms of visualisation, *Plotly* has been utilised where the value of *KMI* is displayed using an *Angular Gauge Chart*, the average values of the individual indicators I_1 , I_2 and I_3 are displayed using *Bullet Gauge Charts*, whereas the average values of confidence assessment criteria (i.e. Data Reliability, Model Realism, Expert Agreement) are displayed in a *Radar Chart*.

The *Assumption Matrix* Python script allows to visualise the Assumption Matrix (Section 5.3) corresponding to the *DBN* created earlier. This script reads the Pickle file corresponding to the *DBN* to extract the assumption nodes and their dependencies. These assumption nodes are then placed in their corresponding priority section of a *Heatmap* from *Plotly*. The prioritisation is performed according to Algorithm 4.

The *Margin Allocation* Python script implements Algorithm 5 to identify the assump-

⁵<https://plotly.com/python/> (Accessed: 26/11/2021)

tions that have not been explicitly associated with a margin and detect instances of margin redundancy, in addition to visualising the association between assumptions and margins for which they are explicitly mitigating the risk. This visualisation is implemented as a Sankey diagram from *Plotly*. Once the assumptions that have not been explicitly associated with a margin are identified, they are ranked in terms of their prioritisation from the *Assumption Matrix* script. Regarding margin redundancy detection, it is necessary to distinguish between parameters that are independent (i.e. root nodes) and dependent (i.e. intermediate or leaf nodes). To this end, the *predecessors()* method from *NetworkX* is used to determine whether a particular node has a parent node (i.e. predecessor), meaning that it is dependent.

The *Change Absorption* Python script, implementing Algorithm 6, allows to locate the closest margin that could act as a change absorber and inform about requirements, functions, solutions and other assumptions that could be directly affected by the assumption initiating a change. The user first selects from a drop-down list the assumption that is considered to be initiating a change (A_{CI}). A_{CI} is then passed as an input to a Python function, which computes the shortest path from A_{CI} to every margin node (if any) using the *NetworkX* implementation of Dijkstra's algorithm [181]. Once all such paths have been generated, the shortest of these is considered to lead to the closest change absorber. To verify that this shortest path is realistic, a *for loop* iterates over the *Solution* nodes of the shortest path to make sure they are hierarchically related. This is to prevent situations where, for instance, a margin on the fuel tank volume would be suggested to absorb change initiated by an assumption affecting the tail size. Checking hierarchy is possible due to the fact that each *Solution* node has a 'parent' attribute. Finally, the script returns both the shortest path and the set of assumptions associated with elements of the shortest path. The shortest path contains information about which solutions have been frozen, and the set of assumptions includes also assumptions conflicting with A_{CI} .

The *Margin Revision* Python script allows to notify the user about instances of margin revision through monitoring changes in assumptions. This is implemented via the

Observer pattern (illustrated in Figure 6.4), which is a software design pattern where an object (referred to as subject or publisher) notifies its dependents (referred to as observers or subscribers) whenever there are any changes in the subject [183]. A subject can have any number of observers. In the context of margin revision, all *Assumption* objects act as subjects, and the explicitly associated margins act as observers. Notifications are triggered when changes occur in an assumption's status, level of confidence and dependencies. This process is described in Section 5.6 which presents the corresponding Algorithm 7.

If the margin is to be increased while satisfying constraints, the *Margin Space* method is used. To this end, a design of experiment is generated in AirCADia Explorer to populate the margin space, which in turn can be visualised in AirCADia Vision as a contour plot. Each contour represents a constraint, which informs about the limit for increasing margins.

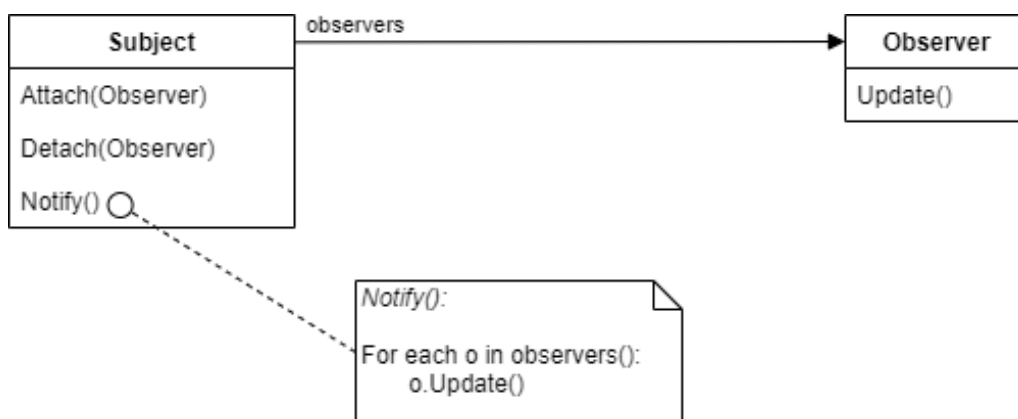


Figure 6.4: UML class diagram of an observer pattern (adapted from [183])

Detection of Conflicting Assumptions

The method described in Section 4.4 was implemented into a Python script, where the `Graph.reverse()` function from *NetworkX* is used to enable a reversed traversal of the computational workflow. Graph traversal, using DFS, is implemented with the `dfs_tree(Graph, Source)` function from *NetworkX*. During the search, the nodes are checked for their type, so that encountered *Assumption* nodes are added to the list of conflicting assumptions.

All the aforementioned software implementation and visualisation approaches are demonstrated in Section 6.4. Appendix A contains source code excerpts relating to the aforementioned Python implementation.

6.3 Demonstration Use Cases

6.3.1 Use Case 1: Design of a Lightweight Fighter Aircraft

Evaluating the developed methods required demonstrating them first to assess whether they work as intended. To this end, a use case about the early design of a fighter aircraft was developed.

The use case built upon Raymer’s conceptual design of a lightweight fighter [41], which was extended to include additional assumptions and their dependencies, uncertainty assessment, and technical risk management.

The use case was about the design of a fighter aircraft, using the F-16 as a baseline configuration while implementing newer technologies and additional capabilities, including a variable dihedral vertical tail (VDVT) [41]. The latter consisted of an unproven technology to “*control the rearward shift in aerodynamic center as the aircraft accelerates to supersonic flight by converting from a V tail subsonically to upright vertical tails supersonically*”. The purpose of the VDVT is to decrease trim drag and improve manoeuvrability [41]. This is an example of considering novel technologies at early design, for which no historical data or prior experience can be used to support decision-making. Performance requirements were defined as follows:

- Take-off and landing: 1000 ft ground roll
- Approach Speed \leq 130 kts
- Maximum Mach \geq 1.8
- Accelerate from Mach 0.9 to Mach 1.4 in 30 sec at 35000 ft

- Specific Excess Power $P_s = 0$ at 5g at 30000 ft at Mach 0.9 and Mach 1.4
- Turn Rate $\dot{\psi} \geq 20^\circ/\text{sec}$ at 350 kts at 20000 ft
- Unrefueled ferry range with overload internal fuel shall be no less than 2300 nmi

The payload consisted of two advanced missiles (200 lb), an advanced gun (400 lb), 750 rounds ammo (440 lb), and a pilot (220lb).

The computational workflow for sizing the fighter is shown in Figure 6.5, where the statistical models were derived from [41] and some intermediate calculations were omitted for simplification and clarity. Note that the *Missiles Bay* sizing model was added to illustrate assumptions conflict. The assumptions underlying the design are listed in Table 6.1. High-level functional and logical views of the system architecture are shown in Figure 6.8.

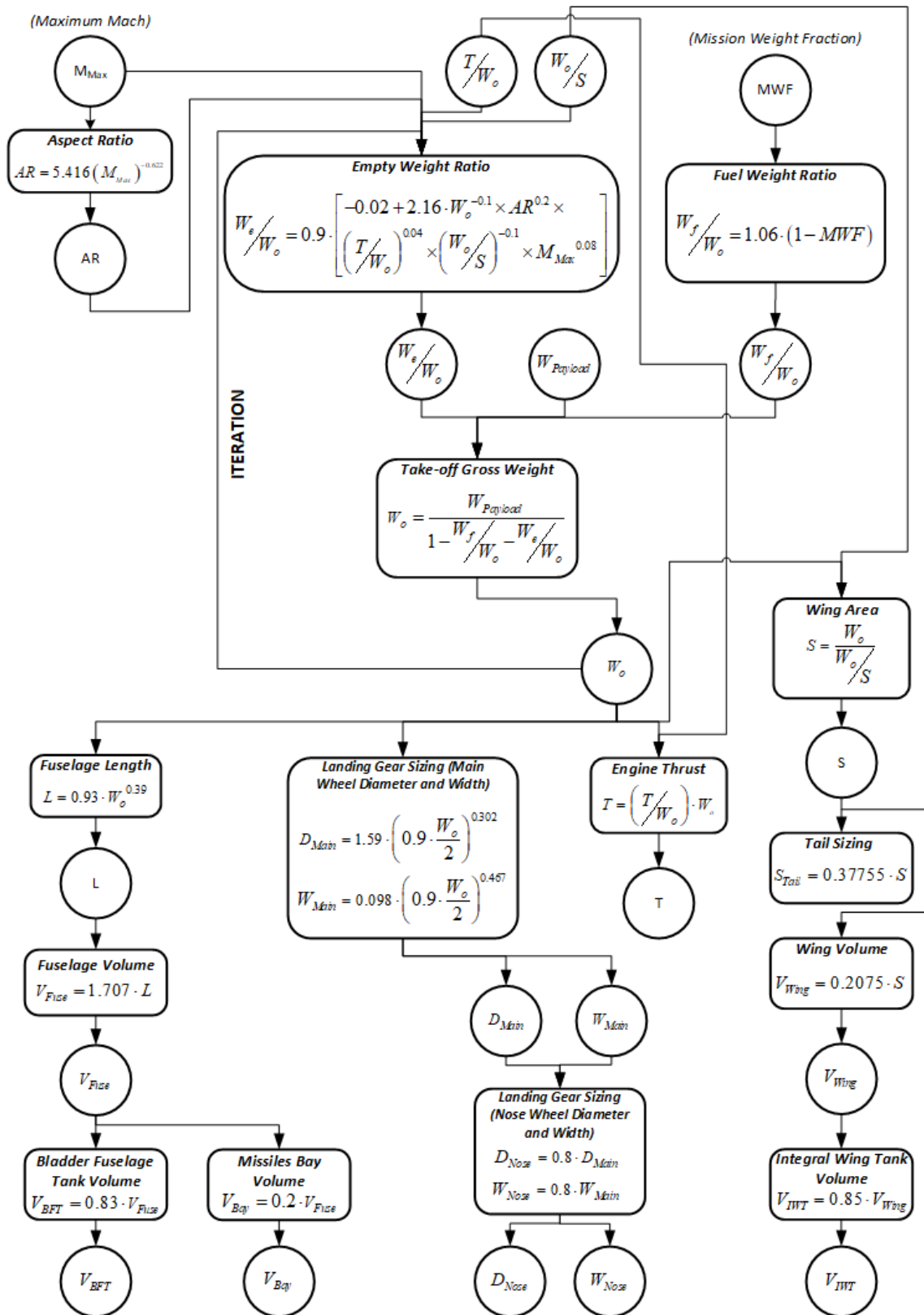


Figure 6.5: Computational Workflow for Aircraft Sizing

Assumption		Description
α_1	Tail Convert	Tail configuration is currently unknown. Conversion from V-tail (subsonic) to vertical tail (supersonic) is assumed to improve overall aerodynamic efficiency of the system
α_2	Unproven Tech	Readiness of the unproven technology cannot be accurately predicted currently. The variable dihedral vertical tail is assumed to be realisable and ready for when the system is to be integrated
α_3	Overload Range	Fuel tanks have not been sized yet. It is assumed that fuel tanks will store enough fuel for the overload range
α_4	Turbofan Engine	The rubber engine is not yet fixed in size and thrust. An existing afterburning turbofan engine is assumed for initial design analysis of the aircraft
α_5	SFC Adjustment	The rubber engine is expected to incorporate advanced technology. For initial design analysis, the SFC of the rubber engine is assumed to be 80% of the SFC of the selected (existing) engine in order to adjust for advanced technology
α_6	Airfoil Selection	Airfoil optimisation cannot be performed at this stage. <i>NACA 64 A006</i> is assumed to be the most suitable (existing) airfoil type for initial sizing and analysis
α_7	AR Estimation	Trade study to determine the aspect ratio has not been conducted yet. Statistical model based on existing aircraft is assumed to be applicable for initial wing layout
α_8	Composite Structure	Materials selection is not finalised at this stage. 30% of the structure is assumed to be made of composite material
α_9	W_e Adjustment	The impact of the variable dihedral vertical tail on the aircraft empty weight is currently unknown. The empty weight is assumed to be 200 lb heavier than the estimated empty weight corresponding to a conventional tail
α_{10}	BFT Sizing	The internal layout of the fuselage is currently unknown. The bladder fuselage tanks are assumed to occupy 83% of the fuselage usable volume
α_{11}	Missile Bay Sizing	The internal layout of the fuselage is currently unknown. The missiles bay is assumed to occupy 20% of the fuselage usable volume
α_{12}	Mmax Definition	Maximum Mach is assumed to be higher than that of the baseline configuration, thus satisfying the current mission requirement
α_{13}	Nose Tire Sizing	Loads to be supported by the nose wheel are currently unknown. Nose tire is assumed to be 80% the size of the main tires

Table 6.1: List of assumptions corresponding to Use Case 1

6.3.2 Use Case 2: Conflict between Collaborating Teams

An example of conflict between collaborating teams is considered in this use case for the purpose of demonstrating the conflict detection method (Section 4.4). This example (derived from [184]) involves two collaborating teams: High-Lift Devices and Structure. The objective of the High-Lift Devices team is to size the leading and trailing edge high-lift devices (which include the control surfaces and the actuation systems). The objective of the Structure team is to define the structural layout (e.g. location of ribs and spars). However, these two teams interact through the interface between the high-lift systems and the structural layout, where the spars should be located so that both the control surfaces and their actuation systems can be accommodated. In Figure 6.6, the chord-wise length of the actuator (L_{act}) should be smaller than the gap between the slat and the front spar.

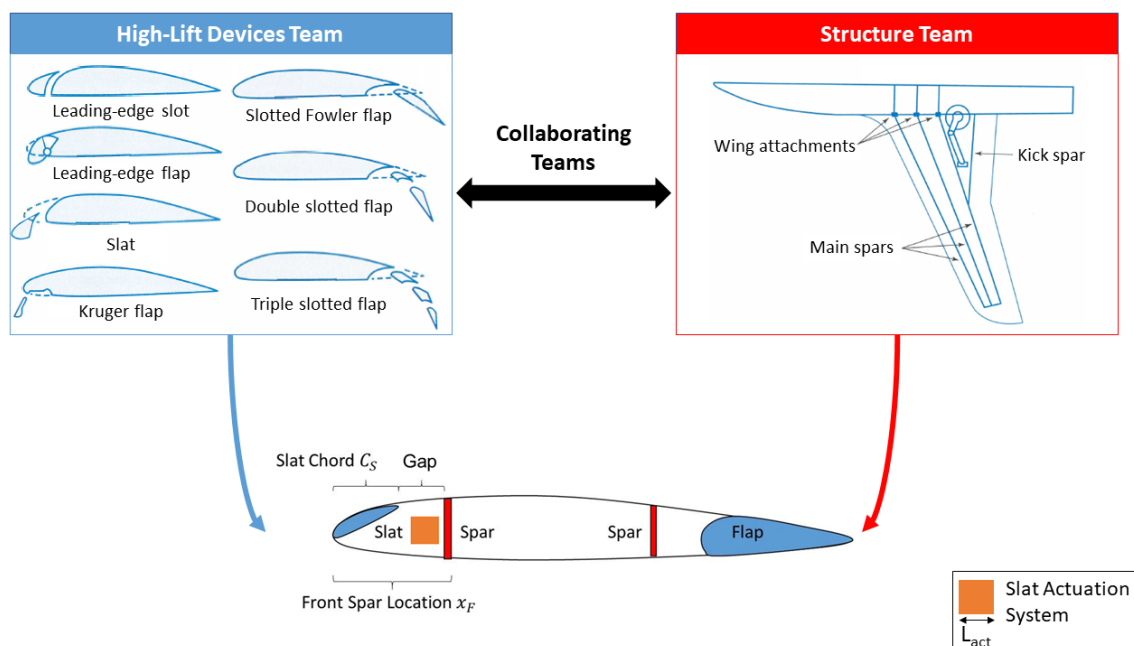


Figure 6.6: Interface between high-lift devices and structural layout (adapted from [41], [184])

In order to meet the stiffness requirement, the Structure team would ideally put the two spars further apart. However, this implies that the chord-wise dimensions of the high-lift devices are restricted, which would adversely affect the maximum lift coefficient [184].

In this use case, the perspective of the High-Lift Devices team is considered, where

the latter has to assume the location of the spars (e.g. by using design rules), while the Structure team has yet to finalise the structural layout. A conflict is encountered where the design rule used to assume the front spar location resulted in a gap that cannot accommodate the actuation system of the movable slat. In Figure 6.6, the length of the actuator (L_{act}) turned out to be larger than the gap between the slat and the front spar.

Therefore, the demonstration of the conflict detection method shall be about finding the set of assumptions leading to the constraint violation (i.e. the size of the actuator exceeding the available gap). To enable such demonstration, a computational workflow consisting of 15 models and 57 parameters is used. The computational workflow includes models for sizing the actuation system and estimating the front gap, which then allow to check whether the gap can accommodate the actuation system. Additionally, models for aerodynamic performance are included to estimate the maximum lift coefficient when high-lift devices are either retracted or deployed. Models for structural analysis are also included to estimate wing tip deflection (which is of interest to the Structure team). For better readability, a simplified representation of the computational workflow is illustrated by Figure 6.7, and only the parameters that are the most relevant to the demonstration are described in Table 6.2. All the parameters and models constituting the computational workflow are described in Appendix B.

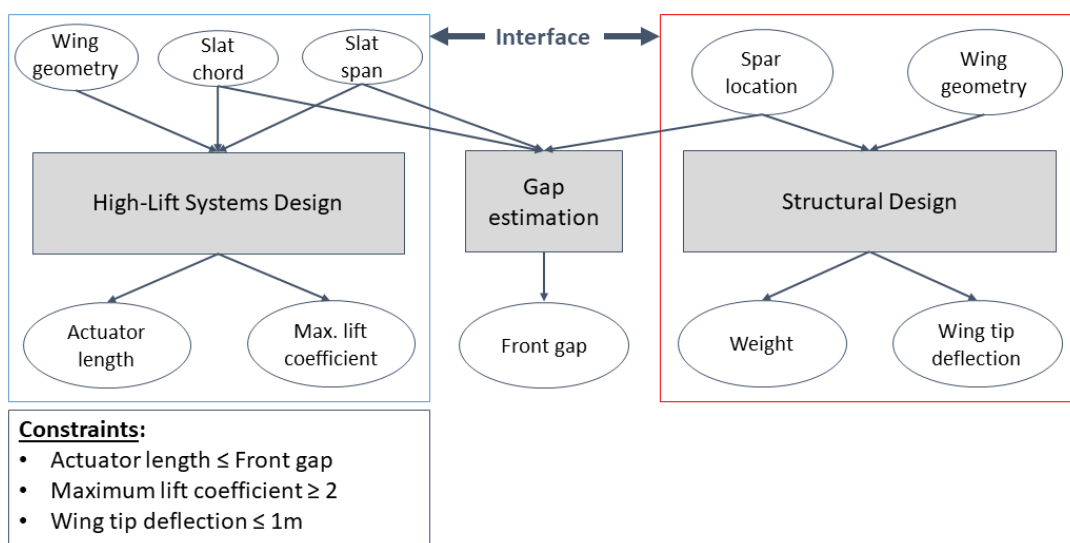


Figure 6.7: Simplified, high-level representation of the computational workflow

Parameter	Description
c_s/c	Slat chord to wing chord ratio
Δc	Chord extension due to deployment of slat
<i>Front_Gap</i>	Gap between front spar and slat
I	Bending inertia
L_{act}	Length of linear hydraulic actuator
L_{stroke}	Actuator's stroke length
Λ_{25}	Sweep angle of the wing quarter chord
Λ_{LE}	Sweep angle of the wing leading edge
S_{ref}	Wing reference area
W_{wing}	Wing weight
x_{FS}	Ratio of front spar location to wing chord
$y_{slat,in}$	Inboard spanwise location of the slat

Table 6.2: Description of key parameters in Use Case 2

The assumptions underlying the computational workflow are described in Table 6.3.

	Description of the Assumption	Associated Parameter
A_1	Wing reference area was estimated based on an assumed C_{Lmax} in the constraint analysis	S_{ref}
A_2	Stroke length is assumed from existing linear hydraulic actuators used for slat actuation	L_{stroke}
A_3	Chord extension due to deployment of slat is assumed to not interfere with the engine pylon	Δc
A_4	Inboard spanwise location of the slat is assumed based on historical data	$y_{slat,in}$
A_5	The wing weight is assumed based on the wingspan only	W_{wing}
A_6	The beam curvature is assumed to be constant for the estimation of the bending inertia	I
A_7	Ratio of slat chord to wing root chord is initially assumed based on previous experience	c_s/c
A_8	Sweep of the quarter-chord line was assumed based on a simplified geometric relationship, where the kink is ignored	Λ_{25}
A_9	Design rule of 25% for the front spar location is assumed to provide sufficient space for fitting the slat actuation system	x_{FS}
A_{10}	Leading edge sweep angle is assumed from a historical trend as a function of the maximum Mach number	Λ_{LE}

Table 6.3: List of assumptions corresponding to Use Case 2

6.4 Demonstration of the Developed Methods

6.4.1 Design Belief Network

First of all, the system architecture is defined using AirCADia Architect under the RFLP paradigm, as illustrated in Figure 6.8. On the top-left of Figure 6.8 is the Requirements Domain, at the bottom is the Functional Domain, and on the top-right is the Logical Domain. Figure 6.9 illustrates the Computational Domain in AirCADia Explorer. The Computational Domain consists of the computational workflow for initial sizing and performance assessment (cf. Figure 6.5).

During the design of the bladder fuel tank, an assumption has been made regarding its sizing (see assumption α_{10} in Table 6.1). To capture this assumption, right-clicking on the corresponding sizing model opens a new interface allowing the user to describe the assumption. Figure 6.10 illustrates the user interface. The name and description of assumption α_{10} (from Table 6.1) are entered in the corresponding fields. The status of the assumption is set to *Awaiting Evaluation*. Finally, the confidence in the assumption is assessed using the three assessment criteria (cf. Section 4.3.3). Since there was access to historical data from previous similar aircraft, Data Reliability is assessed as *High*. Since the sizing model is actually an empirical one that was created based on older aircraft, Model Realism is assessed as *Low*. Expert Agreement is set as *Moderate* for demonstration purposes.

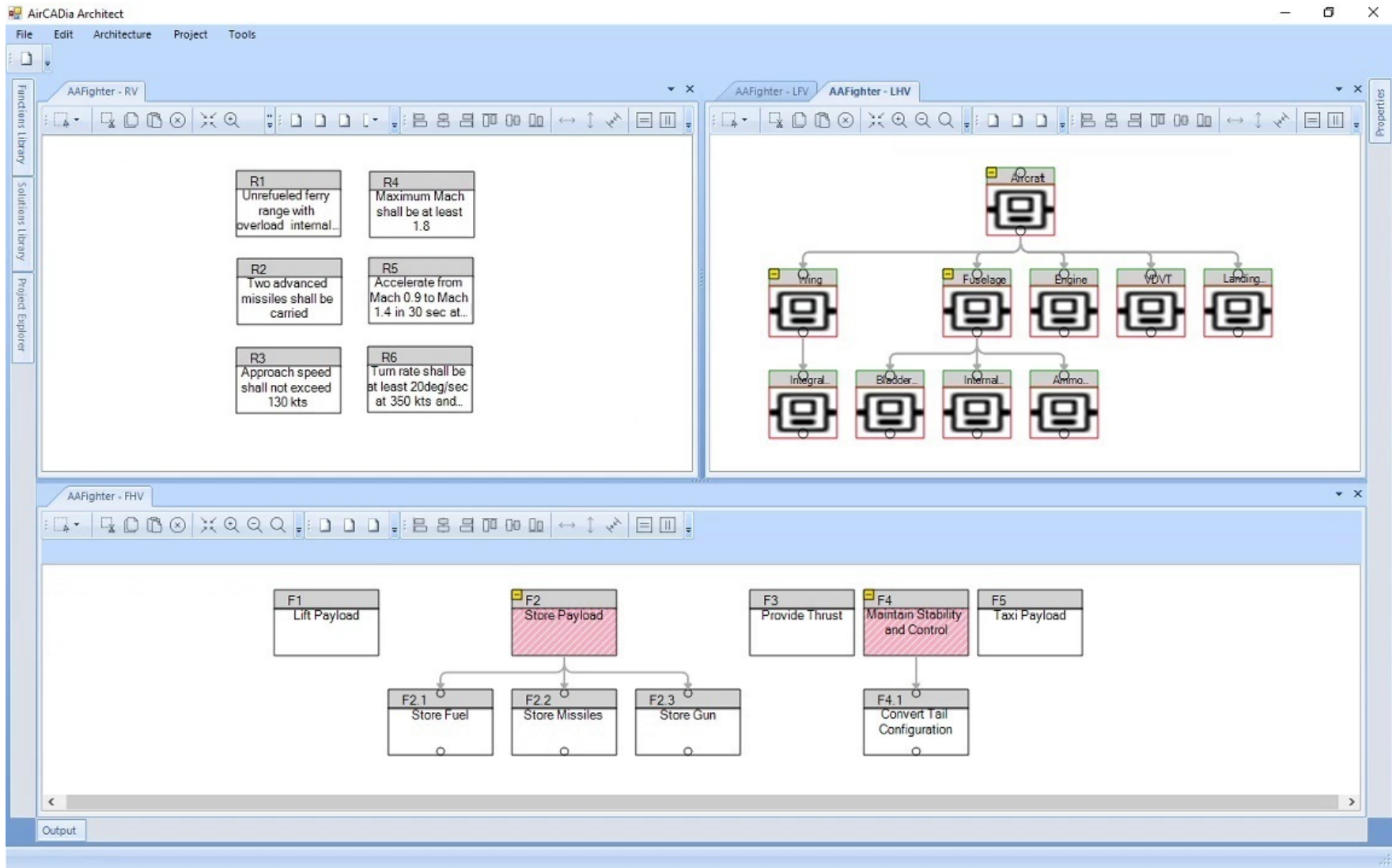


Figure 6.8: R-F-L domains (AirCADia Architect)

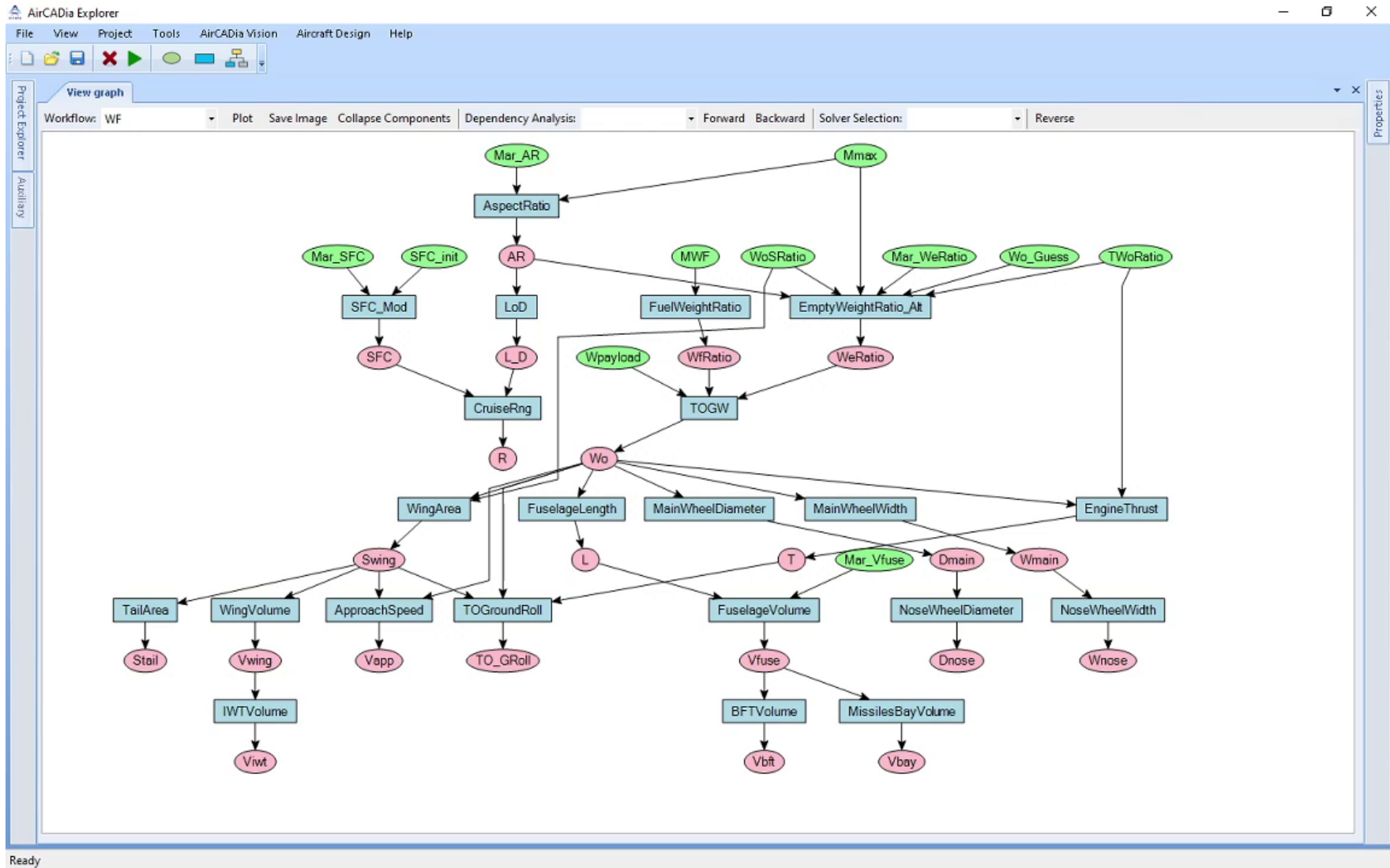
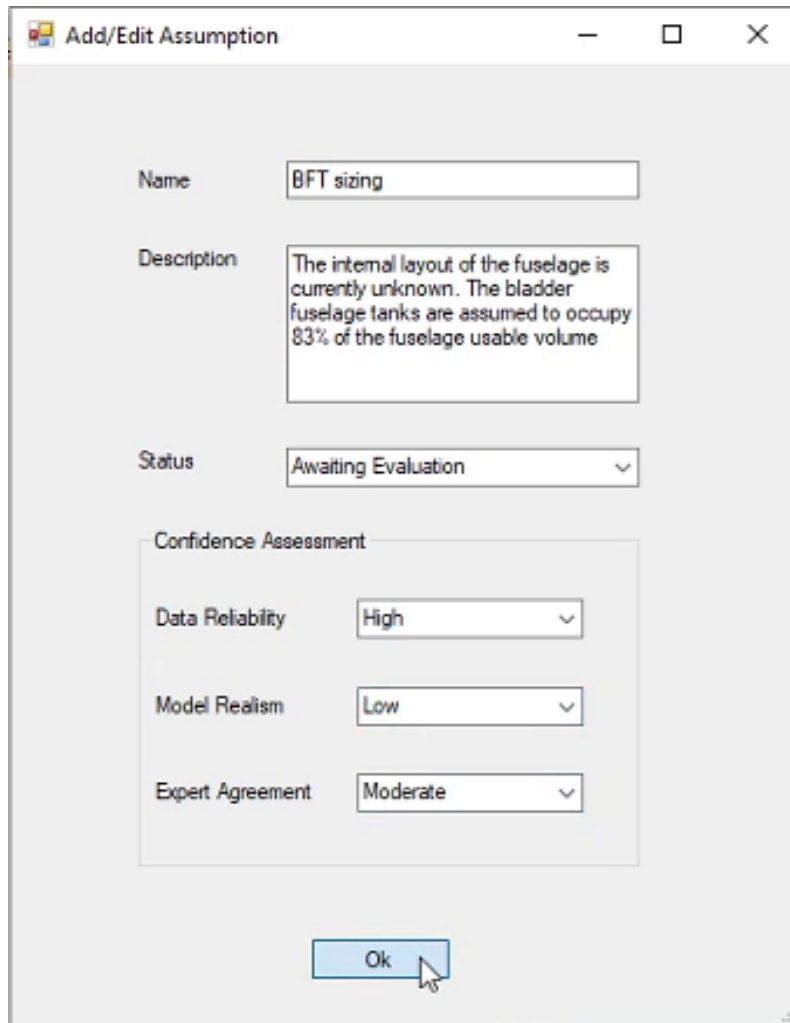


Figure 6.9: Computational workflow for sizing and performance assessment (AirCADia Explorer)



The screenshot shows a window titled "Add/Edit Assumption". It contains the following fields:

- Name:** BFT sizing
- Description:** The internal layout of the fuselage is currently unknown. The bladder fuselage tanks are assumed to occupy 83% of the fuselage usable volume
- Status:** Awaiting Evaluation
- Confidence Assessment:**
 - Data Reliability:** High
 - Model Realism:** Low
 - Expert Agreement:** Moderate

An "Ok" button is located at the bottom of the window.

