TU Delft

Delft University of Technology

A Graph-based, Probabilistic Framework for Novel Aerospace Technology Evaluation and Selection

Roelofs, M.N.

DOI
10.4233/uuid:fad6b4e3-5cbe-4451-b6e4-b40160617488

Publication date
2021

Document Version
Final published version

Citation (APA)
Roelofs, M. N. (2021). *A Graph-based, Probabilistic Framework for Novel Aerospace Technology Evaluation and Selection*. [Dissertation (TU Delft), Delft University of Technology].
https://doi.org/10.4233/uuid:fad6b4e3-5cbe-4451-b6e4-b40160617488

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

A Graph-based, Probabilistic Framework for Novel Aerospace Technology Evaluation and Selection

by Martijn Roelofs

A Graph-Based, Probabilistic Framework for Novel Aerospace Technology Evaluation and Selection

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates,
to be defended publicly on Monday 29 November 2021 at 12:30 o'clock

by

Martijn Nico Roelofs

Master of Science in Aerospace Engineering,
Delft University of Technology, the Netherlands,
born in Leidschendam, the Netherlands.

This dissertation has been approved by the promotors.

promotor: prof. dr. ir. L.L.M. Veldhuis
copromotor: dr. ir. R. Vos

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. L.L.M. Veldhuis	Delft University of Technology, promotor
Dr. R. Vos	Delft University of Technology, copromotor

Independent members:

Prof. dr. G.C.H.E. de Croon	Delft University of Technology
Prof. dr. D. Mavris	Georgia Institute of Technology
Dr. D. Soban	Queen's University of Belfast
Dr. B. Smith	University at Buffalo
Dr. K. Amadori	Linköping University

Reserve members:

Prof. dr. ir. R. Benedictus	Technische Universiteit Delft



Keywords: Aerospace technology, Graph theory, (Bayesian) probability, decision making

Printed by: Gildeprint

Front & Back: Made by the amazing Julieta Bolaños Arriola

Copyright © 2021 by M.N. Roelofs

ISBN 978-94-6419-392-3

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

CONTENTS

Summary vii

Samenvatting xi

1 Introduction 3
- 1.1 Technology Selection . . . 5
- 1.2 Technology Definition . . . 7
- 1.3 Research context: project MANTA. . . . 9
- 1.4 Research Objectives. . . . 9
- 1.5 Research Question . . . 15
- 1.6 Thesis Outline . . . 15

2 Background 19
- 2.1 Technology Evaluation and Selection . . . 20
- 2.2 Analysis Frameworks for Novel Aircraft Configurations and Technologies. . . 23
- 2.3 State-of-the-art . . . 24
- 2.4 Ontology . . . 27
- 2.5 First-Order Logic . . . 31
- 2.6 Graph Theory . . . 33
- 2.7 Uncertainty Definitions . . . 41
- 2.8 Uncertainty Modeling. . . . 42
- 2.9 Uncertainty Propagation . . . 46
- 2.10 Sensitivity Analysis . . . 47
- 2.11 Conclusion . . . 49

3 Engineering systems ontology 51
- 3.1 Requirements for the ontology . . . 53
- 3.2 Basic Formal Ontology . . . 54
- 3.3 Information Artifact Ontology . . . 56
- 3.4 Physics-based Simulation Ontology. . . . 60
- 3.5 Extending BFO, PSO and IAO . . . 61
- 3.6 Technologies as Graph Transformations . . . 64
- 3.7 Granularity . . . 66
- 3.8 Discussion on BFO and PSO . . . 67
- 3.9 Discussion on the Use of an Ontology. . . . 68
- 3.10 Conclusion . . . 68

4 Technology portfolio generation **71**
4.1 Methodology Overview . . . 73
4.2 Technology Compatibility and Incompatibility . . . 73
4.3 Technology Enabling . . . 76
4.4 Generating portfolios . . . 79
4.5 Maximum Dissimilarity . . . 80
4.6 Case study: Aircraft Technologies . . . 82
4.7 Case Study: Industry Technology Set . . . 86
4.8 Case Study: Factorio . . . 88
4.9 Discussion . . . 95
4.10 Conclusion . . . 96

5 Technology portfolio evaluation **99**
5.1 Methodology . . . 102
5.2 Application . . . 112
5.3 Discussion . . . 123
5.4 Conclusion . . . 124

6 Probabilistic inversion **127**
6.1 Methodology . . . 128
6.2 Application . . . 136
6.3 PI for a single technology . . . 145
6.4 PI for technology and portfolio selection . . . 148
6.5 Discussion . . . 153
6.6 Conclusion . . . 155

7 Synthesis **157**
7.1 Method overview and comparison with state-of-the-art . . . 158
7.2 The MANTA project . . . 162
7.3 Application of methodology to MANTA . . . 173
7.4 Conclusion . . . 179

8 Conclusion **181**
8.1 Conclusions . . . 182
8.2 Recommendations . . . 184
8.3 Outlook . . . 185

Acknowledgements **187**

References **189**

A Constraint Satisfaction Problem **203**

B Subgraph Isomorphism **207**

C Graph Edit Distance **209**
C.1 Fast GED Implementation Details for Graph Transformation Rules . . . 210

D Enabling technology set **213**

E Maximum dissimilarity **215**

F Portfolio Generation **217**

G Factorio Results **219**

H Dependency graph algorithm **223**

I Computation graph algorithm **227**

J System Flow Directions **231**

K Dependency Graphs Case Studies **233**

Curriculum Vitæ **237**

List of Publications **239**

SUMMARY

Technology selection is ubiquitous in all manners of complex systems engineering, design and everyday business operations. Technologies are often complex entities shrouded in uncertainty, assumptions and interpretation. Therefore, quantifying their effect (i.e. outcome) becomes challenging for several reasons. First, as simulations may not be available for novel technologies, experts have to estimate their impacts. Secondly, as the technologies may not be well-defined, different experts have different interpretations and will assign different outcomes. Thirdly, even if analysis methods are available, there is epistemic and model uncertainty in the outcome. Finally, available analysis frameworks are typically application-specific, non-extensible and inflexible.

It is the objective of this thesis to pave the way towards a technology selection methodology that offers a structured, repeatable and traceable way to represent technologies and consecutively quantify their impacts on an engineering system. Such a methodology is implemented as a decision support system, i.e. a computer program that assists a decision maker throughout the decision-making process. Three components of said methodology can be identified: technology representation and portfolio generation, technology (portfolio) evaluation and technology (portfolio) selection.

From these three components, the following three research objectives are distilled:

- Establish a formal knowledge representation of engineering systems, such that technologies can be described.
- Specify, develop and demonstrate a methodology that enables automatic inference of technology (parameter) dependencies, using the knowledge representation established earlier.
- Specify, develop and demonstrate a methodology that estimates uncertainty distributions and subsequently quantifies uncertainty in the system, based on a given dependency structure, as well as support inverse uncertainty quantification.

These objectives reflect that technology evaluation in engineered systems is extremely complex, especially in high-tech industries, such as the aerospace or automotive sectors. They do that by aiming for automation and incorporating uncertainty into the selection process.

A formal knowledge representation is achieved through an ontology that aims to capture engineering systems as realistically as possible. The basis of this ontology is formed by a combination of three upper ontologies: Basic Formal Ontology (BFO), Information Artifact Ontology (IAO) and Physics-based Simulation Ontology (PSO). With the ontology at the foundation, the information that describes an engineering system is captured in a knowledge graph. Such a graph contains the particulars (individuals) as nodes, and their relations as edges. A node has a type and possibly any number of attributes. Edges

also have a type and may contain attributes. Knowledge graphs are very flexible data structures and, therefore, support the flexibility, extensibility and applicability of the method.

Technology is defined in this dissertation as a materialized form of knowledge of a system in order to alter the system's form or behavior to satisfy certain requirements. The key insight following from this definition is to model technologies as graph transformation rules. Such a rule takes part of a knowledge graph and transforms it into another knowledge graph. Because the knowledge graphs describe systems, the transformation describes the change in that system. The graph transformation rules enjoy the expressiveness and consistency of the underlying knowledge graphs, because a rule only consists of two knowledge graphs: a pattern graph and an effect graph.

When combining technologies into technology portfolios, technology incompatibility and technology enablers have to be considered. Technology incompatibility is considered through two mechanisms: the graph transformation rules and first-order logic (FOL) based on the ontological description. The graph transformation rules have formal notions of independence associated with them that are translated into technology compatibility. The latter approach is more flexible and extensible than the FOL-based approach. As a result, it is advised to use that approach, unless a clear definition of technology compatibility is available.

Any framework for technology evaluation has to be modular and extensible. There is simply no analysis method that could capture all possible systems and technologies. Modularity is achieved by specifying separate analysis models for different aspects of a system. Again, the ontology allows for providing context to the analysis models. A knowledge graph specifies the pattern in a system that the analysis model applies to.

From the knowledge graph of a system (including a certain technology portfolio), and the application of a set of analysis models, a dependency graph is generated. The dependency graph contains all possible computation directions between the variables in the system. From this graph, a computation graph is distilled given the quantities of interest (QoIs) and known input variables. The computation graph contains one specific possible computation sequence to compute the QoI from the inputs.

When a probabilistic calculation is performed, the joint probability distribution of two dependent variables should be correctly characterized. To assist the practitioner, a first-order-logic-based rule set is created that enables inference of dependent variables. It works by stating generalized causal influences between properties of physical entities. A practitioner can view which variables are considered as dependent by the rules, and specify joint dependency structures for those.

The probabilistic inversion (PI) technique is proposed to compute a preference over technology portfolios when their effects are uncertain. PI requires samples of the input and output variables of an analysis model. Therefore, it poses no restriction on the type of model, except that it is computationally tractable to generate thousands of samples. Then, constraints (i.e. target distributions) on any of the input and output variables have to be specified. The PI problem then aims to find a new distribution over the samples such that all constraints are satisfied. The redistributed variables (after PI) show how the technology variables need to change in order to achieve certain goals. Similarly, when multiple technologies are combined into portfolios, the combined effects are quantified,

and probabilistic inversion shows how the frequency of the technologies should change to achieve certain goals. This frequency shift provides a preference order for selection. Multi-objective goals can be imposed on PI, without having to be converted to a single-objective function, as usually is done in design optimization. This comes at the cost that each goal is treated with equal weight.

A comparison is made between the state-of-the-art method and the herein developed method towards technology evaluation and selection. Three key differences between these methods have been identified. First, the proposed method features significantly more automation than the state-of-the-art approach. Second, in the proposed approach, all data is either stored in a knowledge graph, or is numeric. This makes it machine-interpretable and removes all the disadvantages that textual data has in the state-of-the-art method. Third, the approach is more flexible and extensible than the state-of-the-art, because the dependency and computation graphs are generated on-the-fly and uniquely for each technology portfolio. This also removes the dependency of the impact factors on the analysis tool. Concluding, the graph-based, probabilistic framework for technology evaluation and selection provides a robust, consistent and traceable process to evaluate and rank technology portfolios.

SAMENVATTING

Selectie van technologie is alomtegenwoordig op alle manieren van complexe systeembouw, ontwerp en dagelijkse bedrijfsvoering. Technologieën zijn vaak complexe entiteiten gehuld in onzekerheid, aannames en interpretaties. Daarom wordt het kwantificeren van hun effect (d.w.z. uitkomst) om verschillende redenen een uitdaging. Ten eerste, aangezien simulaties mogelijk niet beschikbaar zijn voor nieuwe technologieën, moeten experts hun impact inschatten. Ten tweede, aangezien de technologieën misschien niet goed gedefinieerd zijn, hebben verschillende experts verschillende interpretaties en zullen ze verschillende resultaten toewijzen. Ten derde, zelfs als er analysemethoden beschikbaar zijn, is er epistemische en modelonzekerheid in de uitkomst. Ten slotte zijn de beschikbare analysekaders typisch toepassingsspecifiek, niet uitbreidbaar en inflexibel.

Het is het doel van dit proefschrift om de weg vrij te maken voor een technologieselectiemethodologie die een gestructureerde, herhaalbare en traceerbare manier biedt om technologieën weer te geven en achtereenvolgens hun impact op een technisch systeem te kwantificeren. Een dergelijke methodologie wordt geïmplementeerd als een beslissingsondersteunend systeem, d.w.z. een computerprogramma dat een besluitvormer bijstaat tijdens het besluitvormingsproces. Drie componenten van de genoemde methodologie kunnen worden onderscheiden: technologierepresentatie en portfoliogeneratie, technologie (portfolio) evaluatie en technologie (portfolio) selectie.

Uit deze drie componenten worden de volgende drie onderzoeksdoelstellingen gedestilleerd:

- Breng een formele kennisweergave van technische systemen tot stand, zodat technologieën kunnen worden beschreven.

- Specificeer, ontwikkel en demonstreer een methodologie die automatische inferentie van technologische (parameter) afhankelijkheden mogelijk maakt, gebruikmakend van de eerder vastgestelde kennisrepresentatie.

- Specificeer, ontwikkel en demonstreer een methodologie die onzekerheidsverdelingen schat en vervolgens de onzekerheid in het systeem kwantificeert, op basis van een gegeven afhankelijkheidsstructuur, en die ook de inverse onzekerheidskwantificatie ondersteunt.

Deze doelstellingen weerspiegelen dat de evaluatie van technologie in technische systemen buitengewoon complex is, vooral in hightechindustrieën, zoals de lucht- en ruimtevaart- of automobielsector. Dat doen ze door te streven naar automatisering en door onzekerheid mee te nemen in het selectieproces.

Een formele kennisrepresentatie wordt bereikt door middel van een ontologie die erop gericht is technische systemen zo realistisch mogelijk vast te leggen. De basis van

deze ontologie wordt gevormd door een combinatie van drie bovenste ontologieën: Basic Formal Ontology (BFO), Information Artifact Ontology (IAO) en Physics-based Simulation Ontology (PSO). Met de ontologie aan de basis, wordt de informatie die een engineeringssysteem beschrijft vastgelegd in een kennisgrafiek. Zo'n grafiek bevat de bijzonderheden (individuen) als knooppunten en hun relaties als randen. Een knoop heeft een type en mogelijk een willekeurig aantal attributen. Randen hebben ook een type en kunnen attributen bevatten. Kennisgrafieken zijn zeer flexibele datastructuren en ondersteunen daarom de flexibiliteit, uitbreidbaarheid en toepasbaarheid van de methode.

Technologie wordt in dit proefschrift gedefinieerd als een gematerialiseerde vorm van kennis van een systeem om de vorm of het gedrag van het systeem te veranderen om aan bepaalde eisen te voldoen. Het belangrijkste inzicht dat uit deze definitie volgt, is om technologieën te modelleren als regels voor grafiektransformatie. Zo'n regel maakt deel uit van een kennisgrafiek en zet deze om in een andere kennisgrafiek. Omdat de kennisgrafieken systemen beschrijven, beschrijft de transformatie de verandering in dat systeem. De grafiektransformatieregels genieten de expressiviteit en consistentie van de onderliggende kennisgrafieken, omdat een regel slechts uit twee kennisgrafieken bestaat: een patroongrafiek en een effectgrafiek.

Bij het combineren van technologieën in technologieportfolio's moet rekening worden gehouden met incompatibiliteit met technologie en technologie-enablers. Technologie-incompatibiliteit wordt beschouwd via twee mechanismen: de grafiektransformatieregels en eerste-orde logica (FOL) op basis van de ontologische beschrijving. Aan de transformatieregels voor grafieken zijn formele noties van onafhankelijkheid verbonden die worden vertaald in technologiecompatibiliteit. De laatste benadering is flexibeler en uitbreidbaar dan de op FOL gebaseerde benadering. Daarom wordt geadviseerd om die benadering te gebruiken, tenzij er een duidelijke definitie van compatibiliteit van technologie beschikbaar is.

Elk raamwerk voor technologie-evaluatie moet modulair en uitbreidbaar zijn. Er is gewoon geen analysemethode die alle mogelijke systemen en technologieën kan vastleggen. Modulariteit wordt bereikt door afzonderlijke analysemodellen te specificeren voor verschillende aspecten van een systeem. Nogmaals, de ontologie maakt het mogelijk om context te bieden aan de analysemodellen. Een kennisgrafiek specificeert het patroon in een systeem waarop het rekenmodel van toepassing is.

Vanuit de kennisgrafiek van een systeem (inclusief een bepaald technologieportfolio) en de toepassing van een set analysemodellen wordt een afhankelijkheidsgrafiek gegenereerd. De afhankelijkheidsgrafiek bevat alle mogelijke berekeningsrichtingen tussen de variabelen in het systeem. Uit deze grafiek wordt een berekeningsgrafiek gedestilleerd gezien de hoeveelheden van belang (QoI's) en bekende invoervariabelen. De berekeningsgrafiek bevat een specifieke mogelijke berekeningsreeks om de QoI uit de ingangen te berekenen.

Wanneer een probabilistische berekening wordt uitgevoerd, moet de gezamenlijke kansverdeling van twee afhankelijke variabelen correct worden gekarakteriseerd. Om de beoefenaar te helpen, wordt een eerste-orde-logica-gebaseerde regelset gecreëerd die het mogelijk maakt afhankelijke variabelen af te leiden. Het werkt door gegeneraliseerde causale invloeden tussen eigenschappen van fysieke entiteiten te vermelden. Een beoe-

fenaar kan zien welke variabelen door de regels als afhankelijk worden beschouwd, en hiervoor structuren voor gezamenlijke afhankelijkheid specificeren.

De probabilistische inversie (PI) techniek wordt voorgesteld om een voorkeur boven technologieportfolio's te berekenen wanneer hun effecten onzeker zijn. PI heeft steekproeven nodig van de input- en outputvariabelen van een analysemodel. Daarom stelt het geen beperking op het type model, behalve dat het rekenkundig traceerbaar is om duizenden monsters te genereren. Vervolgens moeten beperkingen (d.w.z. doelverdelingen) voor elk van de invoer- en uitvoervariabelen worden gespecificeerd. Het PI-probleem beoogt vervolgens een nieuwe verdeling over de monsters te vinden, zodat aan alle beperkingen wordt voldaan. De herverdeelde variabelen (na PI) laten zien hoe de technologievariabelen moeten veranderen om bepaalde doelen te bereiken. Evenzo, wanneer meerdere technologieën worden gecombineerd in portfolio's, worden de gecombineerde effecten gekwantificeerd, en de probabilistische inversie laat zien hoe de frequentie van de technologieën zou moeten veranderen om bepaalde doelen te bereiken. Deze frequentieverschuiving biedt een voorkeursvolgorde voor selectie. Multiobjectieve doelen kunnen aan PI worden opgelegd, zonder dat ze hoeven te worden geconverteerd naar een enkelvoudige objectieve functie, zoals gewoonlijk wordt gedaan bij ontwerpoptimalisatie. Dit gaat ten koste van het feit dat elk doel met evenveel gewicht wordt behandeld.

Er wordt een vergelijking gemaakt tussen de state-of-the-art methode en de hierin ontwikkelde methode richting technologie-evaluatie en selectie. Er zijn drie belangrijke verschillen tussen deze methoden geïdentificeerd. Ten eerste bevat de voorgestelde methode aanzienlijk meer automatisering dan de state-of-the-art aanpak. Ten tweede worden in de voorgestelde benadering alle gegevens opgeslagen in een kennisgrafiek of zijn ze numeriek. Dit maakt het machinaal interpreteerbaar en neemt alle nadelen weg die tekstuele gegevens hebben in de state-of-the-art methode. Ten derde is de aanpak flexibeler en uitbreidbaar dan de state-of-the-art, omdat de afhankelijkheids- en berekeningsgrafieken on-the-fly en uniek worden gegenereerd voor elke technologieportfolio. Dit neemt ook de afhankelijkheid van de impactfactoren van de analysetool weg. Concluderend biedt het op grafieken gebaseerde, probabilistische raamwerk voor technologie-evaluatie en -selectie een robuust, consistent en traceerbaar proces voor het evalueren en rangschikken van technologieportfolio's.