Figure 6.10: User interface to record assumptions (AirCADia Architect)

All the captured assumptions can then be accessed, as shown in Figure 6.11. Note that the *confidence* attribute of assumption α_{10} has been automatically set to *Low* according to Algorithm 1. Furthermore, the dependency between the assumption and its corresponding computational model was automatically recorded (cf. *BFTVolume* in *Dependencies* field, Figure 6.11).

It is possible to associate any element from the R-F-L-C domains to an already existing assumption, and this is in an effort to reduce assumption duplication. For example, by right-clicking on requirement R4 (Maximum Mach shall be at least 1.8), we can select the corresponding assumption α_{12} , and then click on *Link*. This is illustrated in Figure 6.12. This functionality can also be used to record identified conflicts between assumptions.

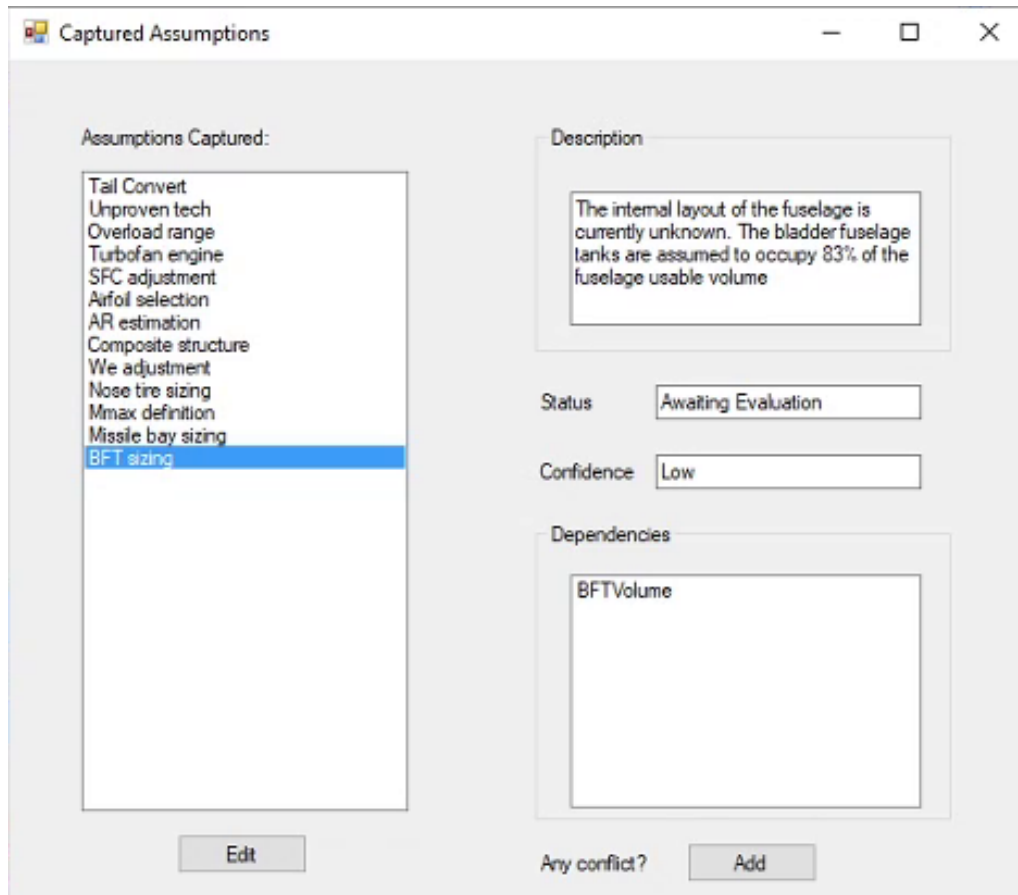


Figure 6.11: User interface to access captured assumptions (AirCADia Architect)

To illustrate this, let us consider assumption α_{11} (Table 6.1), where the missiles bay is assumed to occupy 20% of the fuselage usable volume. However, this conflicts with assumption α_{10} since we are now exceeding 100% of the fuselage usable volume. Since there is not enough knowledge at this stage to resolve the conflict, the latter should be recorded for future resolution. The *Add* button on the bottom-right in Figure 6.11 allows the user to record an identified conflict. In this example, the user can link assumption α_{11} to assumption α_{10} (cf. Figure 6.13), where this assumption intra-domain dependency can be interpreted as a conflicting relationship (as discussed in Section 4.3.2).

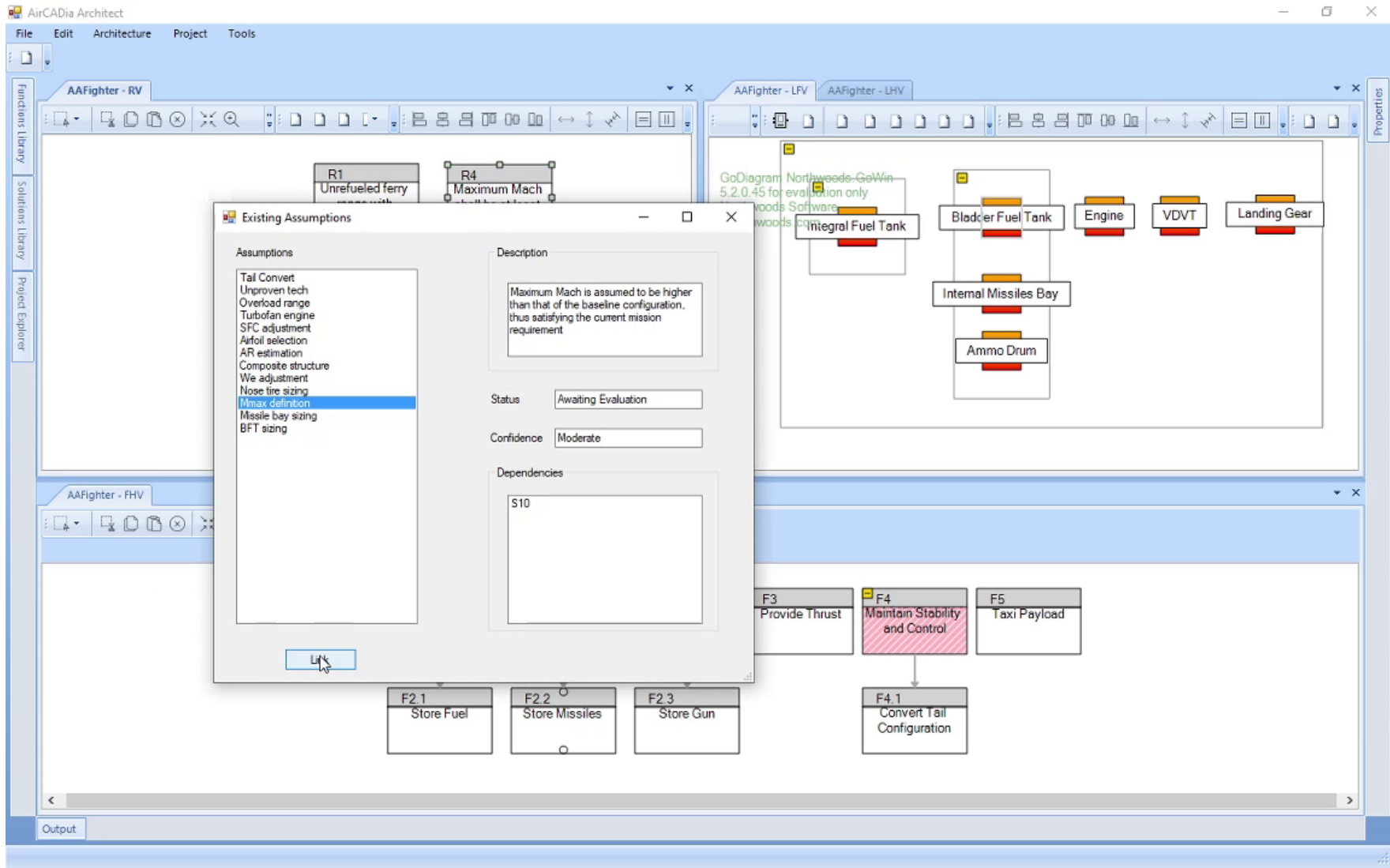


Figure 6.12: User interface to link with existing assumptions (AirCADia Architect)

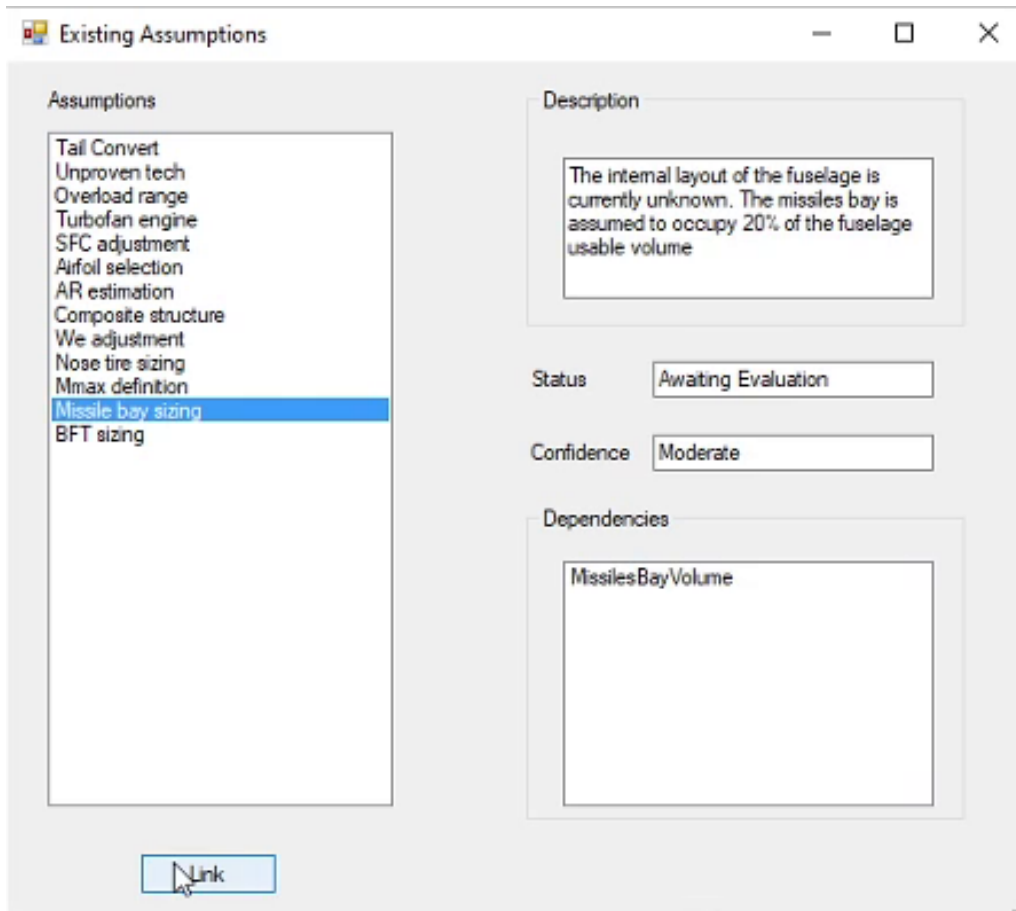


Figure 6.13: Recording the identified conflict between assumptions α_{10} and α_{11} (AirCADia Architect)

Finally, the resulting *DBN* (which is based on the graph-theoretical structure presented in Section 4.3.1) can be visualised using *Plotly*, as shown in Figure 6.14. Figure 6.14 also shows that each element of the system model can be traced back to its associated assumptions.

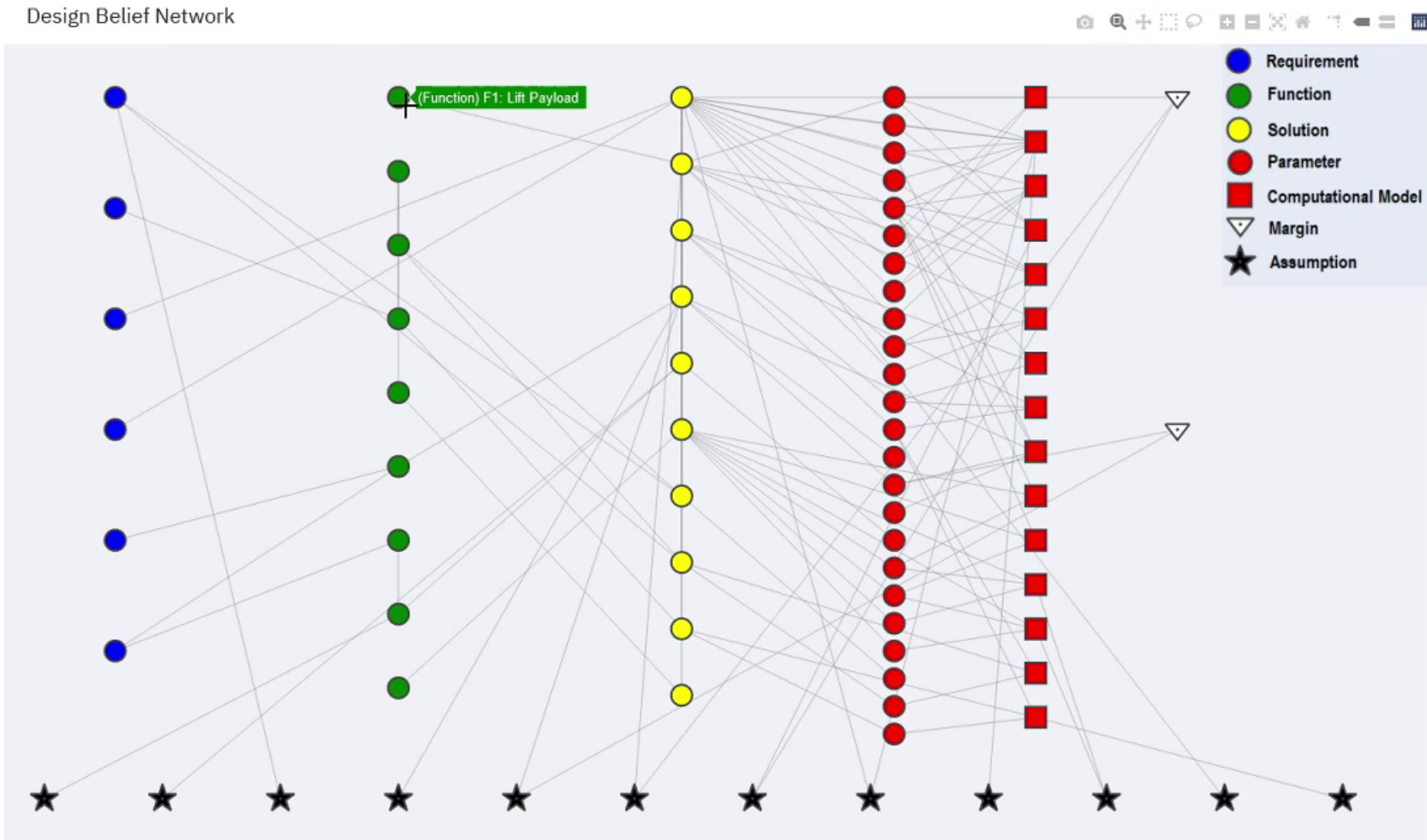


Figure 6.14: DBN corresponding to Use Case 1 (Software prototype tool)

The demonstrated approach to assumption capturing contrasts with the traditional document-centric approach (e.g. using an Assumption Log), which suffers from some limitations especially in the context of large complex systems. Such limitations include the lack of capturing dependencies, where the complexity of the design makes it very challenging to mentally keep track of all dependencies. This would also mean that traceability cannot be as easily and consistently provided. Then, due to this lack of dependency, there is a higher risk of assumption duplication when using a document. Furthermore, since there is no assessment of the confidence in assumptions with the traditional document-centric approach, it can be challenging to prioritise assumptions in terms of their likelihood of being invalid.

However, it can be observed from Figure 6.14 that using the visualisation of the network as a support has its limitations with respect to the number of system elements and assumptions considered. Therefore, scalability (from a visualisation perspective) is identified as a potential limitation.

6.4.2 Managing Risks Associated With Assumptions

Captured assumptions are sources of risks that need to be managed, since invalid assumptions can lead to costly redesigns or failure. In what follows, the risks associated with the captured assumptions are assessed using the two proposed methods *KMI* (Section 5.2) and *Assumption Matrix* (Section 5.3). Then, risks are mitigated through the use of margins to act as change absorbers. Risk mitigation is supported by the two proposed methods *Margin Allocation Analysis* (Section 5.4) and *Change Absorber Localisation* (Section 5.5). Finally, risk monitoring is supported by the *Margin Revision* method (Section 5.6), where changes in assumptions are monitored to automatically suggest margins to be revised.

Risk Assessment

Risk assessment is supported by the proposed methods: *KMI* (Section 5.2) and *Assumption Matrix* (Section 5.3). First, let us demonstrate the *KMI* method, which provides an

indication of the overall risk of change due to lack of knowledge. By selecting the *Knowledge Maturity Assessment* option in the software prototype tool, the values of the KMI and its individual indicators are calculated and can be visualised as shown in Figure 6.15.

The *KMI* was calculated as 15% (according to Equation 5.5), which indicates there is a rather high risk of change to occur due to invalid assumptions. Below the *KMI* are displayed the three individual indicators, where *Validation* has a low value (0.077) due to the fact that only one assumption (α_4) has been recorded as valid so far. A value of 0.333 for *Confidence* means that the LoC of all assumptions have been assessed, on average, as moderate-to-low. Whereas *Consistency* is very high (0.985) due to the fact that only one conflict (between α_{10} and α_{11}) has been captured so far.

Figure 6.16 shows how the *Confidence* indicator can be broken down into its assessment criteria. It can be seen that, on average, both *Data Reliability* and *Model Realism* have been assessed as Moderate, whereas *Expert Agreement* has been assessed as High. This shows that efforts should be focused on *Data Reliability* and *Model Realism* in order to increase knowledge maturity.

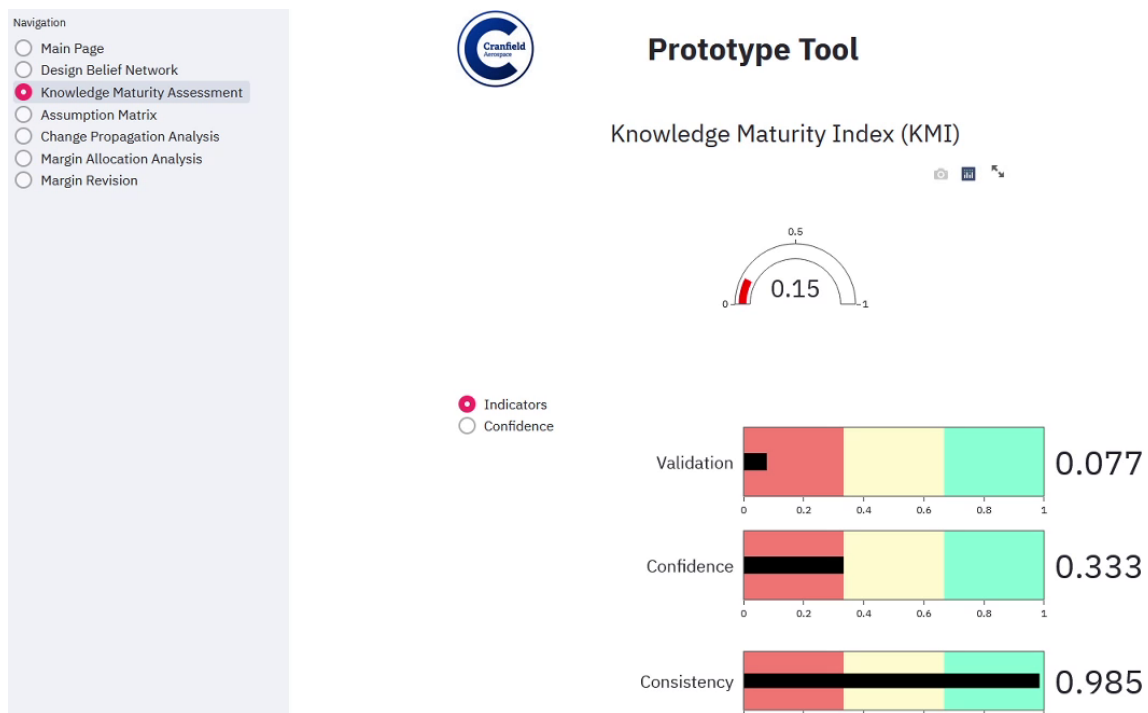


Figure 6.15: Knowledge maturity assessment corresponding to Use Case 1 (Software prototype tool)

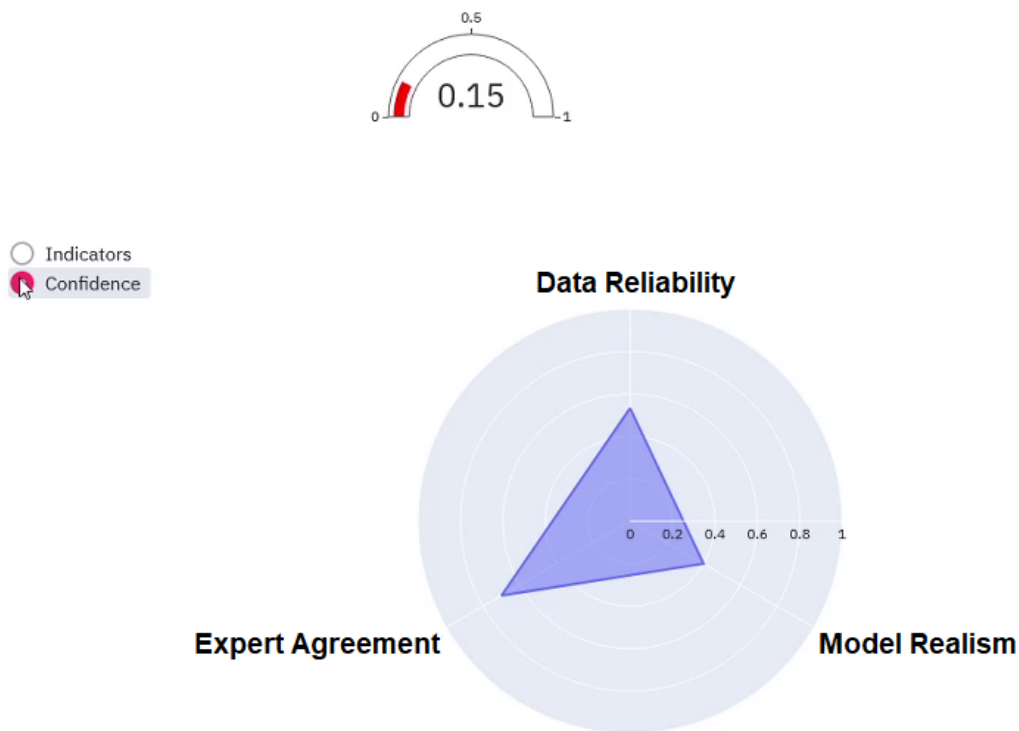


Figure 6.16: Confidence assessment in Use Case 1 (Software prototype tool)

Uncertainty analysis is necessary to gauge the robustness of composite indicators to the assumptions made in constructing them [166]. Regarding the KMI construction in Section 5.2, the choice of the *Intensities of Importance* in the *Weighting* step was based on the author's best understanding and intuition. Therefore, it is important to analyse how the KMI varies with respect to the uncertainty in weights calculation. In the following, ω_1 , ω_2 and ω_3 are treated as the input sources of uncertainty, and KMI as the output of interest. Recall from Section 5.2 that ω_1 , ω_2 and ω_3 are the weights corresponding to the indicators I_1 , I_2 and I_3 , respectively.

Matrix 6.4.2 represents the possible options of *Intensities of Importance* (where each set corresponds to the possible values of *Intensity of Importance* for a particular pairwise comparison), thus capturing the range of uncertainty. Note that, since I_1 is the dominating indicator, it cannot be equally important or of weak importance, compared to I_2 and I_3 . Thus, the *Intensities of Importance* values of 1 and 2 were not included in the uncertainty analysis (see the second and third columns of the first row in Matrix 6.4.2).

Matrix 6.4.2:

$$\begin{matrix} & I_1 & I_2 & I_3 \\ \begin{matrix} I_1 \\ I_2 \\ I_3 \end{matrix} & \left[\begin{array}{ccc} 1 & (3,4,5) & (3,4,5) \\ (\frac{1}{5}, \frac{1}{4}, \frac{1}{3}) & 1 & (1,2,3,4,5) \\ (\frac{1}{5}, \frac{1}{4}, \frac{1}{3}) & (\frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1) & 1 \end{array} \right] \end{matrix}$$

Then, the weights (ω_1 , ω_2 and ω_3) are calculated for the different combinations so that their uncertainty is propagated to the output KMI. To this end, a Monte Carlo Simulation with 10^6 (pseudorandom generated) samples was conducted. The resulting statistics are computed via a software implementation (see Appendix C), and summarised in Table 6.4.

Quantity	Value
I_1	0.077
I_2	0.333
I_3	0.985
KMI	15.03%
Minimum KMI	13.48%
Maximum KMI	17.18%
Mean of KMI	15.17%
Variance of KMI	9.17×10^{-5}
Standard Deviation (SD) of KMI	0.009
Coefficient of Variation (Mean/SD) of KMI	0.063

Table 6.4: Results of the uncertainty analysis

Knowing that the KMI is intended to provide an indication rather than an exact measurement, a *Coefficient of Variation* of 6.3% is highly reasonable, and thus implies an acceptable variability in KMI due to the aforementioned author's uncertainty in the weighting process. However, such uncertainty may become non-negligible when the KMI is evaluated in practice and the *Intensity of Importance* exceeds 5 (cf. Table 5.1). Furthermore, the involvement of industry experts is still needed to validate the pairwise comparisons of importance between the individual indicators.

Figure 6.17 illustrates the ability to assess progress of knowledge maturity over time. Let us consider that we are at the first decision gate, where the *KMI* was calculated as 15%. Over time, the *KMI* would allow us to keep track of progress in knowledge maturity by calculating its value at subsequent decision gates, and comparing variation (e.g. rate of

increase between two consecutive gates). For instance, it could indicate whether strategies and investments to acquire more reliable data lead to increasing knowledge maturity (and by how much it increases). In fact, since the *KMI* can be broken down into its individual constituents (as illustrated by Figure 6.17), we can use that information to guide decisions regarding which particular area should be focused on at each decision gate.

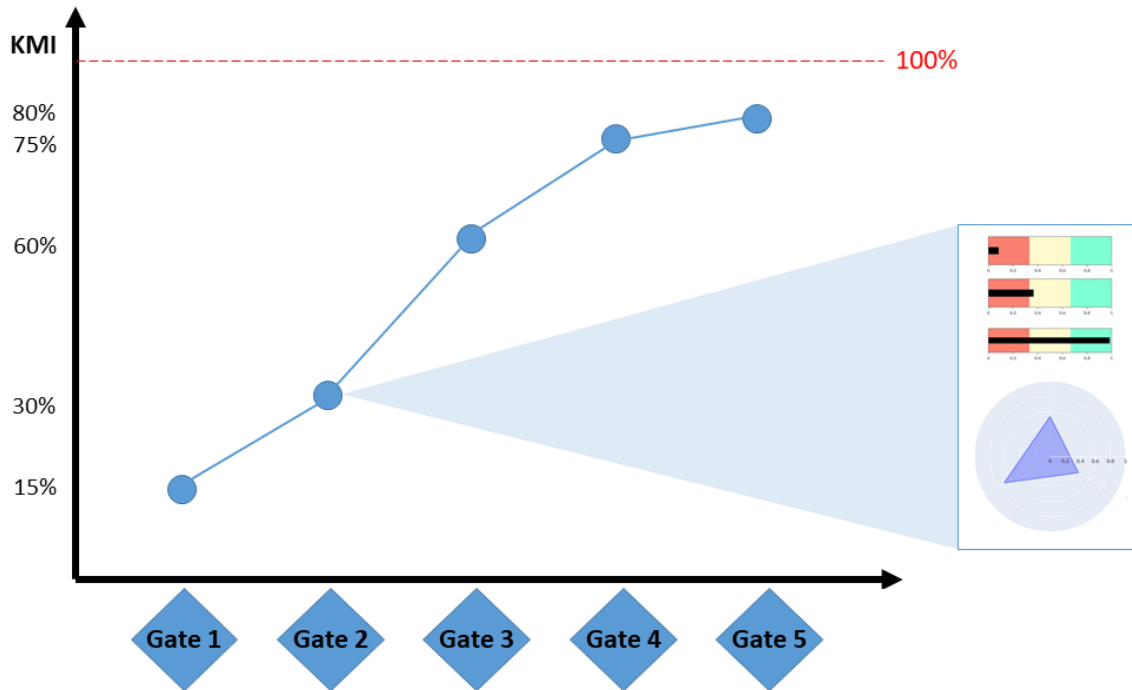


Figure 6.17: Illustration of KMI progress over time

Second, let us demonstrate the *Assumption Matrix* method. Figure 6.18 shows the *Assumption Matrix* corresponding to Use Case 1, and generated according to Algorithm 4, where each star corresponds to an assumption from Table 6.1. For instance, assumption α_{10} that was captured earlier has been categorised with the highest risk priority due to the fact that it has a *Low* LoC and the highest number of dependencies.

Furthermore, this method would enable to visualise the assumptions prioritisation directly from the system model. In Figure 6.19, the different elements in AirCADia Architect have been automatically coloured to match the priority category of the associated assumptions. For instance, the bladder fuel tank from the *Logical View* has been coloured in red to reflect the fact that it is associated with an assumption (α_{10}) with the highest priority.

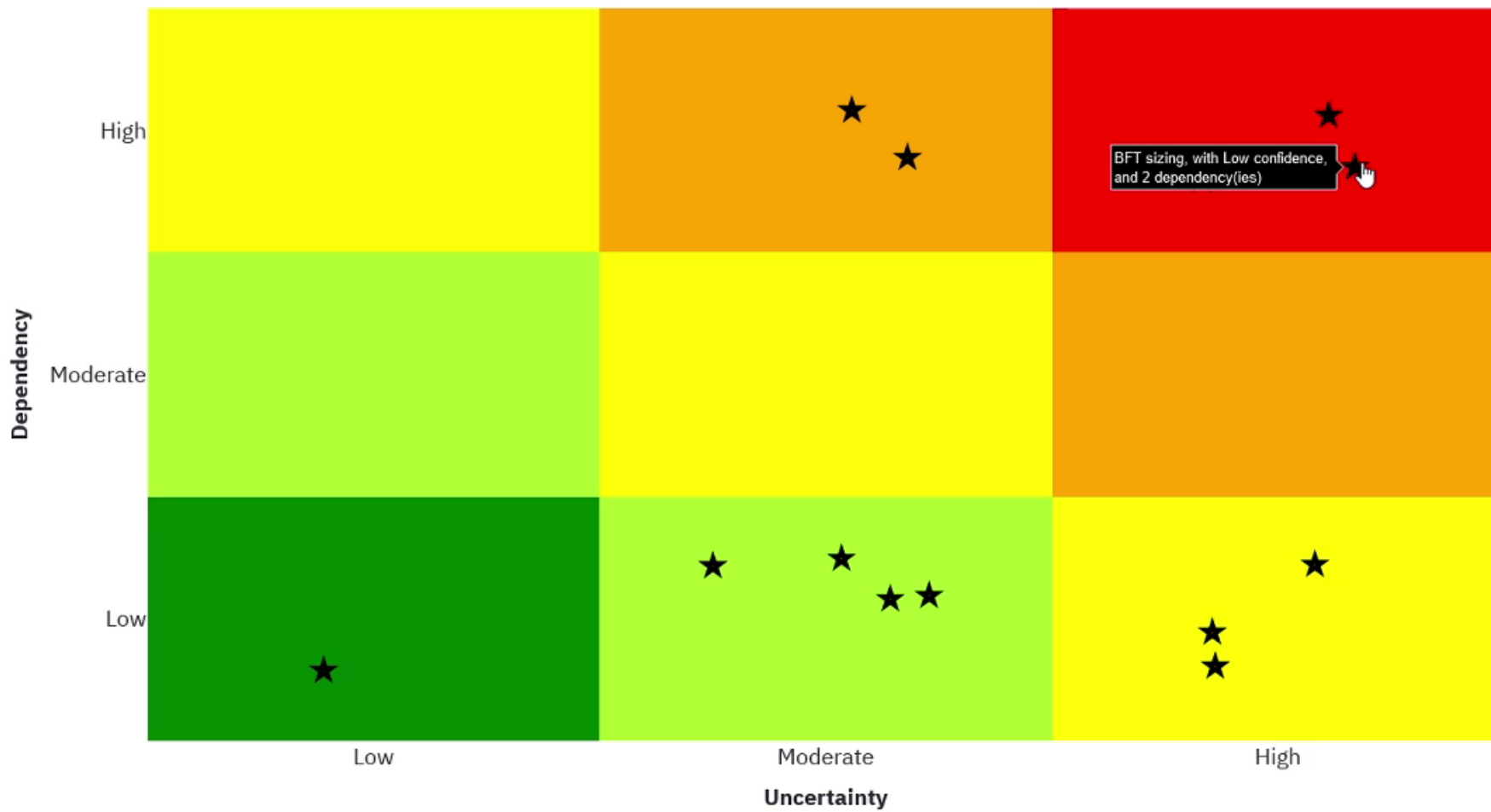


Figure 6.18: *Assumption Matrix* corresponding to Use Case 1 (Software prototype tool)

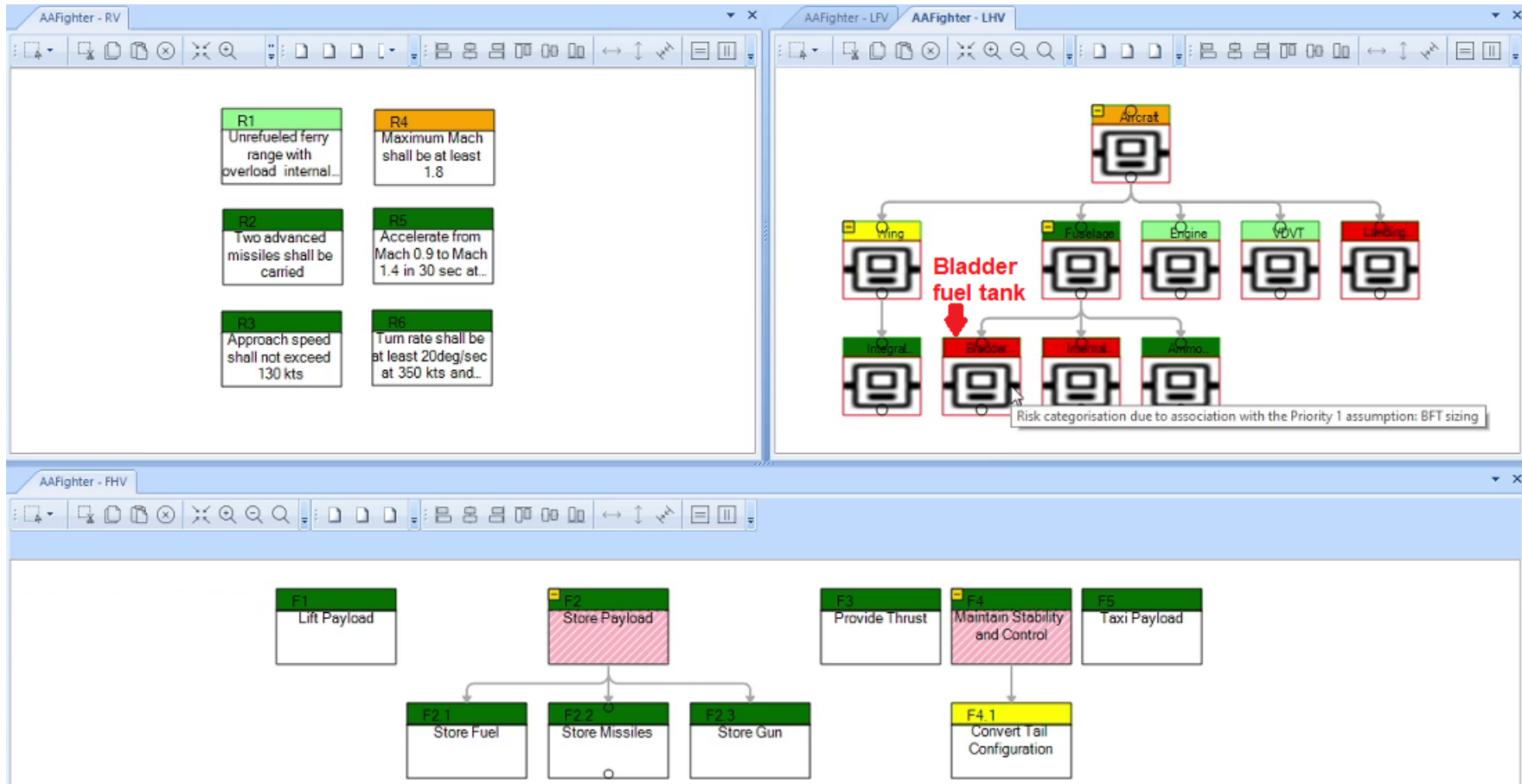


Figure 6.19: System model colouring to match assumption prioritisation in the *Assumption Matrix* (AirCADia Architect)

Risk Mitigation

In order to mitigate risk, margins can be used as a strategy to absorb change. To support this stage of the risk management process, two methods are proposed: *Margin Allocation Analysis* and *Change Absorber Localisation*.

First, *Margin Allocation Analysis* is demonstrated with the software prototype tool (via the implementation of Algorithm 5), which displays the status of margin allocation corresponding to Use Case 1.

Figure 6.20 illustrates the Sankey diagram used to visualise the explicit association between margins and assumptions for which they are mitigating risk.



Figure 6.20: Sankey diagram of margin-assumption dependencies corresponding to Use Case 1 (Software prototype tool)

Below the Sankey diagram, there is the list of assumptions that are not associated with any margin (as shown in Figure 6.21). This list refers to all the assumptions for which the risk has not been explicitly mitigated by a margin, which can inform future needs for margin allocation (especially with respect to the highest priority assumptions). Note that the assumptions are ranked in terms of their risk prioritisation, as defined in the *Assumption Matrix* method (Section 5.3).

At the bottom of Figure 6.21, it can be seen that the tool notifies about no margin

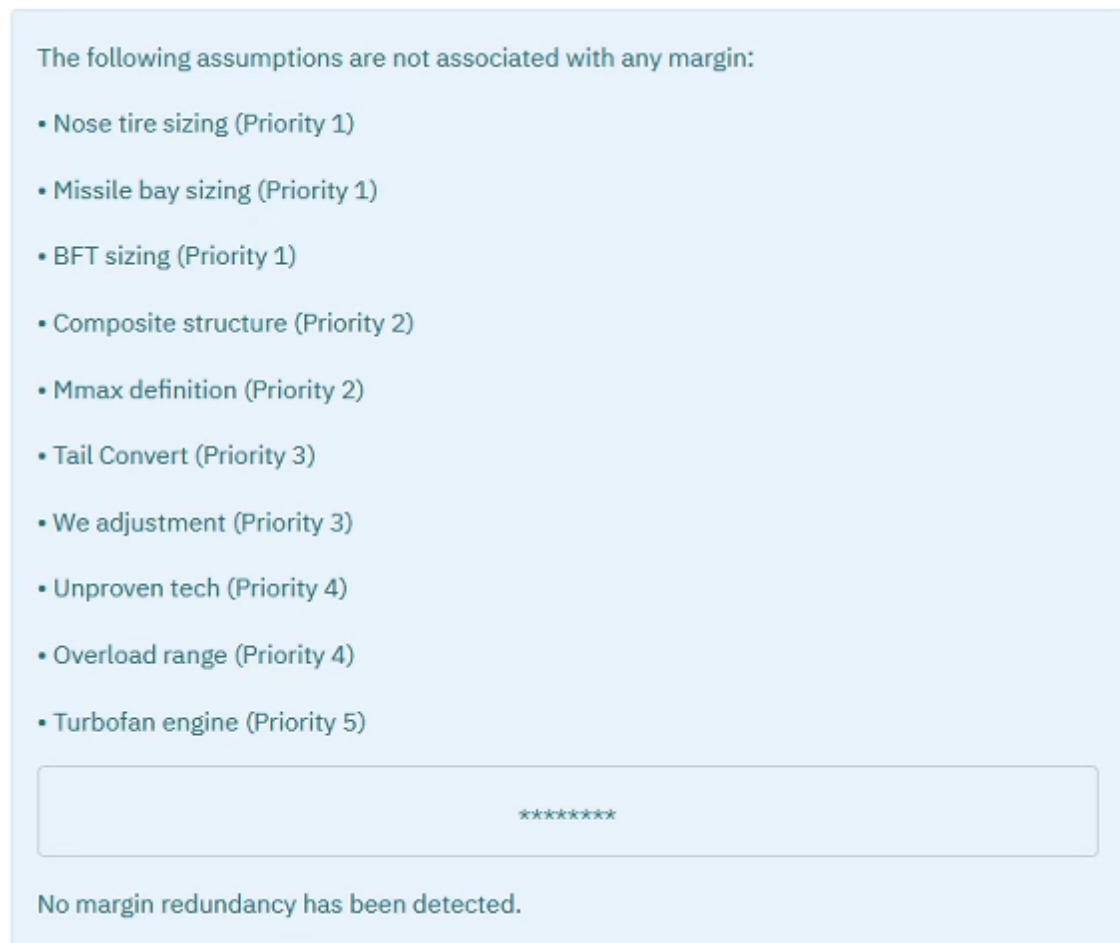


Figure 6.21: Non-mitigated assumptions corresponding to Use Case 1 (Software prototype tool)

redundancy being detected (according to Algorithm 5). To demonstrate the detection of margin redundancy, the example about aspect ratio (AR) estimation in Section 5.4 (and illustrated in Figure 5.5) is used. Since there is already a margin assigned to the aspect ratio parameter and associated with α_7 (AR Estimation), another margin is assigned to the maximum Mach (Mmax) parameter and associated with α_{12} (Mmax Definition). Figure 6.22 shows the updated Sankey diagram, where the new association is displayed at the end. Figure 6.23 shows that the instance of margin redundancy illustrated by the example in Section 5.4 has been detected. Furthermore, it also detected that the margin assigned to Mmax is due to requirement uncertainty (since Mmax is detected as a root node by Algorithm 5), and the margin assigned to AR is due to computational uncertainty (since

AR is detected as an intermediate/leaf node by Algorithm 5).



Figure 6.22: Updated Sankey diagram of margin-assumption dependencies corresponding to Use Case 1 (Software prototype tool)

Notification of potential redundancy:

- 'Margin_Aircraft.Mmax' (due to requirement uncertainty) and 'Margin_AR' (due to computational uncertainty) may be redundant.

Figure 6.23: Instance of margin redundancy detection (Software prototype tool)

Second, *Change Absorber Localisation* is demonstrated with the software prototype tool, where Figure 6.24 shows the user interface allowing to select the assumption that is considered as a change initiator.

Recall that the bladder fuel tank from the *Logical View* has been coloured in red to reflect the fact that it is associated with an assumption (α_{10}) with the highest risk priority (cf. Figure 6.19). Thus, assumption α_{10} (BFT Sizing) is selected as a change initiator in order to locate the closest change absorber. The output is shown in Figure 6.25, where change initiated by the invalidated α_{10} could potentially propagate by first causing change in the bladder fuel tank, which in turn can affect the fuselage, and ultimately lead to re-design at the aircraft level since no margin has been found to mitigate propagation by



Figure 6.24: List of assumptions, which represent potential change initiators (Software prototype tool)

absorbing change. This conforms with the notion of hierarchical relationship implemented in Algorithm 6 (cf. Section 5.5).

In the lower half of the user notification in Figure 6.25 is the list of assumptions that could be affected along the aforementioned propagation path. α_{11} (Missile Bay Sizing) was listed due to the fact that it is conflicting with α_{10} , and therefore a change in the status of α_{10} could affect α_{11} . Both α_8 (Composite Structure) and α_{12} (Mmax Definition) were listed due to the fact that they are associated with the solution element *Aircraft* from the Logical domain. Therefore, a change in *Aircraft* could affect its associated assumptions.

Select Change Initiator:

BFT sizing|

Change initiated by (Assumption) 'BFT sizing' could propagate through the following architectural elements:

- (Solution) Bladder Fuel Tank;
- (Solution) Fuselage;
- and stop at Aircraft level, where there is currently no margin acting as a Change Absorber.

The following assumptions, associated with elements along the propagation path above, may be affected by the change:

- Missile bay sizing (Conflict)
- Composite structure (If change in Solution 'Aircraft')
- Mmax definition (If change in Solution 'Aircraft')

Figure 6.25: Demonstration of *Change Absorber Localisation* with α_{10} (Software prototype tool)

One way of mitigating the risk of change propagating to the aircraft level is to assign a margin to the fuselage volume. The extra volume inside the fuselage would provide some flexibility to redesign the bladder fuel tank if necessary, without resizing the fuselage, which in turn would prevent changes at the aircraft level. For example, a margin of 10% is assigned to the fuselage volume parameter V_{fuse} , as shown in Figure 6.26.

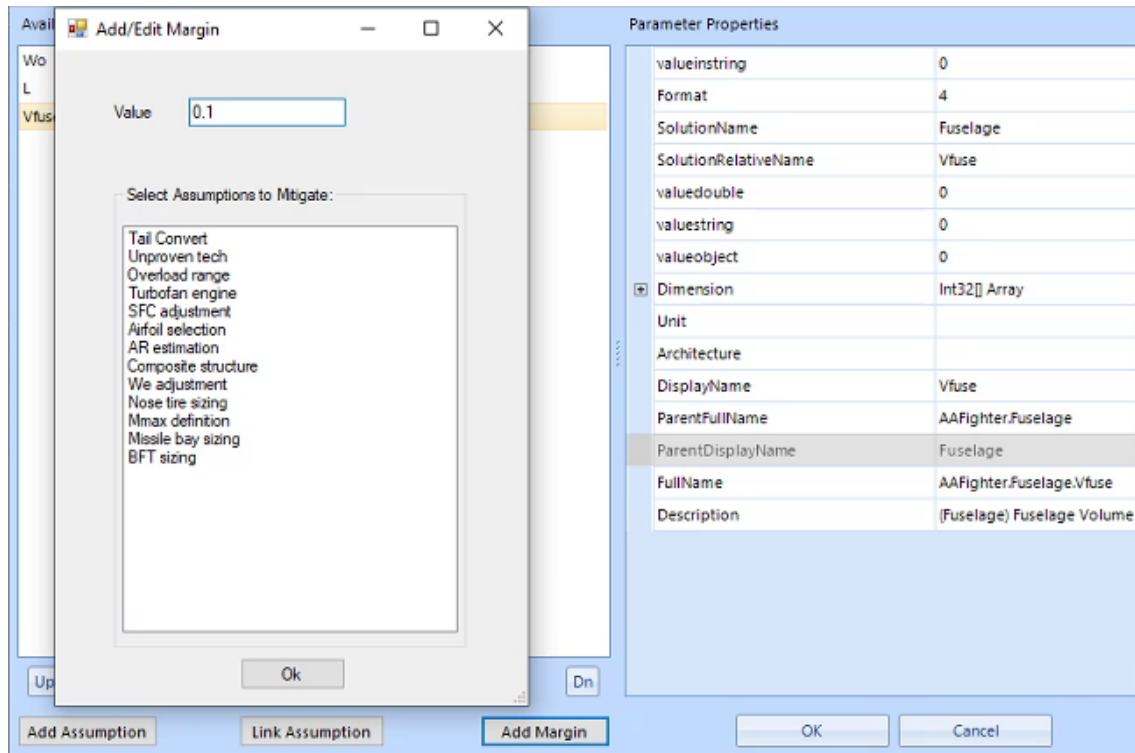


Figure 6.26: Assigning a margin to the fuselage volume (AirCADia Architect)

Figure 6.27 shows that the margin on fuselage volume has been taken into account and a change absorber (i.e. margin on *Vfuse*) has been detected at the fuselage level. Furthermore, note from Figure 6.27 that both assumptions α_8 (Composite Structure) and α_{12} (Mmax Definition) are not listed as potentially affected assumptions since change is no longer suggested to propagate to the aircraft level.

Another functionality is the ability to notify about which solutions have been frozen along the propagation path (cf. Section 5.5). For example, the Fuselage component in the Logical domain is set to be frozen, as shown in Figure 6.28. Figure 6.29 shows that the fuselage is now detected as a *Frozen Solution*. This information is expected to be useful in assessing the impact of change, knowing that frozen components are more costly to change.

Select Change Initiator:

BFT sizing

Change initiated by (Assumption) 'BFT sizing' could propagate through the following architectural elements:

- (Solution) Bladder Fuel Tank;
- and stop at (Solution) 'Fuselage', where there is a margin on parameter 'Vfuse' acting as a Change Absorber.

The following assumptions, associated with elements along the propagation path above, may be affected by the change:

- Missile bay sizing (Conflict)

Figure 6.27: Demonstration of *Change Absorber Localisation* following margin assignment (Software prototype tool)

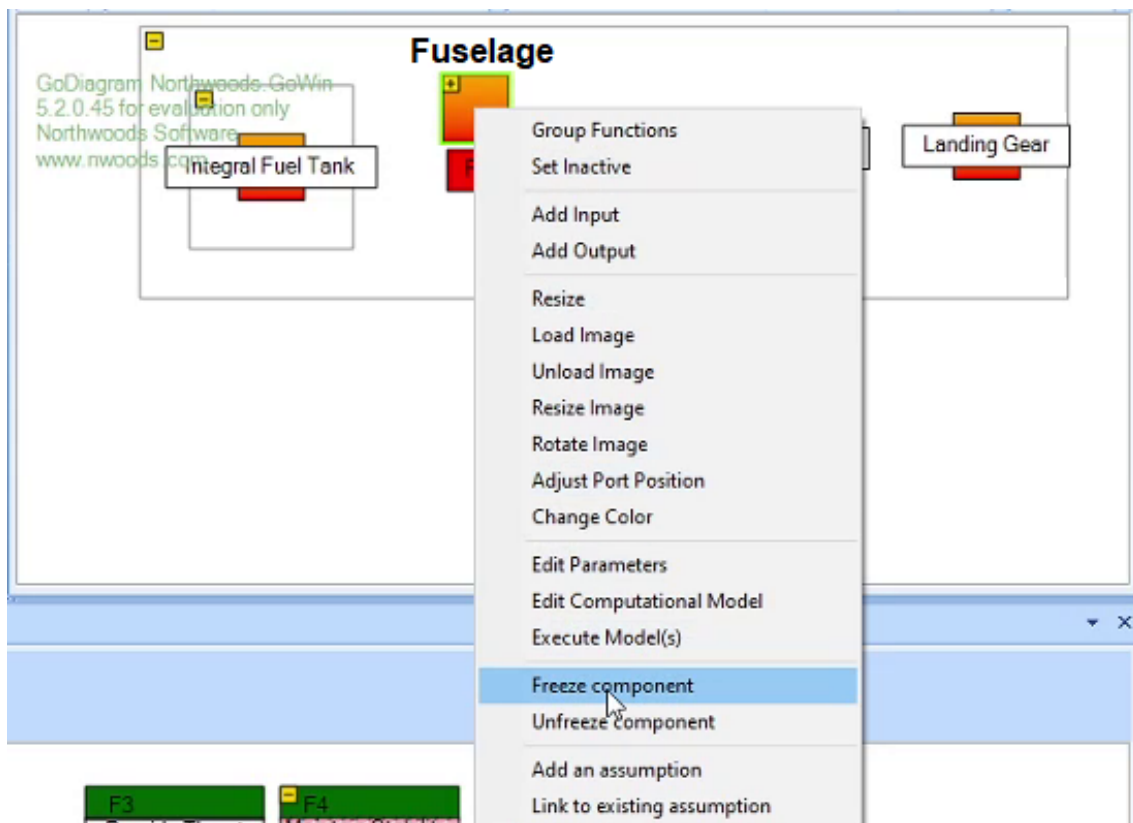


Figure 6.28: Demonstration of component freezing (AirCADia Architect)

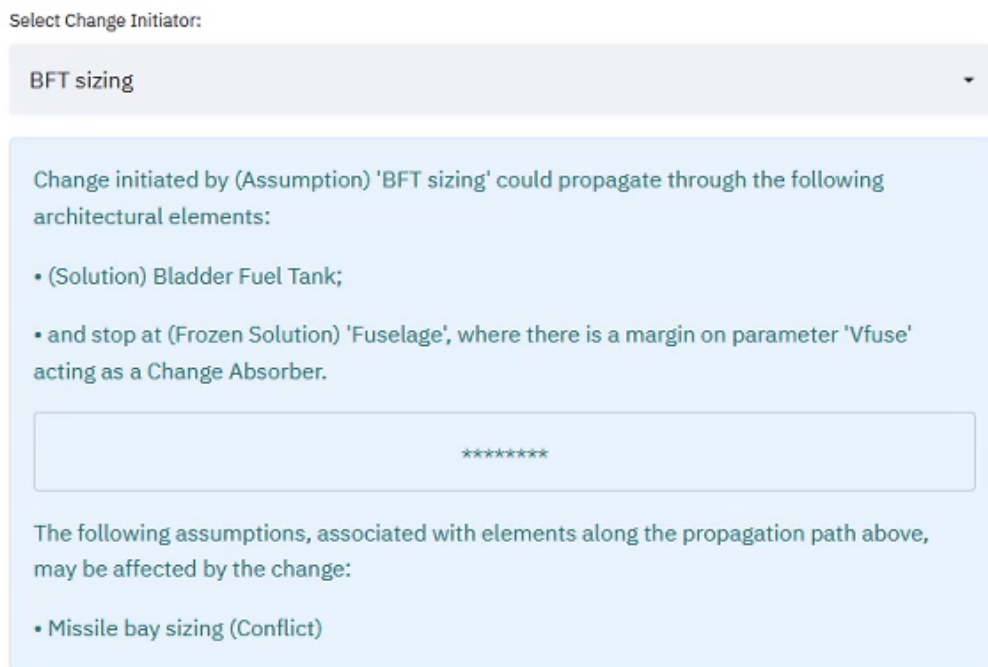


Figure 6.29: Demonstration of *Change Absorber Localisation* following component freezing (Software prototype tool)

Risk Monitoring

After margins are assigned to parameters and associated with assumptions, risk is monitored by detecting changes in the *DBN*, and thus inferring margin revisions as prescribed by Algorithm 7.

Let us consider we are now further in the design process, and therefore acquired more knowledge about the fighter aircraft. Thus, we are able to validate the earlier estimation of aspect ratio (i.e. changing the status of α_7 from *Awaiting Evaluation* to *Valid*). However, the initial adjustment of the SFC (α_5) does not match the latest analysis results. Therefore, the *Model Realism* criterion for α_5 is reassessed from *Moderate* to *Low*. This in turn leads to changing the value of LoC corresponding to α_5 from *Moderate* to *Low*.

Going back to the *Margin Revision* module in the software prototype tool, Figure 6.30 shows that the tool has detected the aforementioned changes. The margin on SFC is suggested to be increased since the decrease in LoC has been detected, which can be interpreted as an increasing risk of α_5 being invalid. Additionally, the margin on AR is

suggested to be reduced since α_7 has been validated, meaning that it no longer poses a risk of initiating change due to being invalid.

However, if it is decided to follow the suggestion and increase the margin on SFC, constraint satisfaction cannot be ensured. Therefore, the *Margin Space* method is suggested to address this situation, as discussed in Section 5.6. Figure 6.31 shows AirCADia Vision, where the plot on the left represents a constraint diagram (thrust-to-weight ratio vs. wing loading), and the plot on the right represents a *Margin Space* (margin on SFC vs. margin on AR). V_{app} refers to the approach speed constraint, TO_GRoll refers to the take-off ground roll constraint, and R refers to cruise range. The “Master Equation” from Mattingly *et al.* [42] for energy-based constraint analysis was used to generate the contour lines corresponding to the aforementioned performance constraints.

As shown in Figure 6.31, the margin on SFC is initially set to 3%, and the margin on AR is initially set to 9%. If it is decided to increase the margin on SFC to 6%, this would violate constraint R , as shown in Figure 6.32 where the design space becomes completely unfeasible (i.e. coloured in red). Since the tool also suggested we can reduce the margin on AR, the latter is first reduced to 6%, and then the margin on SFC is increased to 6%. Figure 6.33 shows that this combination allows to remain in the feasible space. Therefore, such margin trade-off allowed to update the risk mitigation strategy (according to the suggested revisions), while still satisfying the constraints.