NOMENCLATURE

A	Attributes
$\boldsymbol{a}$	Acceleration
C_D	Drag coefficient
C_L	Lift coefficient
D	Dependency Graph
$\boldsymbol{D}$	Drag force
E	Edges
e	Oswald efficiency factor
F	Cumulative density function
$\boldsymbol{F}$	Force
G	(Knowledge) Graph
H	Pattern Graph
$\mathbb{I}$	Indicator function
K	Gluing graph
k	Impact factor
L	Pattern graph
$\boldsymbol{L}$	Lift force
M_{F}	Fuel mass
M_{OE}	Operating empty mass
M_{MTO}	Maximum take-off mass
M_{P}	Payload mass
M_{ZF}	Zero-fuel mass
m	Match, Morphism
$\mathcal{N}$	Gaussian distribution function
P	Analysis Parameters, Power
p	Probability
q	Inter-quantile
R	Effect graph, Range
S	Wing area
s	Sample
SE	Specific Energy
$\boldsymbol{T}$	Thrust force
t	Technology
V	Vertices, Velocity
$\boldsymbol{W}$	Weight
$\boldsymbol{x}$	Sample vector, optimization vector
$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$	Random vectors

Operators

$\forall$	For all
$\exists$	Exists
$\Rightarrow$	Implies
$\Leftrightarrow$	If and only if
$\wedge$	And
$\vee$	Or
$\neg$	Negation
$\parallel$	Compatible
$\perp$	Incompatible
$\prec$	Enables

Greek symbols

α	Copula shape parameter
Δ	Difference
δ	Deflection angle
η	Efficiency
μ	Mean
ρ	Air density
σ	Standard deviation
τ	Kendall's tau

Subscripts

ac	Aircraft
bat	Battery
des	Descent
cl	Climb
cr	Cruise
e	Electric, Oswald factor
f	Fuel, Flap
gb	Gearbox
I	Input
i	i-th entry
M	Mass
max	Maximum
O	Output
OEM	Operating Empty Mass
p	Propeller

req	Requirement
s	Shaft
tot	Total
0	Initial

Acronyms

BFO	Basic Formal Ontology
CAD	Computer Aided Design
CDF	Cumulative Distribution Function
CSP	Constraint Satisfaction Problem
DAG	Directed Acyclic Graph
DOF	Degree-of-Freedom
EM	Electric Machine (Electromotor)
FB	Fuel burn
FD	Functional Decomposition
GB	Gearbox
GDC	Generically Dependent Continuant
GT	Gas Turbine
GUI	Graphical User Interface
IC	Independent Continuant
ICE	Information Content Entity
IPF	Iterative Proportional Fitting
IRL	Integration Readiness Level
MA	Mission Analysis
MDO	Multidisciplinary Design Optimization
PM	Power Module
PDF	Probability Density Function
PI	Probabilistic Inversion
POS	Probability of Success
PRE	Payload Range Efficiency
PSO	Physics-based Simulation Ontology
QoI	Quantity of Interest
SA	Sensitivity analysis
SDC	Specifically Dependent Continuant
SE	Specific energy
S/N	Signal-to-noise ratio
SPPH	Serial-Parallel Partial Hybrid
SRL	System Readiness Level
TCG	Technology Compatibility Graph
TCM	Technology Compatibility Matrix
TIF	Technology Impact Forecasting
TRL	Technology Readiness Level
TSEC	Thrust-specific energy consumption
UQ	Uncertainty Quantification

1

INTRODUCTION

1

Technology selection is ubiquitous in all manners of complex systems engineering, design and everyday business operations. From picking one design alternative over another during the conceptual design of a radical new aircraft configuration, to selecting a new manufacturing process for a small part inside a complex machine. From a research project investigating promising technologies, to the incremental improvement of an existing product by including a new technology.

It comes as no surprise that people have researched extensively how to make good decisions [1, 2]. This has led to the development of Decision Theory [3, 4]. A decision maker has to choose from a set of alternatives that each lead to some outcome, and does so by assigning utility to each outcome. For example, the alternatives could be various aircraft technologies, while the outcome is the aircraft fuel burn for a given mission. However, technologies are often complex entities shrouded in uncertainty, assumptions and interpretation. Therefore, quantifying their utility (i.e. outcome) becomes challenging for several reasons:

1. As simulations may not be available for novel technologies, experts have to estimate their impacts.

2. As the technologies may not be well-defined, different experts have different interpretations and will assign different outcomes.

3. Even if analysis methods are available, they are typically application-specific, non-extensible and inflexible. More often than not, specific analysis methods need to be developed, which are stand-alone, and not integrated with existing methods.

Let us consider the first challenge. Although there is currently no practicable alternative to using expert judgment, there are several problems. Experts may be over-confident (illustrated in Figure 1.1), resulting in inaccurate judgment of the technology effect [5]. Conversely, experts may be non-informative; their estimation may be factually correct but not very informative (e.g. specification of a wide range of values for a variable) [6], as shown in Figure 1.2. The minimum and maximum values are retrieved from all experts' estimates, or could simply be negative and positive infinity, respectively. Finally, experts (inherent to human nature) may be inconsistent [7], leading to different technologies being assessed differently from similar technologies that have the same behavior or function.

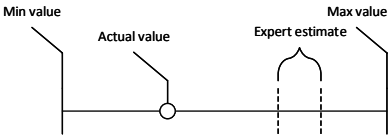


Figure 1.1: Overconfident expert: very informative estimate, but unfortunately far from the truth.

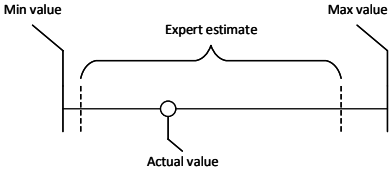
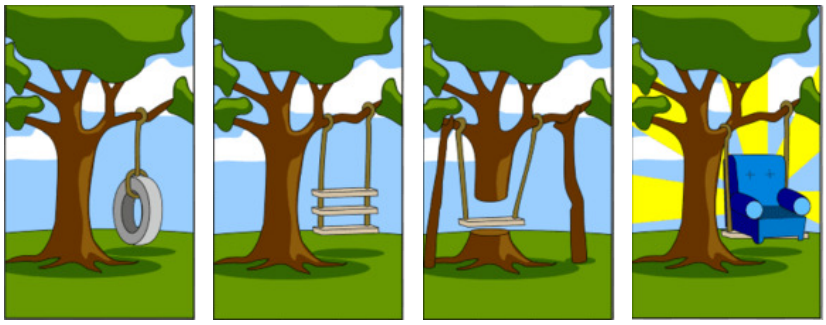


Figure 1.2: Uninformative expert: the truth is within the estimate, but the estimate has a wide range.

The second challenge aggravates the first, in that the inconsistency from a single expert is magnified by the inconsistency between experts as a result of differing views of

what a certain technology entails, how it works and what its effect is on the quantities of interest. Besides such miscommunication, lack of communication or lack of common data may play a role. This problem is demonstrated in Figure 1.3 [1], with a well-known cartoon in systems engineering that illustrates the miscommunication and misinterpretation of different people in a systems engineering project.



(a) The technology (b) The description (c) Understanding expert 1 (d) Understanding expert 2

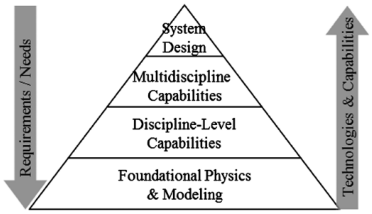
Figure 1.3: Misunderstanding of a technology leading to a mischaracterization of their effects

Regarding the third challenge, physics-based analysis methods are preferred to assess novel technologies with reasonable accuracy. However, there is epistemic uncertainty, i.e. lack of knowledge concerning the physics involved. Additionally, the technology itself may not be matured, hence its behavior and form may change during the development process. Consider the four-level hierarchy proposed by NASA (see Figure 1.4) [8]. Each level expresses the integration of a technology in a multi-disciplinary system, along with the requirements to reach that level. Note how the levels are stacked on top of each other, meaning that to develop a technology at level 4, one has to go through the other three levels first. There usually is a gap between level 1 and 3, since it is difficult to create a method or tool that relates fundamental physics to technology-specific responses. Additionally, for novel technologies, level 1 is also often lacking, if the technology relies on an unexplored aspect of physics. Finally, when combinations of technologies are considered, their compatibility and combined effects need to be assessed. These combined effects might involve synergies or discordance. Synergy means that the multiple technologies affect the same target and the total impact is more than the addition of the individual contributions. Discordance means that the multiple technologies partly or fully negate each other's impacts. Both need to be properly defined.

1.1. TECHNOLOGY SELECTION

As we'll see in Chapter 2, technology selection is prevalent in aircraft design practices. It comes as no surprise that it has received considerable attention from academic re-

[1] https://hackernoon.com/how-to-accept-over-engineering-for-what-it-really-is-6fca9a919263, Retrieved 7 April 2021



Level 4	Development of system-level capabilities, integration of multidisciplinary technologies
Level 3	Integration of methods and technologies to develop multidisciplinary solutions
Level 2	Development of discipline-specific technologies and tools
Level 1	Foundational research to further fundamental understanding of underlying principles

Figure 1.4: Levels of technnology reasearch and system development [8]. Lower levels have an increasing number of requirements and needs, and focus more and more on fundamental principles. Traversing to higher levels increases the amount of integration and builds on lower levels to create multidisciplinary technologies and capabilities.

searchers. A common technique to account for the impact of technologies is through assigning a difference to a certain parameter that describes the aircraft system. For example, the effect of an entire flap system can be represented by a change in maximum lift capability and subsystem mass. The Technology Identification, Evaluation and Selection (TIES) [9–11] methodology, developed at Georgia Tech, works this way. A probabilistic approach towards uncertainty is adopted and technology selection is done based on the highest probability of meeting objectives [12]. The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) is used to create a ranking of technology portfolios. The impacts of technologies are called k-factors, because they are multiplicative factors to an attribute of the aircraft system that is affected by the technology. The advantage of using k-factors is that key disciplinary metrics are taken into account and no commitment needs to be made to modeling a technology [13]. It is therefore a straightforward approach, and can quickly and easily identify the impact of certain technologies. Epistemic uncertainty with regard to subsystem assessment and state of the art (SOTA) assumptions may also be dealt with by using k-factors [14, 15]. However, subtle characteristics specific to a certain technology may not be taken into account and therefore distinguishing effects may be overlooked. For example, a certain technology may require a cutout in an existing structural member. Then the structural assessment of that member needs to be modified to take this into account. A k-factor could perhaps only be applied to the system mass of which the structural member is a part. Estimating the effect of the technology, especially when the system has to be re-sized, then becomes incredibly difficult.

Simulation environments to analyze novel technologies have also received considerable attention. Two major limitations of current conceptual design environments were identified by Lu et al. [16]:

1. The application of design tools is limited to a specific vehicle type, because the sizing method is fixed

2. Flexibility and scalability of disciplinary analysis tools is lacking

Both of these limitations should be overcome for effective evaluation of novel aircraft concepts and technologies. However, creating a flexible and modular design environment is challenging due to the coupling of operational and systems capabilities [17].

Nonetheless, some libraries exist (e.g. Modelica) that focus on flexibility in disciplinary analysis. The goal is to create such solutions for all disciplines. Another important challenge for designing unconventional aircraft is consistent geometry representation. As of now, no suitable geometry definition tool or CAD package is available that allows geometry to be used continuously from conceptual design into detailed design [18].

Despite these challenges, several efforts have been made to arrive at a conceptual aircraft design framework that is suitable for technology evaluation [19, 20]. Some include uncertainty quantification [21–23]. Additionally, NASA has developed several physics-based analysis tools for different disciplines, which were used in an aircraft design study for the Next Generation Air Transportation System [24]. A practical technology evaluation study was performed by Heinemann et al. [25] to investigate the feasibility of meeting the goals set for a year 2050 tube-and-wing aircraft. Several technologies are discussed, including maneuver and gust load alleviation, a hybrid laminar-flow wing, variable camber and active flow control.

As we can see, technology evaluation focuses on representing technologies quantitatively with k-factors, whilst obscuring the intricacies of the technologies themselves. Additionally, analysis frameworks have been developed to investigate particular, novel technologies and designs. Even uncertainty is adopted in a multitude of research efforts. However, the issue of consistency is systematically overlooked or avoided. Furthermore, proper expert elicitation procedures are often not employed. There is neither a clear-cut method to represent and capture technologies, nor a framework that enables quantifying their effects consistently. When considering uncertainty, most efforts only include forward propagation of uncertainty. While this is already valuable, the inverse problem is of interest as well. Furthermore, while it is known that dependencies between random variables need to be considered for a proper evaluation of the impact of technologies, this is hardly ever done in practice.

1.2. TECHNOLOGY DEFINITION

Technology evaluation is not a straight-forward process as has been clarified above. Therefore, this work aims to contribute to the available knowledge on how to quantify the impact of a technology or technology portfolio on an engineering system. However, up to now, we have not clarified what a technology is perceived to be within this work. In fact, the term technology is rather broad and has enjoyed many different definitions. Marx [26] even emphasizes the vagueness of the term:

> The curious fact is that the discursive triumph of the concept of technology is in large measure attributable to its vague, intangible, indeterminate character — the fact that it does not refer to anything as specific or tangible as a tool or machine.

As an example of a definition, Bush [27] states:

> “Technology is a form of human cultural activity that applies the principles of science and mechanics to the solutions of problems. It includes the resources, tools, processes, personnel, and systems developed to perform tasks and create immediate particular, and personal and/or competitive advantages in a given ecological, economic, and social context.”

Although it is a more socio-cultural definition, it expresses the things that a technology encompasses. Furthermore, it highlights that a technology serves a purpose, i.e. has a function or goal. A similar conclusion is obtained from the definition by Merrill [28]:

> "Technology in its broad meaning connotes the practical arts. These arts range from hunting, fishing, gathering, agriculture, animal husbandry, and mining through manufacturing, construction, transportation, provision of food, power, heat, light, etc., to means of communication, medicine, and military technology. Technologies are bodies of skills, knowledge, and procedures for making, using and doing useful things. They are techniques, means for accomplishing recognized purposes"

Feibleman [29] approaches the definition of technology from a broader perspective, including pure science, applied science, technology and engineering. Historically, applied science and technology were quite distinct, being practiced by natural philosphers and articans, respectively. However, over the past few centuries, the two have merged to a certain extent Feibleman [29]:

> "There is now only the smallest distinction between applied science, the application of the principles of pure science, and technology. The methods peculiar to technology: trial-and-error, invention aided by intuition, have merged with those of applied science: adopting the findings of pure science to the purposes of obtaining desirable practical consequences. Special training is required, as well as some understanding of applied and even of pure science."

Mitcham and Schatzberg [30] provide a similar review of and distinction between pure science, applied science, technology and engineering.

Although much more can be said about what technology is and how it relates to science and engineering, the purpose of this section is to provide a working definition within the context of this work. Therefore, the following definition is proposed:

> **A technology is a materialized form of knowledge applied to a given system in order to alter the system's form or behavior to satisfy certain requirements.**

Let us further expand this definition. The *materialized form of knowledge* ranges from implementing a process, incorporating a protocol, or instantiating or modifying a material entity. Thus, when the knowledge is only implicit (e.g. exists as a thought), it is not considered materialized. The *system* can be any system, including social, biological and physical systems, as well as engineered systems. For the present work, only engineered systems are considered. The *system behavior* involves all processes that the system participates in. Its *form* is the physical, tangible object that makes up the system in reality. Finally, a technology has a purpose, as Bush [27] and Merrill [28] conclude. Therefore, it has to satisfy certain requirements that capture the function or goal of the technology.

Concretely, in the aircraft domain, a technology according to this definition could be a new material for the wingbox that makes the wing more flexible and lighter, in order

to reduce fuel consumption. Another example would be software that enables the airplane to be statically unstable by correcting for deviations from equilibrium through a controller, allowing the horizontal tailplane to be smaller and reduce fuel burn in this way. Finally, a technology could be an operational procedure where the airspace is more efficiently utilized, such that airplanes can fly more direct routes and have shorter loiter times, reducing the fuel burn of a fleet of airplanes at airline level.

1.3. RESEARCH CONTEXT: PROJECT MANTA

The research presented in this thesis has been conducted as part of the MANTA project, which is part of the European Commission CleanSky II Joint Undertaking Work Plan. Clean Sky 2 is a Public-Private Partnership (PPP) between the European Commission and the EU aviation industry, aiming to reduce aviation environmental impact by accelerating development and deployment of cleaner air transport technologies [2]. MANTA stands for Movables for Next Generation Aircraft and was started to develop and demonstrate innovative multifunctional movables that increase airframe efficiency over the complete flight envelope of business jets (BJ) and large passenger aircraft (LPA) as contribution to the societal challenge for this topic to reduce 3 to 5% CO_2. The MANTA consortium consists of two industrial partners, Fokker Aerostructures and ASCO, as well as two research centres, NLR and DLR and one university, TUD. The project is guided by three topic leaders: Airbus, Dassault Aviation and Saab Aeronautics.

Technologies introduced in the MANTA project are used within this thesis to conduct experiments and exemplify the introduced methods. Unfortunately, the method developed herein was not applied to MANTA due to chronological misalignment, as is discussed in Chapter 7. Not insignificantly, the MANTA project also provided valuable insights into the state-of-practice of industry technology selection, which incentivized parts of the present work.

1.4. RESEARCH OBJECTIVES

The aim of any technology selection methodology is a structured, repeatable and traceable way to exploit all information available on applicable technologies and extracting useful information about the effect of the technologies in order to reduce subjectivity in the technology selection process [32]. It is the objective of this thesis to pave the way towards a technology selection methodology that offers a structured, repeatable and traceable way to represent technologies and consecutively quantify their impacts on an engineering system. Such a methodology can be implemented as a decision support system, i.e. a computer program that assists a decision maker throughout the decision-making process. Three components of said methodology can be identified:

1. Technology representation and portfolio generation
2. Technology (portfolio) evaluation
3. Technology (portfolio) selection.

These are elaborated below.

[2] www.cleansky.eu

Technology representation and portfolio generation A formal knowledge representation method needs to be in place to describe technologies. Capturing technologies formally has several benefits: knowledge capturing, knowledge sharing, consistency, and automation. These benefits serve to build a foundation for addressing the challenges introduced at the start of this chapter. Automation further enables us to replace human experts by expert systems, which may improve reliability, robustness and consistency. When human experts are still employed, a structured knowledge representation may align their interpretations and reduce diverging assessments. A formal representation offers a language flexible enough for practitioners to express what a certain technology entails, while being restrictive enough such that ambiguity is avoided. If such a language can be encoded in a structured data format, the information can be easily shared and integrated. If only one valid way exists to express a certain concept, the lack of ambiguity also improves consistency — no matter how one starts to describe a technology, they end up with the same description. When human experts are replaced with machine experts, we can avoid human pitfalls, such as overconfidence or bias. Of course, machines are not perfect either. Deep neural networks may, for example, be able to learn very complex tasks, but we have very little insight into the things they learn and how they even work. Nonetheless, we can more easily measure the reliability and precision of a machine expert than that of a human one.

Technology (portfolio) evaluation Novel technologies may rely on physics for which empirical or computationally tractable estimations or simulations are unavailable. Therefore, it is hard to quantify their impacts and effects. This problem is aggravated by the lack of knowledge on the final form, function and behavior of the technologies. However, we require some manner of quantification of effects for an unambiguous decision under risk/uncertainty, rather than a decision under ignorance. Technology selection usually takes place in the conceptual design stage or early phase of a research project, when the combinatorial design space is large. Flexible, detailed, physics-based analysis methods are too computationally expensive to run for such a vast design space. Thus, we need a way to easily model novel technologies for which new analysis methods have to be developed, and have the flexibility of combining those analysis methods with our current simulation tools. The aim is to reuse existing analysis methods and facilitate the creation and inclusion of new ones. An evaluation framework that satisfies these needs tackles the third challenge exposed earlier. Moreover, new disciplinary analysis methods themselves should be more flexible in the variety of systems they can analyze. The graph-based approach introduced in this thesis enables the creators of these analysis methods to meet this requirement.

Technology (portfolio) selection Decision Theory offers many guidelines on selecting the best rational option from a set of alternatives. Even in presence of uncertainty, Decision Theory can be employed. However, technology selection is sometimes not the goal of the exercise. Rather, we wish to figure out more about the technologies we are investigating, and may select a technology that seems to have a higher Technology Readiness Level (TRL), or is easier to research and implement. Thus, Decision Theory is not the only tool we need. A method that offers inverse inference in an uncertain environment

is required. Through such a method, decision-makers can base their judgment on numbers and mathematical truths. Furthermore, it allows them to quickly investigate the impact of inputs to the evaluation, and, thus, to quickly check the versatility (or robustness) of their decisions. Robust decisions help mitigate the adverse effects of the first two challenges in technology selection.

While these three components each have shortcomings that are addressed in this dissertation, it is useful to consider the types of queries we would like to pose to the decision support system, to guide us in the technology selection decision-making process. These queries help to shape what the methodology should be capable of.

"Which is more beneficial for reducing QoI q, technology t_1 or technology t_2?" This is a typical forward query:

$$\text{argmin}_{t\in\mathcal{T}}\ q(S+t) \tag{1.1}$$

where $\mathcal{T}$ is the set of technologies, q is the quantity of interest (QoI) to be minimized, S is the baseline system and t a technology in $\mathcal{T}$ that is applied to S. The baseline system is always required, as it establishes a common ground with respect to which the technologies are compared. Furthermore, note that system means any system, but this dissertation solely focuses on engineered systems. In a deterministic setting, this query may be solved using standard optimization techniques. Even simpler, assuming each technology is quantified with fixed values, the quantity q is just computed for each technology t and the smallest is the answer to the query.

If multiple quantities of interest are considered, the targets on each QoI may result in a different technology t being optimal to reach those goals. Then, one can use multi-objective optimization techniques, such as weighted (linear) combinations of the objectives, or a Pareto search. However, we can also frame the problem probabilistically as:

$$P(t|\boldsymbol{q}) = \prod_i P(t|q_i) = \prod_i \frac{P(q_i|t)P(t)}{P(q_i)} \tag{1.2}$$

Now, each target $P(q_i)$ is a probability distribution rather than a fixed value and we can observe the posterior probability that a given technology satisfies that target. The conditional probability $P(q_i|t)$ is simply the result of a simulation that computes q_i with technology t present. Finally, $P(t)$ is $1/|\mathcal{T}|$, because we give each technology an equal chance.

"What change in the technology design variables $\boldsymbol{x}$ is needed for technology t_1 to have a significant effect on the outputs $\boldsymbol{y}$?" In general, if we have input variables $\boldsymbol{x} \in \mathbb{R}^N$ and output variables $\boldsymbol{y} \in \mathbb{R}^M$ and a model $F : \mathbb{R}^N \mapsto \mathbb{R}^M$, we want to compute $\boldsymbol{x} = F^{-1}(\boldsymbol{y})$. Therefore, this is an inverse query. In a Bayesian setting, we would like to compute:

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\boldsymbol{x}) \cdot P(\boldsymbol{x})}{P(\boldsymbol{y})} = \frac{F \cdot P(\boldsymbol{x})}{P(\boldsymbol{y})} \tag{1.3}$$

This equation is very similar to Equation 1.2. However, instead of expressing the probability of a technology given some distribution of the QoIs, here we compute the probabil-

1

ity of observing some distribution $P(\boldsymbol{x}|\boldsymbol{y})$ over the input variables $\boldsymbol{x}$ given some desired distribution $P(\boldsymbol{y})$ over the output variables $\boldsymbol{y}$.

"Which technology portfolio leads to most reduction in QoI q?" The difference between this query and the first is the fact that portfolios are considered instead of individual technologies. The challenge lies in that the set of technology portfolios is the power set of technologies: $\mathscr{P} = 2^{\mathscr{T}}$. That also means that the amount of portfolios is the power of two of the number of technologies: $|\mathscr{P}| = 2^{|\mathscr{T}|}$. We seek a solution to the combinatorial problem:

$$\text{argmin}_P \; \boldsymbol{J}(S_P) \tag{1.4}$$

where $\boldsymbol{J}$ is the objective function and S_P is the system after applying the technology portfolio. If a fixed target on $\boldsymbol{J}$ is set, then the problem becomes very similar to Equation 1.2. The only difference is that the technologies $\mathscr{T}$ are replaced with the technology portfolios $\mathscr{P}$. The first task is to find the set of feasible technology portfolios, i.e. those portfolios for which the technologies are both mutually compatible as well as fulfilling the constraint inequalities. The second task is to find those portfolios that realize the objective. Note the similarity between the probabilistic formulations of the first two queries and the fact that the third query is just an extension of the first. This suggests that a single framework could solve all three of these queries.

Such a probabilistic framework for technology evaluation and selection requires, in the ideal case, the following capabilities. It can:

1. Observe the system of interest (SoI). The SoI is an input and the framework should comprehend what the SoI's composition, behavior and function are.

2. Observe the technologies. Similar as with the SoI, the framework should be able to digest whatever technology it is presented with, and make sense of how it affects the SoI. The framework understands how technologies interact physically, with minimal prior knowledge encoded by experts. Thus, it should comprehend physics through learning autonomously.

3. Figure out how the technologies and combinations thereof affect the objective J. Obviously, one could naively compute all possible technology portfolios and then compute J for each of them. However, that becomes intractable for a moderate set of technologies, or when the simulation models are computationally expensive. Therefore, a qualitative sense of physics should enable the system to pinpoint which portfolios are worthwhile to simulate and for which it can make a reasonable estimate. Then, a quantitative sense of physics in the form of simulation models have to be mapped to the remaining portfolios in order to compute J for all of them.

4. Assign probability distributions to uncertain input variables. Based on an understanding of physics, much like human experts do, the framework can use past experiences to make assertions about a current problem. For example, when a novel technology introduces some variable for which no previous data exists, similar variables may be found by comparing the novel technologies with previous

technologies. Then, the data about those former technologies can be extrapolated to the novel technology, with some additional uncertainty.

5. Understand how to rank technology portfolios even when the objective J does not evaluate to a clear ranking. This actually is a ubiquitous problem in decision theory. In the end, the problem comes down to a preference of decision and policy makers, for example, whether they are risk-averse, risk-neutral or risk-seeking.

Some of these items are too far fetched and out of scope for this dissertation. However, it is worthwhile to consider what an ideal probabilistic framework for technology evaluation and selection should be capable of.

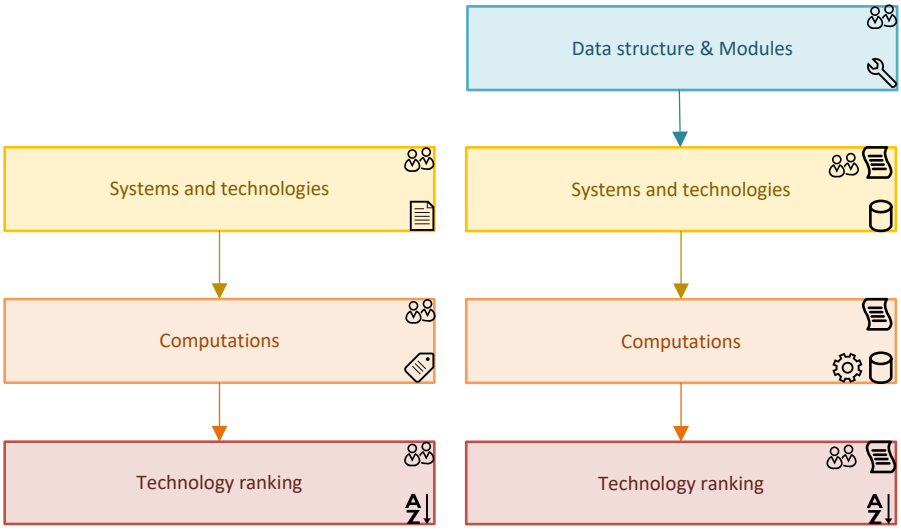
From the aforementioned three components — technology representation and portfolio generation, technology (portfolio) evaluation and technology (portfolio) selection — and ideal framework, the following three research objectives are distilled:

- **Establish a formal knowledge representation of engineering systems, such that technologies can be described.**
- **Specify, develop and demonstrate a methodology that enables automatic inference of technology (parameter) dependencies, using the knowledge representation established earlier.**
- **Specify, develop and demonstrate a methodology that estimates uncertainty distributions and subsequently quantifies uncertainty in the system, based on a given dependency structure, as well as support inverse uncertainty quantification.**

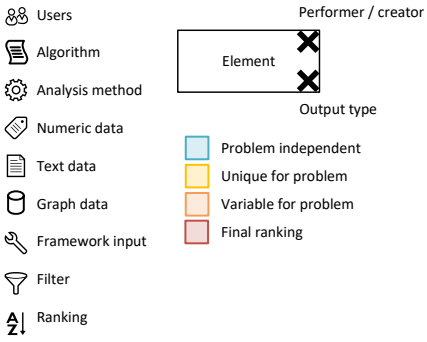
Let us compare how these objectives lead to a method that tackles the problems with the state-of-the-art approach. The state-of-the-art situation as sketched in Section 1.1 is notionally displayed in Figure 1.5(a). Systems and technologies are described with text and figures, and then analyzed using problem-specific analysis methods. The users have to manually map the technologies to the inputs of these analysis methods. In fact, most operations, except for the actual computations, are performed by humans.

This situation is contrasted by the method shown in Figure 1.5(b), which includes a problem independent portion that formalizes the data structure with which systems and technologies can be described. Moreover, analysis methods are included here as well, in such a way that they can automatically be applied to any system or technology. Note that the output types in this method are mostly in the form of graph data (which is explained in subsequent chapters) and most operations are performed by computer algorithms, rather than human experts. In summary, the method should bring two key benefits: knowledge formalization and automation.

1



(a) State-of-the-art (b) Proposed method



(c) Legend

Figure 1.5: Comparison between state-of-the-art method and present method ideal. Most significantly, the new method establishes problem-independent data structures, modules and knowledge that is reused. Furthermore, technologies and systems are represented with graph data structures, instead of text and figures. Finally, many parts of the technology evaluation and selection process are automated. The state-of-the-art is a summarization (and simplification) of the approaches introduced in section 2.1.

1.5. RESEARCH QUESTION

From the research objectives and the exposition of queries the framework has to answer, it does not become clear how these objectives are to be attained. In order to guide the research towards a suitable solution that satisfies the above research objectives, a main research question is formulated as follows:

> **How is an aircraft technology portfolio selected in a robust, consistent and traceable manner to minimize an objective function, given uncertain technology metrics, while including technology dependencies?**

Some of the terms used in this research question deserve further explanation. *Robust* implies that the method is applicable to any technology/configuration, produces the same result for the same analysis, and improves with new data. *Consistent* implies that technologies are represented unambiguously and when different analysis methods and assumptions are present, these are applied to all technologies equally. Finally, *traceable* implies that the method supports explicit knowledge capturing and sharing.

The main research question is broken down into the following sub-questions:

1. How to represent engineering systems and technologies consistently and robustly, allowing for knowledge capturing, reuse and sharing?

2. How to define dependencies between technologies, and how to characterize these dependencies based on the physical behavior of the technologies?

3. How to analyze (novel) technologies in a consistent, reliable and robust manner, such that their (combined) effects are characterized, with uncertain input metrics/parameters?

4. How should a ranking of a set of technologies or technology portfolios be obtained for multiple, conflicting, uncertain quantities of interest?

Each of these is addressed in one of the chapters in this thesis. Note that some chapters address multiple sub-questions.

Although originating from the MANTA project, this thesis research addresses the problem of representing technologies and quantifying their effects in terms of quantities of interest. The scope is widened to any complex system and novel technology, instead of specifically aircraft and movable technologies. The reason is that aircraft and the associated technologies are sufficiently complex that only a holistic approach is believed to tackle the indicated problems in a consistent, reusable manner.

1.6. THESIS OUTLINE

This thesis is structured to follow the steps in a technology evaluation and selection workflow. However, Chapter 2 first presents the prerequisite knowledge and background information that supports all elements of the following research chapters. Additionally, Chapter 2 presents a literature review of relevant research fields. How this knowledge is fed into the subsequent chapters is illustrated in Figure 1.6.

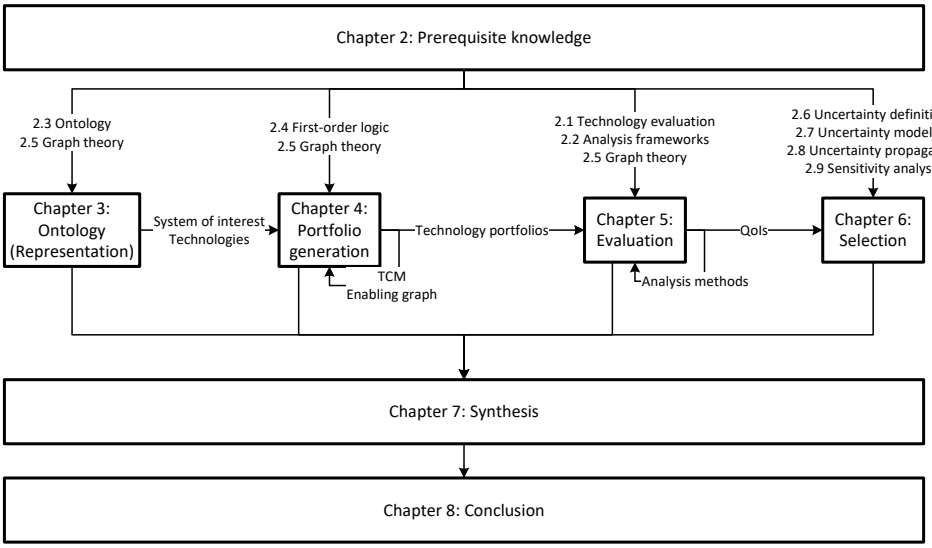


Figure 1.6: Thesis outline following the four steps of the method: technology representation, portfolio generation, evaluation and selection. The arrows indicate how knowledge from the chapters flows from one to another; in some cases including a specification of that knowledge.

An ontology to represent engineering systems and technologies is presented in Chapter 3. This ontology forms the knowledge representation that is used by the three subsequent chapters, as can also be seen in Figure 1.6.

Chapters 4 through 6 discuss the following three steps of the technology evaluation and selection process: portfolio set generation, portfolio evaluation and portfolio selection. Thus, Chapter 4 presents multiple methods to generate the technology compatibility matrix (TCM) in a structured, repeatable and traceable manner. The TCM is one of the elements used to reduce the set of portfolios, by excluding those which contain incompatible technologies. Furthermore, a method to deduce technology enabling is discussed, which further reduces the set of possible portfolios. Finally, an auxiliary technique — maximum dissimilarity — is presented to find a representative set of portfolios, such that only those would have to be analyzed, rather than all remaining portfolios after pruning through the TCM.

After the portfolios that have to be evaluated are generated, a simulation model should be constructed to compute the quantities of interest, based on the known input variables. Because each portfolio differs and different technologies may be captured with different parameterizations, a tailor-made simulation graph is constructed in Chapter 5 for each portfolio. Because some of the inputs may be uncertain variables, their dependency structure should also be indicated. This is a challenging task for engineers or decision makers non-versed in probabilistic simulations. Therefore, the sub-task of finding which pairs of variables have a dependency structure is automated by an inference mechanism, thus alleviating the effort.

Chapter 6 introduces probabilistic inversion (PI) as a means to use the data gener-

ated in Chapter 5 as to answer the various queries on the technologies and portfolios. PI is an alternative to Bayesian approaches and computationally very attractive, although it is a sample-based method. Another advantage of PI is that it works with any black-box simulation model. Most selection queries are of an inverse nature: given a goal for a quantity of interest, one wants to know a distribution over the inputs that satisfies that goal. PI is able to do exactly that, when both the outputs and inputs are random variables.

To synthesize the methods and ideas presented in Chapters 4 through 6, Chapter 7 presents how the proposed methodology may have been applied to the MANTA project. It also shows the limitations of the present framework and where opportunities lie for further research.

The thesis is concluded in Chapter 8 by answering the four research sub-questions, along with the main research question. Recommendations for improving on and extending the presented framework are exposed as well. Finally, an outlook towards future research is presented.

2

BACKGROUND

This chapter presents an overview of the several topics that this dissertation builds upon. They are briefly addressed in the following paragraphs. Then, the subsequent sections highlight each of these topics and explain them in more detail.

Ontology Human–machine communication is an active field of research and understandably so. Natural language is intrinsically ambiguous, which allows humans to communicate efficiently [33]. The ambiguous terms can be efficiently reused and are disambiguated based on context. However, natural language lacks the precision a computer requires to operate on (i.e. computers do not do well with ambiguity). Ontologies are conceptualizations of a portion of the world around us, and aim to dissect it into concepts and the relations between them. They provide a framework to formally capture knowledge and are machine-interpretable.

First-order logic First-order logic (FOL) is a formal language consisting of variables, relations and quantifiers to construct sentences that express propositions. There are different types of formal languages, of which zero-order logic or propositional logic is the least expressive. First-order logic extends propositional logic to include quantifiers and variables, making it more expressive and most commonly used. Typically, rules are established in FOL that construct clauses from atoms and define an implication direction between these clauses. For example: if x is a human, x has a brain. However, the concept of rules extends to other areas (such as graphs) as well.

Graph theory Graphs are used in this work as a means to represent knowledge about systems and technologies. Graph transformation rules are used to modify a system with a technology. Therefore, we need the graph theoretic concepts such as (sub)graph isomorphism and maximum common subgraphs. Furthermore, for creating computation graphs, we require the notions of (directed) cycles and topological sorting of graphs. Some graphs usually follow a certain schema, i.e. a type graph, which is induced by an ontology. Such graphs are called knowledge graphs.

Uncertainty quantification Because the quantification of technology qualities and impacts is subject to uncertainty, their evaluation should be conducted probabilistically. That is to say, the uncertainty has to be propagated through the simulation environment, to obtain uncertainty on the quantities of interest as well. There are multiple ways to perform such propagation, as well as models to represent uncertainty in the first place.

Before these topics are treated, however, a literature review of state-of-the-art technology evaluation and selection practices is provided in section 2.1. Consecutively, a literature review of analysis frameworks is presented in section 2.2. Ontology is discussed in section 2.4, followed by a description of first-order logic in section 2.5. Then, graph theory is briefly introduced in section 2.6, while uncertainty is discussed in section 2.7 through section 2.9. Lastly, sensitivity analysis is touched upon in section 2.10, after which the chapter is concluded.

2.1. TECHNOLOGY EVALUATION AND SELECTION

In order to make a well-informed decision as to which technologies have the most impact at the lowest risk, different methods can be employed. These are discussed below, however, in general such technology evaluation and selection is paired with uncertainty and as such this should be incorporated in the analysis as well [13].

Any technology selection methodology should exploit all data and information available on applicable technologies and extract useful information to reduce subjectivity in the technology selection process in a structured, repeatable and traceable way [32]. Addressing the benefit of possible novel and immature technologies during the conceptual design phase to understand impact on design and top-level requirements is important. Additionally, a maturity level needs to be addressed to keep overall acquisition cost in check and finally, computation time during design space exploration should be reduced. There is a trend towards using physics-based analyses, which are more expensive than semi-empirical models. A methodology is shown in Figure 2.1 that combines the Technology Readiness Level (TRL), compatibility matrix (TCM), Integration Readiness Level (IRL), sensitivity analysis and System Readiness Level (SRL). Uncertainty associated with new technology can usually be derived based on the TRL [34]. To associate TRL with uncertainty, the work of Kirby and Mavris [9] may be used. An elaborate characterization of technology readiness levels and technology integration is made by Jimenez *et al.* [8], who conclude that technology integration is part of technology readiness and should be accounted for as such. Integration of software in technology readiness is considered by Hantos [35], who proposes a TRL scale for this purpose. Additionally, SRL is stated as a more sophisticated measure, but the difficulties in assigning IRL are exposed, as Jimenez et al. [8] also pointed out. Besides the state-of-the-art (SOTA), business strategy and value of a project (and hence, technologies) should be taken into account when performing technology assessment and selection [36].

A common technique to account for the impact of technologies is through assigning a difference to a certain parameter that is present in the aircraft system. For example, the effect of an entire flap system can be represented by a change in $C_{L_{\max}}$ and subsystem mass. An example for a vehicle sizing environment is shown in Figure 2.2. The Technology Identification, Evaluation and Selection (TIES) [9–11] methodology, developed at

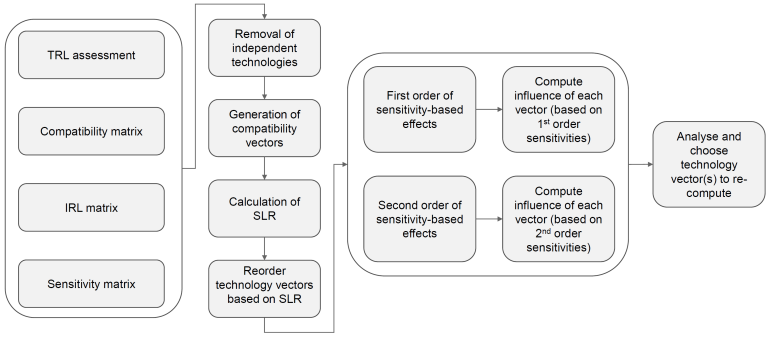


Figure 2.1: Flow diagram of technology investigation method to assess impact on aircraft measures of effectiveness [32]

Georgia Tech, works this way. Technology selection is done based on the highest probability of meeting objectives. Some improvements to TIES have been proposed, including a probabilistic evaluation of the technologies [10] and a bi-level optimization strategy to reduce computation time [37]. The impacts of technologies are called k-factors. The advantage of using k-factors is that key disciplinary metrics are taken into account and no commitment needs to be made to model a technology [13]. It is therefore a straightforward approach, and can quickly and easily identify the impact of certain technologies. However, subtle characteristics specific to a certain technology may not be taken into account and therefore certain effects may be overlooked.