Prototype Tool

The following margins are suggested for revision:

- Margin_SFC is suggested to be increased due to decrease in the Level of Confidence of assumption 'SFC adjustment' from Moderate to Low
- Margin_AR is suggested to be reduced due to validation of assumption 'AR estimation'

Figure 6.30: Suggested margin revisions in the presence of change (Software prototype tool)

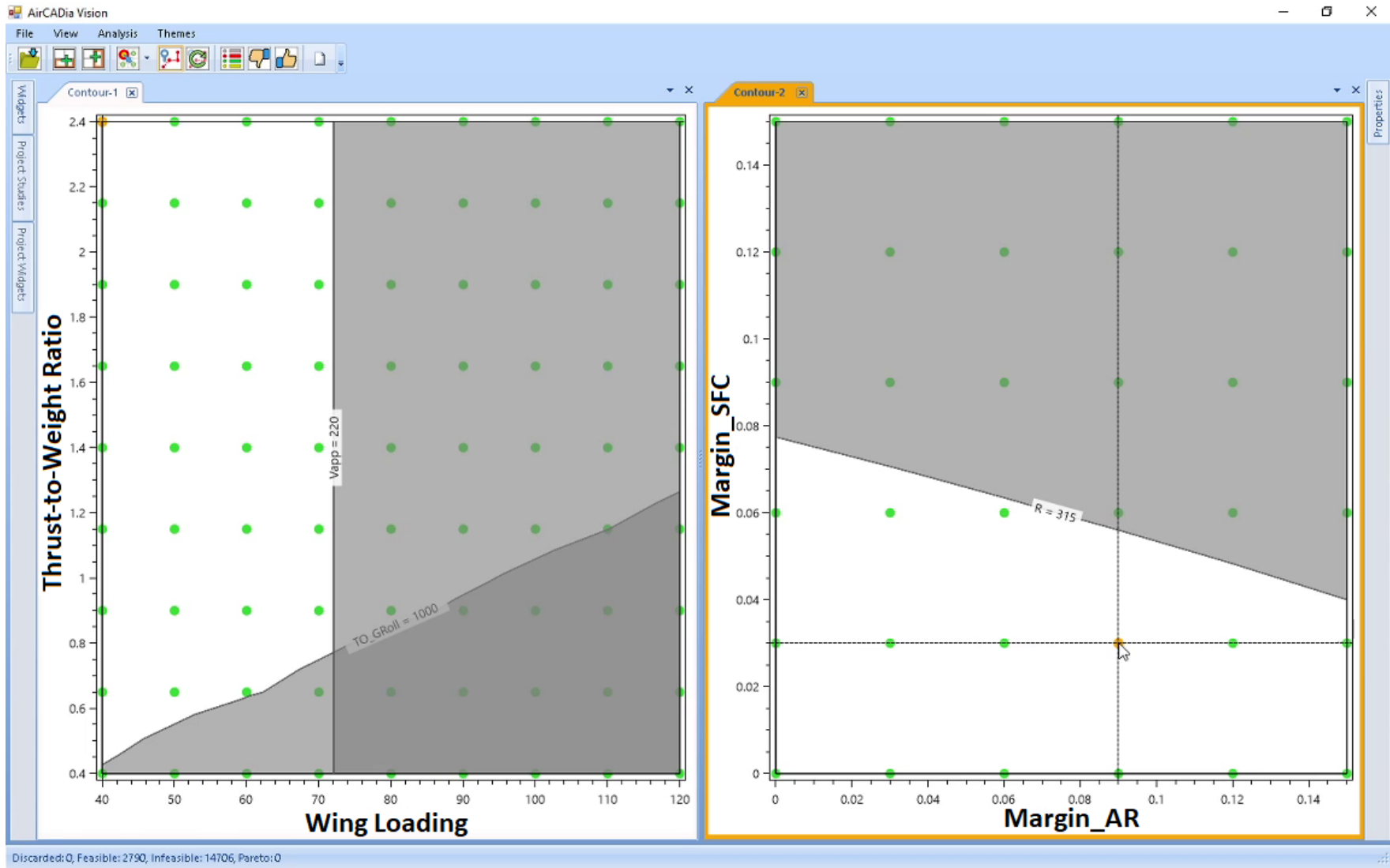


Figure 6.31: Margin Space (AirCADia Vision)

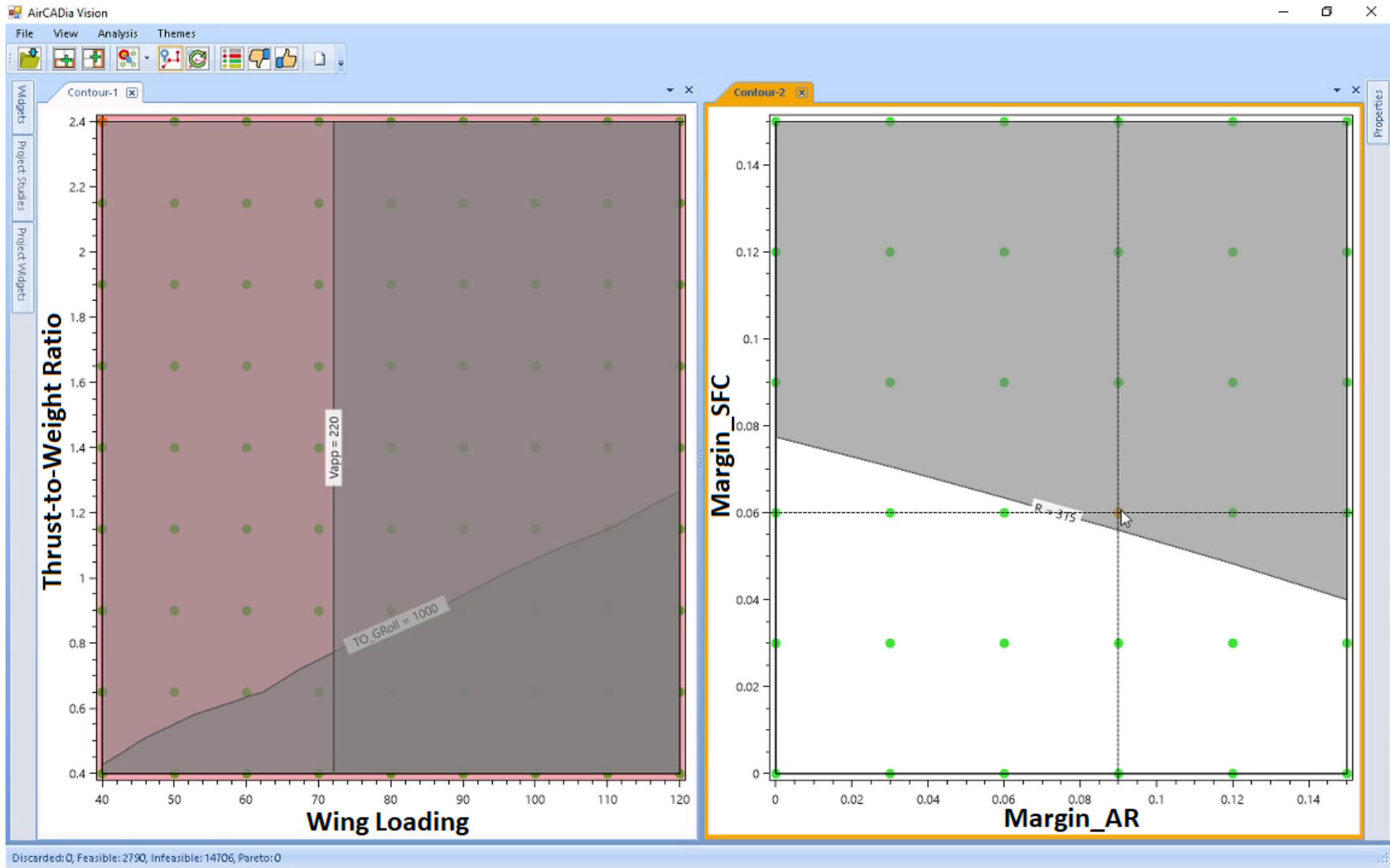


Figure 6.32: Constraint violation (AirCADia Vision)

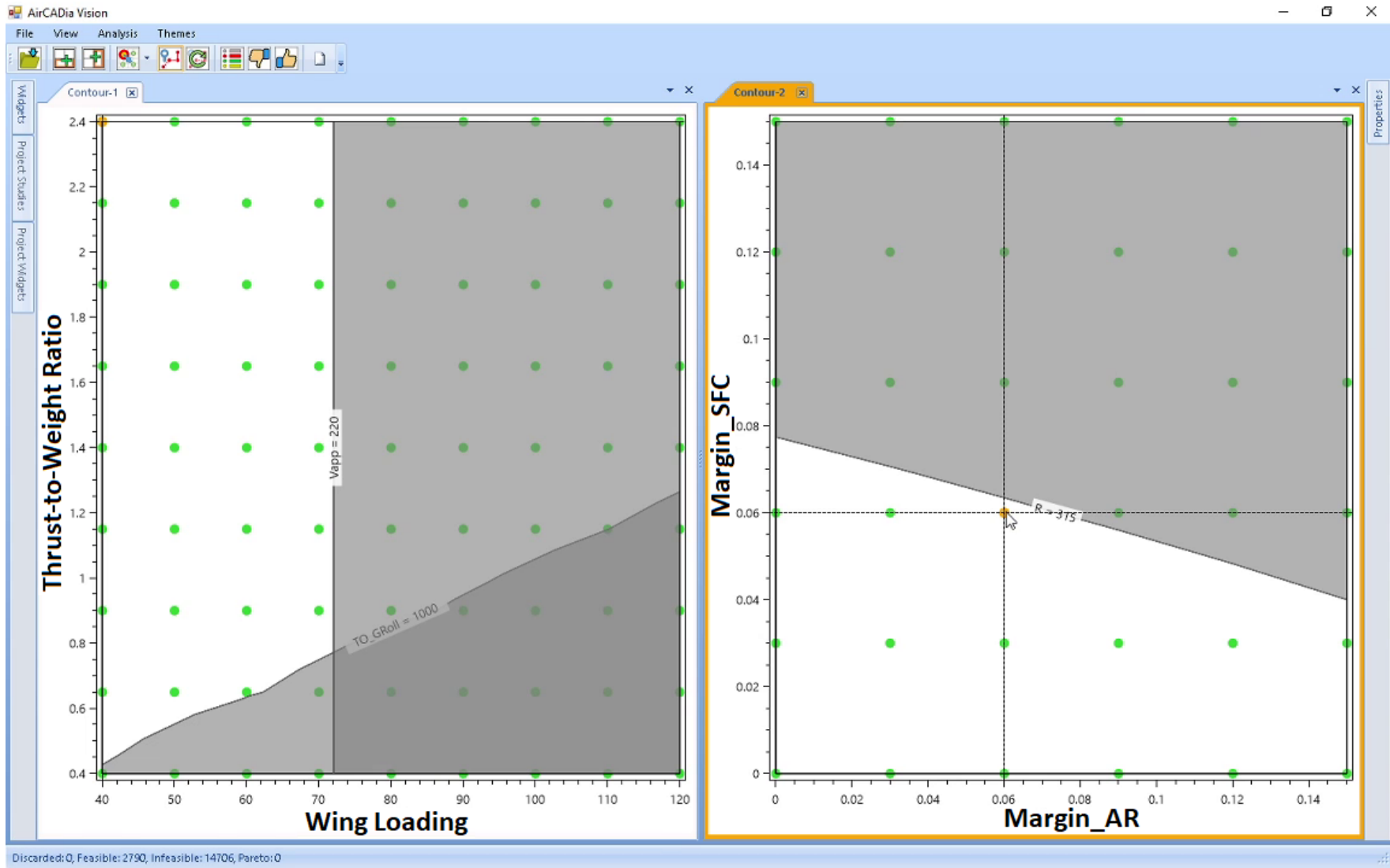


Figure 6.33: Constraint satisfaction (AirCADia Vision)

6.4.3 Detection of Conflicting Assumptions

To demonstrate the detection of conflicting assumptions, the method presented in Section 4.4 is applied to Use Case 2 (as described in Section 6.3.2). Executing the computational workflow corresponding to Use Case 2 (cf. Appendix B) led to the length of the actuator ($L_{act} = 0.8574m$) being estimated as larger than the available gap between the slat and the front spar ($Front_Gap = 0.6405m$). Therefore, the spatial constraint for accommodating the slat actuation system is violated.

According to the proposed method (Section 4.4), assumptions contributing to the constraint violation are collected via traversing the computational workflow. Such graph traversal shall start from both L_{act} and $Front_Gap$, as these two parameters are directly involved in the constraint formulation (i.e. $L_{act} \leq Front_Gap$). This is illustrated by Figure 6.34, where the detected set of conflicting assumptions was returned as follows: $\{A_2, A_4, A_7, A_9, A_{10}\}$. The assumptions in this set are associated with the parameters L_{stroke} , $y_{slat,in}$, c_s/c , x_{FS} and Λ_{LE} , respectively. Since all the aforementioned parameters are inputs to single-output models, there is no need to vary them to determine whether they influence the constraint violation (recall Step 3 in Section 4.4.1). Therefore, the set of conflicting assumptions that needs to be resolved in order to satisfy the constraint $L_{act} \leq Front_Gap$ is the following: $S_{CA} = \{A_2, A_4, A_7, A_9, A_{10}\}$.

To resolve the conflict, one or more of the assumptions in S_{CA} can be modified so that the new combination of assumptions leads to constraint satisfaction. To explore the different combinations, a full-factorial, 6-level Design of Experiment (DoE) involving the parameters L_{stroke} , $y_{slat,in}$, c_s/c , x_{FS} and Λ_{LE} is performed according to the ranges in Table 6.5. This leads to a total of $6^5 = 7776$ combinations. Out of these 7776 combinations, only 1904 lead to $L_{act} \leq Front_Gap$, which can be visualised using a parallel coordinates plot (as shown in Figure 6.35). A parallel coordinates plot is a visualisation technique for multidimensional data, where each polyline represents one combination.

```

51 T1 = nx.dfs_tree(WF_Rev, source="L_act")
52 for e in list(T1.edges()):
53     try:
54         if WF_Rev.nodes[e[1]]['type'] == 'Assumption':
55             Conflict_Asmpts.append(e[1])
56     except:
57         continue
58
59 T2 = nx.dfs_tree(WF_Rev, source="Front_Gap")
60 for e in list(T2.edges()):
61     try:
62         if WF_Rev.nodes[e[1]]['type'] == 'Assumption':
63             Conflict_Asmpts.append(e[1])
64     except:
65         continue
66
The detected set of conflicting assumptions: ['A_2', 'A_10', 'A_9', 'A_4', 'A_7']

```

Graph traversal starting from L_act

Graph traversal starting from Front_Gap

Figure 6.34: Detected set of conflicting assumptions in Use Case 2 (Prototype Tool)

Parameter	Range
L_{stroke}	[0.2, 0.4]
$y_{slat,in}$	[10, 15]
c_s/c	[0.1, 0.2]
x_{FS}	[0.2, 0.3]
Λ_{LE}	[30, 35]

Table 6.5: Full-factorial DoE setup

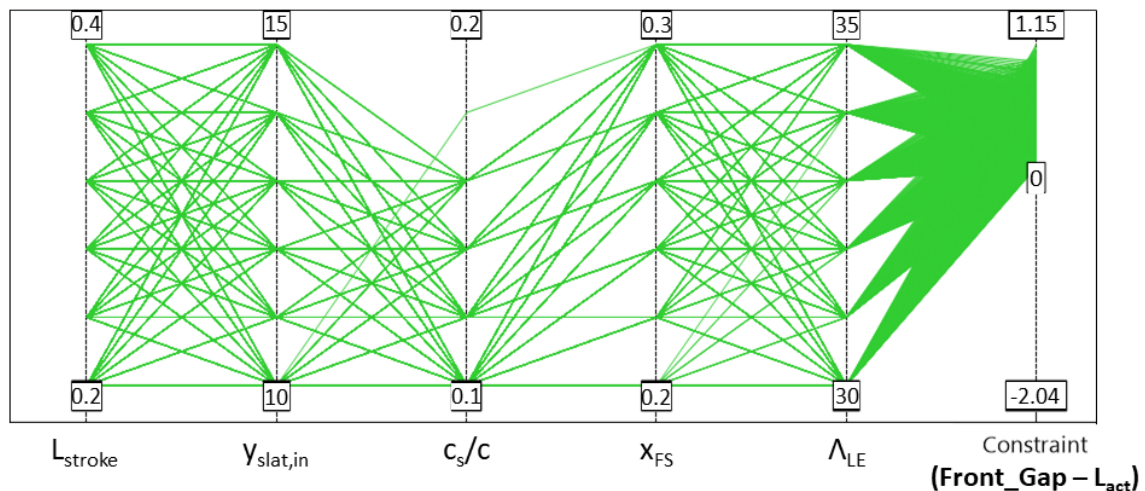


Figure 6.35: Parallel coordinates plot showing feasible combinations (AirCADia Vision)

As mentioned earlier, one or more of the assumptions in S_{CA} can be modified so that the new combination of assumed values leads to constraint satisfaction. For instance, the assumptions underlying x_{FS} and c_s/c shall be revised for demonstration purposes. The

High-Lift Devices team initially assumed c_s/c to be 15% based on previous experience. The value of c_s/c can be modified since it is a design variable for the High-Lift Devices team. In contrast, x_{FS} is not a design variable for the High-Lift Devices team, and so its value had to be initially assumed based on a design rule. The fact that A_9 was detected as a conflicting assumption shows that the design rule is not applicable in this particular context. Thus, a new estimation of the front spar location is needed. Figure 6.36 shows a contour plot, where the red dot refers to the originally assumed values (i.e. $x_{FS} = 0.25$ and $c_s/c = 0.15$). Since this combination violates the spatial constraint for accommodating the slat actuation system, the assumed values can be revised as $x_{FS} = 0.26$ and $c_s/c = 0.14$, where Figure 6.36 shows that this new combination (represented by the blue dot) satisfies the constraint. In practice, constraints would not be considered independently, since the objective is to find a solution that satisfies all the constraints simultaneously. To illustrate this, an additional constraint can be considered in this example. Figure 6.37 shows the addition of the constraint on maximum lift coefficient (C_{Lmax}), which shall be at least 2. This shows that the aforementioned combination (represented by the blue dot) satisfies the constraint on maximum lift coefficient as well.

Therefore, by revising the assumptions on both x_{FS} and c_s/c (such that $x_{FS} = 0.26$ and $c_s/c = 0.14$), while the assumptions on L_{stroke} , $y_{slat,in}$ and Λ_{LE} remain unchanged, the spatial constraint for accommodating the slat actuation system is now satisfied.

It is important to note that the number of generated combinations (i.e. 7776 in this demonstration) is dictated by the choice of the sampling approach. Therefore, the computational cost of the proposed method can be reduced by using other sampling strategies such as Latin hypercube [185], which are more efficient than full-factorial sampling. This constitutes an avenue for future work.

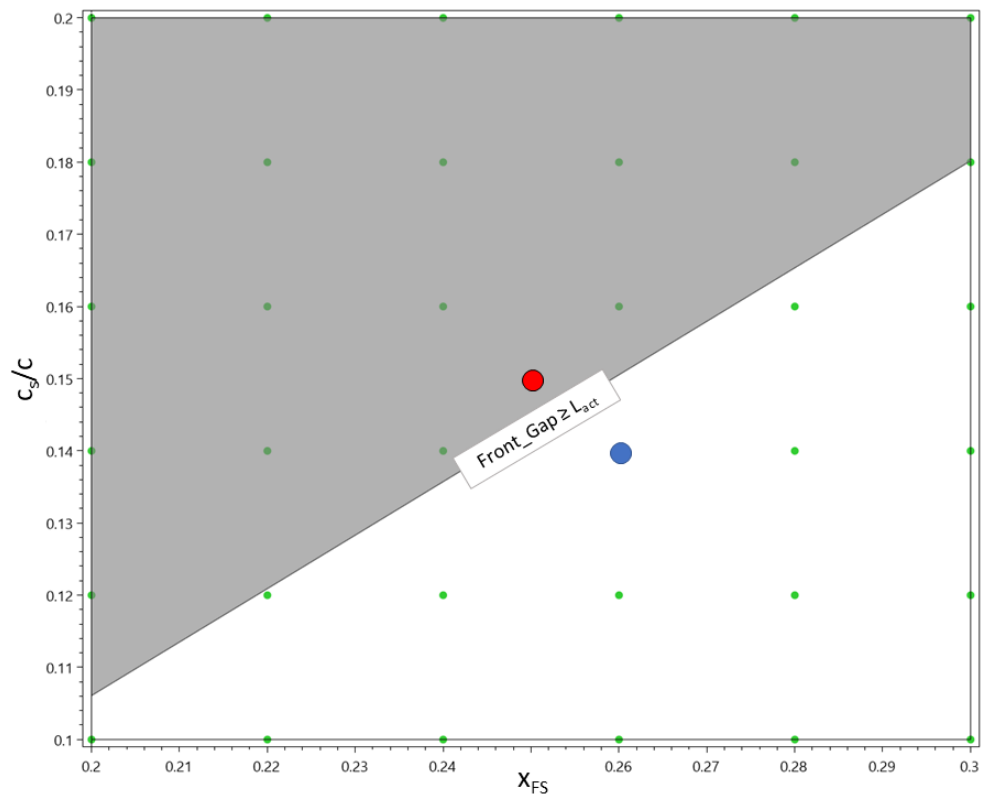
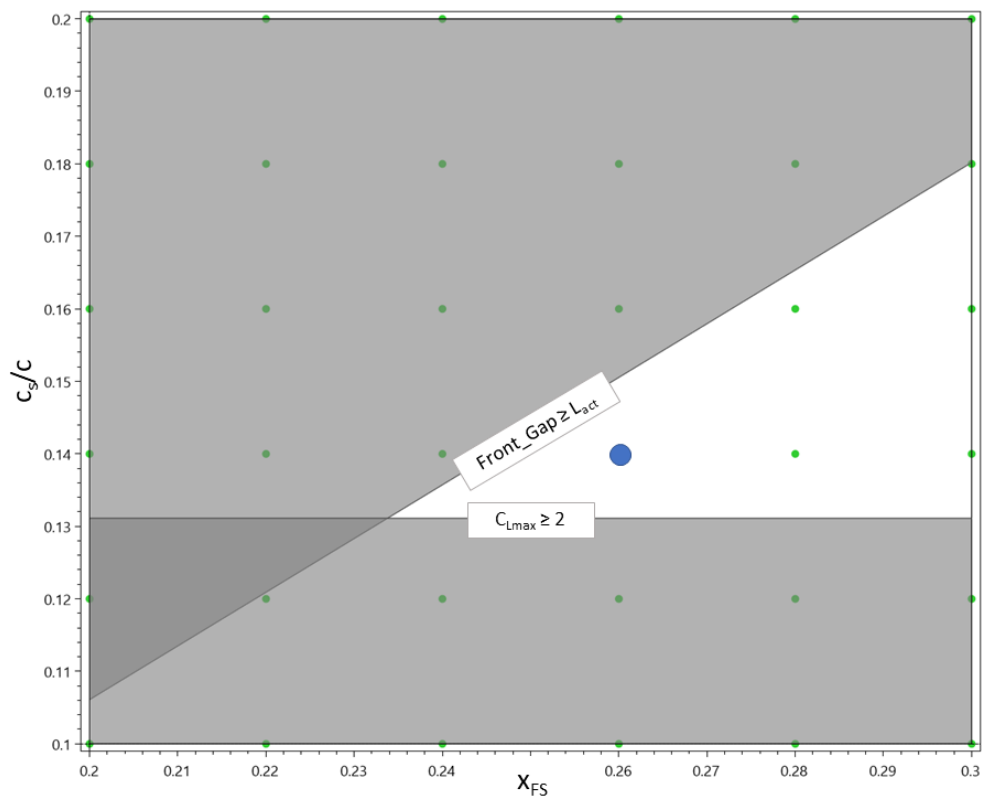
Figure 6.36: Contour plot of c_s/c vs. x_{FS} (AirCADia Vision)

Figure 6.37: Contour plot with an additional constraint (AirCADia Vision)

6.5 Industry Feedback

6.5.1 Purpose

The final part of the evaluation consisted of an industrial feedback session, where the objectives were to:

1. Introduce the research to a group of experts from industry.
2. Obtain feedback regarding the usefulness and industrial relevance of the proposed approach, and suggestions for further improvement.

6.5.2 Approach

Before the evaluation session took place, all the participants were asked to complete and sign a consent form, where they are informed about their right to withdraw, the anonymisation of collected data as well as the security of data storage.

The evaluation session took place on the 26th of May 2021 via an online videoconference. This session lasted for an hour and fifteen minutes, and was structured as follows:

1. Introductory presentation and demonstration (30 min), where background to my research was provided and the developed methods (except Algorithm 2) were demonstrated through Use Case 1 (Section 6.3.1).
2. Questions and discussion (30 min), to further elaborate on my methods and discuss the participants' comments.
3. Questionnaire (15 min), where the evaluation participants were invited to fill out an online questionnaire (Appendix D) consisting of Likert-type and open-ended questions. This was facilitated by the online tool *Microsoft Forms*⁶.

Four experts from Airbus provided feedback through the online questionnaire. The credentials of the experts are summarised in Table 6.6.

⁶<https://forms.office.com/> (Accessed: 26/11/2021)

Table 6.6: Participants information

	Job title	Years of relevant experience
Participant 1	Special Advisor to the Head of Engineering Strategy Process Methods and Tools	35
Participant 2	Expert in Modelling and Simulation - Engineering Methods Department	35
Participant 3	Expert in Modelling and Simulation, Systems Incubation & Integration	25+
Participant 4	Research Project Leader	30

6.5.3 Results

What follows are the collected responses to the Likert-type questions. The possible answers to any of the questions are *Strongly disagree*, *Disagree*, *Neither agree nor disagree*, *Agree* and *Strongly agree*.

The question numbers are consistent with the original form (cf. Appendix D), which explains starting the question enumeration at 6.1. The colour-code in Figures 6.38-6.42 allows to distinguish between the participants' answers.

Design Belief Network

Questions 6.1, 6.2, 6.3 and 6.4, presented in Table 6.7, are related to the *DBN* method. The answers to these questions are summarised in Figure 6.38.

Table 6.7: Likert-type questions related to DBN

Question No.	Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option
6.1	The proposed network contributes to the industrial need for formal (model-based) assumption management.
6.2	The proposed network improves the integration of assumption management within the design environment.
6.3	The proposed network is useful at providing traceability to systems engineering activities (such as Requirements Management).
6.4	The three criteria: <i>data reliability</i> , <i>model realism</i> , and <i>expert agreement</i> provide a reasonable indication of the practitioner's confidence in making an assumption.

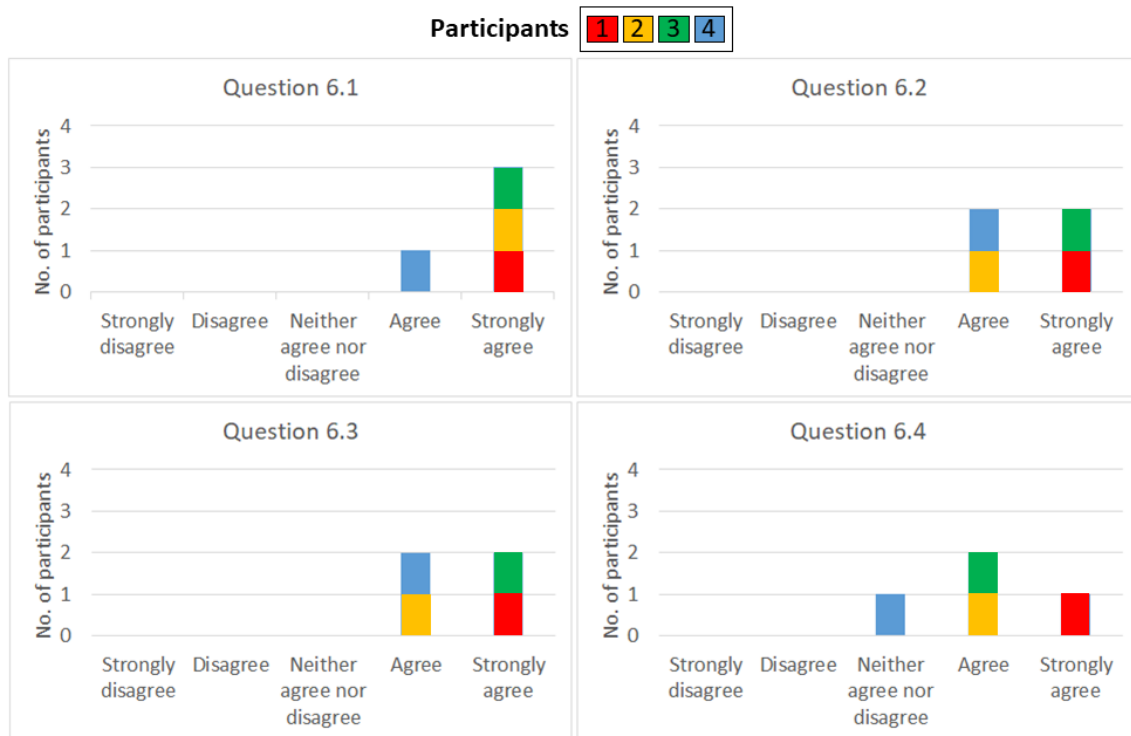


Figure 6.38: Answers to Questions 6.1, 6.2, 6.3 and 6.4

Responses to Questions 6.1, 6.2 and 6.3 show that all participants either *Agree* or *Strongly agree*. This supports the claims regarding the *DBN* contributing to the industrial need for model-based assumption management, improving the integration of assumption management within the design environment, and providing traceability to systems engineering activities. For Question 6.4, all participants but one either *Agree* or *Strongly agree* with the claim that *data reliability*, *model realism*, and *expert agreement* provide a reasonable indication of the practitioner's confidence in making an assumption. Participant 4 provided a neutral response.

Knowledge Maturity Assessment

Questions 7.1 and 7.2, presented in Table 6.8, are related to the knowledge maturity assessment approach. The answers to these questions are summarised in Figure 6.39.

Responses to Question 7.1 show that all participants but one either *Agree* or *Strongly agree* with the claim that the knowledge maturity assessment approach provides a reasonable indication of the overall risk of change due to lack of knowledge. Participant 4

Table 6.8: Likert-type questions related to Knowledge Maturity Assessment

Question No.	Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option
7.1	Knowledge Maturity assessment provides a reasonable indication of the overall risk of change due to lack of knowledge.
7.2	The presented ability to assess progress of Knowledge Maturity over time is useful for supporting decisions at gate reviews.