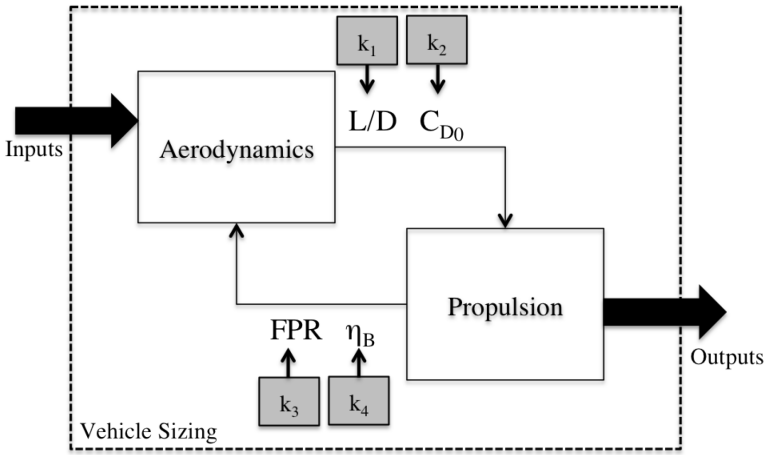


Figure 2.2: Example of k-factor mapping to a vehicle sizing environment [12]. The k-factors are k_1 through k_4.

It is important to incorporate information about the uncertainty of quantities of interest when making decisions based on them. Gatian [12] developed a technology portfolio selection process that takes into account the uncertainty associated with technol-

ogy impacts. Probability theory is used for modeling the uncertainty. Technologies are first evaluated individually, followed by a global sensitivity analysis to identify the most influential k-factors. Monte Carlo simulations propagate the impacts and uncertainties to system level and consecutively identify technology portfolio performance, as shown in Figure 2.3. When displaying the results, two metrics can be used for comparison: probability of success (POS) and signal-to-noise ratio (S/N). S/N represents both performance and uncertainty and therefore offers a dimensionality reduction [12]. The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) is used to create a ranking of portfolios. It is concluded that S/N is better to differentiate between clumped together portfolios, while POS is better when its values are not similar and a direct comparison of performance is required, which S/N is unable to provide. Alternative selection methodologies employ a Pareto front approach, which enable a posteriori articulation of preferences [38, 39].

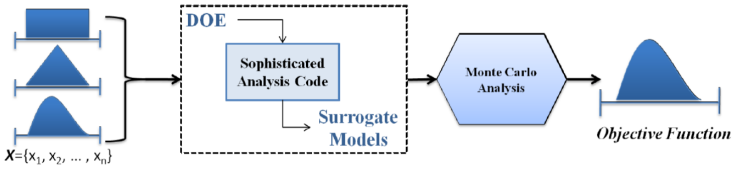


Figure 2.3: Monte Carlo uncertainty propagation method [12]. The inputs are the variables modified by the technology k-factors. The objective function is input to TOPSIS.

Epistemic uncertainty with regard to subsystem assessment and state-of-the-art (SOTA) assumptions can also be dealt with using k-factors, which are applied to sources of epistemic uncertainty and consecutively modified in a sensitivity study [14, 22]. In a related study, the method is applied to the assessment of more electric subsystems, comparing these to hydraulic and pneumatic subsystems. Among others, the environmental control system and control surface actuators are modeled and the benefits of electrification of these systems is investigated [15]. In terms of uncertainty, it is recognized that most modern techniques only take into account the variance of variables, while a distinction should be made between positive and negative uncertainty, which should be characterized by an asymmetric distribution [36].

A study conducted at Delft University of Technology used interval analysis (using the efficient global optimization (EGO) approach) to propagate uncertainties through a conceptual design tool, with the DAKOTA platform, and verified the method by a test case for a Wing Ice Protection System (WIPS) [40]. A practical technology evaluation study was performed by Heinemann *et al.* [25] to investigate the feasibility of meeting the goals set for a year 2050 tube-and-wing aircraft. Several technologies are discussed, including maneuver and gust load alleviation, a hybrid laminar-flow wing, variable camber and active flow control.

This section presented the state-of-the-art techniques towards technology evaluation and selection. Typically, existing analysis methods are employed to quantify the effects of technologies. Inputs to those methods are changed from the baseline through the use of so-called k-factors. This approach also allows aspects such as TRL and SRL to

be taken into account. A technology compatibility matrix is helpful to reduce the amount of possible combinations of technologies. The k-factor approach is simple to implement and operate, but lacks detail that novel technologies may require. Because the k-factors are strongly coupled to available analysis methods, the scope of technologies they can represent is limited.

Uncertainty with respect to technology impacts is taken into account in some studies, but the joint dependencies between technologies is not taken into account, while this is of importance, as will be discussed in subsection 2.8.3. Furthermore, only forward propagation of uncertainty is performed, whereas, as discussed in Chapter 1, inverse problems have to solved as well. This constitutes an opportunity for the present dissertation to complement existing research.

2.2. ANALYSIS FRAMEWORKS FOR NOVEL AIRCRAFT CONFIGURATIONS AND TECHNOLOGIES

The previous section focused on the evaluation and selection of technologies, but skipped over what frameworks have been developed to analyze the technologies. This section reviews what analysis frameworks are used in contemporary literature. Physics-based analysis and simulation are becoming more prevalent in engineering design, even during conceptual design. This is a result of the ever-growing complexity of the engineered systems. Especially when new technologies are applied, for which no existing data exists, we need to resort to physics to model and evaluate them.

Technology assessment of aerospace vehicles requires an integrated, multidisciplinary platform, with parametric geometry and sensitivity analysis [41]. Refs. [42–46] are examples of frameworks specifically created to investigate certain novel technologies. Such rigorous analysis improves the decisions that are made during early design stages, which usually allocate most of the resources of the entire design process [9]. Uncertainty quantification is more often employed as well, to make these decisions resilient to unknowns regarding technological developments, physical understanding or socio-economic situations [47, 48].

There is still a plethora of issues to be overcome before physics-based analysis becomes the norm, especially in early design stages. First of all, the application of design tools is limited to a specific vehicle type, because the sizing method is fixed [16]. Secondly, flexibility and scalability of disciplinary analysis tools is lacking [16]. Thus, there is a need for modular analysis frameworks, which allow for the analysis of widely varying system architectures and behaviors [49, 50]. Evaluating different technologies or combinations thereof requires all of them to be captured in a single computational framework.

Creating a flexible and modular design environment is challenging due to the coupling of operational and systems capabilities [17]. Another important issue for designing unconventional aircraft is geometry representation. As of now, no suitable geometry definition tool or CAD package is available that allows geometry to be reused from conceptual design into detailed design [18]. Some efforts were made to apply knowledge-based engineering to carry over geometry among different design stages [51].

Despite these challenges, several efforts have been made to arrive at a conceptual aircraft design framework, some including uncertainty quantification. A generalized

methodology for sizing unconventional aircraft and unconventional propulsion was proposed by Bucsan et al. [20], building on previous work [19, 21]. The work by Nam [21] includes uncertainty quantification in the form of the Probabilistic Aircraft Sizing Method (PASM) and recommends inclusion of evidence theory or the Bayesian approach into the method (these are discussed in section 2.8). It is remarked that sizing depends greatly on component model accuracy, which would be an area where uncertainty quantification can help.

The Design Engineering Engine (DEE) and Multi-Model Generator (MMG) are based around the idea of a holistic analysis framework where different analysis methods are coupled through one data structure from which multiple models can be derived (e.g. an aerodynamic model, a structure model) [49]. [52] presents a method with a Common Computational Model (CCM) that is intended to contain different aspects of objects at different levels of fidelity, as well as standardizing the interfaces between analysis and design models. Thus, the method appears very similar to the DEE and MMG.

Efforts for automated execution of analyses have also been conducted. Mainly, an M.Sc. thesis by Ramakers [53] and the work by Van Gent et al. [54, 55] are of particular interest. Both of these refer to the study of Pate et al. [56]. That study describes the construction of a Fundamental Problem Graph (FPG) to deduce relations between analysis tools and the variables of a Multidisciplinary Design Optimization (MDO) problem. From this FPG a Problem Solution Graph (PSG) is derived that defines how a particular MDO problem should be solved.

All in all, there is a need for modular, extensible analysis frameworks that employ physics-based analysis methods. However, currently, none of the presented frameworks satisfactorily captures a novel engineering system incorporating novel technologies and is then able to quantify its effects. This is because most analysis methods still focus on a specific vehicle type, as Lu et al. [16] points out. These issues are addressed in Chapter 5.

Apart from the physics-based analysis of a novel technology or system, most frameworks do not incorporate uncertainty quantification. And if they do, the same drawbacks apply as were observed in the previous section on technology selection methods: dependencies are not taken into account and inverse queries are not supported. These issues are addressed in Chapter 6.

2.3. STATE-OF-THE-ART

The state-of-the-art in technology evaluation and selection is discussed in the previous two sections. An overview of that discussion is presented in Figure 2.4. It is a more detailed version of Figure 1.5(a). The method takes in a set of technologies and a system of interest, which are combined into technology portfolios. These are subsequently described with impact factors, which are chosen among a set defined by the available analysis tool. The analysis tool then computes several quantities of interest, which are combined with qualitative selection criteria, if there are any. These criteria are then scored and weighted by a group of experts to produce the final technology ranking.

Notice how all the steps in this methodology are performed by the users of the method, except for the computation of the quantities of interest. Furthermore, the technologies, system of interest and technology portfolios are all represented in a textual format. Only the impact factors and quantities of interest form numeric data. Also notice how the

state-of-the-art method does not contain any steps that are problem-independent. Indeed, all steps have to be fully repeated for each technology evaluation and selection problem.

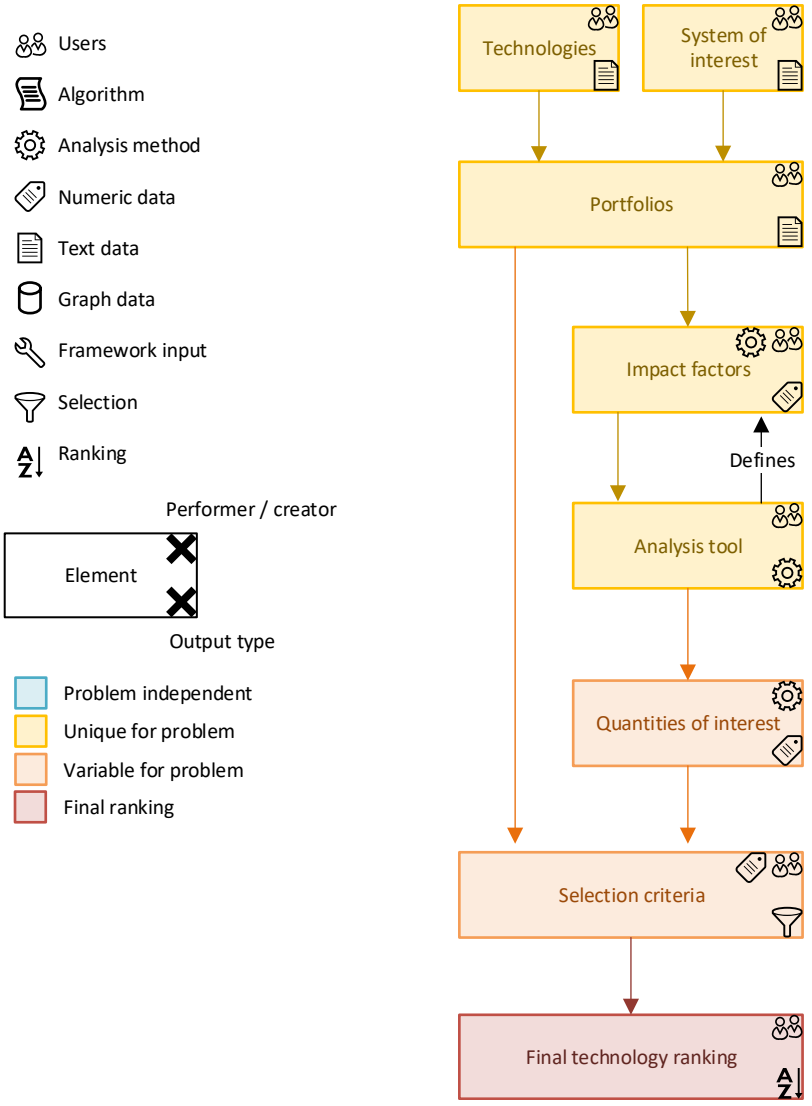


Figure 2.4: Overview of state-of-the-art method for technology evaluation and selection, including legend to explain symbols and colors.

2.4. ONTOLOGY

Ontology (uncountable noun) is a branch of philosophy that is concerned with grouping entities into categories or classes and stating various relations between those categories and entities. An ontology (countable noun) is a collection of such classes and relations, that is used to represent knowledge in a certain domain in a consistent manner. The reason why an ontology is needed in this dissertation is discussed in Chapter 3. It proves difficult to construct an ontology that captures any form of knowledge about any sort of entity. However, it is a useful endeavour to generalize the concepts in an ontology, such that they become domain-independent, and may be used and extended in other ontologies. An ontology that contains such generalizations is called an upper ontology.

While the concept ontology is a philosophical endeavor, it applies to computer science as well. In fact, ontologies are pervasive in the current web. Known as the Semantic Web, a set of tools and methods have been implemented that allow websites to present data in a semantic form, using ontologies. For example, Facebook uses ontologies to represent social networks and forms knowledge graphs with data about their users. The Semantic Web mainly consists of a data format, called the Resource Description Framework (RDF) and a language to specify ontologies, called the Web Ontology Language (OWL). RDF represents data in the form of triples (s, p, o), where s is the subject, p is a property or relation, and o is the object. Graph theoretically, such a triple forms a typed, directed edge (with s and o being the nodes, and p the edge type). Additionally, RDF contains specifications of the basic vocabulary, which constrains the types of subjects and objects, and relations. OWL extends RDF to capture more high-level concepts required to express ontological statements.

There are some concepts closely related to ontologies. The two most relevant ones are taxonomy and mereology. A taxonomy is a classification of concepts, into classes and subclasses, forming a hierarchical structure. An example is shown in Figure 2.5(a). Mereology is the study of parts and the wholes they form, see Figure 2.5(b). Where a taxonomy classifies discrete sets of concepts, a mereology offers universal statements, such as: airplanes have wings as parts. Mereology is an important concept in systems engineering, which mostly focuses on dividing systems into subsystems and components. An ontology may subsume both taxonomy and mereology, see Figure 2.5(c).

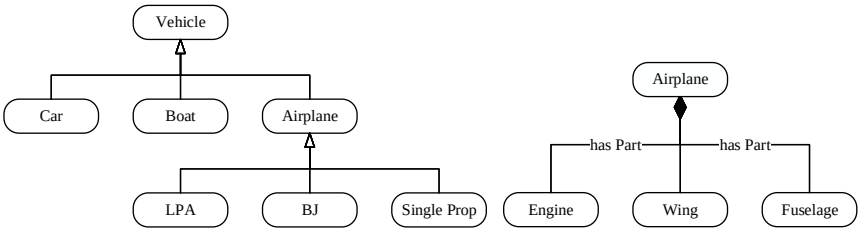
Facts represented in an ontology can be classified in two categories: universal and existential statements. These are also known as the TBox (terminological component) and ABox (assertion component). TBox statements typically apply to classes and are therefore universal for all individuals belonging to those classes. For example, all aircraft are vehicles:

$$\forall x \in \text{aircraft} : \text{vehicle}(x) \tag{2.1}$$

which is analogous to:

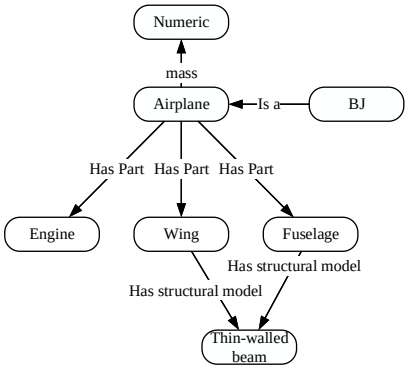
$$\text{aircraft} \subseteq \text{vehicle} \tag{2.2}$$

Conversely, ABox statements are TBox-compliant and are associated with instances (i.e. individuals or particulars) of classes. For example, the fact that the Wright Flyer is an aircraft is stated as: aircraft(*Wright Flyer*). An example of an ontology with both universal and existential statements is depicted in Figure 2.6. The Falcon 7X is a particular instance of the class of business jets (BJ), just as the PW307A is an instance of the engine class.



(a) Example of a taxonomy classifying types of vehicles

(b) Example of a mereology describing an airplane



(c) Example of an ontology describing an airplane. It combines elements from both the taxonomy and mereology in figures 2.5(a) and 2.5(b).

Figure 2.5: Examples of a taxonomy, mereology and ontology.

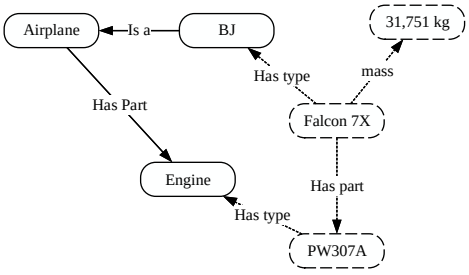


Figure 2.6: Example of ontology with particulars. Particulars and their relations are distinguished with a dashed line.

2.4.1. OPEN AND CLOSED WORLD ASSUMPTION

Equivalence is an important concept that enables categorizing individuals into classes. Deciding equivalence between entities is difficult, because it depends on context. Usually, a workaround solution is to introduce classes for each kind of entity in a domain ontology. However, that is labor-intensive and even then, an entity could belong to multiple classes and, therefore, a distinction between those classes is difficult to make, because without a proper definition a class is merely an ambiguous label. The main culprit is the Open World Assumption (OWA). Under the OWA, any statement that is not explicitly included in the knowledge base is neither treated as true or false, but rather as unknown. Thus, suppose one defines a large aircraft class as an aircraft having exactly four engines. If a reasoner is to infer that a particular aircraft is a large aircraft, because it has four engines, the following statements are necessary:

- Engines one to four are engines
- The aircraft is an aircraft
- The aircraft contains engines one to four
- Engines one to four are different individuals
- The aircraft only contains those four engines, and no others

If any one of these statements are missing, the OWA prevents inference of the fact that the aircraft is a large aircraft. Conversely, the Closed World Assumption (CWA) assumes that any statement that is not explicitly known is false. Additionally, the CWA includes the unique name assumption that implies any two individuals with different names are actually different individuals. Using the CWA, a reasoner would infer that the aircraft is a large aircraft even without the last two statements.

2.4.2. ONTOLOGIES FOR ENGINEERING SYSTEMS

Naturally, efforts to develop an ontology for engineering design have been conducted. An early effort towards this goal is the PhysSys ontology [57, 58]. It comprises three ontologies addressing the systems layout (component ontology), the physical processes underlying behavior (process ontology) and the descriptive mathematical relations (EngMath ontology [59]) [60]. The former two build on three ontologies describing mereology, topology and systems theory. Unfortunately, neither the ontology itself nor its documentation seem to be available. Recently, the Physics-based Simulation Ontology (PSO) is developed for the aforementioned goal [61]. It is based on Basic Formal Ontology (BFO) for ontological realism and the classical-mechanics view on physics to describe physical processes using partial differential equations. In their work, Cheong and Butscher [61] mention many related approaches mainly for computer aided design (CAD), computer aided engineering (CAE) and product lifecycle management (PLM). Štorga et al. [62] developed an ontology to capture product designs. However, the method fails to capture how these can be modeled and analyzed. Finally, there are several efforts to develop ontologies specific to a certain target domain, such as Ref. [42] for flexible part design.

In addition to modeling the physical reality of systems, an important aspect is the teleological view: their functioning. Kitamura and Mizoguchi [63] develop an ontology specifically to capture functional knowledge. As such, it may be used as an extension to ontologies like PSO, PhysSys and the one employed in this work. Other approaches focus on functional decomposition [64–69], which is mainly used in design synthesis. This allows automatic selection of components to fulfill certain requirements. Additionally, representing a system in terms of its behavior instead of a certain fixed parameterization gains popularity in model-based systems engineering [65, 70–73], which may be attributed to the formalization of behavioral descriptions. While these allow for a structured approach to describe systems and enable automated design synthesis, they all rely on human specified functions. Hence, physical realism is not achieved through these approaches.