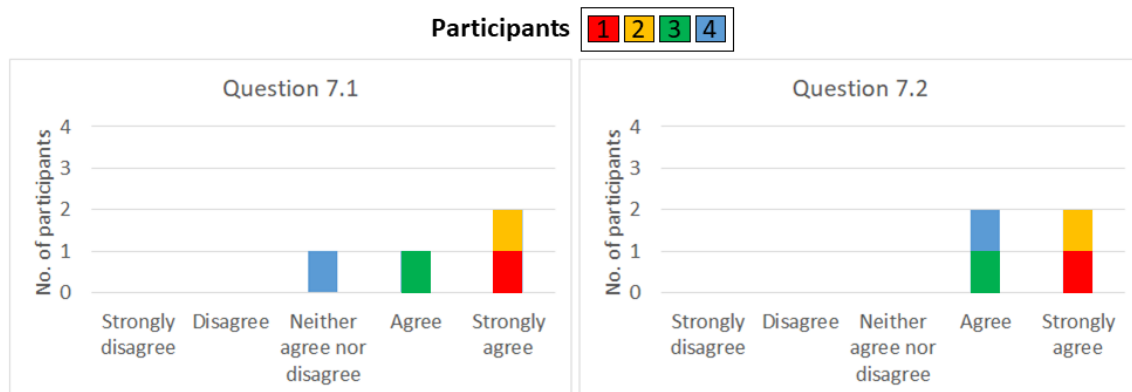


Figure 6.39: Answers to Questions 7.1 and 7.2

provided a neutral response. For Question 7.2, all participants either *Agree* or *Strongly agree* with the claim that the presented ability to assess progress of Knowledge Maturity over time is useful for supporting decisions at gate reviews.

Assumption Matrix and Change Absorber Localisation

Questions 8.1, 8.2 and 8.3, presented in Table 6.9, are related to the *Assumption Matrix* and *Change Absorber Localisation* methods. The answers to these questions are summarised in Figure 6.40.

Responses to Question 8.1 show a high degree of agreement, thus supporting the claim that the *Assumption Matrix* provides a reasonable means of prioritising assumptions in terms of their risk to initiate change. For Questions 8.2 and 8.3, all participants but one either *Agree* or *Strongly agree* with the claims that (i) the ability to suggest a change propagation path up to the closest change absorber is useful for supporting risk assessment, and (ii) the presented approach to identify design decisions and assumptions, potentially affected by change propagation, is useful for supporting risk assessment. There

Table 6.9: Likert-type questions related to Assumption Matrix and Change Absorber Localisation

Question No.	Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option
8.1	The Assumption Matrix provides a reasonable means of prioritising assumptions in terms of their risk to initiate change.
8.2	The ability to suggest a change propagation path up to the closest change absorber is useful for supporting risk assessment.
8.3	The presented approach to identify design decisions and assumptions, potentially affected by change propagation, is useful for supporting risk assessment.

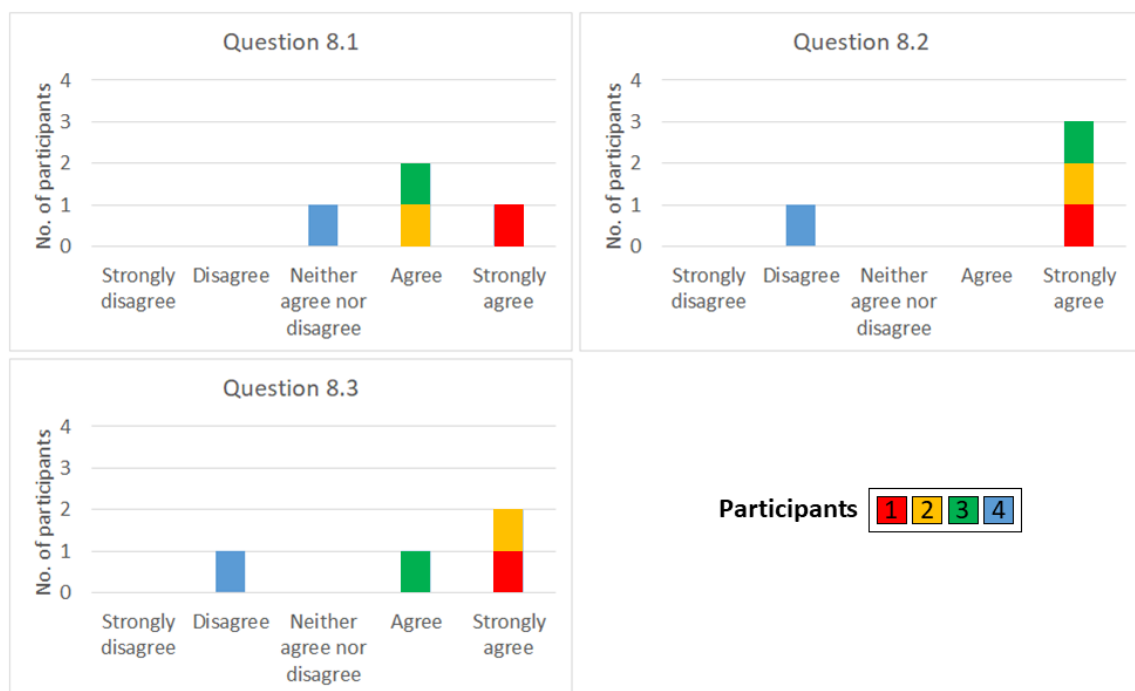


Figure 6.40: Answers to Questions 8.1, 8.2 and 8.3

was only one *Disagree* response to Questions 8.2 and 8.3 (Participant 4).

Margin Allocation and Revision

Questions 9.1, 9.2 and 9.3, presented in Table 6.10, are related to the *Margin Allocation and Revision* methods. The answers to these questions are summarised in Figure 6.41.

Responses to Question 9.1 show a high degree of agreement, thus supporting the claim that providing the status of margin allocation, using the explicit association with assumptions, is useful for supporting margin management. For Questions 9.2 and 9.3, all parti-

Table 6.10: Likert-type questions related to Margin Allocation and Revision

Question No.	Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option
9.1	Providing the status of margin allocation, using the explicit association with assumptions, is useful for supporting margin management.
9.2	The presented approach to detect margin redundancy is reasonable.
9.3	The presented approach for suggesting margins to be revised, is useful for supporting margin management.

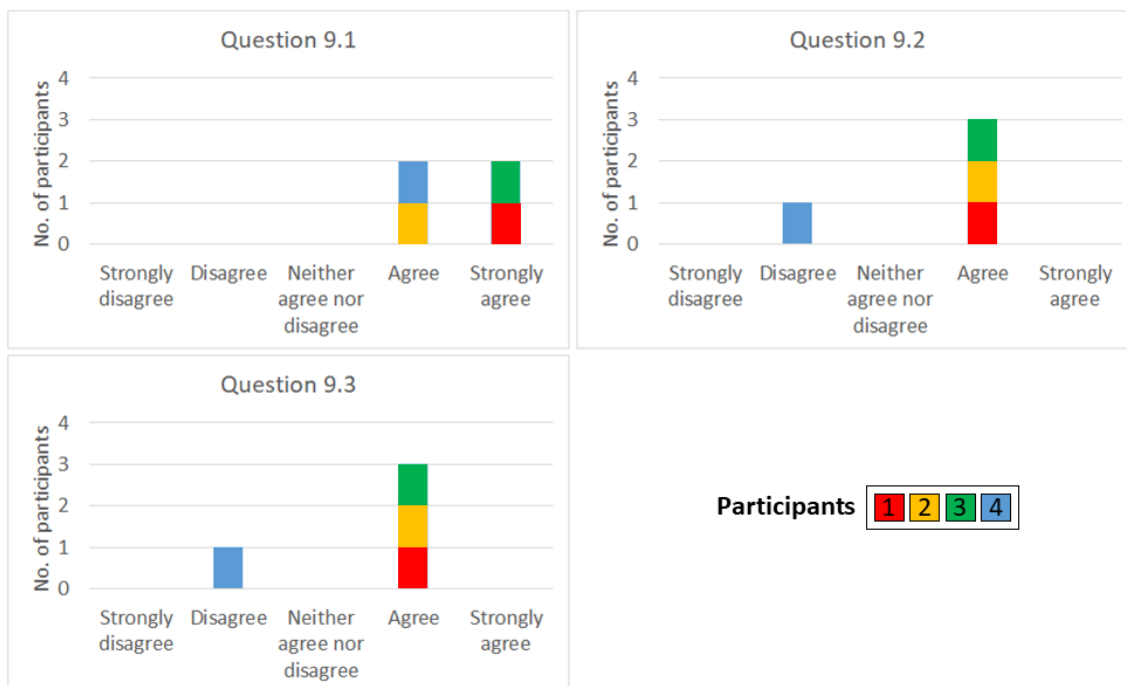


Figure 6.41: Answers to Questions 9.1, 9.2 and 9.3

participants but one (Participant 4) *Agree* with the claims that the presented approach to detect margin redundancy is reasonable, and the presented approach for suggesting margins to be revised is useful for supporting margin management. There was only one *Disagree* response to Questions 9.2 and 9.3 (Participant 4). This may indicate that other factors exist which could be affecting margins concurrently with assumptions.

Overall Approach

Questions 10.1, 10.2 and 10.3, presented in Table 6.11, are related to the overall approach. The answers to these questions are summarised in Figure 6.42.

Table 6.11: Likert-type questions related to the overall approach

Question No.	Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option
10.1	The presented methods can lead to fewer undesired iterations, due to earlier identification and management of risks associated with assumptions.
10.2	The presented methods can lead to a better margin balance, due to timely and interactive margin revision.
10.3	An interactive software tool, similar to the one presented, would be a useful addition to a company's suite of tools.

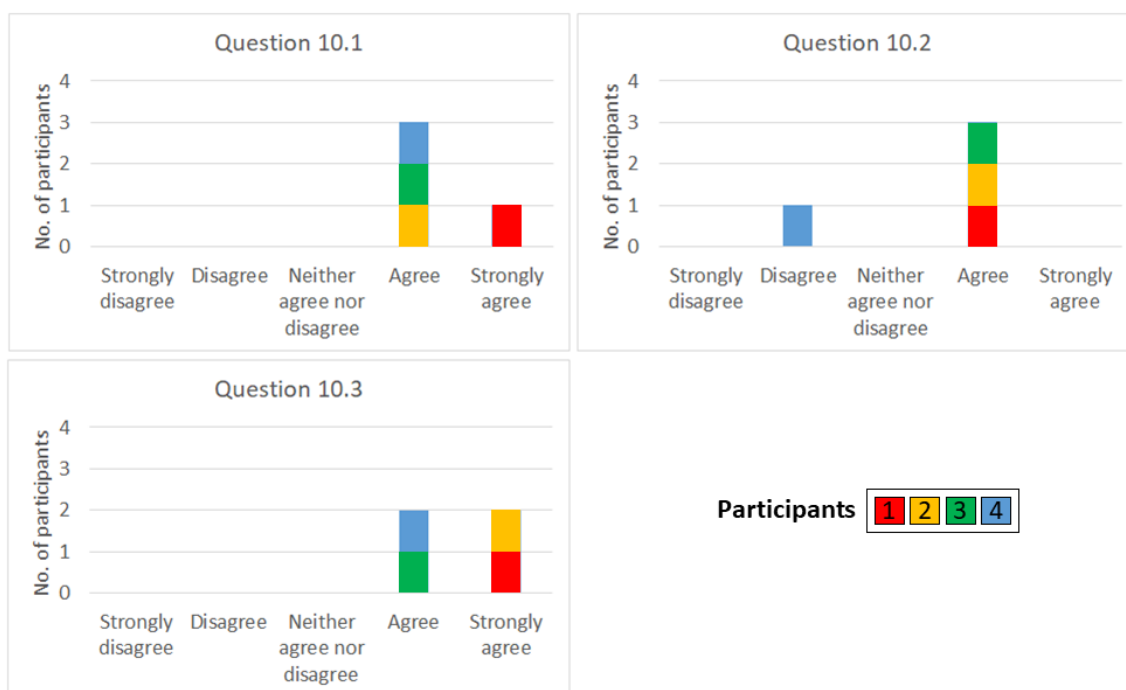


Figure 6.42: Answers to Questions 10.1, 10.2 and 10.3

Responses to Questions 10.1 and 10.3 show a high degree of agreement, thus supporting the claims that: (i) all the presented methods can lead to fewer undesired iterations, due to earlier identification and management of risks associated with assumptions; and (ii) an interactive software tool, similar to the one presented in the demonstration, would be a useful addition to a company's suite of tools. For Question 10.2, all participants but one *Agree* with the claim that all the presented methods can lead to a better margin balance, due to timely and interactive margin revision. There was only one *Disagree* response to Question 10.2 (Participant 4), which follows from the disagreement of Participant 4 with

the claims in Questions 9.2 and 9.3 (Figure 6.41).

Open Questions

The three open-ended questions that close the questionnaire are presented in Table 6.12. The responses to Question 11 were all positive. Participant 1 wrote: *“Really very good approach on assumptions management which in the industry is not enough explicit. Here it will be and will support right risks analysis.”* Participant 2 thought *“it was a well presented set of methods that would add value to [their] business”*. Participant 3 commented it was a *“good, clear presentation of both theory and implementation (demo)”*. Whereas Participant 4 wrote: *“the tool produced seems to be very useful, intuitive and simple for users”*.

Regarding Question 12, Participant 1 thinks that *“the perspective of designing A/C [aircraft] but any complex or simple system design should include [the presented] approach”*. Participant 2 believes that a *“further exploitation of the [Assumption] matrix”* would add value. Whereas Participant 3 sees value in *“augmenting the existing MBSE tools as an integral aspect of Systems Engineering”*. Participant 4 did not respond to Question 12.

The responses to Question 13 were useful for obtaining ideas for further improvement. Participant 1 suggested to *“include the automation and the modeling of assumptions in the same trend as the requirements modeling”* in the implementation, and commented that *“[the presented approach] is really innovative and relevant”*. Participant 2 suggested *“it might be interesting to see if [the presented] definition of assumptions can be mapped to the definitions in the AP243 (MoSSEC) standard”*, adding that *“this would help ensure [the presented] methods could be used to collate/share assumption data across multiple platforms”*. Participant 4 wrote: *“My comments are on the accuracy of the computational values but I don’t know if for early usage, it is a real problem or not”*, which may explain some of the neutral/negative responses of Participant 4 to previous questions. Additionally, Participant 4 suggested to *“take into account [...] stochastic dependency”*.

Participant 3 did not respond to Question 13.

Table 6.12: Open-ended questions

Question No.	Question Text
11	Is there anything that you particularly liked/disliked about any of the presented methods?
12	In what other ways do you think the presented methods could add value?
13	Do you have any other comments or suggestions?

According to the presented results, the industrial evaluation was successful in supporting the expected impact of the proposed methods and their usefulness to industry. Some suggestions from the participants for further improvement include more automation in assumption capturing, more alignment with existing data sharing standards, and the consideration of stochastic dependency. Further evaluation, particularly in an industrial setting with the intended users involved, is desirable. This would lead to further refinement, and is thus the subject of future work.

6.6 Summary and Conclusions

This chapter covered the evaluation of the research, in accordance with a *Type 3* study of the DRM [20]. This preliminary evaluation consisted of two parts:

- A *demonstration*, which consisted of applying the developed methods to: (1) the hypothetical design of a fighter aircraft and (2) conflicting assumptions in collaborative design, in order to assess whether the methods work as intended.
- An *industry feedback* session, which consisted of demonstrating the developed methods to industry experts to obtain feedback on expected usefulness in practice, thus assessing the impact of this research.

The results from the demonstration, as enabled by a software implementation, indicate that the proposed methods function correctly and provide expected results. Therefore, the applicability of the developed methods was successfully assessed. An industry feedback

session was conducted with a panel of experts from Airbus, where the positive overall feedback allowed to successfully assess the usefulness and industrial relevance of the proposed approach.

The demonstration of the *Design Belief Network* method showed an approach to capturing assumptions, their uncertainty and dependencies as part of a model-based design environment. The positive expert feedback supports the claims regarding the *DBN* contributing to the industrial need for model-based assumption management, improving the integration of assumption management within the design environment, and providing traceability to systems engineering activities. Furthermore, there was a high degree of agreement amongst experts regarding the claim that *data reliability*, *model realism*, and *expert agreement* provide a reasonable indication of the practitioner's confidence in making an assumption.

The demonstration of *Conflict Detection* showed a computational strategy to automatically detect a subset of assumptions leading to constraint violation. Therefore, the applicability of the proposed method was successfully assessed. However, the usefulness and industrial relevance still remain to be assessed via seeking expert feedback.

The demonstration of the *Knowledge Maturity Assessment* approach showed an assessment of the overall risk associated with assumptions. The positive expert feedback supports the claims that knowledge maturity assessment provides a reasonable indication of the overall risk of change due to lack of knowledge, and that the presented ability to assess progress of Knowledge Maturity over time is useful for supporting decisions at gate reviews.

The demonstration of the *Assumption Matrix* method showed a prioritisation of assumptions in terms of their risk to initiate change, in addition to the ability to visualise the assumptions prioritisation directly from the system model. The panel of experts agreed that the *Assumption Matrix* provides a reasonable means of prioritising assumptions.

The demonstration of the *Margin Allocation* method showed the status of allocated margins, which includes margin-assumptions associations and detected instances of mar-

gin redundancy. Expert feedback was positive overall, therefore supporting the claims that: (i) providing the status of margin allocation, using the explicit association with assumptions, is useful for supporting margin management; and (ii) the presented approach to detect margin redundancy is reasonable. A negative response was though collected, which may indicate that other (unconsidered) factors may be affecting margins concurrently with assumptions. Thus, further research regarding the explicit relationship between margins and assumptions, while considering the different systems engineering activities, may be needed to identify such extra factors.

The demonstration of the *Change Absorber Localisation* method showed the detection of a change absorber to mitigate the risk of a selected assumption, along with the potential propagation path from the assumption to the change absorber. Although there was a high degree of agreement regarding the proposed method being useful for supporting risk management, one negative response pointed to the lack of considering stochastic dependency. This shall be explored as part of future work.

The demonstration of the *Margin Revision* method showed suggestions for margin revision following changes made in the assumptions, in addition to a margin trade-off that allowed to update the risk mitigation strategy (according to the suggested revisions), while still satisfying the constraints. Although there was a high degree of agreement regarding the proposed method being useful for supporting margin management, there was one negative response which could also indicate that other factors may be affecting margins concurrently with assumptions.

Overall, all the proposed methods were deemed innovative, useful and relevant to industry by the panel of experts, where the presented methods can lead to: (i) fewer undesired iterations, due to earlier identification and management of risks associated with assumptions; and (ii) a better margin balance, due to timely and interactive margin revision. Additionally, an interactive software tool, similar to the one presented in the demonstration, would be a useful addition to a company's suite of tools. Some suggestions for further improvement include more automation in assumption capturing, more alignment

with existing data sharing standards, and the consideration of stochastic dependency.

In conclusion, the developed methods have been assessed through a successful preliminary evaluation, thus achieving both research objectives. Expert feedback on the assumption management process description (Section 4.2.2) and Algorithm 2 for conflict detection shall be collected as part of future work.

Chapter 7

Summary and Conclusions

7.1 Introduction

In this chapter, the main body of this thesis is concluded. First, a summary of the research is provided in Section 7.2. This is followed by a discussion of the contributions to knowledge and engineering practice in Section 7.3. Finally, the limitations of the research and suggestions for future work are provided in Section 7.4.

7.2 Research Summary

The research followed the four stages of the Design Research Methodology (DRM) [20]. The first stage, *Research Clarification*, consisted of an initial literature analysis to clarify the overall research aim, develop a research plan, and frame the subsequent stages. The literature was analysed for an initial understanding of the context in which this research is taking place, with a focus on assumptions. *Research Clarification* led to formulating the aim, objectives and scope of this research. The aim was to *develop a computational approach to support assumption management in model-based systems engineering, with an explicit consideration of the uncertainty in assumptions*. The objectives were to (1) *devise methods to enable assumption management in a model-based design environment*; and (2) *devise methods to manage risk of change due to invalid assumptions, with an ex-*

plicit consideration of both assumptions and margins. The scope was limited to the early design stages of aircraft and technical risk management. Furthermore, organisational aspects were considered outside the scope as the focus was on computational support.

The second stage, *Descriptive Study I*, consisted of a more in-depth literature review in order to improve the understanding of assumptions in the early design of complex systems, in addition to identifying state-of-the-art academic research and current industrial practice in managing assumptions. This led to highlighting the limitations (and associated opportunities for improvement) of existing approaches to epistemic uncertainty management, and also identifying methods and tools from different fields to inform support development. The identified limitations and opportunities for improvement were summarised in Section 3.6.

The third stage, *Prescriptive Study*, consisted of developing the computational support to achieve the research aim. The synthesis, informed by *Descriptive Study I*, involved adapting and extending concepts and methods from different fields such as engineering design, systems engineering, mathematics and computer science. This led to devising novel methods that achieve the research objectives.

To address the first objective, a graph-theoretical structure to capture assumptions, their uncertainty and dependencies in a model-based manner was proposed. Additionally, an algorithm to detect conflicting assumptions that lead to constraint violation was developed. To address the second objective, a set of methods was proposed to: (i) assess the risk of change due to lack of knowledge; (ii) prioritise assumptions in terms of their risk to initiate change; (iii) explicitly associate margins with assumptions for risk mitigation; (iv) detect the closest margin to an assumption for change absorption; and (v) suggest margin revisions following changes in assumptions.

The fourth and final stage, *Descriptive Study II*, involved two forms of assessment. First, a demonstration, which consisted of applying the developed methods to: (1) the hypothetical design of a fighter aircraft and (2) conflicting assumptions in collaborative design, in order to assess whether the methods work as intended. Second, an industry

feedback session, which consisted of demonstrating the developed methods to industry experts to obtain feedback on expected usefulness in practice, thus assessing the impact of this research. To enable research evaluation, the developed methods were implemented into a prototype software tool that interfaced with AirCADia [182], a model-based design and analysis tool.

7.3 Research Contributions

Contributions to knowledge and engineering practice resulting from this research are as follows:

1. A standardised description of the assumption management process, in accordance with ISO/IEC/IEEE 24774:2021 [21]. This description explicitly accounts for the uncertainty in assumptions, and takes into consideration the implications of changes in assumptions. Such aspects were found to be missing in existing descriptions of the assumption's lifecycle.

Additionally, it is argued in this thesis that assumption management should not be limited to the requirements domain, but rather consider as well assumptions related to the functional, logical and computational domains (this is reflected in the Design Belief Network).

2. A graph-theoretical structure (*Design Belief Network*) to capture assumptions, their uncertainty and dependencies in a model-based manner (consistent with the RFLP model). This is in response to an identified need in industry for model-based assumption management, as simply documenting assumptions proved to have little benefit.

Additionally, the assumption dependency types proposed in the literature have been simplified in this research, since these types were found to be non-mutually exclusive, thus inducing a form of redundancy. Such redundancy could result in unnecessarily increasing the effort to capture dependencies, and also potentially confusing

practitioners.

Furthermore, although assumptions have varying degrees of confidence, a lack of assessment of the uncertainty inherent in assumptions was identified in Section 3.2.3. Thus, the proposed method includes an assessment of the level of confidence in assumptions, which is based on the strength of background knowledge.

The positive expert feedback supports the claims regarding the *DBN* contributing to the industrial need for model-based assumption management, improving the integration of assumption management within the design environment, and providing traceability to systems engineering activities.

3. An algorithm to detect conflicting assumptions that lead to constraint violation. The proposed approach is, to some extent, comparable to the one adopted in the TMS (i.e. the 'NOGOOD' set). However, the contribution made in this research is making use of an extra-logical factor (i.e. the Computational Domain) to support the detection of semantic inconsistencies, instead of focusing solely on logical contradictions (as in the TMS). This is expected to reduce the time and cost of identifying conflicting assumptions, especially when it has to be done manually in large scale projects.
4. A composite indicator, *Knowledge Maturity Index (KMI)*, to assess the overall risk of change due to lack of knowledge. The proposed method is based on the explicit use of assumptions as a proxy for estimating the extent of the knowledge gap, where validated assumptions indicate progress towards closing the gap. This is in an attempt to address a limitation identified in the literature review, where although assumption management plays a crucial role in knowledge maturity, no method has been previously proposed to assess knowledge maturity with an explicit consideration of assumptions.

The positive expert feedback supports the claim that the knowledge maturity assessment approach provides a reasonable indication of the overall risk of change due to

lack of knowledge, and that the presented ability to assess progress of Knowledge Maturity over time is useful for supporting decisions at gate reviews.

5. A method (*Assumption Matrix*) to prioritise individual assumptions in terms of their risk to initiate change. Such prioritisation is enabled by both the level of confidence and the dependencies of the captured assumptions. One practical implication of this method is to visualise assumption prioritisation directly from a system model. The *Assumption Matrix* was evaluated by experts as providing a reasonable means of prioritising assumptions.
6. An algorithm to provide the status of margin allocation, with an explicit consideration of assumptions. The status of margin allocation includes the following information: (i) captured associations between margins and assumptions; (ii) assumptions, prioritised according to the *Assumption Matrix*, for which the risk has not been explicitly mitigated by margins; and (iii) detected instances of margin redundancy. This tackles a limitation identified in the literature review, where the explicit relationship between assumptions and margins has not been explored in the published literature.

Positive expert feedback supports the claims that: (i) providing the status of margin allocation, using the explicit association with assumptions, is useful for supporting margin management; and (ii) the presented approach to detect margin redundancy is reasonable.
7. An algorithm to detect the closest margin to an assumption for change absorption. A potential change propagation path is suggested, which starts from an assumption (acting as a *Change Initiator*), and ends at a margin (acting as a *Change Absorber*). This capability is made possible because both assumptions and margins (along with their dependencies) are explicitly captured as part of the *DBN*. This method is in response to limitations identified in the literature review, where: (i) a lack of explicitly considering margins in existing approaches for change propagation prediction

has been identified; and (ii) no approach has been proposed to explicitly relate assumptions to change propagation analysis (although assumptions are known to cause change propagation in engineering design). Positive expert feedback on the proposed method supports the claim about it being useful for supporting risk management.

8. An algorithm to suggest margin revisions following changes in assumptions. This allows to monitor risks associated with assumptions, while updating the risk mitigation strategy accordingly. The *Margin Revision* method can be seen as providing justification for margin revision as the design progresses and more knowledge is acquired, which addresses a limitation identified in the literature review, i.e. a lack of tools that track margins along with the rationale underlying their change. To ensure constraint satisfaction while revising margins, the *Margin Space* concept [161] was successfully applied to this end. This approach is expected to provide an interactive and dynamic revision of margins, with the goal of improving margin balance. Positive expert feedback on the proposed method supports the claim about it being useful for supporting margin management.
9. From a Belief Revision perspective, limitations of existing belief revision approaches were identified, which include: (a) a lack of assessing the uncertainty in assumptions; (b) the fact that TMS must be used in conjunction with an artificial problem-solver; (c) the fact that TMS considers logical contradiction only (i.e. belief set contains both a fact and its negation); (d) dependencies amongst assumptions are not considered; and (e) lack of extra-logical factors to prioritise assumptions for revision. Limitation (a) has been addressed by the LoC assessment criteria. Limitation (b) has been addressed by the interactivity offered by the proposed methods and their implementation. Limitation (c) has been addressed by the *Conflict Detection* method, which can detect conflicting assumptions that lead to constraint violation, and thus goes beyond the limitation of purely logical contradictions. Limitation

(*d*) has been addressed by the assumption intra-domain dependency (i.e. when assumptions are conflicting). Finally, limitation (*e*) has been addressed by (i) the *Assumption Matrix* method, which uses the LoC and dependencies as extra-logical factors; and (ii) the detection of conflicting assumptions, which uses the Computational Domain as an extra-logical factor.

10. Overall, all the proposed methods were deemed innovative, useful and relevant to industry by a panel of experts, where the presented methods can lead to: (i) fewer undesired iterations, due to earlier identification and management of risks associated with assumptions; and (ii) a better margin balance, due to timely and interactive margin revision. Additionally, an interactive software tool, similar to the one presented in the demonstration, would be a useful addition to a company's suite of tools.

7.4 Limitations and Future Work

There are a number of limitations associated with the proposed methods, which could be addressed in future work. These limitations and avenues for future research can be summarised as follows:

1. Regarding the explicit association between margins and assumptions, and the fact that margin revisions are suggested based on changes in assumptions, there could be other factors that can affect margins concurrently with assumptions but were not identified by the author. Thus, exploring such additional factors constitutes an avenue for future work. Furthermore, many concurrent changes in assumptions could potentially lead to conflicting suggestions for margin revision. To address this limitation, belief merging and judgment aggregation [186] seem to be promising approaches to aggregate conflicting margin revision suggestions into a consistent one.

2. Future work shall include considering stochastic dependency, as pointed out in the industry feedback (Section 6.5). The *Change Absorber Localisation* method could potentially be extended to include the likelihood of change to propagate from one specific component/subsystem to another, or how likely a margin is to fully absorb change. Although it may not be possible to estimate such likelihoods, one avenue that could be worth exploring is the use of historical data or expert opinion. For instance in [145], propagation likelihoods between sub-systems were elicited from deputy chief engineers involved in developing the EH101 helicopter.
3. The algorithm to detect conflicting assumptions necessitates making explicit (with a mathematical description) some interfaces between disciplines. However, such interfaces usually remain implicit in practice, which implies that applying the proposed algorithm introduces an extra cost to make domain interfaces explicit. Therefore, future work includes studying the trade-off between the benefit from the proposed approach and the effort needed.
Furthermore, the use of full-factorial sampling while demonstrating the method (Section 6.4.3) was associated with a high computational cost. Therefore, part of future work consists of exploring more efficient sampling techniques such as Latin hypercube.
4. Both the assumption management process description and the conflict detection approach were not evaluated as part of the industry feedback session, which means that their industrial relevance was not validated. Future work therefore includes seeking expert feedback to assess their industrial relevance.
5. The scope of this research was limited to the early-design stages of aircraft and technical risk management. It would be interesting to not only consider detailed design, but also other systems engineering activities such as product verification and validation. It would be also interesting to provide a more detailed description of how assumption evaluation is to be carried out.

Other complex systems could also be considered such as space vehicles, naval ships and automobiles.