Hirtz et al. [64] established a frequently used functional basis to describe the functions of systems and components. The functional basis should be interpreted as a taxonomy and although Hirtz et al. provide descriptions of each class, these are based on natural language and open to interpretation. A functional decomposition (FD) describing aircraft system architectures was developed by Judt and Lawson [65, 70], to consecutively enumerate system architectures and search for the best solution regarding some quantity of interest (QoI) with a hybrid heuristic optimization. Their FD is problem-specific, as is the analysis method, and therefore not easily generalized. The same holds for AirCADia [71], which breaks down the system description into a functional and logical domain and proceeds by mapping functions to means to arrive at a system synthesis. Sen et al. [67, 68, 74] use function–structure graphs to describe the behavior of a system enabling physics-based reasoning on it. Although effective, it appears to only be applicable to mechanical and electrical engineering domains, while continuum mechanics seem to pose a problem to this approach. The BeCoS tool [72, 73] uses an ontology to describe systems semantically rigorous in terms of their behavior. State-machines describe transitions within the behavior and, combined with equations, enable analysis of the system. An approach particularly aimed at capturing functional design knowledge with an ontology is presented by Kitamura and Mizoguchi [63]. It complements the device-centric approach PhysSys, because PhysSys has no ontology for functions from the teleological viewpoint.

2.4.3. QUALITATIVE PHYSICS

In contrast with ontological realism, qualitative physics is studied to capture causal relationships in physics without fully explicating them. For example, De Kleer and Brown [75] develop qualitative physics to describe, predict and explain the behavior of systems. Using causal analysis and teleological reasoning, based on the qualitative physics, electronic circuits are analyzed [76]. Here, function is defined as a causal pattern between variables. Qualitative physics relates structure to behavior, whilst teleology relates behavior to function. Other approaches include the process ontology [77] and bond graph theory [78, 79]. The latter is used in PhysSys to describe processes [58]. The representation methods chosen for naïve physics tended to simplify the details required in classical physics, while focusing more on the formalisms required for efficient reasoning [61]. Ideas from qualitative physics are used in this dissertation to identify causal relation-

ships between physical qualities, but not what these relationships look like.

2.5. FIRST-ORDER LOGIC

First-Order Logic (FOL) is a formal language consisting of variables, relations and quantifiers to construct sentences that express propositions. It is called first-order logic, because it includes the concepts of variables and quantifiers — something which zeroth-order logic, or propositional logic, does not. Higher-order logic allows predicates to have predicates as arguments, while FOL does not. FOL follows a very clearly defined syntax and semantics, making every sentence uniquely interpretable.

2.5.1. SYNTAX

There are two types of expression in FOL: terms and formulas. A term describes an object, or fact, while a formula expresses a predicate that evaluates to either true or false. The syntax of FOL is made up of symbols, which can either be logical or non-logical. The logical symbols always have the same meaning, while non-logical symbols depend on an interpretation.

LOGICAL SYMBOLS

A small set of logical symbols exist in FOL. They are the quantifier symbols, logical connectives, parentheses and variables.

Quantifiers There are two quantifiers in FOL: the universal quantifier $\forall$ and the existential quantifier $\exists$. The former can be read as "for all" and expresses some statement about all entities satisfying some condition. For example: the statement $\forall x \in PhDCandidate : x \in Human$ reads that all PhD candidates are humans. In contrast, the statement $\exists x \in Human : x \in PhDCandidate$ reads that there exists (at least one) human that is a PhD candidate.

The operator : should be interpreted as "such that". Thus, $A : B$ reads that the statement A is such that B is true. It can mostly be omitted, but helps to clarify some sentences.

Quantifiers can be nested to create more complicated sentences. However, the order in which quantifiers are used can significantly alter the meaning of the statement. Take, for example, the statement $\forall x \exists y : \text{Loves}(x, y)$. This implies that everyone loves someone. However, if stating $\exists y \forall x : \text{Loves}(x, y)$, the the meaning is changed to: there is someone who is loved by everyone.

Logical Connectives These are the logical connectives used in this thesis:

- $\Rightarrow$ or $\Leftarrow$: Implication. $A \Rightarrow B$ reads: if A then B, which is equivalent to $B \Leftarrow A$.
- $\Leftrightarrow$: Biconditional. $A \Leftrightarrow B$ reads: A if and only if B. This implies that A can only be true if B is also true. Hence, the implication goes in both directions.
- $\wedge$: Conjunction. $A \wedge B$ reads: A and B.
- $\vee$: Disjunction. $A \vee B$ reads: A or B.

- $\neg$: Negation. $\neg A$ reads: not A.

Sometimes, the $\equiv$ symbol is used. It implies a definition, which has the same meaning as the $\Leftrightarrow$ symbol.

Parentheses While parentheses are not required by FOL, they are used to make sentences more readable. Furthermore, in this thesis the : symbol is used frequently, which may be interpreted as: such that. For example, the formula $\forall x \exists y : P(x, y)$ reads that for each x there exists a y such that the relation P is true for the assignment of those two variables. Now, consider the example from before: $\forall x \exists y : \text{Loves}(x, y)$ and $\exists y \forall x : \text{Loves}(x, y)$. The meaning of either sentence can be clarified by inserting parentheses to obtain $\forall x(\exists y : \text{Loves}(x, y))$ and $\exists y(\forall x : \text{Loves}(x, y))$. In these latter expressions, the : symbol could have been omitted.

Variables Variables are also logical symbols in FOL, and usually are denoted using lowercase letters, such as x, y, z. Effectively, variables are placeholders for propositions. That is to say, a variable represents some object, but that object is assigned to the variable based on the formula expressing the context. The term Father(x) could either be about my father or your father, depending on who is assigned to x.

NON-LOGICAL SYMBOLS

The non-logical symbols are made up of predicates (or relations), functions and constants. These are dependent on the domain of application.

Predicates A predicate symbol has an arity or valence. A predicate is a relation that receives a number of arguments (which are terms) and evaluates them to either true or false. A predicate of arity 0 is effectively a propositional variable, meaning, for example, "It is raining". A unary predicate takes in one term; for example: HasFather(x). This extends to any n-ary form. A binary predicate could be HasSon(x, y).

Functions Similar to predicates, functions have an arity. In contrast to predicates, however, functions evaluate to another term. For example, Father(x) is a function that retrieves the father object of x. A binary function $f(x, y)$ could, in arithmetic, denote the summation of x and y. When the arity is zero, a function symbol is a constant.

2.5.2. FORMAL GRAMMAR

The formal grammar of FOL dictates that terms can be formed in two ways:

- With variables: any variable is a term.
- With functions: a function $f(\boldsymbol{x})$ is a term.

One can repeatedly apply these rules to obtain other terms. Alternatively, one can form formulas through one of the following five ways:

- With predicates: a predicate $P(\boldsymbol{x})$ is a formula.

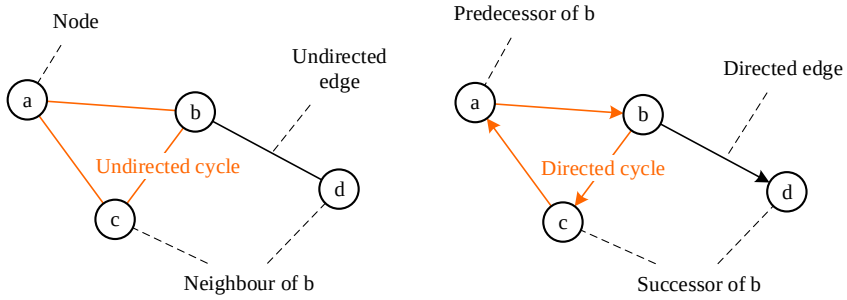
- Through equality: for the terms t_1 and t_2, the statement $t_1 = t_2$ is a formula.
- Through negation: a statement of the form $\neg A$ is a formula, given that A is a formula.
- With logical connectives: a statement $A \cdot B$, where $\cdot$ can be any of the aforementioned logical connectives, is a formula, provided A and B are formulas.
- With quantifiers: a statement $\forall x A$, or $\exists x A$ is a formula, when x is a variable and A a formula.

Again, these rules can be applied consecutively to form more complicated formulas.

2.6. GRAPH THEORY

Graph theory is the study of graphs, which are mathematical structures to capture pairwise relations between objects. Graphs are an extremely flexible data structure, and form the basis of the Semantic Web as introduced in section 2.4. This dissertation builds on graphs to represent engineering systems and on graph transformations to represent technologies. As such, graph theory is a key enabler for this dissertation. In this section, an comprehensive overview of graph theory is presented.

A graph is a tuple $G = (V, E)$ where V are the nodes (vertices) and $E \in V \times V$ the edges, each of which is a tuple (s, t) where s is the source node and t the target node, in the case of a directed edge. If the edge is undirected, there is no distinction between s and t (see Figure 2.7(a)). We can denote the source vertex of a certain edge $e \in E$ as $s(e) \in V$ and similarly for the target node: $t(e) \in V$. Let's define $v(e) = \{s(e), t(e)\}$ to be the collection of vertices connected by the edge e. For common edges, this collection is just the source and target nodes. However, hyperedges are undirected edges with more than two endpoints, i.e. $|v(e)| > 2$. Now let us define the following graph theoretic concepts.



(a) Undirected graph (b) Directed graph

Figure 2.7: Examples of an undirected and directed graph with some key concepts

Directed Graph A directed graph is a graph that contains only directed edges, as in Figure 2.7(b).

Neighbours Denote the edges connected to a certain vertex n as $E(n)$, such that $e \in E(n) \Leftrightarrow n \in v(e)$. Then the neighbours of n are all the nodes it is connected to: $\text{nb}(n) = v(E(n)) \setminus n$. An example of neighbours is shown in Figure 2.7(a), although it also applies to directed graphs.

Predecessors The notion of predecessors only exists in a directed graph. Denote the incoming edges $e \in E^-(n) \Leftrightarrow t(e) = n$. Then the predecessors of n are found as $\text{pred}(n) = s(E^-(n))$. See Figure 2.7(b) for an example.

Successors Similar to predecessors, successors only exist in a directed graph. Denote the outgoing edges $e \in E^+(n) \Leftrightarrow s(e) = n$. Then the successors of n are found as $\text{succ}(n) = t(E^+(n))$. See Figure 2.7(b) for an example.

Walk, Trail and Path (directed or undirected) A walk is a sequence of edges $(e_1, ..., e_n)$ which joins a sequence of vertices (v_1, v_{n+1}), such that $v(e_i) = \{v_i, v_{i+1}\}$. A trail is a walk in which all edges are distinct. A path is a trail in which all vertices are also distinct.

Cycle (directed or undirected) A cycle is a trail where the first vertex is the same as the last, and no further vertices are repeated. See Figure 2.7 for examples in both a directed and an undirected graph.

Directed Acyclic Graph Following the definition of a cycle, a directed acyclic graph (DAG) is a graph with only directed edges that contains no cycles.

Tree A tree is a connected acyclic undirected graph. A connected graph means that there exists a path between any pair of vertices in the graph. In a tree, there is exactly one such path for a given pair of vertices. A polytree is a DAG with a tree as underlying undirected graph.

Type Graph A type graph defines types of nodes and edges and specifies how those nodes may be connected by those edges. An ontology can be mapped onto such a type graph, allowing information adhering to the ontology to be represented as a knowledge graph.

Attributed Graph An attributed graph adds two elements to the tuple describing the graph: (A, pa). A are attributes, typically consisting of a name–value pair. We define $\text{pa}(a) \in V \cup E$ to be the parent element (either node or edge) of an attribute $a \in A$. With this definition, it is not possible to assign attributes to attributes.

Subgraph A subgraph is a graph that is part of another graph. Thus the graph H is a subgraph of G when $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G) \cap (V(H) \times V(H))$. We'll use the notation $H \subseteq G$ to indicate that H is a subgraph of G.

Induced Subgraph An induced subgraph H is a subgraph that is formed by a subset of vertices V_H from a graph G and all edges connecting pairs of vertices in that subset. Thus, $V_H \subseteq V(G)$ and $\forall e \in E(G) : e \in E(H) \Leftrightarrow v(e) \in (V_H \times V_H)$.

Bipartite graph A bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets, such that every edge connects a vertex in one subset to one in the other subset.

2.6.1. (SUB)GRAPH ISOMORPHISM

The notion of (sub)graph isomorphism is pervasive throughout this thesis and forms an important corner-stone of the proposed method. Two graphs G and H are isomorphic when there is a bijection between the vertex sets of G and H:

$$m : V(G) \mapsto V(H) \tag{2.3}$$

such that m is edge-preserving, meaning that $\exists f : E(G) \mapsto E(H)$ such that $\forall e_G \in E(G) : m(s(e_G)) = s(f(e_G)) \wedge m(t(e_G)) = t(f(e_G))$. For a type graph (or labeled graph) the morphisms m and f should be label-preserving as well. Define a label function $l_G : V(G) \mapsto \mathscr{L}$ that maps the vertices in G to a set of labels $\mathscr{L}$. Given such a function for H as well, we require $\forall v_G \in V(G) : l_G(v_G) = l_H(m(v_G))$. The same can be defined for edge labels. When the bijections m and f exist, G and H are isomorphic, denoted as $G \cong H$.

The graph isomorphism definition can be extended to include attributes as well, i.e. making m and f attribute-preserving. Define $b : A(G) \mapsto A(H)$ such that $\forall a \in A(G) : m(\text{pa}(a)) = \text{pa}(b(a)) \vee f(\text{pa}(a)) = \text{pa}(b(a))$. The function pa returns the parent of the attribute, i.e. either a node or an edge.

It is not common to have to find the isomorphism between two graphs. Instead, we are more often concerned with the subgraph isomorphism problem. The subgraph isomorphism problem states that given two graphs G and H, we seek to find if there exists a subgraph $G_0 \subseteq G$ that is isomorphic to H, i.e. $G_0 \cong H$. This is illustrated in Figure 2.8, where it also becomes clear that a certain graph can have multiple subgraph isomorphisms in another graph.

There are multiple ways in which the subgraph isomorphism can be solved. One is through a constraint satisfaction problem (CSP), which is explained in Appendix A. The intricacies of that approach are detailed in Appendix B, along with pseudo-code for the algorithm.

2.6.2. MAXIMUM COMMON SUBGRAPH

There are two types of maximum common subgraph (MCS): the maximum common induced subgraph, or the maximum common edge subgraph. Whenever a MCS is mentioned in this thesis, it is meant to be the maximum common induced subgraph. The MCS of two graphs G and H is a graph K for which there exists a subgraph isomorphism in both G and H, such that it is an induced subgraph in both graphs, with as many vertices as possible. This concept is illustrated in Figure 2.9. An example of a MCS between these two graphs is shown, but it should be noted that this MCS is not unique. Different node mappings are possible, leading to different, but equally valid, MCSs.

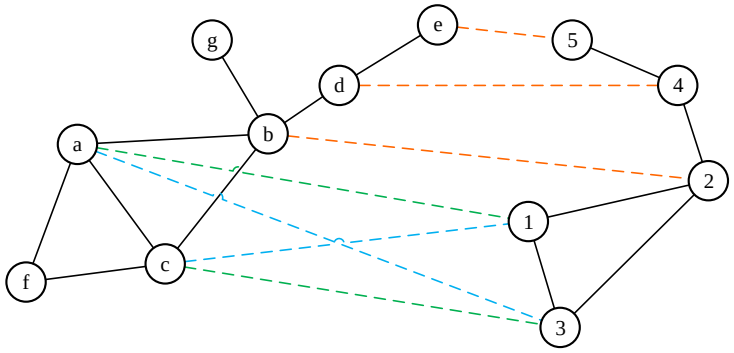


Figure 2.8: Example of subgraph isomorphism: the graph on the right has two congruent subgraphs in the left graph. The node mappings are shown with the dashed lines. The orange maps are the same for both matches, whereas the blue and green pairs are different solutions for those nodes. A full morphism is the combination of orange + green, or orange + blue.

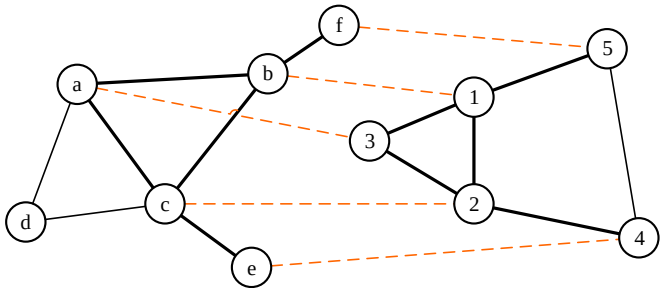


Figure 2.9: Example of a maximum common subgraph. The orange, dashed lines indicate the node mappings. The thicker edges indicate the induced edges that make up the MCS. Note that this MCS is not unique: node d (instead of node e) could also have been mapped to node 4, for example.

To find the MCS of two graphs, one can also employ a CSP, as is done for the subgraph isomorphism problem. However, it turns out one can also use the graph edit distance (GED) algorithm (see subsection 2.6.4 and Appendix C) and specify the edit costs in such a way that the edit path contains the MCS [80].

2.6.3. GRAPH TRANSFORMATION

Specifying changes to graphs formally is done with graph transformations. Graph transformations in the form of graph transformation rules are part of graph grammars, which have been around since the 1970s and have seen some use in design automation and synthesis [81–83].

A graph transformation $r : (L \leftarrow K \rightarrow R)$ is a construct consisting of three graphs: the pattern L, the gluing graph K and the replacement graph or effect graph R. The pattern L is to be matched inside a certain graph (the system in this case), while the replacement graph R replaces the matched instance of L, if it exists. Determining what elements to remove or add is done through the gluing graph K, which contains the corresponding nodes and edges of L and R. See Figure 2.10 for an illustration. Here, the three graphs

are shown as the letters corresponding to our notation. The dotted nodes and edges in the K are just there to complete the K, but have to actual meaning. Thus, K only consists of the nodes a, b and c, and the two edges between them. The node e is removed from the pattern L, along with the edge connected to it. Furthermore, the nodes f and g are added in the effect graph R, along with the three edges connected to them.

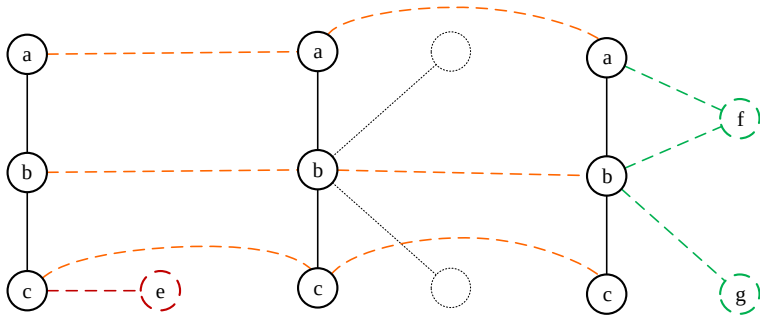


Figure 2.10: Example of a graph transformation rule. The orange dashed lines indicate mappings from the pattern L to the gluing graph K, and from that graph to the effect graph R. The dashed, red edge and node are removed by the rule, whereas the dashed, green nodes and edges are added by the rule. The dotted, black nodes and edges have no meaning, and are just there to complete the letter K.

Note that K can also be empty when L and R have no common substructure. In that case, all elements in L are removed from a graph along with any edges connected to the removed nodes. Then R is substituted in, but will have no connections to the rest of the graph, resulting in a disconnected graph. If K is non-empty, the difference between L and K, denoted by $L-K$, are the nodes and edges which are removed by the graph transformation. Similarly, the difference between R and K, i.e. $R-K$ are the nodes and edges which are added by the graph transformation.

The concept of graph matching is more formally explained using graph morphisms. Let G be the system graph that the rule r is to be applied to. When there is a match of the pattern L in G, there is a graph morphism $m : L \mapsto G$. A graph morphism m consists of two functions $f_V : V_L \mapsto V_G$ and $f_E : E_L \mapsto E_G$, such that $s_G \circ f_E = f_V \circ s_L$ and $t_L \circ f_E = f_V \circ t_L$ [84].

APPLICATION CONDITIONS

Application conditions (AC) are additional constraints placed on the pattern L that have to be satisfied in order for a match m to be found. Two types of AC can be distinguished: positive and negative ACs. A positive application condition (PAC) has to be satisfied for a valid match, while a negative application condition (NAC) prevents a valid match when it is satisfied. In theory, either AC can be any function over the host graph G. For example, it may express a constraint over the total number of matches of L in G, or ensure the absence of a certain graph element in G.

INDEPENDENCE

Graph transformation rules may interfere with one another when applied to the same graph. Therefore, two types of independence between these rules are defined: parallel

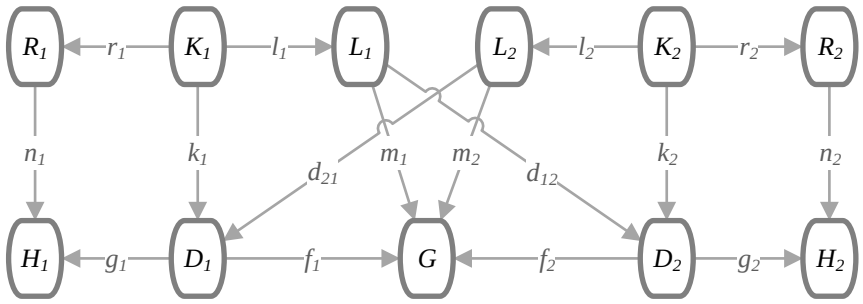


Figure 2.11: Parallel independent graph transformation rules. Adapted from [84]

independence and sequential independence [84]. The former implies that two transformations can be applied simultaneously. The latter implies that the transformations can be applied in any order and produce an identical end-result.

Parallel independence Refer to Figure 2.11. Two transformations $t_1 : G \overset{m_1}{\rightarrowtail} H_1$ and $t_2 : G \overset{m_2}{\rightarrowtail} H_2$ are parallel independent if there are morphisms $d_{12} : L_1 \rightarrowtail D_2$ and $d_{21} : L_2 \rightarrowtail D_1$, such that $f_2 \circ d_{12} = m_1$ and $f_1 \circ d_{21} = m_2$ [84].

While this definition by itself is hard to follow, it can be rephrased as: two transformations are parallel independent if neither creates or deletes something the other uses, or invalidates the application conditions of the other. Two rules are parallel dependent when one of the following conditions is true [84, p. 362]:

1. Delete–use conflict. Rule t_1 deletes a graph element that is in the match m_2.

2. Produce–forbid conflict. Rule t_1 creates graph elements in a way that violates a NAC of t_2.

3. Change–use conflict. Rule t_1 changes attribute values to something non-commensurate with the pattern L_2.

4. Change–forbid conflict. Rule t_1 changes attribute values and thereby violates a NAC of t_2.

If none of these conflicts occur for a pair of transformation rules, they are parallel independent.

Sequential independence Refer to Figure 2.12. Two transformations $t_1 : G \overset{m_1}{\rightarrowtail} H_1$ and $t_2 : H_1 \overset{m_2}{\rightarrowtail} H_2$ are sequentially independent if there are morphisms $d_{12} : R_1 \rightarrowtail D_2$ and $d_{21} : L_2 \rightarrowtail D_1$, such that $f_2 \circ d_{12} = n_1$ and $g_1 \circ d_{21} = m_2$ [84].

Thus, two rules are sequential independent if neither creates something the other uses, deletes something the other uses or creates, and neither creates or deletes something that validates or invalidates the application conditions of the other. They are sequential dependent when one of these conditions is met [84, p. 362]:

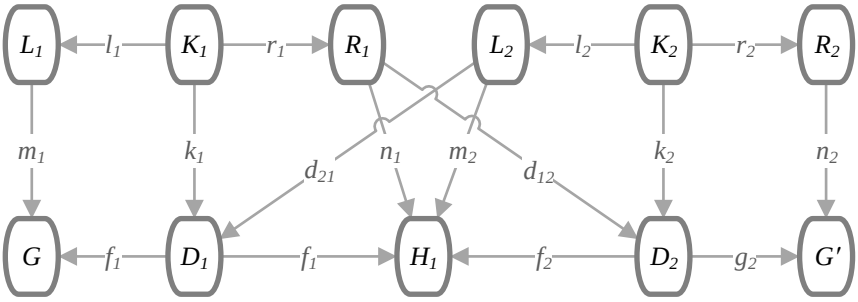


Figure 2.12: Sequential independent graph transformation rules. Adapted from [84]

- Produce–use dependency. Rule t_1 creates graph elements that are in the match m_2.
- Delete–forbid dependency. Rule t_1 deletes graph elements such that a NAC of t_2 is validated.
- Change–use dependency. Rule t_1 changes attribute values that are in the match m_2.
- Change–forbid dependency. Rule t_1 changes attribute values to validate a NAC of t_2.

When no such dependency exists, the two rules are sequential independent. Note that these dependencies are uni-directional, so t_1 can be sequential independent from t_2, while t_2 may sequentially depend on t_1. In Chapter 3 a stronger notion of sequential dependence is introduced: rule enabling. This notion not only indicates that there is a sequential dependency, but that the application of t_1 ensures that t_2 can be applied as well, while it first could not.

2.6.4. GRAPH EDIT DISTANCE

While graph isomorphism is a very strict definition of similarity between two graphs, it is sometimes more useful to have a measure of similarity. Graph edit distance (GED) is such a measure, and is often used in inexact graph matching. The idea is that a graph G can be transformed into another graph H through a series of graph edit operations, which collectively form an edit path. Each edit operation e_i has an associated cost $c(e_i) \geq 0$. Then, each edit path $p = (e_1, ..., e_n)$ has an associated cost

$$C(p) = \sum_{i=1}^{n} c(e_i) \tag{2.4}$$

and the GED is defined as the minimum cost of all possible paths transforming G into H:

$$GED(G,H) = \text{argmin}_{p \in \mathscr{P}} C(p) \quad . \tag{2.5}$$

The following edit operations are included in this thesis: vertex insertion, vertex deletion, vertex substitution, edge insertion, edge deletion, edge substitution, attribute insertion, attribute deletion and attribute substitution. Typically, the insertion and deletion cost for a certain graph element are equal and the substitution cost is either zero when the two substituted items are equivalent, or equal to the sum of the insertion and deletion cost if not. However, the edit costs depend on the application of the GED.

2.6.5. NODE, EDGE AND ATTRIBUTE EQUIVALENCE

Graph isomorphism, maximum common subgraphs and graph edit distance all have in common that we seek to find a correspondence between two graphs. For unlabeled graphs, any vertex can be matched to any other vertex, and edges can be matched when they preserve the structure of the graphs. The same extends to attributes. For labeled graphs, the label of a node, edge or attribute should equal the label of its matched counterpart in the other graph. Note that, while a label appears to be some string (a sequence of characters), it can be any complex piece of information. For example, it can define a node class. Furthermore, depending on that class, a different notion of equivalence may exist, such that certain vertices can be matched to one another, while others cannot. The same holds for edges and attributes as well.

Complicating matters further is that it depends on the application of the subgraph isomorphism or MCS problem how these forms of equivalence are defined. Let us clarify these issues.

- Hierarchy equivalence. Suppose we have a node in a pattern graph that matches any type of vehicle. Thus, its type is set to Vehicle. Now, any node in a host graph that has a type deriving from Vehicle, may be matched with the pattern node. The other way around, however, is not valid: when a node of type Vehicle is present in the host graph, it cannot be matched to a node in the pattern graph with a more specific type, e.g. Aircraft.

- Attribute equivalence. In several cases, a pattern has to match an attribute of a certain type, but does not care about the value. For example, we require a material with an electrical conductance property, regardless of its value. In other cases, the value does matter: a material with a specific density is sought.

- Geometry equivalence. Matching geometry is actually an open area in research, which has received attention from machine learning communities that view it as the next step after image classification. Congruence in geometry is independent of scale, rotation and translation, which complicates algorithms to find it. Furthermore, there is no single representation of geometry that assists in finding equivalence. On the contrary, the pervasive triangular mesh representation actually deteriorates notions of equivalence, because the mesh points and edges may be rather arbitrary.

Thus, depending on the problem, the appropriate form of equivalence has to be implemented in order for the graph matching algorithm to find the correct corresponding graphs.

2.6.6. KNOWLEDGE GRAPHS

An ontology defines the universals and relations that are present in a certain domain. Describing individuals adhering to an ontology can conveniently be done using a graph. Such a graph is called a knowledge graph. A knowledge graph contains entities as nodes and their relationships as typed edges. By having the ontology as a schema layer, logical inference may be applied on a knowledge graph to retrieve implicit knowledge that is not explicitly encoded in the graph. Knowledge graphs are commonly used nowadays by the large tech companies, such as Facebook, LinkedIn, Microsoft, Amazon, etcetera.

2.7. UNCERTAINTY DEFINITIONS

More robust and reliable design decisions may be made when uncertainty is included in the analyses [85]. Chakraborty and Mavris [86] confirms that statement in practice, by comparing electric system architectures in commercial aircraft to hydraulic and pneumatic ones, while including probability distributions on the technology parameters. Most research focuses on including uncertainty as probability distributions, but exclude dependencies among random variables [12, 13, 39, 47, 87]. Zaidi et al. [88] does emphasize the importance of dependencies and includes them in the form of copulas for aircraft technology assessment. They propose a structured form of querying experts to determine which copula[1] to use, to describe the dependency between two variables. However, that approach is not automated.

Uncertainty can be described in different ways. This section presents a common classification of uncertainty. Additionally, model uncertainty is looked at in more detail, since this is a form of uncertainty that is very important in technology evaluation. However, model uncertainty is hard to quantify.

Uncertainty can be defined as the incompleteness in knowledge and the inherent variability of the system and its environment [89]. A common distinction is the division into two classes:

1. **Aleatory** (or statistical) **uncertainty**, which can be seen as the inherent variation in variables. Other ways to define aleatory uncertainty is as type A or stochastic uncertainty. It is an inevitable, irreducible and uncontrollable form of uncertainty, but well identifiable.

2. **Epistemic uncertainty**, i.e. uncertainty due to lack of knowledge. It is also known as cognitive, type B, reducible or subjective uncertainty.

Additional classifications can be made to characterize the origin of uncertainty. Model form uncertainty, or alternatively model structure, non-parametric or structural uncertainty, model bias, model discrepancy or unmodeled dynamics, is associated with the ability of a model to accurately describe the physics involved. Model parameter uncertainty, or parametric uncertainty then stems from estimation of model parameters. An interesting subdivision is into parametric and parameter uncertainty, where the former is concerned with the inputs to the model (e.g. the design variables) and the latter with

[1] A copula is a mathematical function that presents an elegant method to define the joint distribution of two variables. It breaks that distribution up into a dependency structure and two marginal probability distributions of the variables.

the (fixed) parameters of the model. Finally, error and model form uncertainty can be classified as numerical or algorithmic uncertainty. Numerical error is added as a separate category by Oberkampf et al. [90, 91, 92]. Conversely, experimental uncertainty deals with uncertainty in measured data and estimation of parameters from populations of samples. Aleatory uncertainty and variability are examples of this classification.

2.8. UNCERTAINTY MODELING

The most well-known model for uncertainty is probability theory. However, there is a wide variety of different theories, for which Oberkampf et al. [90] provides an overview, based on the work of Klir and Smith [93] and Choquet [94]. These alternative theories include Dempster-Schafer theory, possibility theory, Bayesian probability theory and more.

Aleatory uncertainty is often modeled using probability theory, but representing epistemic uncertainty using probability is questionable since there is no reason to prefer one probability distribution over another [95]. A literature review on uncertainty quantification metrics for whole product life cycle cost in aerospace innovation is presented by Schwabe et al. [96], who indicate that the probability density function is still the most used metric, quantification of uncertainty is still largely subjective (i.e. expert judgment of uncertainty) and no commonly accepted cost estimation methodologies exist for research and development projects.

Bayesian theory is attractive, because it extends probability theory and enables us to specify a prior belief and consecutively update that belief through evidence. Both probability theory and Bayesian probability theory are discussed in the following two sub-sections.

2.8.1. PROBABILITY THEORY

Probability theory is a commonly used technique for representation of uncertainty, since it is relatively easy to implement and is well understood by engineers. A probability density function (PDF) is assigned to uncertain variables, which assigns a probability to each value the variable can attain. With sufficient data available, a PDF can easily be fitted. The PDF model can be chosen depending on uncertainty characteristics and its parameters can be estimated using the method of moments or maximum likelihood method. However, usually during conceptual design little information is available and the probability model has to be assumed by engineers (commonly uniform [97]), which adds to the uncertainty of the design results [98].

In aerospace applications, random variables are typically assumed independent, which can lead to inaccuracies and therefore poor decision making and flawed design. Additionally, poor input distributions of random variables may be assumed. Copulas theory can be used to mitigate both these problems [88]. When uncertain variables are dependent and the joint distribution does not follow a specific type, mixtures are an effective method to represent these joint distributions. Additionally, they are well capable of representing multimodality or tail characteristics, such as leptokurtosis or skewness [99].

2.8.2. BAYESIAN PROBABILITY THEORY

Bayesian theory (BT) is an extension of probability theory that includes evidence to support some hypothesis. Essentially, the theory revolves around Bayes' rule which is stated as follows [100]:

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)} \tag{2.6}$$

where $P(\cdot)$ is the probability of the contained statement, a and b are some variables and the operator | should be interpreted as *"given"* ($P(\cdot|\cdot)$ is called a conditional probability). The $P(b)$ is called the prior, i.e. an estimation of the probability of b without evidence, while $P(b|a)$ is called the posterior, i.e. the updated probability of b now that some evidence has become available. It's perhaps easier to interpret b as *cause* and a as *effect*, such that Equation 2.6 now gives a diagnosis; i.e. the probability of a certain cause given an observed effect. Bayes' rule is often helpful. The *cause→effect* is usually easily quantified, whereas the other way around usually is very hard to quantify [101].

A more general form of Bayes' rule is:

$$\boldsymbol{P}(Y|X,\boldsymbol{e}) = \frac{\boldsymbol{P}(X|Y,\boldsymbol{e})\boldsymbol{P}(Y|\boldsymbol{e})}{\boldsymbol{P}(X|\boldsymbol{e})} \tag{2.7}$$

where $\boldsymbol{e}$ is some background evidence. Using this formulation it is clear that Bayesian theory allows for revision of the probability of some condition when new evidence becomes available.

Although the Bayesian (subjectivist) view of probability has its merits over the classical (frequentist) probability theory, one of its largest drawbacks is through the "Principle of Insufficient Reason" [102]. Basically, the Bayesian approach relies on a complete probabilistic model of the domain, or in other words, a frame of discernment. This frame of discernment sometimes has to be chosen arbitrarily, while it has a major impact on the resulting probabilities. Therefore, the Bayesian approach needs to distinguish between uncertainty and ignorance [102]. A similar argument is given by Soundappan et al. [103]: if the evidence is imprecise, assumptions need be made to estimate the likelihood of the evidence. The posterior probability can be sensitive to these assumptions.

2.8.3. DEPENDENCY MODELING

Not all random events have an equal probability of occurring simultaneously. For example, if the ground is wet, the probability that it has been raining increases (given that knowledge). Therefore, the probability of rain has a positive correlation with the probability of wet ground. To encode such a relationship, we need the notion of a joint probability distribution. This is simply a probability distribution over two or more random variables. Note that a joint probability distribution (or just joint distribution for short) is neither the same as a correlation, nor the same as a causal dependency. These three concepts should not be confused. A correlation only measures similarity between certain trends in two sets of data. It does, however, often specify the structure of a joint distribution. A causal dependency relates two random variables through a cause-effect relationship, such as the ground–rain example. Such a relationship may be encoded using a joint distribution. Finally, a joint distribution represents the knowledge of how

probable the simultaneous occurrence of two (or more) events is. In other words, the probability of multiple statements being true (e.g. ground is wet = true and rain = true).

In this section, two methods to specify joint distributions are discussed: copulas and Bayesian networks. The former may be included in the latter to specify local joint distributions.

COPULAS

Copulas offer a general framework for specifying such multivariate distributions using any univariate marginals and a copula function C that specifies the dependency structure between the marginals [104]. The copula function C relates a set of real random variables $U_1, ..., U_N$ with standard uniform[2] margins as:

$$C(U_1, ..., U_N) = P(U_1 \leq u_1, ..., U_N \leq u_N) \quad . \tag{2.8}$$

The significance of copulas is a result of Sklar's theorem [105] that states that any multivariate distribution H can be represented as a copula function of its marginals:

$$H(X_1, ..., X_N) = C(F_1(X_1), ..., F_N(X_N)) \quad , \tag{2.9}$$

where $F_i(X_i) = P(X_i \leq x_i)$, i.e. it is the cumulative density function of the variable X_i and represents its marginal distribution. Because $F_i : X_i \mapsto [0,1]$, they are the U_i in Equation 2.8.

To specify a multivariate distribution H, a copula should be specified that captures the dependency as required. Because it has standard uniform margins, sampling from a copula is straightforward. Then, knowing for each variable X_i the CDF F_i and its inverse F_i^{-1}, each sampled variable U_i can be transformed into the values for X_i as follows:

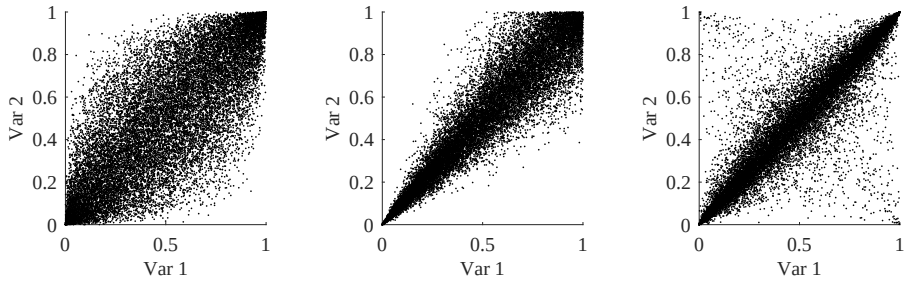
$$X_i = F_i^{-1}(U_i) \quad . \tag{2.10}$$

The copula of a set of samples of X_i can be obtained by transforming each X_i into copula space, i.e. its corresponding U_i. When the CDF F_i is known, this is trivial. However, when it is not known, it may be estimated through kernel density smoothing, for example. Alternatively, the ranks of the values in X_i can be computed and divided by the amount of samples in X_i plus one:

$$U_i \sim \frac{\text{rank}(X_i)}{|X_i| + 1} \quad . \tag{2.11}$$

Not any function is suitable as the copula function C. The ones that have been developed are roughly divided into two families: Gaussian and Archimedean. The Archimedean family contains, among others, the Clayton, Gumbel and Frank copulas. Some notional examples are depicted in Figure 2.13. It is not relevant for this text to further elucidate these copulas and their differences. However, they are so-called parametric copulas, because they are governed by a parameter that indicates the dependence strength. In the following, this parameter is called α and can be computed from common correlation coefficients, such as Kendall's τ.

[2] A standard uniform distribution is a uniform distribution on $[0,1]$.



(a) Gaussian copula (b) Clayton copula (c) t copula

Figure 2.13: Examples of three different copula families

BAYESIAN NETWORKS

Suppose you have a set of M discrete random variables. Each discrete variable has K values. Then the total amount of possible combinations equals K^M. Thus, the joint distribution over all these variables grows exponentially with the amount of variables, because for each combination of values we have to specify the probability of those values occurring simultaneously. In many practical situations, however, it turns out that the joint distribution over a set of random variables is rather sparse. Many random variables are independent, because there is no direct cause-effect relationship between them. Bayesian networks exploit this fact by including the variables as nodes in a DAG and only include directed edges between those variables that have an causal influence on another. Then, for each node, we only have to specify the local joint probability distribution, i.e. the distribution over the predecessors (parents) of a node and that node itself.

Figure 2.14 shows a very simply Bayesian network for the ground–rain example. The problem is modeled as the rain having a causal influence on the ground being wet. However, theoretically, we could have just as easily oppositely directed the edge. Mathematically speaking we would be able to perform the same inferences, even though the input data would be different.

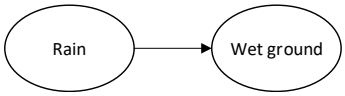


Figure 2.14: Example Bayesian network with two random variables

Bayesian networks rely on conditional probability distributions, which are a special type of joint probability distribution. Specifically, CPDs specify the probability of an event occurring given the knowledge that some other event has occurred (or not). In table form, see Table 2.1, this implies that each row sums to 1. This is because for each possible value of it raining (either true or false), the combined probability of the ground being wet or not should be one (there are no other options).

A discussion of inference in Bayesian networks is beyond the scope of this thesis. However, the interested reader is referred to Ref. [106], which provides an exhaustive

Table 2.1: Example of conditional probability table; T = True and F = False

	Wet ground	
Rain	T	F
T	0.9	0.1
F	0.3	0.7

introduction to Probabilistic Graphical Models (PGM), which is the overarching term for type of mathematical formalisms of which Bayesian networks are a part. Suffice it to say that Bayesian networks offer the possibility to reason in any direction. For example, we can state some evidence about whether it rains and compute the probability of the ground being wet. Conversely, we can enter that the ground is wet, and infer the probability that it has been raining. Therefore, BNs offer a powerful framework to perform reasoning in uncertain domains, which appears enticing for technology selection under uncertainty as well.

Particularly, one could create a Bayesian network with several discrete, binary nodes representing the technologies. They are either included (1) or not (0), with equal prior probability. Then, they affect the variables in the engineering system through a BN. The leaf nodes are the QoIs. By inserting evidence (goals) on the QoI variables, backward inference would give updated distributions on the technology variables. Then, each technology has an associated posterior probability of it being included or not, which should give a decision maker information as to which technologies have the most potential of reaching the set goal.