6. The physical domain of the RFLP model was not considered as it is outside the scope of this research. However, it would be possible to consider it in early design by using simple shapes such as cuboids and cylinders. This can be used to extend the proposed methods such as the Assumption Matrix, which can then allow for sub-systems to be highlighted in the physical domain (i.e. similar to highlighting elements in the R-F-L domains as demonstrated in Section 6.4.2).
7. Applying the developed methods in a real industrial project, with the intended users involved, would constitute an important part of future work. First, it would allow to evaluate the scalability of the methods since the demonstration in this thesis considered only a limited number of assumptions and a simplified system architecture. Note that the assumption prioritisation provided by the *Assumption Matrix* can help with addressing the issue of scalability when the developed methods are applied in a real industrial setting. Thus, an organisation can decide to focus on high priority assumptions only in order to reduce to reasonable amounts the time and cost associated with the proposed methods. Second, applying the developed methods in a real industrial project would allow to evaluate evolution over time. For example, it would be interesting to study how the KMI actually progresses during the development of a product, or how margins are used up as a response to changes in assumptions.
8. To increase the likelihood of industry adopting the proposed methods, future work includes (a) more automation in assumption capturing to reduce the initial cost of applying the proposed approach; (b) extending SysML to include the *Assumption* class along with its dependencies, since the methods were described in this thesis in a language-agnostic manner, or even considering a new *Assumption Diagram* in SysML to visualise all the assumptions, their dependencies and validation con-

ditions; and (c) more alignment with existing data sharing standards such as the ISO STEP AP243 (MoSSEC) [157] (as suggested by one of the industrial evaluation participants), which would allow to share assumption data across multiple platforms.

References

- [1] B. W. McCormick, *Aerodynamics, Aeronautics, and Flight Mechanics*, 2nd edn. John Wiley & Sons, Inc., 1995.
- [2] E. Torenbeek, *Advanced Aircraft Design*. John Wiley & Sons, Ltd, 2013. DOI: [10.1002/9781118568101](https://doi.org/10.1002/9781118568101).
- [3] S. Ghosh, ‘Concurrent Optimization Using Probabilistic Analysis of Distributed Multidisciplinary Architectures for Design Under Uncertainty,’ PhD Thesis, Georgia Institute of Technology, 2016. [Online]. Available: <http://hdl.handle.net/1853/56302>.
- [4] A. D. Kiureghian and O. Ditlevsen, ‘Aleatory or epistemic? Does it matter?’ *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009. DOI: [10.1016/j.strusafe.2008.06.020](https://doi.org/10.1016/j.strusafe.2008.06.020).
- [5] INCOSE, *INCOSE Systems Engineering Handbook*, D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin and T. M. Shortell, Eds. John Wiley & Sons, Inc., 2015.
- [6] E. Crawley, B. Cameron and D. Selva, *System Architecture: Strategy and Product Development for Complex Systems*. Pearson Education Limited, 2016.
- [7] National Science Foundation, ‘Research Opportunities in Engineering Design, NSF Strategic Planning Workshop Final Report,’ National Science Foundation, Tech. Rep., 1996.

- [8] J. M. Mines, 'A bi-Level framework for aircraft design uncertainty quantification and management,' PhD Thesis, Georgia Institute of Technology, 2019. [Online]. Available: <http://hdl.handle.net/1853/61281>.
- [9] D. J. Singer, N. Doerry and M. E. Buckley, 'What is set-based design?' *Naval Engineers Journal*, vol. 121, no. 4, pp. 31–43, 2009.
- [10] National Research Council, *Software for Dependable Systems: Sufficient Evidence?* D. Jackson, M. Thomas and L. I. Millett, Eds. Washington, D.C.: National Academies Press, 2007. DOI: [10.17226/11923](https://doi.org/10.17226/11923).
- [11] C. Yang, P. Liang and P. Avgeriou, 'Assumptions and their management in software development: A systematic mapping study,' *Information and Software Technology*, vol. 94, pp. 82–110, 2018. DOI: [10.1016/j.infsof.2017.10.003](https://doi.org/10.1016/j.infsof.2017.10.003).
- [12] M. Lehman, 'Software Evolution: Cause or Effect,' in *Stevens Memorial Lecture, ICSM 2003*, Amsterdam, 2003.
- [13] NTSB, 'ASR-19-01: Assumptions Used in the Safety Assessment Process and the Effects of Multiple Alerts and Indications on Pilot Performance,' National Transportation Safety Board, Washington, DC, Tech. Rep., 2019.
- [14] A. Shore, 'How assumptions can undermine safety,' in *IET Seminar on Safety Assessments: when is enough enough?*, London: IET, 2010. DOI: [10.1049/ic.2010.0003](https://doi.org/10.1049/ic.2010.0003).
- [15] ESA, 'Ariane 5 Inquiry Board Report,' European Space Agency, Paris, Tech. Rep., 1996.
- [16] ISO, *ISO/IEC/IEEE 15288: Systems and software engineering - System life cycle processes*. IEEE, 2015. [Online]. Available: <https://standards.ieee.org/standard/15288-2015.html>.
- [17] NASA, *NASA SP-2016-6105 Rev2: NASA Systems Engineering Handbook*. NASA, 2016.

- [18] A. Sadlauer, P. Hehenberger and K. Zeman, 'The influence of documenting assumed values of product properties on the number of iterations in the design process - first observations,' *International Journal of Information Technology and Management*, vol. 16, no. 1, pp. 73–90, 2017. DOI: [10.1504/IJITM.2017.080951](https://doi.org/10.1504/IJITM.2017.080951).
- [19] A. M. Madni and M. Sievers, 'Model-Based Systems Engineering: Motivation, Current Status, and Needed Advances,' in *Disciplinary Convergence in Systems Engineering Research*, Cham: Springer International Publishing, 2018, pp. 311–325. DOI: [10.1007/978-3-319-62217-0_22](https://doi.org/10.1007/978-3-319-62217-0_22).
- [20] L. T. Blessing and A. Chakrabarti, *DRM, a Design Research Methodology*. Springer-Verlag London, 2009. DOI: [10.1007/978-1-84882-587-1](https://doi.org/10.1007/978-1-84882-587-1).
- [21] ISO, *ISO/IEC/IEEE 24774:2021: Systems and software engineering — Life cycle management — Specification for process description*. ISO, 2021. [Online]. Available: <https://www.iso.org/standard/78981.html>.
- [22] International Council on Systems Engineering, *Systems Engineering Definition*. [Online]. Available: <https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition> (Accessed: 08/10/2021).
- [23] ISO, *ISO/IEC/IEEE 24748-1: Systems and software engineering — Life cycle management — Part 1: Guidelines for life cycle management*. ISO, 2018. [Online]. Available: <https://www.iso.org/standard/72896.html>.
- [24] R. G. Cooper, 'The new product process: A decision guide for management,' *Journal of Marketing Management*, vol. 3, no. 3, pp. 238–255, 1988. DOI: [10.1080/0267257X.1988.9964044](https://doi.org/10.1080/0267257X.1988.9964044).
- [25] ———, 'Perspective: The Stage-Gate Idea-to-Launch Process—Update, What's New, and NexGen Systems,' *Journal of Product Innovation Management*, vol. 25, no. 3, pp. 213–232, 2008. DOI: [10.1111/j.1540-5885.2008.00296.x](https://doi.org/10.1111/j.1540-5885.2008.00296.x).

- [26] C. Johansson, 'Knowledge Maturity as Decision Support in Stage-Gate Product Development: A Case From the Aerospace Industry,' PhD Thesis, Luleå University of Technology, 2009. [Online]. Available: <http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Abth-12129>.
- [27] B. Bagdatli, F. Karagoz, K. A. Reilley and D. N. Mavris, 'MBSE-enabled Interactive Environment for Aircraft Conceptual Sizing & Synthesis,' in *AIAA Scitech 2019 Forum*, Reston, Virginia: American Institute of Aeronautics and Astronautics, 2019. DOI: [10.2514/6.2019-0497](https://doi.org/10.2514/6.2019-0497).
- [28] Y. Umeda and T. Tomiyama, 'Functional reasoning in design,' *IEEE Expert*, vol. 12, no. 2, pp. 42–48, 1997. DOI: [10.1109/64.585103](https://doi.org/10.1109/64.585103).
- [29] S. Kleiner and C. Kramer, 'Model Based Design with Systems Engineering Based on RFLP Using V6,' in *Smart Product Engineering*, M. Abramovici and R. Stark, Eds., Springer, Berlin, Heidelberg, 2013, pp. 93–102. DOI: [10.1007/978-3-642-30817-8_10](https://doi.org/10.1007/978-3-642-30817-8_10).
- [30] VDI, 'VDI 2206: Design methodology for mechatronic systems,' Düsseldorf, Tech. Rep., 2004.
- [31] Y. Bile, A. Riaz, M. D. Guenov and A. Molina-Cristobal, 'Towards Automating the Sizing Process in Conceptual (Airframe) Systems Architecting,' in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Reston, Virginia: AIAA, 2018. DOI: [10.2514/6.2018-1067](https://doi.org/10.2514/6.2018-1067).
- [32] M. D. Guenov, A. Riaz, Y. H. Bile, A. Molina-Cristobal and A. S. Heerden, 'Computational framework for interactive architecting of complex systems,' *Systems Engineering*, vol. 23, no. 3, pp. 350–365, 2020. DOI: [10.1002/sys.21531](https://doi.org/10.1002/sys.21531).
- [33] G. Yue, J. Liu and Y. Hou, 'Design Rationale Knowledge Management: A Survey,' in 2018, pp. 245–253. DOI: [10.1007/978-3-030-00560-3_33](https://doi.org/10.1007/978-3-030-00560-3_33).
- [34] J. Lee, 'Design rationale systems: understanding the issues,' *IEEE Expert*, vol. 12, no. 3, pp. 78–85, 1997. DOI: [10.1109/64.592267](https://doi.org/10.1109/64.592267).

- [35] D. Brown, 'Assumptions in Design and Design Rationale,' in *DCC'06 Workshop on Design Rationale: Problems and Progress*, 2006.
- [36] S. Jimeno, A. Riaz, M. Guenov and A. Molina-Cristobal, 'Enabling Interactive Safety and Performance Trade-offs in Early Airframe Systems Design,' in *AIAA Scitech 2020 Forum*, Reston, Virginia: American Institute of Aeronautics and Astronautics, 2020. DOI: [10.2514/6.2020-0550](https://doi.org/10.2514/6.2020-0550).
- [37] S. Jimeno Altelarrea, 'Building safety into the conceptual design of complex systems. An aircraft systems perspective,' PhD Thesis, Cranfield University, 2021.
- [38] K. R. Saoub, *A Tour Through Graph Theory*. Boca Raton, FL: CRC Press, 2018. DOI: [10.1201/9781315116839](https://doi.org/10.1201/9781315116839).
- [39] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd edn. The MIT Press, 2009.
- [40] D. Bell, *The class diagram: An introduction to structure diagrams in UML 2*, 2004. [Online]. Available: <https://developer.ibm.com/articles/the-class-diagram/> (Accessed: 14/10/2021).
- [41] D. Raymer, *Aircraft Design: A Conceptual Approach*, 6th edn. Washington, DC: American Institute of Aeronautics and Astronautics, Inc., 2018. DOI: [10.2514/4.104909](https://doi.org/10.2514/4.104909).
- [42] J. D. Mattingly, W. H. Heiser and D. T. Pratt, *Aircraft Engine Design*, 2nd edn. American Institute of Aeronautics and Astronautics, Inc., 2002. DOI: [10.1037/023990](https://doi.org/10.1037/023990).
- [43] SRA, 'Society for Risk Analysis Glossary,' Society for Risk Analysis, Tech. Rep., 2018.
- [44] U.S. DoD, *Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs*. Washington, D.C.: United States Department of Defense, 2017.

- [45] L. Aristodemou, F. Tietze and M. Shaw, 'Stage Gate Decision Making: A Scoping Review of Technology Strategic Selection Criteria for Early-Stage Projects,' *IEEE Engineering Management Review*, vol. 48, no. 2, pp. 118–135, 2020. DOI: [10.1109/EMR.2020.2985040](https://doi.org/10.1109/EMR.2020.2985040).
- [46] E. Nikolaidis, D. M. Ghiocel and S. Singhal, Eds., *Engineering Design Reliability Handbook*. CRC Press, 2005.
- [47] D. N. Mavris, D. A. DeLaurentis, O. Bandte and M. A. Hale, 'A Stochastic Approach to Multi-disciplinary Aircraft Analysis and Design,' in *36th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 1998. DOI: [10.2514/6.1998-912](https://doi.org/10.2514/6.1998-912).
- [48] T. A. Zang, M. J. Hensch, M. W. Hilburger, S. P. Kenny, J. M. Luckring, P. Maghami, S. L. Padula and W. J. Stroud, *NASA/TM-2002-211462: Needs and opportunities for uncertainty-based multidisciplinary design methods for aerospace vehicles*. Hampton, VA: NASA Langley Research Center, 2002. [Online]. Available: <https://ntrs.nasa.gov/search.jsp?R=20020063596>.
- [49] L. Huyse, 'Solving problems of optimization under uncertainty as statistical decision problems,' in *19th AIAA Applied Aerodynamics Conference*, ser. Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, 2001. DOI: [10.2514/6.2001-1519](https://doi.org/10.2514/6.2001-1519).
- [50] H. McManus and D. Hastings, 'A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems,' in *15th Annual International Symposium of the INCOSE*, vol. 15, Rochester, NY: Wiley Online Library, 2005, pp. 484–503.
- [51] E. de Rocquigny, N. Devictor and S. Tarantola, Eds., *Uncertainty in Industrial Practice: A guide to Quantitative Uncertainty Management*. Chichester, UK: John Wiley & Sons, Ltd, 2008. DOI: [10.1002/9780470770733](https://doi.org/10.1002/9780470770733).

- [52] Centre for Modelling & Simulation, *UQ&M - The New Kid on the Block*, 2015. [Online]. Available: <https://cfms.org.uk/news-events/opinion-articles/2015/september/uqm-the-new-kid-on-the-block/>.
- [53] R. C. Smith, *Uncertainty quantification: theory, implementation, and applications*. Society for Industrial and Applied Mathematics, 2013.
- [54] ———, *Uncertainty Quantification from a Mathematical Perspective*. Isaac Newton Institute for Mathematical Sciences, 2018.
- [55] J. C. Helton, *SAND2009-3055: Conceptual and computational basis for the quantification of margins and uncertainty*. Albuquerque, NM, and Livermore, CA: Sandia National Laboratories, 2009. DOI: [10.2172/958189](https://doi.org/10.2172/958189).
- [56] T. Aven, *Quantitative Risk Assessment: The Scientific Platform*. Cambridge University Press, 2011.
- [57] F. Mangeant, ‘Joined initiatives around uncertainty management: Generic methodologies, mathematical challenges, and numerical implementations,’ *Annales des Telecommunications/Annals of Telecommunications*, vol. 66, pp. 397–407, 2011. DOI: [10.1007/s12243-011-0266-7](https://doi.org/10.1007/s12243-011-0266-7).
- [58] S. Marelli and B. Sudret, ‘UQLab: A Framework for Uncertainty Quantification in Matlab,’ *Vulnerability, Uncertainty, and Risk*, no. October, pp. 2554–2563, 2014. DOI: [10.1061/9780784413609.257](https://doi.org/10.1061/9780784413609.257).
- [59] B. Iooss and P. Lemaître, ‘A Review on Global Sensitivity Analysis Methods,’ in *Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications*, G. Dellino and C. Meloni, Eds. Boston, MA: Springer US, 2015, pp. 101–122. DOI: [10.1007/978-1-4899-7547-8_5](https://doi.org/10.1007/978-1-4899-7547-8_5).
- [60] R. Ghanem, D. Higdon and H. Owhadi, Eds., *Handbook of Uncertainty Quantification*. Springer, 2017. DOI: [10.1007/978-3-319-12385-1](https://doi.org/10.1007/978-3-319-12385-1).

- [61] M. Roelofs and R. Vos, 'Technology Evaluation and Uncertainty-Based Design Optimization: A Review,' in *2018 AIAA Aerospace Sciences Meeting*, Reston, Virginia: American Institute of Aeronautics and Astronautics, 2018. DOI: [10.2514/6.2018-2029](https://doi.org/10.2514/6.2018-2029).
- [62] A. Shapiro, 'Monte Carlo Sampling Methods,' in *Stochastic Programming*, vol. 10, Elsevier, 2003, pp. 353–425. DOI: [10.1016/S0927-0507\(03\)10006-0](https://doi.org/10.1016/S0927-0507(03)10006-0).
- [63] S. Yang, F. Xiong and F. Wang, 'Polynomial Chaos Expansion for Probabilistic Uncertainty Propagation,' in *Uncertainty Quantification and Model Calibration*, InTech, 2017. DOI: [10.5772/intechopen.68484](https://doi.org/10.5772/intechopen.68484).
- [64] W. Qi, S. Tian and Z. Qiu, 'A novel stochastic collocation method for uncertainty propagation in complex mechanical systems,' *Science China Physics, Mechanics and Astronomy*, vol. 58, no. 2, pp. 1–8, 2015. DOI: [10.1007/s11433-014-5525-y](https://doi.org/10.1007/s11433-014-5525-y).
- [65] D. P. Thunnissen, 'Uncertainty Classification for the Design and Development of Complex Systems,' in *Third Annual Predictive Methods Conference*, 2003.
- [66] G. J. Klir and T. A. Folger, *Fuzzy sets, uncertainty, and information*. Prentice Hall, 1988.
- [67] S. Ove Hansson, 'Decision Making Under Great Uncertainty,' *Philosophy of the Social Sciences*, vol. 26, no. 3, pp. 369–386, 1996. DOI: [10.1177/004839319602600304](https://doi.org/10.1177/004839319602600304).
- [68] W. Oberkampf, S. DeLand, B. Rutherford, K. Diegert and K. Alvin, 'A new methodology for the estimation of total uncertainty in computational simulation,' in *40th Structures, Structural Dynamics, and Materials Conference and Exhibit*, ser. Structures, Structural Dynamics, and Materials and Co-located Conferences, Reston, Virginia: American Institute of Aeronautics and Astronautics, 1999. DOI: [10.2514/6.1999-1612](https://doi.org/10.2514/6.1999-1612).
- [69] H. M. Regan, M. Colyvan and M. A. Burgman, 'A taxonomy and treatment of uncertainty for ecology and conservation biology,' *Ecological Applications*, vol. 12,

- no. 2, pp. 618–628, 2002. DOI: [10.1890/1051-0761\(2002\)012\[0618:ATATOU\]2.0.CO;2](https://doi.org/10.1890/1051-0761(2002)012[0618:ATATOU]2.0.CO;2).
- [70] W. D. Rowe, ‘Understanding Uncertainty,’ *Risk Analysis*, vol. 14, no. 5, pp. 743–750, 1994. DOI: [10.1111/j.1539-6924.1994.tb00284.x](https://doi.org/10.1111/j.1539-6924.1994.tb00284.x).
- [71] S. Hawer, A. Schönmann and G. Reinhart, ‘Guideline for the Classification and Modelling of Uncertainty and Fuzziness,’ *Procedia CIRP*, vol. 67, pp. 52–57, 2018. DOI: [10.1016/j.procir.2017.12.175](https://doi.org/10.1016/j.procir.2017.12.175).
- [72] M. Padulo, ‘Computational Engineering Design under Uncertainty. An Aircraft Conceptual Design Perspective,’ PhD Thesis, Cranfield University, 2009. [Online]. Available: <http://hdl.handle.net/1826/4462>.
- [73] M. Padulo and M. D. Guenov, ‘A Methodological Perspective on Computational Engineering Design Under Uncertainty,’ in *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, Vienna, 2012.
- [74] C. L. Berner, ‘Contributions to improved risk assessments: to better reflect the strength of background knowledge,’ PhD Thesis, University of Stavanger, 2017.
- [75] A. Ramsey, *Formal Methods in Artificial Intelligence*. Cambridge University Press, 1988.
- [76] Society of Automotive Engineers, *SAE ARP4754A: Guidelines for Development of Civil Aircraft and Systems*. SAE International, 2010. [Online]. Available: <https://www.sae.org/standards/content/arp4754a/>.
- [77] C. Yang, P. Liang and P. Avgeriou, ‘Evaluation of a process for architectural assumption management in software development,’ *Science of Computer Programming*, vol. 168, pp. 38–70, 2018. DOI: [10.1016/j.scico.2018.08.002](https://doi.org/10.1016/j.scico.2018.08.002).
- [78] D. Jenkins, R. Woolston and M. Boyd, ‘Designing for uncertainty (Assumption-based design),’ *newdesign*, pp. 22–25, 2019.

- [79] S. El Fassi, M. D. Guenov and A. Riaz, ‘An Assumption Network-Based Approach to Support Margin Allocation and Management,’ in *Proceedings of the Design Society: DESIGN Conference*, vol. 1, Cambridge University Press, 2020, pp. 2275–2284. DOI: [10.1017/dsd.2020.25](https://doi.org/10.1017/dsd.2020.25).
- [80] I. Ostacchini, ‘Managing assumptions during agile software development,’ Master Thesis, The Open University, 2009.
- [81] S. Matthiesen and T. Nelius, ‘Managing Assumptions during Analysis - Study on successful Approaches of Design Engineers,’ in *NordDesign*, Linköping, 2018.
- [82] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Heldal and P. Pelliccione, ‘Architectural Assumptions and Their Management in Industry – An Exploratory Study,’ in *Software Architecture*, 5, vol. 3, 2017, pp. 191–207. DOI: [10.1007/978-3-319-65831-5_14](https://doi.org/10.1007/978-3-319-65831-5_14).
- [83] S. van Lieshout, ‘Towards design productivity improvement: An explorative research into the enhancement of design processes’ productivity,’ Master Thesis, Delft University of Technology, 2018. [Online]. Available: <http://resolver.tudelft.nl/uuid:6a6454f0-496f-426f-ba59-dc740cd4eb1d>.
- [84] B. Fieggen, ‘Assumptions improve project and product quality,’ *The Quality Assurance Journal*, vol. 7, no. 2, pp. 92–95, 2003. DOI: [10.1002/qaj.221](https://doi.org/10.1002/qaj.221).
- [85] EIA, *EIA-632: Processes for Engineering a System*. Electronic Industries Alliance, 1999.
- [86] ISO, *ISO/IEC 26702: Standard for Systems Engineering - Application and Management of the Systems Engineering Process*. IEEE, 2007. [Online]. Available: <https://standards.ieee.org/standard/26702-2007.html>.
- [87] J. Chelini, J. Camus, C. Comar, D. Brown, A.-P. Porte, M. de Almeida and H. Delseny, ‘Avionics Certification: Back to Fundamentals with Overarching Properties,’ in *Proceeding of the 9th European Congress on Embedded Real Time Software and Systems*, Toulouse, 2018.

- [88] C. M. Holloway, *NASA/TM–2019–220292: Understanding the Overarching Properties*. Hampton, Virginia: National Aeronautics and Space Administration, 2019.
- [89] K. Anzengruber, P. Hehenberger, S. Boschert, R. Rosen and K. Zeman, ‘Development And Usage Of A Mechatronic Design Process Model With Focus On Assumptions,’ in *10th International Workshop on Integrated Design Engineering*, Gommern, 2014.
- [90] M. M. Lehman and J. Fernández-Ramil, ‘The Role and Impact of Assumptions in Software Engineering and its Products,’ in *Rationale Management in Software Engineering*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 313–328. DOI: [10.1007/978-3-540-30998-7_15](https://doi.org/10.1007/978-3-540-30998-7_15).
- [91] I. Ostacchini and M. Wermelinger, ‘Managing assumptions during agile development,’ in *2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, IEEE, 2009, pp. 9–16. DOI: [10.1109/SHARK.2009.5069110](https://doi.org/10.1109/SHARK.2009.5069110).
- [92] G. A. Lewis, T. Mahatham and L. Wrage, *CMU/SEI-2004-TN- 021: Assumptions Management in Software Development*. Carnegie Mellon University, 2004.
- [93] M. A. Al Mamun, M. Tichy and J. Hansson, ‘2012:02: Towards Formalizing Assumptions on Architectural Level: A Proof-of-Concept,’ Chalmers University of Technology and University of Gothenburg, Tech. Rep., 2012.
- [94] A. S. Tirumala, ‘An assumptions management framework for systems software,’ PhD Thesis, University of Illinois at Urbana-Champaign, 2006.
- [95] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Heldal, P. Pelliccione and T. Bi, ‘An industrial case study on an architectural assumption documentation framework,’ *Journal of Systems and Software*, vol. 134, pp. 190–210, 2017. DOI: [10.1016/j.jss.2017.09.007](https://doi.org/10.1016/j.jss.2017.09.007).
- [96] M. Törngren and U. Sellgren, ‘Complexity Challenges in Development of Cyber-Physical Systems,’ in *Principles of Modeling*, M. Lohstroh, P. Derler and M. Sir-

- jani, Eds., Springer, Cham, 2018, pp. 478–503. DOI: [10.1007/978-3-319-95246-8_27](https://doi.org/10.1007/978-3-319-95246-8_27).
- [97] C. Berner and R. Flage, ‘Strengthening quantitative risk assessments by systematic treatment of uncertain assumptions,’ *Reliability Engineering and System Safety*, vol. 151, pp. 46–59, 2016. DOI: [10.1016/j.ress.2015.10.009](https://doi.org/10.1016/j.ress.2015.10.009).
- [98] H. Eriksson, M. Morin, J. Ekberg, J. Jenvald and T. Timpka, ‘Assumptions management in simulation of infectious disease outbreaks,’ *AMIA Annual Symposium proceedings*, vol. 2009, pp. 173–177, 2009.
- [99] A. Basu and R. W. Blanning, ‘The Analysis of Assumptions in Model Bases Using Metagraphs,’ *Management Science*, vol. 44, no. 7, pp. 982–995, 1998. DOI: [10.1287/mnsc.44.7.982](https://doi.org/10.1287/mnsc.44.7.982).
- [100] C. L. Mason, ‘NASA-TM-108117: A beginner’s guide to belief revision and truth maintenance systems,’ NASA Ames Research Center, Moffett Field, CA, Tech. Rep., 1992. [Online]. Available: <https://ntrs.nasa.gov/api/citations/19930006101/downloads/19930006101.pdf>.
- [101] E. Fermé and S. O. Hansson, *Belief Change: Introduction and Overview*, ser. SpringerBriefs in Intelligent Systems. Cham: Springer International Publishing, 2018. DOI: [10.1007/978-3-319-60535-7](https://doi.org/10.1007/978-3-319-60535-7).
- [102] J. P. Delgrande, P. Peppas and S. Woltran, ‘General Belief Revision,’ *Journal of the ACM*, vol. 65, no. 5, pp. 1–34, 2018. DOI: [10.1145/3203409](https://doi.org/10.1145/3203409).
- [103] J. Martins, ‘Belief Revision,’ in *Encyclopedia of Artificial Intelligence*, S. C. Shapiro, Ed., John Wiley & Sons, 1987, pp. 58–62.
- [104] J. Doyle, ‘A truth maintenance system,’ *Artificial Intelligence*, vol. 12, no. 3, pp. 231–272, 1979. DOI: [10.1016/0004-3702\(79\)90008-0](https://doi.org/10.1016/0004-3702(79)90008-0).
- [105] J. Martins and S. C. Shapiro, ‘Reasoning in Multiple Belief Spaces,’ in *Proceedings of the 8th IJCAI*, Karlsruhe, 1983, pp. 370–373.

- [106] D. A. McAllester, 'AIM-667: Reasoning Utility Package User's Manual, Version One,' MIT AI Laboratory, Cambridge, MA, Tech. Rep., 1982.
- [107] J. de Kleer, 'An assumption-based TMS,' *Artificial Intelligence*, vol. 28, no. 2, pp. 127–162, 1986. DOI: [10.1016/0004-3702\(86\)90080-9](https://doi.org/10.1016/0004-3702(86)90080-9).
- [108] H. Graves and Y. Bijan, 'Using formal methods with SysML in aerospace design and engineering,' *Annals of Mathematics and Artificial Intelligence*, vol. 63, no. 1, pp. 53–102, 2011. DOI: [10.1007/s10472-011-9267-5](https://doi.org/10.1007/s10472-011-9267-5).
- [109] J. E. Burge, 'Software Engineering Using design RATIONALE,' PhD Thesis, Worcester Polytechnic Institute, 2005.
- [110] J. E. Burge, J. M. Carroll, R. McCall and I. Mistrik, *Rationale-Based Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. DOI: [10.1007/978-3-540-77583-6](https://doi.org/10.1007/978-3-540-77583-6).
- [111] P. Gärdenfors, 'Belief revision: An introduction,' in *Belief Revision*, Cambridge University Press, 1992, pp. 1–28. DOI: [10.1017/CBO9780511526664.001](https://doi.org/10.1017/CBO9780511526664.001).
- [112] P. Gärdenfors and D. Makinson, 'Revisions of knowledge systems using epistemic entrenchment,' in *Proceedings of the 2nd conference on Theoretical aspects of reasoning about knowledge*, 1988, pp. 83–95.
- [113] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Elsevier, 1988. DOI: [10.1016/C2009-0-27609-4](https://doi.org/10.1016/C2009-0-27609-4).
- [114] D. Dubois and H. Prade, 'Belief change and possibility theory,' in *Belief Revision*, P. Gärdenfors, Ed., Cambridge University Press, 1992, pp. 142–182. DOI: [10.1017/CBO9780511526664.006](https://doi.org/10.1017/CBO9780511526664.006).
- [115] J. R. Galliers, 'Autonomous belief revision and communication,' in *Belief Revision*, P. Gärdenfors, Ed., Cambridge University Press, 1992, pp. 220–246. DOI: [10.1017/CBO9780511526664.009](https://doi.org/10.1017/CBO9780511526664.009).

- [116] R. Carnota and R. Rodríguez, ‘AGM Theory and Artificial Intelligence,’ in *Belief Revision meets Philosophy of Science*, E. Olsson and S. Enqvist, Eds., Springer, Dordrecht, 2010, pp. 1–42. DOI: [10.1007/978-90-481-9609-8_1](https://doi.org/10.1007/978-90-481-9609-8_1).
- [117] B. Malheiro, ‘Reason Maintenance Systems: Tools for Foundations-Based Belief Revision,’ in *Wiley Encyclopedia of Computer Science and Engineering*, 8, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2008. DOI: [10.1002/9780470050118.ecse440](https://doi.org/10.1002/9780470050118.ecse440).
- [118] C. E. Alchourrón, P. Gärdenfors and D. Makinson, ‘On the logic of theory change: Partial meet contraction and revision functions,’ *Journal of Symbolic Logic*, vol. 50, no. 2, pp. 510–530, 1985. DOI: [10.2307/2274239](https://doi.org/10.2307/2274239).
- [119] A. del Val, ‘Non monotonic reasoning and belief revision: syntactic, semantic, foundational and coherence approaches,’ *Journal of Applied Non-Classical Logics*, vol. 7, no. 1-2, pp. 213–240, 1997. DOI: [10.1080/11663081.1997.10510906](https://doi.org/10.1080/11663081.1997.10510906).
- [120] J. Doyle, ‘Reason maintenance and belief revision: Foundations versus coherence theories,’ in *Belief Revision*, P. Gärdenfors, Ed., Cambridge University Press, 1992, pp. 29–51. DOI: [10.1017/CBO9780511526664.002](https://doi.org/10.1017/CBO9780511526664.002).
- [121] D. Zowghi and R. Offen, ‘A logical framework for modeling and reasoning about the evolution of requirements,’ in *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, IEEE Comput. Soc. Press, 1997, pp. 247–257. DOI: [10.1109/ISRE.1997.566875](https://doi.org/10.1109/ISRE.1997.566875).
- [122] G. Haas, ‘Four Ways in Which Theories of Belief Revision Could Benefit from Theories of Epistemic Justification,’ *Erkenntnis*, vol. 85, no. 2, pp. 295–316, 2020. DOI: [10.1007/s10670-018-0028-2](https://doi.org/10.1007/s10670-018-0028-2).
- [123] R. Bañares-Alcántara and H. Lababidi, ‘Design support systems for process engineering—II. KBDS: An experimental prototype,’ *Computers and Chemical Engineering*, vol. 19, no. 3, pp. 279–301, 1995. DOI: [10.1016/0098-1354\(94\)00051-O](https://doi.org/10.1016/0098-1354(94)00051-O).

- [124] J. Keppens and Q. Shen, 'Finding Faults with Uncertain Assumptions,' in *Proceedings of the 9th International Workshop on Principles of Diagnosis*, London, 1998, pp. 183–197.
- [125] S. Jakob, S. Opfer, A. Jahl, H. Baraki and K. Geihs, 'Handling Semantic Inconsistencies in Commonsense Knowledge for Autonomous Service Robots,' in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, IEEE, 2020, pp. 136–140. DOI: [10.1109/ICSC.2020.00026](https://doi.org/10.1109/ICSC.2020.00026).
- [126] J. Liu and Z. Sun, 'Representing Design Intent for Computer-supported Consistency Maintenance,' in *2008 International Conference on Computer Science and Information Technology*, IEEE, 2008, pp. 561–565. DOI: [10.1109/ICCSIT.2008.147](https://doi.org/10.1109/ICCSIT.2008.147).
- [127] C. Johansson, A. Larsson, T. Larsson and O. Isaksson, 'Gated maturity assessment: supporting gate review decisions with knowledge maturity assessment,' in *CIRP Design Conference*, 2008.
- [128] C. Johansson, B. Hicks, A. C. Larsson and M. Bertoni, 'Knowledge Maturity as a Means to Support Decision Making during Product-Service Systems Development Projects in the Aerospace Sector,' *Project Management Journal*, vol. 42, no. 2, pp. 32–50, 2011. DOI: [10.1002/pmj.20218](https://doi.org/10.1002/pmj.20218).
- [129] European Commission, *Value Improvement through a Virtual Aeronautical Collaborative Enterprise (VIVACE)*, 2004. [Online]. Available: <https://cordis.europa.eu/project/id/502917> (Accessed: 19/04/2021).
- [130] K. Dalkir, *Knowledge Management in Theory and Practice*, 2nd edn. The MIT Press, 2011.
- [131] N. Khatibian, T. Hasan gholoi pour and H. Abedi Jafari, 'Measurement of knowledge management maturity level within organizations,' *Business Strategy Series*, vol. 11, no. 1, pp. 54–70, 2010. DOI: [10.1108/17515631011013113](https://doi.org/10.1108/17515631011013113).

- [132] V. Kochikar, 'The knowledge management maturity model — A stage framework for leveraging knowledge,' in *Proceedings of the KMWorld 2000 Conference*, New Jersey, 2000.
- [133] APQC, *Knowledge Management Capability Assessment Tool*. [Online]. Available: <https://www.apqc.org/what-we-do/benchmarking/assessment-survey/knowledge-management-capability-assessment-tool> (Accessed: 08/02/2021).
- [134] K. Ehms and M. Langen, 'Holistic Development of Knowledge Management with KMMM,' Siemens AG/Corporate Technology, Munich, Tech. Rep., 2002.
- [135] C. Eckert, O. Isaksson and C. Earl, 'Design margins: a hidden issue in industry,' *Design Science*, vol. 5, pp. 1–24, 2019. DOI: [10.1017/dsj.2019.7](https://doi.org/10.1017/dsj.2019.7).
- [136] ———, 'Product property margins: An underlying critical problem of engineering design,' in *Proceedings of TMCE 2012*, 2012.
- [137] S. Coggon, 'Setting of Wing Target Loads,' in *Second Uncertainty Quantification and Management for High Value Manufacturing study group with Industry*, Liverpool: Innovate UK, 2017.
- [138] J. D. Anderson, *Aircraft Performance and Design*. McGraw-Hill, 1999.
- [139] D. P. Thunnissen, 'Method for Determining Margins in Conceptual Design,' *Journal of Spacecraft and Rockets*, vol. 41, no. 1, pp. 85–92, 2004. DOI: [10.2514/1.9211](https://doi.org/10.2514/1.9211).
- [140] J. Yuan, M. Savill, T. Kipouros, S. Coggon and F. Scarpa, 'Probabilistic margin assessment of aircraft conceptual design using a modified reliability based design optimization methodology,' *Proceedings of ISMA 2016*, pp. 4463–4477, 2016.
- [141] C. Eckert, C. Earl, S. Lebjoui and O. Isaksson, 'Components Margins through the Product Lifecycle,' in *IFIP Advances in Information and Communication Technology*, vol. 409, 2013, pp. 39–47. DOI: [10.1007/978-3-642-41501-2_5](https://doi.org/10.1007/978-3-642-41501-2_5).

- [142] A. Touboul, J. Reygner, F. Mangeant and P. Benjamin, *A formal framework to define margins in industrial processes*, 2019. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02156493>.
- [143] P. A. Gale, 'Margins in Naval Surface Ship Design,' *Naval Engineers Journal*, vol. 87, no. 2, pp. 174–188, 1975. DOI: [10.1111/j.1559-3584.1975.tb03728.x](https://doi.org/10.1111/j.1559-3584.1975.tb03728.x).
- [144] C. Eckert, O. Isaksson, S. Lebjoui, C. F. Earl and S. Edlund, 'Design margins in industrial practice,' *Design Science*, vol. 6, e30, 2020. DOI: [10.1017/dsj.2020.19](https://doi.org/10.1017/dsj.2020.19).
- [145] P. J. Clarkson, C. Simons and C. Eckert, 'Predicting Change Propagation in Complex Design,' *Journal of Mechanical Design*, vol. 126, no. 5, pp. 788–797, 2004. DOI: [10.1115/1.1765117](https://doi.org/10.1115/1.1765117).
- [146] O. de Weck, 'Strategic Engineering: Designing Systems for an Uncertain Future,' in *Pratt & Whitney Fellows Lecture*, East Hartford, Connecticut, 2009.
- [147] D. Long and S. Ferguson, 'A Case Study of Evolvability and Excess on the B-52 Stratofortress and F/A-18 Hornet,' in *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 4, Cleveland, Ohio: American Society of Mechanical Engineers, 2017. DOI: [10.1115/DETC2017-68287](https://doi.org/10.1115/DETC2017-68287).
- [148] C. Eckert, P. J. Clarkson and W. Zanker, 'Change and customisation in complex engineering domains,' *Research in Engineering Design*, vol. 15, no. 1, pp. 1–21, 2004. DOI: [10.1007/s00163-003-0031-7](https://doi.org/10.1007/s00163-003-0031-7).
- [149] H. Cheng and X. Chu, 'A network-based assessment approach for change impacts on complex product,' *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 1419–1431, 2012. DOI: [10.1007/s10845-010-0454-8](https://doi.org/10.1007/s10845-010-0454-8).
- [150] B. Hamraz, N. H. Caldwell and P. J. Clarkson, 'A Matrix-Calculation-Based Algorithm for Numerical Change Propagation Analysis,' *IEEE Transactions on Engineering Management*, vol. 60, no. 1, pp. 186–198, 2013. DOI: [10.1109/TEM.2012.2203307](https://doi.org/10.1109/TEM.2012.2203307).

- [151] E. C. Koh, N. H. Caldwell and P. J. Clarkson, ‘A technique to assess the changeability of complex engineering systems,’ *Journal of Engineering Design*, vol. 24, no. 7, pp. 477–498, 2013. DOI: [10.1080/09544828.2013.769207](https://doi.org/10.1080/09544828.2013.769207).
- [152] D. Long and S. Ferguson, ‘Studying Dynamic Change Probabilities and Their Role in Change Propagation,’ *Journal of Mechanical Design*, vol. 142, no. 10, 2020. DOI: [10.1115/1.4046674](https://doi.org/10.1115/1.4046674).
- [153] A. Brahma and D. C. Wynn, ‘A Study on the Mechanisms of Change Propagation in Mechanical Design,’ *Journal of Mechanical Design*, vol. 143, no. 12, 2021. DOI: [10.1115/1.4050927](https://doi.org/10.1115/1.4050927).
- [154] T. A. Zang, S. Mahadevan, J. C. Tai and D. N. Mavris, ‘A Strategy for Probabilistic Margin Allocation in Aircraft Conceptual Design,’ in *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Reston, Virginia: AIAA, 2015. DOI: [10.2514/6.2015-3443](https://doi.org/10.2514/6.2015-3443).
- [155] R. M. Cooke, T. A. Zang, D. N. Mavris and J. C. Tai, ‘Sculpting: A Fast, Interactive Method for Probabilistic Design Space Exploration and Margin Allocation,’ in *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Reston, Virginia: AIAA, 2015. DOI: [10.2514/6.2015-3440](https://doi.org/10.2514/6.2015-3440).
- [156] K. Hall, P. Schroll and S. Sharma, ‘Managing Margins Under Uncertainties Surrogate Modelling and Uncertainty Quantification,’ in *Complex Systems Design and Management*, Cham: Springer International Publishing, 2020, pp. 49–63. DOI: [10.1007/978-3-030-34843-4_5](https://doi.org/10.1007/978-3-030-34843-4_5).
- [157] ISO, *ISO/DIS 10303-243: Industrial automation systems and integration — Product data representation and exchange — Part 243: Application protocol: For modeling and simulation information in a collaborative systems engineering context (MoSSEC)*. ISO, 2018. [Online]. Available: <https://www.iso.org/standard/72491.html>.

- [158] M. Ghosn and F. Moses, 'Reliability Calibration of Bridge Design Code,' *Journal of Structural Engineering*, vol. 112, no. 4, pp. 745–763, 1986. DOI: [10.1061/\(ASCE\)0733-9445\(1986\)112:4\(745\)](https://doi.org/10.1061/(ASCE)0733-9445(1986)112:4(745)).
- [159] E. Alfred Mohammed, S. Benson, S. Hirdaris and R. Dow, 'Design safety margin of a 10,000 TEU container ship through ultimate hull girder load combination analysis,' *Marine Structures*, vol. 46, pp. 78–101, 2016. DOI: [10.1016/j.marstruc.2015.12.003](https://doi.org/10.1016/j.marstruc.2015.12.003).
- [160] A. Brahma and D. C. Wynn, 'Margin value method for engineering design improvement,' *Research in Engineering Design*, vol. 31, no. 3, pp. 353–381, 2020. DOI: [10.1007/s00163-020-00335-8](https://doi.org/10.1007/s00163-020-00335-8).
- [161] M. D. Guenov, X. Chen, A. Molina-Cristóbal, A. Riaz, A. S. J. van Heerden and M. Padulo, 'Margin Allocation and Tradeoff in Complex Systems Design and Optimization,' *AIAA Journal*, vol. 56, no. 7, pp. 2887–2902, 2018. DOI: [10.2514/1.J056357](https://doi.org/10.2514/1.J056357).
- [162] ISO, *IEEE/ISO/IEC 12207-2017: Systems and software engineering - Software life cycle processes*. IEEE, 2017. [Online]. Available: <https://standards.ieee.org/standard/12207-2017.html>.
- [163] F. Mas, J. Menéndez, M. Oliva and J. Ríos, 'Collaborative Engineering: An Airbus Case Study,' *Procedia Engineering*, vol. 63, pp. 336–345, 2013. DOI: [10.1016/j.proeng.2013.08.180](https://doi.org/10.1016/j.proeng.2013.08.180).
- [164] R. Flage and T. Aven, 'Expressing and communicating uncertainty in relation to quantitative risk analysis,' *Reliability: Theory and Applications*, vol. 4, no. 2 (13), pp. 9–18, 2009.
- [165] L. Wang, W. Shen, H. Xie, J. Neelamkavil and A. Pardasani, 'Collaborative conceptual design—state of the art and future trends,' *Computer-Aided Design*, vol. 34, no. 13, pp. 981–996, 2002. DOI: [10.1016/S0010-4485\(01\)00157-9](https://doi.org/10.1016/S0010-4485(01)00157-9).

- [166] OECD, *Handbook on Constructing Composite Indicators: Methodology and User Guide*. OECD Publishing, 2008. [Online]. Available: <https://www.oecd.org/sdd/42495745.pdf>.
- [167] ———, *OECD Glossary of Statistical Terms - Composite Indicator Definition*, 2004. [Online]. Available: <https://stats.oecd.org/glossary/detail.asp?ID=6278> (Accessed: 18/11/2020).
- [168] T. F. Coleman and J. J. Moré, ‘Estimation of Sparse Jacobian Matrices and Graph Coloring Problems,’ *SIAM Journal on Numerical Analysis*, vol. 20, no. 1, pp. 187–209, 1983. DOI: [10.1137/0720013](https://doi.org/10.1137/0720013).
- [169] X. Gan, I. C. Fernandez, J. Guo, M. Wilson, Y. Zhao, B. Zhou and J. Wu, ‘When to use what: Methods for weighting and aggregating sustainability indicators,’ *Ecological Indicators*, vol. 81, pp. 491–502, 2017. DOI: [10.1016/j.ecolind.2017.05.068](https://doi.org/10.1016/j.ecolind.2017.05.068).
- [170] E. H. Forman, ‘The analytic hierarchy process as a decision support system,’ *Proceedings of the IEEE Computer society*, 1983.
- [171] S. Greco, A. Ishizaka, M. Tasiou and G. Torrisi, ‘On the Methodological Framework of Composite Indices: A Review of the Issues of Weighting, Aggregation, and Robustness,’ *Social Indicators Research*, vol. 141, no. 1, pp. 61–94, 2019. DOI: [10.1007/s11205-017-1832-9](https://doi.org/10.1007/s11205-017-1832-9).
- [172] T. L. Saaty and L. G. Vargas, *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*, ser. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2012, vol. 175. DOI: [10.1007/978-1-4614-3597-6](https://doi.org/10.1007/978-1-4614-3597-6).
- [173] A. Ishizaka and M. Lusti, ‘How to derive priorities in AHP: a comparative study,’ *Central European Journal of Operations Research*, vol. 14, no. 4, pp. 387–400, 2006. DOI: [10.1007/s10100-006-0012-9](https://doi.org/10.1007/s10100-006-0012-9).

- [174] G. Beliakov, A. Pradera and T. Calvo, *Aggregation Functions: A Guide for Practitioners*, ser. Studies in Fuzziness and Soft Computing. Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2007, vol. 221. DOI: [10.1007/978-3-540-73721-6](https://doi.org/10.1007/978-3-540-73721-6).
- [175] P. Zhou and B. W. Ang, ‘Comparing MCDA Aggregation Methods in Constructing Composite Indicators Using the Shannon-Spearman Measure,’ *Social Indicators Research*, vol. 94, no. 1, pp. 83–96, 2009. DOI: [10.1007/s11205-008-9338-0](https://doi.org/10.1007/s11205-008-9338-0).
- [176] D. R. Katz, S. Sarkani, T. Mazzuchi and E. H. Conrow, ‘The Relationship of Technology and Design Maturity to DoD Weapon System Cost Change and Schedule Change During Engineering and Manufacturing Development,’ *Systems Engineering*, vol. 18, no. 1, pp. 1–15, 2015. DOI: [10.1111/sys.21281](https://doi.org/10.1111/sys.21281).
- [177] GAO, ‘GAO-12-400SP: Defense Acquisitions - Assessments of Selected Major Weapon Programs,’ U.S. Government Accountability Office, Washington, DC, Tech. Rep., 2012.
- [178] N. Harrison, ‘Warship Design Maturity Measurement and Analysis Throughout the Lifecycle,’ in *International Conference on Systems Engineering in Ship and Offshore Design*, Bath, 2010.
- [179] B. Sauser, D. Verma, J. Ramirez-Marquez and R. Gove, ‘From TRL to SRL: The Concept of Systems Readiness Levels,’ in *Conference on Systems Engineering Research*, Los Angeles, 2006, pp. 1–10.
- [180] T. Eger, C. Eckert and P. J. Clarkson, ‘The Role of Design Freeze in Product Development,’ in *DS 35: Proceedings ICED 05, the 15th International Conference on Engineering Design*, A. Samuel and W. Lewis, Eds., Melbourne: Design Society, 2005.
- [181] E. W. Dijkstra, ‘A note on two problems in connexion with graphs,’ *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. DOI: [10.1007/BF01386390](https://doi.org/10.1007/BF01386390).

- [182] M. D. Guenov, M. Nunez, A. Molina-Cristóbal, V. Datta and A. Riaz, ‘AirCADia – an Interactive Tool for the Composition and Exploration of Aircraft Computational Studies At Early Design Stage,’ *29th Congress of the International Council of the Aeronautical Sciences*, pp. 1–12, 2014.
- [183] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.
- [184] X. Chen, A. Riaz, M. Guenov and A. Molina-Cristobal, ‘A Set-based Approach for Coordination of Multi-level Collaborative Design Studies,’ in *AIAA Scitech 2020 Forum*, Reston, Virginia: American Institute of Aeronautics and Astronautics, 2020. DOI: [10.2514/6.2020-0320](https://doi.org/10.2514/6.2020-0320).
- [185] M. D. McKay, R. J. Beckman and W. J. Conover, ‘A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code,’ *Technometrics*, vol. 21, no. 2, p. 239, 1979. DOI: [10.2307/1268522](https://doi.org/10.2307/1268522).
- [186] G. Pigozzi, ‘Belief Merging and Judgment Aggregation,’ in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Spr 2021, Metaphysics Research Lab, Stanford University, 2021. [Online]. Available: <https://plato.stanford.edu/archives/spr2021/entries/belief-merging/>.
- [187] Mrl5, *Python-design-patterns/observer*, 2018. [Online]. Available: <https://github.com/mrl5/python-design-patterns/blob/master/observer.py> (Accessed: 16/11/2021).
- [188] ESDU, *ESDU 99031: Computer program for estimation of lift curve to maximum lift for wing-fuselage combinations with high-lift devices at low speeds: Amendment (B)*. London: ESDU, 2009. [Online]. Available: https://www.esdu.com/cgi-bin/ps.pl?t=doc&p=esdu_99031b.
- [189] G. X. Dussart, M. M. Lone and C. O’Rourke, ‘Size Estimation Tools for Conventional Actuator System Prototyping in Aerospace,’ in *AIAA Scitech 2019 Forum*,

- Reston, Virginia: American Institute of Aeronautics and Astronautics, 2019. DOI: [10.2514/6.2019-1634](https://doi.org/10.2514/6.2019-1634).
- [190] M. Drela, *Lab 10 Lecture Notes: Area and Bending Inertia of Airfoil Sections*, 16.01-04. Massachusetts Institute of Technology, 2006. [Online]. Available: <https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-01-unified-engineering-i-ii-iii-iv-fall-2005-spring-2006/systems-labs-06/spl10b.pdf>.
- [191] ———, *Lab 10 Lecture Notes: Wing Bending Calculations*, 16.01-04. Massachusetts Institute of Technology, 2006. [Online]. Available: <https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-01-unified-engineering-i-ii-iii-iv-fall-2005-spring-2006/systems-labs-06/spl10.pdf>.
- [192] J. Carreyette, ‘Aircraft Wing Weight Estimation,’ *Aircraft Engineering and Aerospace Technology*, vol. 22, no. 1, pp. 8–11, 1950. DOI: [10.1108/eb031848](https://doi.org/10.1108/eb031848).

Appendix A

Evaluation: Software Implementation

The following are excerpts from the source code for some main features presented in Section 6.2.

Parsing the architecture's XML file

```
import xml.etree.ElementTree as ET

tree = ET.parse(aapath) #'aapath' refers to the path of the
                        architecture's XML file

root = tree.getroot()

#Location of relevant information in architecture's XML file
requirements = root[4][0][1]
functions = root[4][0][2]
solutions = root[4][0][3]
assumptions = root[4][0][4]
margins = root[4][0][5]

'''NOTE: There is an additional step to extract information on
parameters and computational models from 'solutions', which is
not shown for simplification'''
```

```
return requirements, functions, solutions, parameters, models, \  
        assumptions, margins
```

Design Belief Network

```
import networkx as nx  
  
#Create empty Design Belief Network  
DBN = nx.Graph()  
  
'''Add nodes from parsed information: requirements, functions,  
solutions, parameters, models, assumptions, margins:'''  
  
DBN.add_nodes_from(requirements)  
DBN.add_nodes_from(functions)  
DBN.add_nodes_from(solutions)  
DBN.add_nodes_from(parameters)  
DBN.add_nodes_from(models)  
DBN.add_nodes_from(assumptions)  
DBN.add_nodes_from(margins)  
  
#Add edges from an 'edgeList' extracted from parsing  
DBN.add_edges_from(edgeList)  
  
#Export the DBN as a Python Pickle file  
nx.write_gpickle(DBN, "DBN_File.gpickle")
```

Knowledge Maturity Index

```
'''Read DBN Pickle file to extract data that will serve to  
calculate individual indicators'''
```

```
DBN = nx.read_gpickle("DBN_File.gpickle")

# Calculate Validation Indicator
nb_asmps = 0 #Initialise total number of assumptions
nb_validated = 0 #Initialise number of validated assumptions
for node in DBN.nodes():
    if DBN.nodes[node]['type'] == 'Assumption':
        nb_asmps += 1
        if DBN.nodes[node]['status'] == 'Valid':
            nb_validated += 1
I1 = nb_validated/nb_asmps

# Calculate Confidence Indicator
nb_asmps = 0 #Initialise total number of assumptions
LoC_Total = 0 #Initialise total Level of Confidence
for node in DBN.nodes():
    if DBN.nodes[node]['type'] == 'Assumption':
        if DBN.nodes[node]['status'] == 'Awaiting Evaluation':
            nb_asmps += 1
            if DBN.nodes[node]['confidence'] == 'Low':
                LoC_Total += 1/10
            elif DBN.nodes[node]['confidence'] == 'Moderate':
                LoC_Total += 5/10
            elif DBN.nodes[node]['confidence'] == 'High':
                LoC_Total += 9/10
I2 = LoC_Total/nb_asmps

# Calculate Consistency Indicator
nb_asmps_aw = 0 #Initialise number of assumptions awaiting
                    evaluation
nb_conflicts = 0 #Initialise number of conflicts between
                    assumptions
for node in DBN.nodes():
    if DBN.nodes[node]['type'] == 'Assumption':
```

```

        if DBN.nodes[node]['status'] == 'Awaiting Evaluation':
            nb_asmps_aw += 1
for edge in DBN.edges():
    if DBN.edges[edge]['type'] == 'Conflict':
        nb_conflicts += 1
if nb_asmps_aw > 1:
    I3 = 1-(2*nb_conflicts)/(nb_asmps_aw*(nb_asmps_aw-1))
else:
    I3 = 1

#Calculate KMI
KMI = I1**(2/3) * I2**(1/6) * I3**(1/6)

```

Margin Allocation

```

#Separate dependent and independent parameters:
Ind_Par = [] #Independent parameters, i.e. root nodes
D_Par = [] #Dependent parameters, i.e. intermediate/leaf nodes
for n in DG.nodes:
    if len(list(DG.predecessors(n))) == 0:
        Ind_Par.append(n)
    else:
        D_Par.append(n)

#Store unmitigated assumptions:
unmit_asmps = [] #Initialise list of unmitigated assumptions
for i in assumptions:
    for j in margins:
        if i in j["mitigatedAssumptions"]:
            i = None
            break
        else: continue
    if i != None: unmit_asmps.append(i)

```

```

#Margin redundancy:
Redund_List = [] #Initialise list of redundant margin tuples
for mar in margins:
    if (mar["parameter"] in Ind_Par) \
    & (mar["value"] != 0):
        #If the margin is assigned to an independent parameter
        for mod in models:
            for mar2 in margins:
                if (mar["parameter"] in mod["inputs"]) & \
                (mod["output"] == mar2["parameter"]):
                    #cf. Algorithm 5, line 13 and description in
                    Section 5.4
                    Redund_List.append((mar,mar2))
#Store redundant margins

```

Change Absorption: Graph Traversal

```

for margin in DBN.nodes():
    if DBN.nodes[margin]['type'] == 'Margin':
        '''Find the shortest path between the input assumption
        and every margin (if it exists)'''
        dict_paths[len(nx.shortest_path(DBN,source=assumption,\
        target=DBN.nodes[margin]['parameter']))]=nx.\
        shortest_path(DBN, source=assumption, target=\
        DBN.nodes[margin]['parameter'])

if len(dict_paths.keys()) != 0: #If at least one 'shortest' path
    has been identified
    shortest = dict_paths[min(dict_paths.keys())] #Find the overall
    shortest path

```


Margin Revision: Observer Pattern

The Python implementation of the *Publisher* class was adapted from [187].

```
class Publisher(object): #Represents the subject being observed
    def __init__(self):
        self._observers = [] #This is where references to
                               #all the observers are stored

    def attach(self, observer):
        if observer not in self._observers: #If the observer is not
                                               already in the observers
                                               list
            self._observers.append(observer) #Add the observer to
                                               the list

    def detach(self, observer): #To remove the observer
        try:
            self._observers.remove(observer)
        except ValueError:
            pass

    def notify(self, alertlistx, modifier=None):
        for observer in self._observers: # For all the stored
                                           observers
            if modifier != observer:
                #The observer that is updating the value is not
                notified
                observer.update(self, alertlistx) #Notify the
                                                    observers

class AsmpsPublisher(Publisher): #Assumptions as publishers
    def __init__(self, name, assumptions):
        Publisher.__init__(self)
        self._name = name #Set the name of the concrete publisher
```

```
self._asmps = assumptions #Initialise the value to the
                        parsed assumptions
```

Conflict Detection: Graph Traversal

```
#The graph corresponding to the computational workflow (WF)
#is reversed via NetworkX to enable reversed graph traversal:
WF_Rev = nx.DiGraph.reverse(WF)

Conflicting_Asmpts = [] #Stores conflicting assumptions

#Perform a depth-first search:
T = nx.dfs_tree(WF_Rev, source="INSERT CHANGE INITIATOR")
for e in list(T.edges()):
    if WF_Rev.nodes[e[1]]['type'] == 'Assumption':
        Conflicting_Asmpts.append(e[1])
```

Appendix B

Evaluation: Computational Workflow in Use Case 2

Computational Workflow

The computational workflow is illustrated in Figure B.1, and the parameters and models are described in Tables B.1 and B.2. Subsequently, each computational model is presented.