The drawback of the described approach using BNs is that they are not efficient with continuous variables. For discrete BNs, there are exact algorithms, of which some are even have linear complexity with respect to the size of the BN. For continuous BNs, however, this is not the case and inference has to resort to sampling-based methods. This is one of the reasons why the described approach is not employed in this thesis, but rather we use a technique called Probabilistic Inversion (PI) in Chapter 6 to achieve the same result.

2.9. UNCERTAINTY PROPAGATION

Different methods are available to propagate uncertainties of input, parameters and modeling to the output quantities. The characterizations and management of uncertainties are required at both the discipline level and integrated system level, such that also the relationship between uncertainties affecting input and those affecting output are involved [107]. Computational efficiency is the main challenge that is to be tackled by these methods. Most methods suffer from the so-called curse of dimensionality, i.e. with an increase in uncertain variables, the algorithmic expense grows exponentially. Additionally, the propagation method can introduce an error in the estimated uncertainty, for example by using too few samples, or simply because it assumes a certain function for the uncertainty. An entire framework for uncertainty quantification, encompassing all forms of uncertainty and error, has been introduced [108]. An example of industrial in-

terest in uncertainty quantification for the evaluation of complex physical systems is the DARPA EQUiPS (Enabling Quantification of Uncertainty in Physical Systems) program [109], which aims to develop a rigorous framework for the propagation and management of uncertainty in modeling and design of complex engineering systems.

Monte Carlo Simulations (MCS) remain a popular method to propagate uncertainty through a system, especially when a black-box system is used. The main reasons are the ease of implementation and the insensitivity to the dimensionality of the problem. The main disadvantage of MCS is the large sample size required to provide an accurate estimate, due to the Central Limit Theorem. To improve the accuracy of the estimation, i.e. reducing its variance, many variance reduction techniques were invented: antithetic variates, control variates, importance sampling, conditional Monte Carlo sampling and stratified sampling, for example Ref. [110]. A review of improved Monte Carlo methods in UMDO for aerospace vehicles is provided by Hu et al. [111]. Monte Carlo simulation appears especially suitable for uncertainty propagation in aeroservoelastic systems [112]. That conclusion may be generalized to apply for any tightly coupled system.

Because of the reasons mentioned before, MCS is the most appropriate tool to use in this dissertation for the propagation of uncertainty. It is used in Chapter 6 to quantify the uncertainty on the QoIs. It should be noted, however, that if only discrete variables are present in the technology selection study, a Bayesian Network could be used instead. The benefit of using a Bayesian approach is that inverse uncertainty propagation is also possible. Unfortunately, the Bayesian approach becomes computationally intractable for continuous variables, unless several assumptions and restrictions on the probability distributions (e.g. Gaussians) are satisfied. As a best-of-both-worlds solution, Chapter 6 introduces probabilistic inversion as a technique to enable the inverse queries, when using MCS for forward uncertainty propagation.

2.10. SENSITIVITY ANALYSIS

Sensitivity analysis provides design insight on a local level, or method insight on a global level. In a deterministic setting, local sensitivity analysis is usually performed to find the impact of a change in input variables on the output variables [113]. This results in partial derivatives of the output. Several techniques can be used to compute these derivatives, which are discussed below. Gradient-based optimizers make use of these gradients to guide a design solution to an optimal point.

In the context of design under uncertainty, input variables have a distribution of possible values, and global sensitivity analysis (GSA) is performed to investigate the effect on output variables with respect to the entire range of input values [113]. GSA can either be used before design to *screen out* those variables that have little influence (i.e. neglect uncertainty in these variables, to reduce dimensionality in the context of uncertainty quantification) and investigate the interaction between design and noise variables or it can be applied after design to determine where efforts should be made to reduce uncertainty.

2.10.1. LOCAL SENSITIVITY ANALYSIS

Different techniques exist for computing gradients, i.e. derivatives, of functions (or equivalently, systems). Symbolic differentiation (which is very similar to differential calculus) produces an exact derivative. However, for even moderately complex functions, the symbolic derivative can amount to several pages of expressions. Nonetheless it has successfully been applied in an aerodynamic shape optimization algorithm [114]. Numerical differentiation, i.e. finite differences or divided differences, is an approximate differentiation technique that is well known and can easily be applied to any function, no matter how complex. However, it only produces an estimate of the derivative and for functions with many inputs and outputs, the computational cost of this technique is considerable, if not prohibitive.

Lastly, automatic differentiation was introduced as a method that has neither of these problems: it produces an exact result at only a small computational cost. Automatic differentiation is in principle the application of the chain rule to computer programs [115]. The technique can either be applied in a forward-mode or reverse-mode fashion (or some hybrid combination thereof). The reverse-mode is closely related to adjoint differential equations [115].

Several papers discuss the principles of automatic differentiation, including forward and reverse accumulation, adjoint program construction and operator overloading, the if-else statement problem and iterative processes [116–119]. A robust optimization of an aircraft concept using automatic differentiation is presented by Su and Renaud [120]. These studies were all performed in the last century and recent research shows no hint of automatic differentiation being used, which may lead to the conclusion that it has lost popularity. A likely explanation is the issues involved with applying automatic differentiation to existing software, i.e. black-box systems.

2.10.2. GLOBAL SENSITIVITY ANALYSIS

GSA can be performed using various methods, but in general, a trade-off is made: accuracy of the solution versus computational efficiency. In order to take the entire probability distribution into account, which provides an accurate description of effects, GSA becomes computationally inefficient. In fact, for a large number of variables it becomes an intractable problem. On the other hand, simplifying the analysis by only considering certain statistical parameters such as variance, GSA can be performed, but with loss of accuracy and it remains computationally expensive. Another important observation is that most GSA techniques assume independence of input variables, which in many cases is warranted, but if not, different approaches should be used, such as copulas [113].

ANOVA (analysis of variance) is a commonly used method for GSA. It decomposes a function into its contributing components, for which the effects and variances can be determined [113]. From the variance, so called sensitivity indexes can be computed: the *main sensitivity index* (MSI), which describes the effect of one variable, and the *total sensitivity index* (TSI), which is the effect of a variable and all its interaction effects combined. ANOVA is used in a study by [121], where it is used to investigate the effect of component confidence intervals on the system confidence interval of Bayesian networks.

The advantage of variance-based sensitivity analysis is that it is easy to implement

and interpret, but it can not sufficiently describe uncertainty of systems with highly skewed or multimodal responses [122]. An alternative to ANOVA, but still a variance-based method, is FAST (Fourier amplitude sensitivity test) [123]. Alternatively, Sobol' indices [124] are anoter variance-based GSA technique. GSA based on Sobol' indices is used by Decarlo and Mahadevan [125] on an aerothermal problem, where an importance sampling-based kernel method is developed to estimate the indices. It allows for time-dependent multidisciplinary analysis of the sensitivity. A study by Chen et al. [113] develops an analytical variance based GSA method, by observing that many surrogate models are in the form of multivariate tensor-product basis functions, for which analytical solutions exist of the integrals needed to compute the sensitivity indices. So the modeling and simulation environment should be represented by metamodels for this to work, but it is shown that the method performs faster than Monte Carlo simulation and avoids sampling error. The Multidisciplinary Statistical Sensitivity Analysis (MSSA) [122] method is a relative-entropy-based SA technique proposed by Jiang et al. [122], which captures the entire distribution of a QoI and is therefore especially suitable for RBDO. Another method, based on Kullback-Leibler entropy, leads to the same conclusion [126]. High-dimensional model representation (HDMR) theory combined with ANOVA was introduced by Opgenoord and Willcox [127] to efficiently compute sensitivities of computationally costly models. The sensitivities are used to update risk and uncertainty budgets, based on which a design can be evaluated. Polynomial Dimensional Decomposition (PDD) can also employed for GSA and UQ of stochastic systems and a sparse representation can be obtained that results in fewer model calls [128]. For a short review of other GSA approaches, this work can also be consulted.

2.11. CONCLUSION

To accurately predict the impact of novel technologies, a generalized sizing and assessment method is deemed appropriate, but unfortunately such a method does not yet exist. This is partly due to the challenge of parameterizing geometry and using generic analysis methods. Another issue is the modeling at technology level. This step is usually avoided and replaced by introducing factors that account for a technology's impact on system parameters. Such approaches require expert knowledge as input. This is not necessarily a problem, although subjectivity is introduced and conservatism may result. The state-of-the-art methods for technology evaluation heavily rely on natural language for storing information, which has to be interpreted by human experts, which may lead to inconsistency. Furthermore, the k-factor approach heavily relies on available analysis methods to dictate what impact factors are available. This limits the expressiveness with which technologies can be modeled and likely omits some of their key characteristics.

Regarding uncertainty modeling and quantification, probability theory is still most widely used. Probability theory requires probability density functions as input, which have to be assumed by experts [98]. However, copulas theory appears promising to mitigate this problem. Another issue with probability theory is the single probability metric it produces, which may over- or underestimate the actual uncertainty and hence may lead to incorrect conclusions. Model form uncertainty remains difficult to quantify and most efforts focus on correction strategies using Bayesian calibration or multi-fidelity approaches. Both require high fidelity data, which may not be available or requires sig-

nificant computational effort.

High dimensionality of practical engineering problems remains an issue to be tackled. Technology evaluation algorithms, including optimization, uncertainty quantification, sensitivity analysis and surrogate modeling are commonly performed using algorithms that scale exponentially with the amount of variables. Techniques to reduce a problem's dimensionality are available, but are not effective enough to tackle this issue entirely.

It may be concluded that for technology evaluation and selection, there is not yet a holistic approach that includes a modular and flexible analysis environment, or uncertainty. Furthermore, there is no link between ontological modeling of engineering systems and the model-based approaches for simulation. If one is to combine these elements, uncertainty may best be modeled using probability theory. The most relevant is epistemic uncertainty regarding technology impact and specification, as well as readiness levels and state-of-the-art assumptions. Joint dependencies may be modeled using copulas, or with more elaborate techniques, such as Bayesian Networks. The propagation of uncertainty is easiest using Monte Carlo Simulation, because it is independent of the employed analysis method. Finally, sensitivity analysis may be used as a first estimate of the effects of technology impacts, either locally or globally, but serves little use for an accurate account of the quantitative impact technologies have.

The observations made in this chapter outline the developments that are made in the subsequent chapters. Concretely, the following chapter discusses how technologies may be represented and modeled using a formal ontology and graph theory. The chapters after the next use those representations to structure the technology evaluation and selection process, to make it more robust, consistent and traceable.

3

ENGINEERING SYSTEMS ONTOLOGY

As set out in the introduction, one of the first research objectives is to develop a formal knowledge representation of engineering systems and technologies. Specifically:

> Establish a formal knowledge representation of engineering systems, such that technologies can be described.

As we've seen in Chapter 2, ontologies offer a formal means to capture and represent domain-specific knowledge. Although it may be established that an ontology is a suitable solution for this research objective, the relevance of the research objective itself has not received attention, yet. In other words, why do we need a formal knowledge representation to start with? The reason is threefold:

- a formal knowledge representation enables us to capture and store knowledge, and
- to reuse and share that knowledge, and
- to ensure consistency and robustness.

Let's discuss each of these in the following three paragraphs.

Without an ontology, the engineering systems and technologies would have to be represented using some other method. Commonly, natural language is used. Extensive reports are drafted in a text editor, with some images to clarify the textual description. Those reports then serve as the knowledge representation. One can imagine that without a clear structure, such reports are hard to parse, and it is difficult to determine what knowledge actually resides within. Even when a consistent structure is followed, the knowledge itself is in the form of text and images, both of which require considerable interpretation to become usable information.

Although sharing a report is just as easy as sharing any digital document, the knowledge contained within is not so easily reused. Say some research project carries over from a prior one, and therefore some of the knowledge generated in the former project is used as a starting point. Typically, only part of that knowledge is relevant to the new

project, and in other areas that knowledge is incomplete, because of additional considerations in the new project. Thus, with knowledge not captured formally, one has to re-digest and reconfigure the existing knowledge to put it into a form that meets the new requirements.

Consistency of knowledge representation is perceived within this thesis as that the same knowledge would be represented in the same way at different points in time. Thus, suppose someone expresses today's temperature in some way in the knowledge base. Then, if tomorrow's temperature is recorded, it has to be done in an analogous way. Robustness is defined herein as a very similar concept, where it is meant that if there are multiple ways to express the same information, these have to be equivalent. That way, multiple users can use the same knowledge representation and avoid ambiguity or miscommunication through interpretation. Clearly, natural language does neither ensure consistency nor robustness.

Even though the preceding remarks presumed natural language as the ubiquitous knowledge representation technique in systems engineering, they still hold when other, more structured languages are used. For example, SysML or UML do a much better job at capturing knowledge about engineering systems formally, but do not provide a true ontological treatment of, at least, physics. Therefore, it is up to the modeler to come up with the foundational concepts that make up their descriptions, and as a result, ambiguity arises. Consider, in contrast, the period table of elements. This is a formal, rigorous basis of elements that are unique and can be combined to form other, more complex materials. However, the basis is fixed (to the extent that every now and then additions to it are found) and serves as a vocabulary to describe materials. It is, in a sense, the same as for mathematics. Mathematics is a language that was developed to be unambiguous: a conclusion formed from mathematical statements will always be true given those statements, and no contradicting conclusion can be drawn from those same statements. Mathematics is based on constructs, which are only virtual (they have no real meaning, per se), but form a basis from which all other laws and relations in mathematics follow. Numbers are the most prominent of these constructs. Conversely, if we consider functions of objects, it proves difficult to establish a basis or vocabulary of elementary functions from which any, more complex, function can be constructed. Even though some have been proposed [64], they are open to interpretation and debatable depending on your viewpoint. The issue with SysML and UML is that they do not include such exhaustive bases, and instead leave that to the practitioner.

Recall that knowledge adhering to an ontology can be captured as a knowledge graph. Graph matching requires a strong notion of node types and edge types. What graph structure is then equivalent to another? To answer that, a type graph is needed that specifies what kind of nodes and relationships are allowed. An ontology, although more widely applicable than graphs, has the exact same purpose. Furthermore, an ontology captures universal relationships between node and edge types, allowing for inference of new facts. A piece of information encoded as a graph is typically just a set of particulars (individuals), thus no general conclusions can be drawn from that without an accompanying ontology.

3.1. REQUIREMENTS FOR THE ONTOLOGY

Now that it is established why an ontology to represent systems and technologies is needed, it should be defined what knowledge needs to be captured, and which inferences the knowledge base should support. The answers to these two questions provide the guidelines on the development of the ontology.

3.1.1. WHAT KNOWLEDGE SHOULD THE ONTOLOGY CAPTURE?

It is impossible as of yet to propose a basis for describing physics and the artifacts that can be composed. This is because the actual building blocks of physics itself have not yet been found, and there is no unified theory of physics (gravity and relativity vs. quantum mechanics). As such, a decomposition as proposed in this research is only valid within a certain scope, which sets limits on certain physical quantities (such as velocity, in the context of classical mechanics versus general relativity). Any "law" that we may describe (e.g. Newton's first law, Maxwell's equations) are not in fact, a law of physics, but only hold true within a limited scope. Only when a unified theory is found, may a basis be developed along with accompanying laws that always hold true. When only focusing on the subset of physics that systems engineering is concerned with (disregarding the development of a Star Trek Voyager), it is perfectly reasonable to form a basis and accompanying laws that hold within the imposed limits. Such is the aim of this study.

We argue that to uniquely define a behavior, three elements need to come together: *why*, *what* and *how*. *Why* are the functions and requirements of the component that exhibits the behavior, *what* is the form of the component that are necessary to exhibit its behavior (which in turn fulfills the function) and *how* is a set of physical phenomena taking place in and around that component. Functions are a subset of the behavior (i.e. the part of the behavior that is intended), while requirements could be added through constraints on attributes associated with the behavior. The form of a component exhibiting the component is easily represented with geometry as a mesh, for example. However, determining equivalence of meshes and geometry in general is extremely difficult. Finally, specifying the physical phenomena involved with a behavior either requires an ontology of those phenomena, or requires specifying an analysis method that represents a subset of physics. In the latter case, different fidelity levels and assumptions in the analyses pose a challenge. Using the form (geometry) of the associated component in the behavior definition is expected to inhibit equality or equivalence determination of behaviors for this reason: comparing form is not a trivial task. Nonetheless, a behavior should be characterized by the form exhibiting it.

3.1.2. WHICH INFERENCES SHOULD THE ONTOLOGY SUPPORT?

Through inference, the knowledge captured using an ontology is leveraged to produce new knowledge. In our case, the ontology should support inferences that support the following two uses: technology portfolio generation and technology portfolio evaluation. The former requires knowledge of how technologies interact and what dependencies arise among them. The latter requires knowledge of how the entities in the ontology may be modeled numerically and consecutively simulated.

The interaction among technologies results from their physical behaviors. Therefore, as discussed before, capturing physical behavior properly is compulsory. The qualita-

tive interaction of physical behaviors has to be known as well, such that the interaction of technologies can then be inferred. The interaction between technologies defines two concepts: whether a set of technologies are incompatible with one another, and whether a set of technologies enables the application of another technology. These concepts require a notion of compatibility and a definition of what it means to enable another technology. The method for these inferences is discussed in Chapter 4.

Simulation models have to be captured by the ontology in such a way that their application to a given system and technology set can be inferred. Therefore, the ontology should offer concepts and rules to associate modelization to realistic entities and a description of simulation methods. How this is approached is discussed in Chapter 5.

Based on the preceding deliberation, this chapter aims to develop an ontology suitable to formally capture knowledge about engineering systems and technologies. It is based on existing upper ontologies: Basic Formal Ontology, Information Artifact Ontology and the Physics-based Simulation Ontology. These are discussed first, and the proposed additions are addressed next. Finally, the issue of granularity is touched upon.

3.2. BASIC FORMAL ONTOLOGY

BFO is developed with the idea of ontological realism, which aims to represent reality as it exists [61]. This allows a situation to be described as realistically as possible, without resorting to assumptions or abstractions of it. Models are described separately and point to such a physical reality. In general, ontological realism ensures perspectivism, which emphasizes that multiple perspectives of the same thing may be valid [129].

BFO divides entities into continuants and occurrents, see Figure 3.1. Continuants are entities that retain their identity throughout time, even though they may change. Occurrents are entities that unfold in time, or temporal regions.

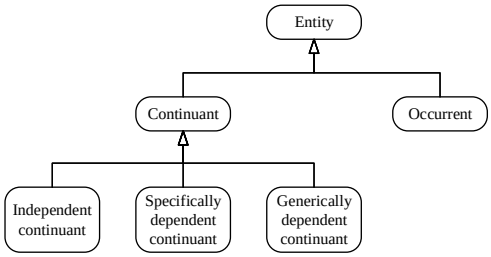


Figure 3.1: Primary classes in BFO, dividing entities into continuants and occurrents

Continuants are divided into independent, generically dependent and specifically dependent continuants, as shown in Figure 3.1. The first — independent continuants (IC) — are entities that exist irrespective of the existence of other entities and are further divided into material and immaterial entities, see Figure 3.2. Generically dependent continuants (GDC) are entities that have an identity, but do not exist concretely unless concretized by some IC. For example, a novel is a GDC, and is concretized by ink printed on paper. Multiple copies of the same GDC can, therefore, exist, and while all of these are different ICs, there is only one GDC: the novel itself. Finally, specifically dependent con-

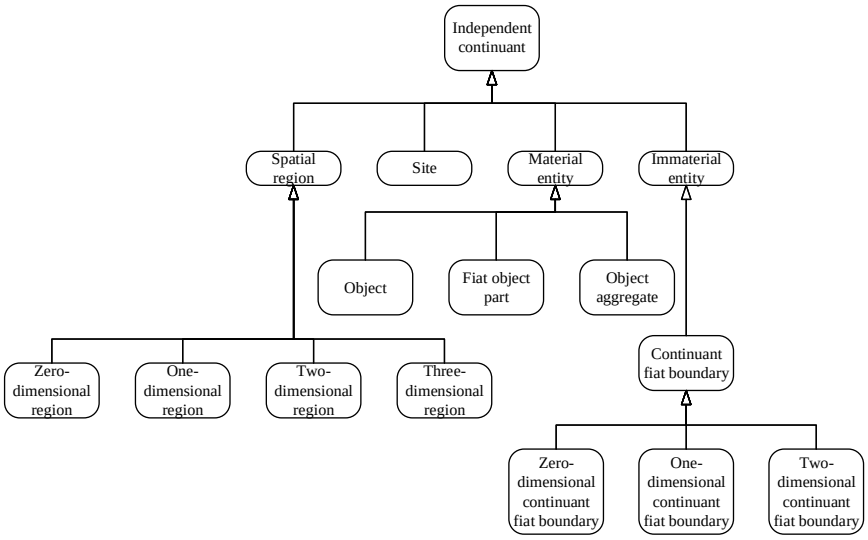


Figure 3.2: Taxonomy of independent continuants in BFO, extends Figure 3.1

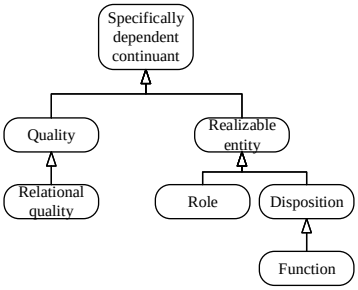


Figure 3.3: Taxonomy of specifically dependent continuants in BFO, extends Figure 3.1

tinuants (SDC) adhere in an IC and cannot be transferred to another IC (like a GDC can). In other words, they specifically belong to that IC. SDCs are sub-divided into qualities and realizable entities, which are roles and dispositions, as shown in Figure 3.3. Thus, the pressure of a portion of air is a SDC adhering in that portion of air (an IC).