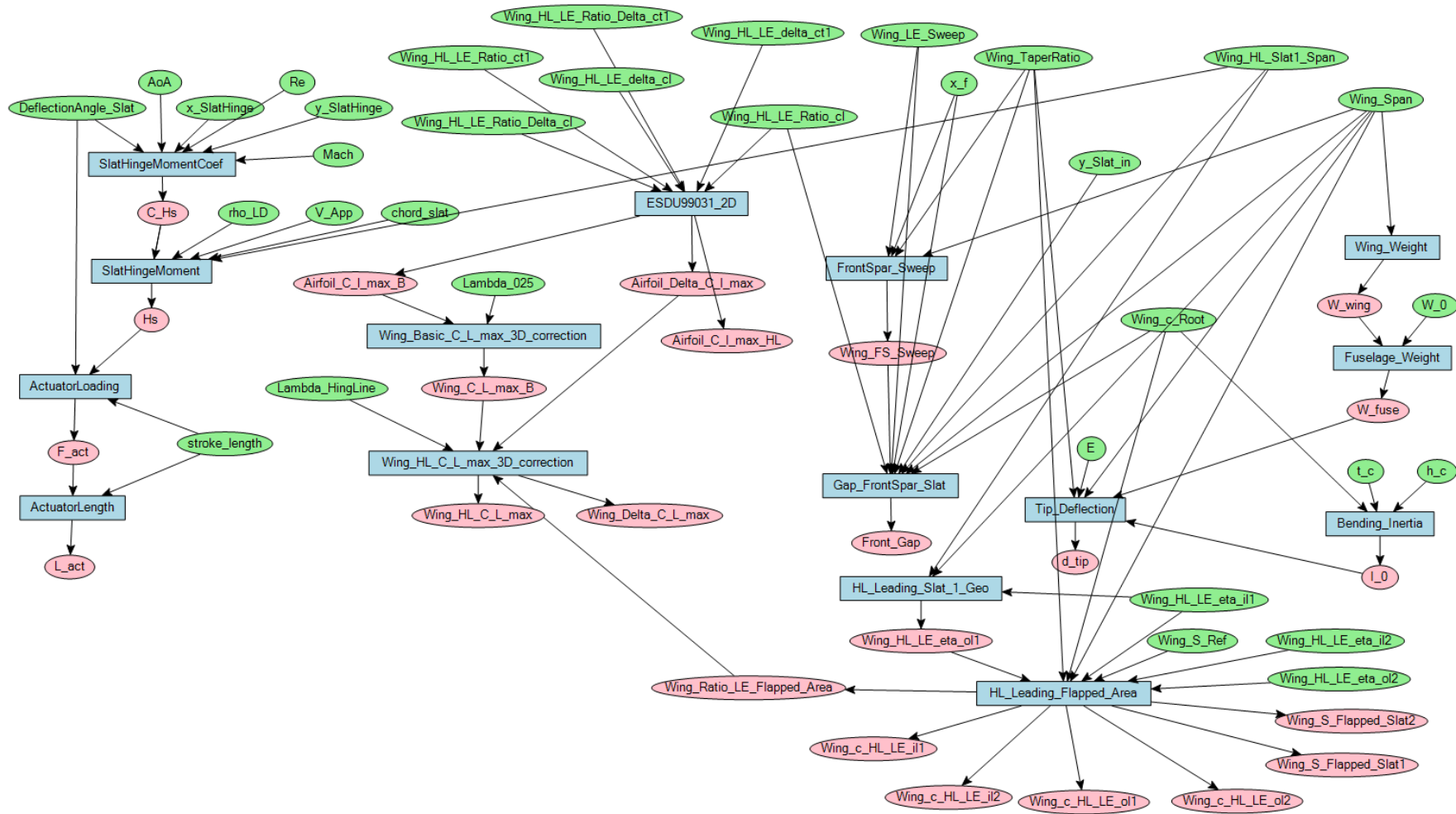


Figure B.1: Computational workflow corresponding to Use Case 2 (AirCADia Explorer)

Table B.1: Description of the parameters in Use Case 2

Parameter	Description
Airfoil_C.l.max_B	Maximum lift coefficient of basic airfoil (high-lift devices undeployed)
Airfoil_C.l.max_HL	Maximum lift coefficient of airfoil with high-lift devices deployed
Airfoil_Delta_C.l.max	Increment in airfoil maximum lift coefficient due to high-lift devices
AoA	Angle of attack (degree)
C_Hs	Slat hinge moment coefficient
chord_slat	Slat chord (ft)
d_tip	Wing tip deflection (m)
DeflectionAngle_Slat	Deflection angle of the slat (degree)
E	Young's Modulus (Pa)
F_act	Force on linear hydraulic actuator (N)
Front_Gap	Gap between front spar and slat (m)
h.c	Camber-to-chord ratio
Hs	Slat hinge moment (N.m)
I.0	Bending inertia (m ⁴)
L_act	Length of linear hydraulic actuator (m)

Parameter	Description
Lambda_025	Sweep angle of the wing quarter chord (degree)
Lambda_Hingline	Sweep angle of the slat hinge line (degree)
Mach	Mach number
Re	Reynold's number
rho_LD	Air density (kg/m ³)
stroke_length	Actuator's stroke length (m)
t_c	Thickness-to-chord ratio
V_App	Approach velocity (kt)
W_0	Take-off gross weight (kg)
W_fuse	Fuselage weight (kg)
W_wing	Wing weight (kg)
Wing_c_HL_LE_il1	Wing chord at spanwise location: inboard of slat 1
Wing_c_HL_LE_il2	Wing chord at spanwise location: inboard of slat 2
Wing_c_HL_LE_ol1	Wing chord at spanwise location: outboard of slat 1
Wing_c_HL_LE_ol2	Wing chord at spanwise location: outboard of slat 2
Wing_C_L_max_B	Maximum lift coefficient of basic wing (high-lift devices undeployed)
Wing_c_Root	Wing root chord (ft)

Parameter	Description
Wing_Delta_C.L.max	Increment in wing maximum lift coefficient due to high-lift devices
Wing_FS.Sweep	Sweep angle of the front spar (degree)
Wing_HL_C.L.max	Maximum lift coefficient of wing with high-lift devices deployed
Wing_HL_LE_delta_cl	Chord extension due to deployment of slat (ft)
Wing_HL_LE_delta_ct1	Chord extension due to deployment of trailing-edge flap (ft)
Wing_HL_LE_eta_il1	Spanwise distance from wing root (centre-line) chord as fraction of semi-span for inboard limit of Slat 1
Wing_HL_LE_eta_il2	Spanwise distance from wing root (centre-line) chord as fraction of semi-span for inboard limit of Slat 2
Wing_HL_LE_eta_ol1	Spanwise distance from wing root (centre-line) chord as fraction of semi-span for outboard limit of Slat 1
Wing_HL_LE_eta_ol2	Spanwise distance from wing root (centre-line) chord as fraction of semi-span for outboard limit of Slat 2
Wing_HL_LE_Ratio_cl	Slat chord to wing chord ratio
Wing_HL_LE_Ratio_ct1	Trailing-edge flap chord to wing chord ratio
Wing_HL_LE_Ratio_Delta_cl	Ratio of chord extension due to deployment of slat to wing chord
Wing_HL_LE_Ratio_Delta_ct1	Ratio of chord extension due to deployment of trailing-edge flap to wing chord
Wing_HL_Slat1_Span	Slat span (ft)
Wing_LE.Sweep	Sweep angle of the wing leading edge (degree)
Wing_Ratio.LE.Flapped_Area	Ratio of slat flapped area to wing area
Wing_S.Flapped_Slat1	Flapped area of Slat 1 (ft ²)

Parameter	Description
Wing_S_Flapped_Slat2	Flapped area of Slat 2 (ft ²)
Wing_S_Ref	Wing reference area (ft ²)
Wing_Span	Wing span (ft)
Wing_TaperRatio	Wing taper ratio
x_f	Ratio of front spar location to wing chord
x_SlatHinge	Ratio of slat hinge x-location to wing chord
y_Slat_in	Inboard spanwise location of the slat (ft)
y_SlatHinge	Ratio of slat hinge y-location as to airfoil thickness

Table B.2: Description of the computational models in Use Case 2

Model	Description
ActuatorLength	Model to estimate the length of a linear hydraulic actuator
ActuatorLoading	Model to estimate the force on a linear hydraulic actuator
Bending_Inertia	Model to estimate the bending inertia of the wing
ESDU99031_2D	Computer program for estimation of lift curve to maximum lift for wing-fuselage combinations with high-lift devices at low speeds [188]
FrontSpar_Sweep	Model to estimate the sweep angle of the front spar
Fuselage_Weight	Model to estimate the weight of the fuselage based on the aircraft empty weight and the wing weight
Gap_FrontSpar_Slat	Model to estimate the gap between the front spar and the slat
HL_Leading_Flapped_Area	Model to estimate the flapped area of the slat
HL_Leading_Slat_1_Geo	Model to estimate the spanwise distance from wing root (centre-line) chord as fraction of semi-span for outboard limit of the slat
SlatHingeMoment	Model to estimate the slat hinge moment
SlatHingeMomentCoef	Model to estimate the slat hinge moment coefficient using XFOIL ¹
Tip_Deflection	Model to estimate the deflection of the wing tip

¹<https://web.mit.edu/drela/Public/web/xfoil/> (Accessed 27/10/2021)

Model	Description
Wing_Basic_C.L_max_3D_correction	Model to estimate the maximum lift coefficient of basic wing (i.e. high-lift devices undeployed)
Wing_HL_C.L_max_3D_correction	Model to estimate the maximum lift coefficient of wing with high-lift devices deployed
Wing_Weight	Model to estimate the weight of the wing based on the wingspan

Description of the Models

ActuatorLength Model

Equation B.1 to estimate the actuator length was proposed by Dussart *et al.* [189], where the authors fitted an empirical database of balanced actuator length on a polynomial surface.

$$L_{act} = \frac{60.66 + 0.8069 \times \frac{F_{act}}{1000} + 2.622 \times (1000 \times stroke_length)}{1000} \quad (B.1)$$

ActuatorLoading Model

Equation B.2 estimates the force acting on the linear hydraulic actuator based on conservation of work during the deployment of the slat. The equation has been simplified for demonstration purposes.

$$F_{act} = \frac{Hs \times DeflectionAngle_Slat \times \frac{\pi}{180}}{stroke_length} \quad (B.2)$$

Bending Inertia Model

Equation B.3 from [190] is an approximation of the bending inertia of airfoil sections. The multiplying factor '0.3048' in Equation B.3 is for converting *Wing_c_Root* from feet to meters.

$$I_0 = 0.036 \times (0.3048 \times Wing_c_Root)^4 \times t_c \times (t_c^2 + h_c^2) \quad (B.3)$$

ESDU99031_2D Model

To analyse the aerodynamic performance in the presence of high-lift devices, an executable file of the “*ESDU 99031: Computer program for estimation of lift curve to maximum lift for wing-fuselage combinations with high-lift devices at low speeds*” [188] was used. The input file to execute the model is the following, where the description of the values

can be found in [188]:

Calculation type CALC=1

1

1

0.2 6E6

1

C

0.10

0

0.421 0.0608

0.0

0.0 0.0 0.0

0.01 0.012 -0.010

0.0125 0.013 -0.0105

0.025 0.018 -0.014

0.05 0.025 -0.019

0.10 0.036 -0.025

0.20 0.0495 -0.0334

0.30 0.0570 -0.0379

0.40 0.0607 -0.0392

0.50 0.0592 -0.0371

0.60 0.0522 -0.0308

0.70 0.0413 -0.0218

0.80 0.0278 -0.0119

0.90 0.0133 -0.0029

0.95 0.0062 0.0001

0.99 0.00124 0.00002

1.0 0.0 0.0

0.402

2 2

5

3

45.0 0.00687 0.2000 0.0800

0.01004 0.0180 0.00751 0.1440 0.0420

30.0 0.00687 0.1500 0.0594

0.01004 0.0360 0.0390 0.1440 0.0420

35.0 0.080

30.0 0.010

0.350

0.900

FrontSpar_Sweep Model

Equation B.5 was used to estimate the sweep angle of the front spar. α in Equation B.4 is an intermediate parameter to calculate *Wing_FS_Sweep*.

$$\alpha = \frac{Wing_Span}{2} \times \tan(Wing_LE_Sweep \times \frac{\pi}{180}) + x_f \times Wing_TaperRatio - x_f \quad (B.4)$$

$$Wing_FS_Sweep = \arctan\left(\frac{\alpha}{\frac{Wing_Span}{2}}\right) \times \frac{180}{\pi} \quad (B.5)$$

Fuselage_Weight Model

In Equation B.6, the fuselage weight is calculated by subtracting the wing weight from the aircraft empty weight, where the latter was estimated from [41].

$$W_fuse = (0.97 \times W_0^{0.94}) - W_wing \quad (B.6)$$

Gap_FrontSpar_Slat Model

Equations B.7-B.9 were used to estimate the gap between the front spar and slat using a geometrical relationship. α and β are intermediate parameters to calculate *Front_Gap*. The multiplying factor '0.3048' is for converting from feet to meters. Figure B.2 illustrates the parameters relating to the geometrical relationship.

$$\alpha = \cos\left(Wing_LE_Sweep \times \frac{\pi}{180}\right) + \sin\left(Wing_LE_Sweep \times \frac{\pi}{180}\right) \times \tan\left((Wing_LE_Sweep - Wing_FS_Sweep) \times \frac{\pi}{180}\right) \quad (B.7)$$

$$\beta = 1 - \left((1 - Wing_TaperRatio) \times \frac{Wing_HL_Slat1_Span \times \cos\left(Wing_LE_Sweep \times \frac{\pi}{180}\right) + y_Slat_in}{\frac{Wing_Span}{2}} \right) \quad (B.8)$$

$$Front_Gap = (\alpha \times \beta \times x_f - Wing_HL_LE_Ratio_cl) \times Wing_c_Root \times 0.3048 \quad (B.9)$$

HL_Leading_Flapped Area Model

Equations B.10-B.16 were used to calculate the ratio of slats flapped area to the wing reference area. Equations B.14-B.16 were derived from [41]. Figure B.3 illustrates the flapped wing area.

Chord at spanwise location - inboard of slat 1:

$$Wing_c_HL_LE_il1 = Wing_c_Root - Wing_HL_LE_eta_il1 \times (1 - Wing_TaperRatio) \times Wing_c_Root \quad (B.10)$$

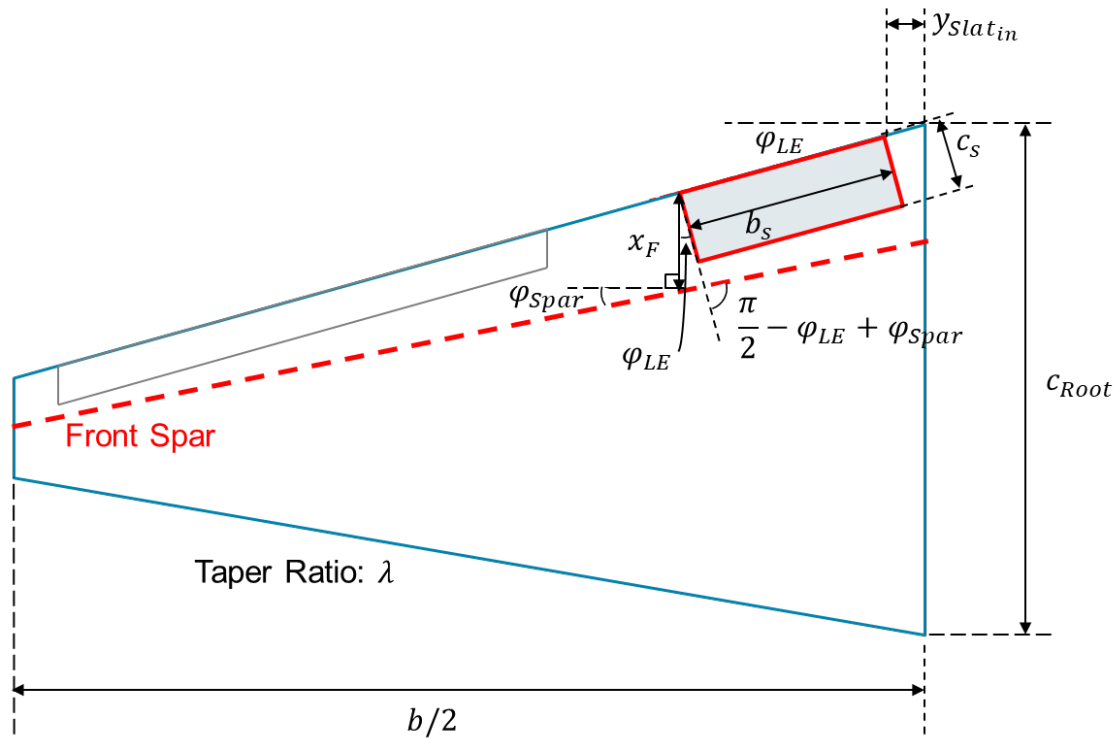


Figure B.2: Interface between the front spar and retracted slat

Chord at spanwise location - outboard of slat 1:

$$Wing_c_HLLE_ol1 = Wing_c_Root - Wing_HLLE_eta_ol1 \times (1 - Wing_TaperRatio) \\ \times Wing_c_Root \quad (B.11)$$

Chord at spanwise location - inboard of slat 2:

$$Wing_c_HLLE_il2 = Wing_c_Root - Wing_HLLE_eta_il2 \times (1 - Wing_TaperRatio) \\ \times Wing_c_Root \quad (B.12)$$

Chord at spanwise location - outboard of slat 2:

$$Wing_c_HLLE_ol2 = Wing_c_Root - Wing_HLLE_eta_ol2 \times (1 - Wing_TaperRatio) \\ \times Wing_c_Root \quad (B.13)$$

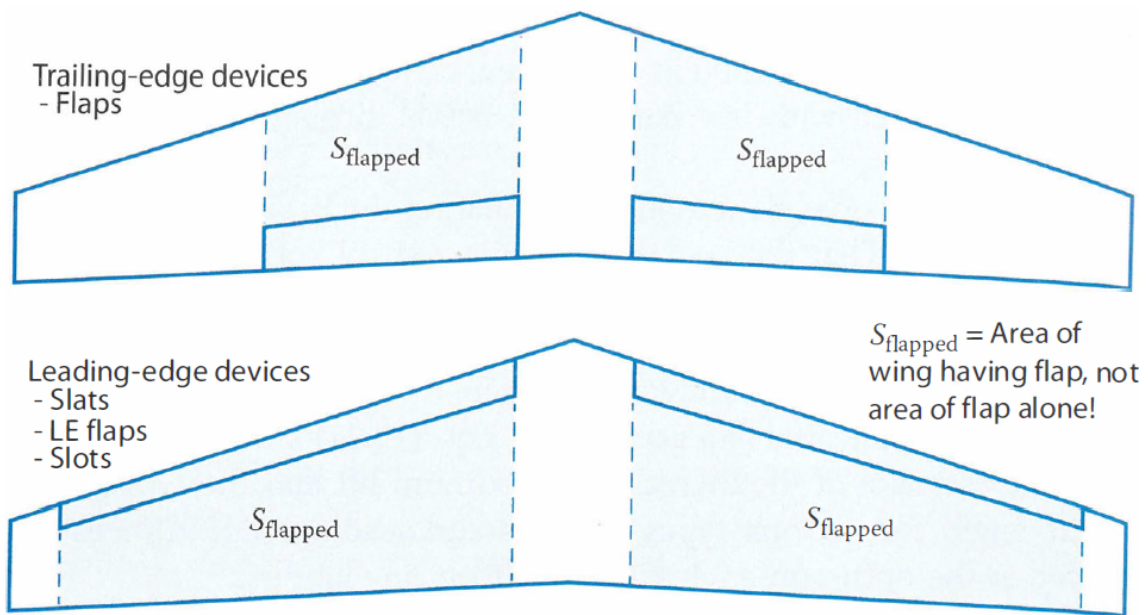


Figure B.3: Illustration of the flapped wing area (from [41])

Flapped Area of slat 1:

$$Wing_S_Flapped_Slat1 = (Wing_c_HLLE_il1 + Wing_c_HLLE_ol1) \times Wing_Span \times \frac{Wing_HLLE_eta_ol1 - Wing_HLLE_eta_il1}{2} \quad (B.14)$$

Flapped Area of slat 2:

$$Wing_S_Flapped_Slat2 = (Wing_c_HLLE_il2 + Wing_c_HLLE_ol2) \times Wing_Span \times \frac{Wing_HLLE_eta_ol2 - Wing_HLLE_eta_il2}{2} \quad (B.15)$$

Ratio of slats flapped area to the wing reference area:

$$Wing_Ratio_LE_Flapped_Area = \frac{Wing_S_Flapped_Slat1 + Wing_S_Flapped_Slat2}{Wing_S_Ref} \quad (B.16)$$

HL Leading Slat 1 Geo Model

Equation B.17 was used to estimate the spanwise distance from wing root (centre-line) chord as fraction of semi-span for outboard limit of the slat.

$$Wing_HL_LE_eta_ol1 = \frac{Wing_HL_Slat1_Span}{0.5 \times Wing_Span} + Wing_HL_LE_eta_il1 \quad (B.17)$$

SlatHingeMoment Model

Equation B.18 calculates the slat hinge moment. The multiplying factor '0.3048' is for converting from feet to meters, whereas the multiplying factor '0.514444' is for converting from knots to m/s.

$$Hs = C_Hs \times 0.5 \times rho_LD \times (0.514444 \times V_App)^2 \times (0.3048 \times chord_slat) \\ \times (0.3048 \times Wing_HL_Slat1_Span) \times (0.3048 \times chord_slat) \quad (B.18)$$

SlatHingeMomentCoef Model

To calculate the hinge moment coefficient of the slat, XFOIL 6.99² was used. XFOIL consists of a collection of menu-driven routines for designing and analysing subsonic isolated airfoils. To calculate the hinge moment coefficient using XFOIL, the following input file was used:

```
NACA 4412 //Airfoil type to be loaded in XFOIL
gdes //Geometry design routine
flap //Add a flap
0.15 //Flap hinge x-location at 15% of airfoil chord (x/c)
999 //To specify flap hinge y-location as % of thickness (y/t)
0.5 //Flap hinge y-location at 50% of thickness
20 //Deflection angle of the flap (degrees)
```

²<https://web.mit.edu/drela/Public/web/xfoil/> (Accessed 27/10/2021)

```

adeg //Command to rotate about origin
-20 //Angle to rotate about origin
eXec //Set Current Airfoil to the new airfoil (i.e. with slat)
//Empty line to go one level up in the menu hierarchy
oper //Direct operating point (for analysis)
r 30000000 //Specify Reynold's number
m 0.3 //Specify Mach number
visc //Turn on viscous mode
iter 250 //Specify number of iterations
alfa 0 //Run analysis at angle of attack = 0
fmom //Calculate hinge moment and forces

```

Tip Deflection Model

Equation B.19 from [191] provides an approximation to estimate the wing tip deflection. The multiplying factor '0.3048' is for converting *Wing_Span* from feet to meters.

$$d_{tip} = \frac{W_{fuse} \times 9.81}{E \times I_0} \times \frac{(Wing_Span \times 0.3048)^3}{96} \times \frac{1 + 2 \times Wing_TaperRatio}{1 + Wing_TaperRatio} \quad (B.19)$$

Wing_Basic_C_L_max_3D_correction Model

Equation B.20 from [41] was used to estimate the maximum lift coefficient of the basic wing (i.e. high-lift devices are not deployed).

$$Wing_C_L_max_B = 0.9 \times Airfoil_C_L_max_B \times \cos\left(Lambda_{025} \times \frac{\pi}{180}\right) \quad (B.20)$$

Wing_HL_CL_max_3D_correction Model

Equation B.21 from [41] was used to estimate the maximum lift coefficient of the wing with high-lift devices deployed.

$$Wing_HL_CL_max = Wing_CL_max_B + (0.9 \times Airfoil_Delta_CL_max \times Wing_Ratio_LE_Flapped_Area \times \cos(Lambda_HingLine \times \frac{\pi}{180})) \quad (B.21)$$

Wing_Weight Model

Equation B.22 from [192] was used to estimate the wing weight. The multiplying factor '0.45359' is for converting from pounds to kilograms.

$$W_wing = (5340 \times (\frac{Wing_Span}{100})^3) \times 0.45359 \quad (B.22)$$

Workflow Execution

Figures B.4 and B.5 show the values of the inputs and outputs, respectively, when executing the computational workflow via AirCADia Explorer.

Name	Value
Wing_HL_LE_eta_il1	0.09700000
Wing_HL_Slat1_Span	15.00000000
Wing_Span	262.00000000
Wing_S_Ref	8000.00000000
Wing_c_Root	46.94000000
Wing_HL_LE_eta_il2	0.27300000
Wing_HL_LE_eta_ol2	0.77300000
Wing_TaperRatio	0.17000000
Wing_HL_LE_Ratio_cl	0.15000000
Wing_HL_LE_Ratio_ct1	0.35000000
Wing_HL_LE_Ratio_Delta_cl	0.08000000
Wing_HL_LE_Ratio_Delta_ct1	0.08000000
Wing_HL_LE_delta_cl	45.00000000
Wing_HL_LE_delta_ct1	35.00000000
Lambda_025	30.00000000
Lambda_HingLine	33.77400000
Wing_LE_Sweep	33.77000000
x_f	0.25000000
y_Slat_in	12.73000000
x_SlatHinge	0.1500
y_SlatHinge	0.5000
DeflectionAngle_Slat	20.0000
Re	30000000.0000
Mach	0.3000
AoA	0.0000
rho_LD	1.22647824
V_App	140.00000000
chord_slat	7.0000
stroke_length	0.3000
E	7310000000.0000
W_0	560000.0000
t_c	0.0800
h_c	0.1500

Figure B.4: Execution of the computational workflow - Input values (AirCADia Explorer)

Name	Value
Wing_HL_LE_eta_ol1	0.21150382
Wing_c_HL_LE_il1	43.16086060
Wing_c_HL_LE_ol1	38.69976900
Wing_c_HL_LE_il2	36.30386540
Wing_c_HL_LE_ol2	16.82376540
Wing_S_Flapped_Slat1	1227.90944395
Wing_S_Flapped_Slat2	3479.85981740
Wing_Ratio_LE_Flapped_Area	0.58847116
Airfoil_C_I_max_B	1.26000000
Airfoil_Delta_C_I_max	2.35000000
Airfoil_C_I_max_HL	3.61000000
Wing_C_L_max_B	0.98207281
Wing_HL_C_L_max	2.01664387
Wing_Delta_C_L_max	1.03457106
Wing_FS_Sweep	33.70724081
Front_Gap	0.3543
C_Hs	0.1627
Hs	10773.0324
F_act	12534.9978
L_act	0.8574
W_wing	43562.0794
I_0	3.4876
W_fuse	201947.5211
d_tip	0.4721

Figure B.5: Execution of the computational workflow - Output values (AirCADia Explorer)

Appendix C

Evaluation: Uncertainty Analysis

The following script was used to generate the uncertainty analysis results in Table 6.4:

```
import numpy as np
import statistics as st
import itertools
import scipy

### Indicators values from use case
I_1 = 0.077
I_2 = 0.333
I_3 = 0.985

### Generate N pseudorandom independent values on interval [0,1
)

N = 1000000
eta = scipy.random.rand(N)

### Intensities of Importance Uncertainty
# Initializing list of possible 'Intensities of Importance'
all_list = [[3, 4, 5], [3, 4, 5], [1, 2, 3, 4, 5]]
# using itertools.product() to compute all possible
permutations

perm = list(itertools.product(*all_list))
```

```
x = 0
y = 0
z = 0
w1_set = [] # Set of resulting w1 from the possible
              combinations in 'perm' and
              value of trigger 'eta'
w2_set = [] # Set of resulting w2 from ...
w3_set = [] # Set of resulting w3 from ...

for i in eta:
    for j in range(len(perm)):
        if i >= (j/len(perm)) and i < ((j+1)/len(perm)):
            x = perm[j][0]
            y = perm[j][1]
            z = perm[j][2]
matrix = np.array([[1, x, y], [1/x, 1, z], [1/y, 1/z, 1]])

### Normalisation of Matrix
col1_sum = 0
col2_sum = 0
col3_sum = 0
for i in range(3):
    col1_sum = col1_sum + matrix[i,0]
    col2_sum = col2_sum + matrix[i,1]
    col3_sum = col3_sum + matrix[i,2]

matrix2 = matrix
for i in range(3):
    matrix2[i,0] = matrix2[i,0]/col1_sum
    matrix2[i,1] = matrix2[i,1]/col2_sum
    matrix2[i,2] = matrix2[i,2]/col3_sum

### Calculation of Weights
```

```
w1 = st.mean(matrix2[0,:])
w1_set.append(w1)
w2 = st.mean(matrix2[1,:])
w2_set.append(w2)
w3 = st.mean(matrix2[2,:])
w3_set.append(w3)

### Uncertainty Analysis for Intensities of Importance
KMI_values = []
for i in range(len(w1_set)):
    KMI = (I_1)**w1_set[i] * (I_2)**w2_set[i] * (I_3)**w3_set[i]
        ]
    KMI_values.append(KMI)

print('Baseline KMI = ', (I_1)**(2/3) * (I_2)**(1/6) * (I_3)**(
        1/6))

print('KMI Min = ', min(KMI_values))
print('KMI Max = ', max(KMI_values))
print('KMI Mean = ', st.mean(KMI_values))
print('KMI Variance = ', st.variance(KMI_values))
print('KMI Std Deviation = ', st.stdev(KMI_values))
print('KMI Coefficient of Variation = ', st.stdev(KMI_values)/
        st.mean(KMI_values))
```


Appendix D

Evaluation: Questionnaire

The following is the questionnaire used as part of the industry feedback session (Section 6.5).



Research Evaluation Questionnaire

Research Title: Systematic Assumption Management in Systems Engineering: An Aircraft Design Perspective

Author: Soufiane El Fassi

* Required

* This form will record your name, please fill your name.

Participant information

1. Name (Optional)

2. Job title and department

*

3. Years of relevant experience

*

4. Participant number

*

5. I confirm that I have read and understand the information provided on the consent form, and give my consent to taking part in this research.

*

Check this item to confirm

Design Belief Network

6. Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option. *

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The proposed network contributes to the industrial need for formal (model-based) assumption management.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The proposed network improves the integration of assumption management within the design environment.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The proposed network is useful at providing traceability to systems engineering activities (such as Requirements Management).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The three criteria: <i>data reliability</i> , <i>model realism</i> , and <i>expert agreement</i> provide a reasonable indication of the practitioner's confidence in making an assumption.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Knowledge Maturity Assessment

7. Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option. *

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Knowledge Maturity assessment provides a reasonable indication of the overall risk of change due to lack of knowledge.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The presented ability to assess progress of Knowledge Maturity over time is useful for supporting decisions at gate reviews.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Assumption Matrix and Change Propagation Analysis

8. Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option. *

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The Assumption Matrix provides a reasonable means of prioritising assumptions in terms of their risk to initiate change.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The ability to suggest a change propagation path up to the closest change absorber is useful for supporting risk assessment.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The presented approach to identify design decisions and assumptions, potentially affected by change propagation, is useful for supporting risk assessment.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Margin Allocation and Revision

9. Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option. *

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Providing the status of margin allocation, using the explicit association with assumptions, is useful for supporting margin management.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The presented approach to detect margin redundancy is reasonable.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The presented approach for suggesting margins to be revised, is useful for supporting margin management.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Overall Approach

10. Please indicate to what extent you agree or disagree with the following statements by choosing the appropriate option. *

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The presented methods can lead to fewer undesired iterations, due to earlier identification and management of risks associated with assumptions.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The presented methods can lead to a better margin balance, due to timely and interactive margin revision.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
An interactive software tool, similar to the one presented, would be a useful addition to a company's suite of tools.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Open questions

11. Is there anything that you particularly liked/disliked about any of the presented methods?

12. In what other ways do you think the presented methods could add value?

13. Do you have any other comments or suggestions?

This content is neither created nor endorsed by Microsoft. The data you submit will be sent to the form owner.

 Microsoft Forms

Appendix E

Publications by the Author

The research in this thesis directly led to the following publications by the author:

1. S. El Fassi, M. D. Guenov, and A. Riaz, “Assumption Management in Model-Based Systems Engineering: A Graph-Theoretical Approach,” **In preparation:** to be submitted for peer review to the journal *Systems Engineering*, 2022.
2. S. El Fassi, M. D. Guenov, and A. Riaz, “A Design Margin Approach to Managing Risks Associated With Assumptions,” **In preparation:** to be submitted for peer review to the *Journal of Mechanical Design*, 2022.
3. S. El Fassi, M. D. Guenov, and A. Riaz, “An Assumption Network-Based Approach to Support Margin Allocation and Management,” in *Proceedings of the Design Society: DESIGN Conference*, vol. 1, pp. 2275–2284, 2020.

The author also co-authored the following publication while a research student at Cranfield University:

X. Chen, A. Riaz, and S. El Fassi, “Application of Artificial Neural Networks for Efficient Reliability-Based Design Optimization,” in *32nd Congress of the International Council of the Aeronautical Sciences*, Shanghai, China, September 6-10, 2021.