Occurrents are (primarily) processes and temporal regions (see Figure 3.4). A process is an occurrent entity that exists in time by occurring or happening, has temporal parts, and always depends on some (at least one) material entity [129]. A temporal region, logically, is either a temporal interval (timespan) or instance (point in time).

All the classes present in BFO are depicted in Figure 3.1, Figure 3.2, Figure 3.3 and Figure 3.4. Refer to Smith [130] for a thorough description and definition of these classes. For a discussion on classifying material entities and processes, refer to Refs. [131, 132].

Table 3.1 lists the relations that BFO contains. Each relation has a domain, which is the set of classes that can act as subject to the relation, and a range, which is the set of classes of objects for that relation. Furthermore, Table 3.1 lists the reflexivity, symmetry,

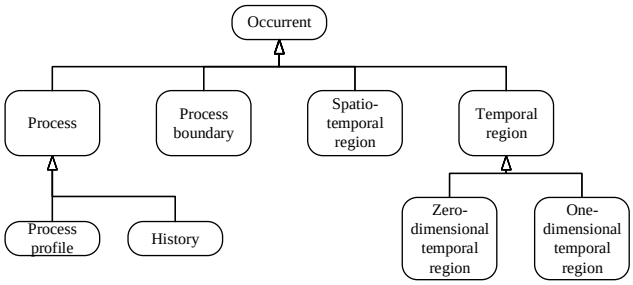


Figure 3.4: Taxonomy of occurrents in BFO, extends Figure 3.1

transitivity and the inverse for each relation. Reflexivity entails that an entity has a relationship to itself, i.e. x **relation** x. Symmetry indicates that x **relation** $y \Rightarrow y$ **relation** x. Reflexivity and symmetry also have inverses: irreflexivity and asymmetry. These mean that the negative of that property holds for the relation. Transitivity propagates a relation through a hierarchical structure; i.e. x **relation** $y \wedge y$ **relation** $z \Rightarrow x$ **relation** z. Finally, the inverse of a relation is simply its counterpart: x **relation** $y \Rightarrow y$ **inverse** x.

3.3. INFORMATION ARTIFACT ONTOLOGY

The Information Artifact Ontology (IAO) was created as a domain-neutral representation of information content entities (ICE), built on top of BFO [133]. It is supposed to be a mid-level ontology that describes information content entities, processes that consume or produce them, material bearers of information and relations with ICEs. The aim is to represent different ways that information relates to the world, and keep a clear distinction between information and what that information is about. As such, IAO is consistent with the realist ontology goal of BFO.

Figure 3.5 shows the taxonomy of classes in IAO, excluding some which are of less relevance to our discussion (denoted by an ellipsis). To illustrate how these classes are used, consider Figure 3.6. The flying laboratory of TUD — the Cessna Citation II with designation PH-LAB — is considered, and the information concerning its length is captured in IAO. The dashed lines and boxes indicate particulars (individuals), whereas the solid boxes are universals. As shown, the particular aircraft that is PH-LAB is an instance of the Cessna Citation II class. PH-LAB has a quality, which is its length. However, the information that represents that length is not directly captured in that quality. Instead, a measurement of that length stores that information, with a combination of a value and measurement unit. Additionally, this measurement may be concretized in some form. In this particular example, it is concretized by the solid state drive of my laptop. Specifically, the Floating-Gate MOSFETS that make up that solid state drive concretize the information. The part of the graph that describes the conretization of the measurement is not strictly necessary, and only serves as a demonstration.

At first, Figure 3.6 may seem too convoluted to state that the length of an aircraft has a certain value. However, after some consideration, this really is the least amount of information necessary to specify such a piece of information. The reason is that we want to be able to specify a certain quality (the length) of an object (PH-LAB), in such a

Table 3.1: Relations in BFO. The columns R, S, and T stand for Reflexive, Symmetric and Transitive, respectively. A • indicates that relation possesses the property, while a ∘ indicates it possesses the inverse of that property

Relation	Domain	Range	R	S	T	Inverse
bearer of		specifically dependent continuant				inheres in
has disposition	independent continuant	disposition				disposition of
has function	independent continuant	function				function of
has quality		quality				quality of
has role	independent continuant	role				role of
concretizes	specifically dependent continuant	generically dependent continuant				is concretized as
contains process	independent continuant	process				occurs in
has part			•		•	part of
has participant	occurrent	continuant				participates in
inheres in						bearer of
disposition of	disposition	independent continuant				has disposition
function of	function	independent continuant				has function
quality of	quality	independent continuant				has quality
role of	role	independent continuant				has role
is concretized as	generically dependent continuant	specifically dependent continuant				concretizes
occurs in	occurrent	independent continuant				
part of			•		•	has part
participates in	continuant	occurrent				has participant
realizes	process	realizable entity				realized in
realized in	realizable entity	process				realizes

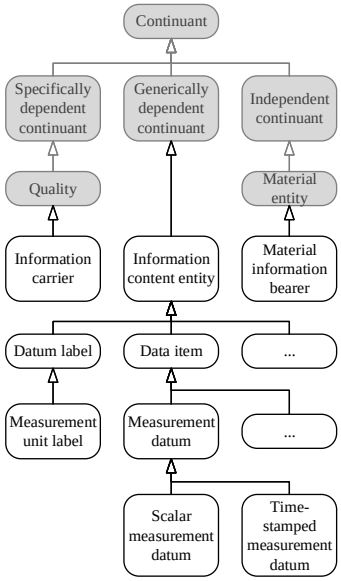


Figure 3.5: Taxonomy of entities in IAO

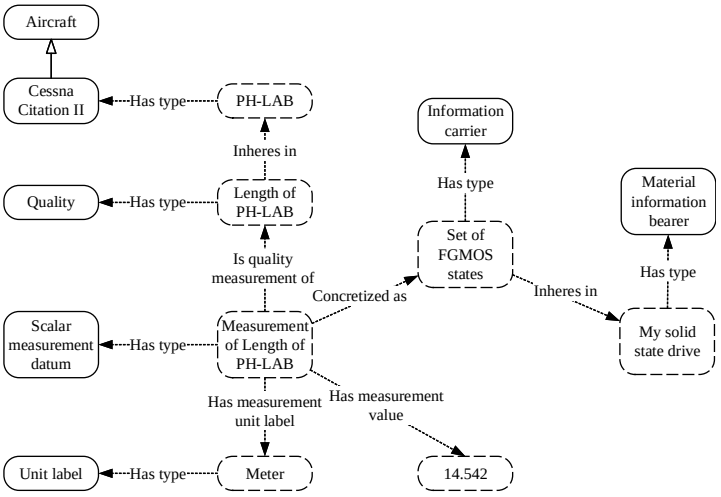


Figure 3.6: Example of usage of IAO to quantify the length of the PH-LAB aircraft. The part that indicates how the length measurement is concretized is not necessary, but demonstrates how that information could be specified.

Table 3.2: Relations in IAO. The columns R, S, and T stand for Reflexive, Symmetric and Transitive, respectively. A • indicates that relation possesses the property, while a ◦ indicates it possesses the inverse of that property

Relation	Domain	Range	R	S	T	Inverse
is about	information content entity					
denotes						
is duration of	time measurement datum	process				
is quality measurement of	measurement datum	quality				is quality measured as
is quality specification of		quality				is quality specified as
mentions						
is quality measured as	quality					is quality measurement of
is quality specified as	quality					is quality specification of

way that we can also capture what that quality is. That is why the "Length of PH-LAB" is a node in the graph. Although not shown in Figure 3.6, we can start adding relations to that node to indicate it is a spatial quality, and indicate the spatial dimension along which it is defined (e.g. from the nose to the tail of the aircraft). The measurement of the value of a quality has to be separate from the quality itself, because the quality's value could change over time. Even when it doesn't, it can be measured in different units. That is why the "Measurement of Length of PH-LAB" is a separate node, with relations to a measurement value and unit label. It's type indicates it's value is interpreted as a scalar value. This could alternatively be a vector, color, image or any other type of data structure.

The relations of IAO are listed in Table 3.2. The most important relation is **is about**. That relation implies that some information content entity **is about** some entity; hence, captures some information about it. Several specifications of this relation exist, as well as their inverses.

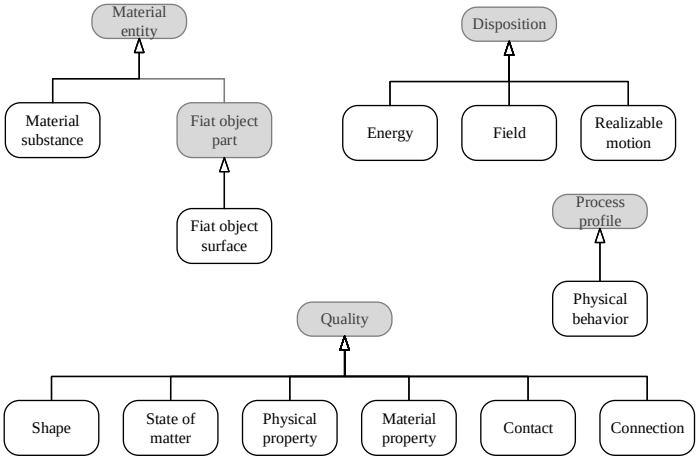


Figure 3.7: Taxonomy of classes in PSO-Physics

3.4. PHYSICS-BASED SIMULATION ONTOLOGY

The Physics-based Simulation Ontology (PSO) [61] was developed to "assist in modeling the physical phenomenon of interest in a veridical[1] manner, while capturing the necessary and reusable information for physics-based simulation solvers". PSO is divided into two ontologies: PSO-Physics and PSO-Sim. The former expresses classes and relations to model physical phenomena, whereas the latter expresses information artifacts about those physical phenomena, in order to model them in simulation software. Therefore, PSO-Physics mainly builds on BFO, while PSO-Sim extends IAO.

An overview of the classes introduced in PSO-Physics is shown in Figure 3.7. PSO adds to BFO the concepts of material substance and the **made of** relation, among other things. A material substance is defined as a chemical substance that a BFO material entity is made of [61]. It is inspired by DOLCE [134] and the material constitution theory [135].

Furthermore, the fiat object surface is added in PSO to denote a fiat object part of a BFO object that is minimal in one spatial dimension. Fiat object surfaces can be used to denote (a part of) the surface of an object, such as the outer surface of a wing, for example.

Qualities are divided into shape, state of matter, physical properties and material properties. Physical properties are defined as a BFO quality that determines the physical state of a material entity, and can be measured as quantitative values based on some measurement units. Material properties are defined as a BFO quality that can be measured to identify the physical characteristics of a PSO material substance. The distinction is made, because material properties can uniquely identify the type of material substance, while physical properties cannot. For example, density can uniquely identify a type of material, while pressure cannot.

Another PSO concept is the physical behavior, which is a subclass of the BFO process

[1] Truthful; veracious; accurate

Table 3.3: Relations in PSO. The columns R, S, and T stand for Reflexive, Symmetric and Transitive, respectively. A • indicates that relation possesses the property, while a ◦ indicates it possesses the inverse of that property

Relation	Domain	Range	R	S	T	Inverse
made of	material entity	material substance				
physically related to	specifically dependent continuant	specifically dependent continuant				

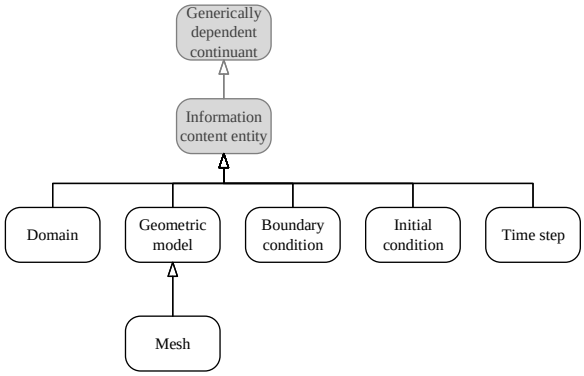


Figure 3.8: Taxonomy of classes in PSO-Sim

profile. [131] lays out the foundations of process profiles, which can be seen as specifications of a process. A process profile demarcates a part of the process that is identified for a particular purpose, much like fiat object surfaces are parts of an object with a certain purpose. The PSO physical behavior then is a process profile that is part of some process while occupying the same temporal region, and follows a specific law of physics [61].

PSO reuses many relations from BFO, and introduces two. These two are listed in Table 3.3. The **physically related to** relation specifies that two SDCs are related to one another through some physical law.

PSO-Sim is not described extensively in Ref. [61]. Therefore, the visualization of its taxonomy in Figure 3.8 is assumed by the present author, and is based on the textual description in Ref. [61]. There are no relations introduced in PSO-Sim. Instead, it makes use of the IAO relation **is about**.

3.5. EXTENDING BFO, PSO AND IAO

The proposed ontology is built with BFO, IAO and PSO as upper ontologies. Table 3.4 shows the classes that are introduced to the current ontology as subclasses of BFO. They are loosely based on works on functional decomposition [64, 67, 68, 74] and should therefore allow to define engineering components and how these interact. Components, fluids and solids are all material entities. A signal generically depends on a carrier and is

Table 3.4: Introduced classes, their superclasses and definitions, if applicable

Class	Superclass	Definition
Component	Object (BFO)	
Fluid	Material substance (PSO)	Gas ∪ Liquid ∪ Plasma
Gas	Fluid	Material substance ∧ **has quality** state of matter = gas
Liquid	Fluid	Material substance ∧ **has quality** state of matter = liquid
Plasma	Fluid	Material substance ∧ **has quality** state of matter = plasma
Interface	Continuant fiat boundary (BFO)	see Equation 3.1
Signal	Generically dependent continuant (BFO)	
Solid	Material substance (PSO)	Material substance ∧ **has quality** state of matter = solid
Boundary flow	Role (BFO)	
Inflow	Role (BFO)	
Outflow	Role (BFO)	

merely the interpretation of a physical quality. The notion of an interface is introduced, which is a continuant fiat boundary that has some dispositions. An interface occupies a spatial region. Examples of interfaces are connection points of pipes or cables, or the surface of an object.

$$\begin{aligned} \text{interface}(i) \equiv\ & i \in \text{ continuant fiat boundary} \wedge \\ & \exists r : i \textbf{ occupies } \text{spatial region}(r) \wedge \\ & \exists e : i \textbf{ part of } \text{material entity}(e) \wedge \\ & \exists d : i \textbf{ has disposition } \text{disposition}(d) \end{aligned} \tag{3.1}$$

Interfaces are used to express interactions between physical (material and immaterial) entities, through dispositions and functions defined on them.

Now, consider a 2D finite volume method. A portion of fluid is described with a control volume, bounded by some polygon, usually a triangle. The fluid flows through the three faces of that triangle, but depending on the actual direction with respect to that fluid element, it either flows into or out of that element. To support this description, several specific subclasses of BFO role are introduced: boundary flow, inflow and outflow. The boundary flow merely indicates that some entity is considered as a boundary to some other entity. Inflow and outflow are subclasses of boundary flow and self-explanatory. These roles do not apply to continuum mechanics specifically; in fact, they are valid for any control volume, where the in- and outflow of a system are of considera-

Table 3.5: Relations in present ontology, both taken from BFO (upper part) or introduced (lower part). The columns R, S, and T stand for Reflexive, Symmetric and Transitive, respectively. A • indicates that relation possesses the property, while a ∘ indicates it possesses the inverse of that property

Relation	Domain	Range	R	S	T	Inverse
assigns role		role				
blocking disposition of	disposition	disposition	∘			
inhibits	process	process	∘			
is derivative of	information content entity	information content entity	∘	∘		
is equal to	information content entity	information content entity		•		
inhibiting process of	process	process	∘			
overlaps			•	•		
occupies	material entity ∪ process	spatial region				
causally influences	information content entity	information content entity			•	

tion.

Table 3.5 shows the relations that are included in the proposed ontology. The relation **occupies** requires some elaboration. When m **occupies** r, it means that the material entity (or process) m is exactly located in spatial region r. Thus, equivalently one could state:

$$m \textbf{ occupies } r \Rightarrow \forall r_1 : \text{spatial region}(r_1) \wedge r_1 \textbf{ part of } r \wedge m \textbf{ occupies } r_1 .$$

The next few rules are necessary to reason about overlapping entities. Because the notion of overlapping is very general (e.g. sets overlap, spatial or temporal regions overlap), it applies to anything, and no class restrictions are included in the rules.

$$r_1 \textbf{ overlaps } r_2 \Leftrightarrow \exists r_3 : r_3 \textbf{ part of } r_1 \wedge r_3 \textbf{ part of } r_2 \tag{3.2}$$

Usually, the **overlaps** relationship applies to spatial regions, in which case CAD or similar software may be used to infer it. For now, a statement is directly added to the ontology for individuals that overlap, such that subsequent inferences can be made. Furthermore, any spatial region that is part of another spatial region, overlaps its parent:

$$r_1 \textbf{ overlaps } r_2 \Leftarrow \forall r_1, r_2 : \text{spatial region}(r_1) \wedge \text{spatial region}(r_2) \wedge r_1 \textbf{ has part } r_2 \tag{3.3}$$

Another spatial reasoning mechanism that is needed is that any process that is part of another process occurs in the same region:

$$p_1 \textbf{ occurs in } r_1 \Leftarrow \forall p_1, p_2, r_1 : \text{process}(p_1) \wedge \text{spatial region}(r_1) \wedge \text{process}(p_2) \wedge p_2 \textbf{ occurs in } r_1 \wedge p_2 \textbf{ has part } p_1 \tag{3.4}$$

A significant portion of domain-specific reasoning involves the interaction of processes. Users may wish to define that a process inhibits other processes of some specific type. This is most easily achieved by introducing a class of processes of which all individuals are inhibiting processes of some other class of process:

$$\forall p_1, p_2 : p_1 \textbf{ inhibiting process of } p_2 \Rightarrow \text{InhibitingProcess}(p_1) \wedge \text{InhibitedProcess}(p_2) \tag{3.5}$$

The relation **inhibiting process of** only informs that its operands have the potential to be inhibitory. However, the involved processes only actually inhibit each other when they occur simultaneously in the same spatial region:

$$\begin{aligned} \forall p_1, p_2 : p_1 \textbf{ inhibits } p_2 \Leftrightarrow\ & \text{process}(p_1) \wedge \text{process}(p_2) \wedge \\ & p_1 \textbf{ inhibiting process of } p_2 \wedge \\ & \exists r_1, r_2 : \text{spatial region}(r_1) \wedge \\ & \text{spatial region}(r_2) \wedge \\ & p_1 \textbf{ occupies } r_1 \wedge p_2 \textbf{ occupies } r_2 \wedge \\ & r_1 \textbf{ overlaps } r_2 \end{aligned} \tag{3.6}$$

Dispositions are realized in processes. When two dispositions are realized by inhibitory processes, these dispositions block each other. Therefore, we borrow the idea of a blocking disposition from [136, 137]:

$$\begin{aligned} \forall d_1, d_2 : d_1 \textbf{ blocking disposition of } d_2 \Leftrightarrow\ & \text{disposition}(d_1) \wedge \text{disposition}(d_2) \wedge \\ & \exists p_1, p_2 : \\ & \text{process}(p_1) \wedge \text{process}(p_2) \wedge \\ & p_1 \textbf{ realizes } d_1 \wedge p_2 \textbf{ realizes } d_2 \wedge \\ & p_1 \textbf{ inhibits } p_2 \end{aligned} \tag{3.7}$$

The relation **causally influences** indicates that the one physical quality causes a change in the other, according to some law of physics. This relation is not symmetric, although it is transitive.

Finally, the **assigns role** relation is introduced between a role some entity. It is meant to clarify which entities assign a certain role. For example, a person may both have the roles of manager and friend. To some colleague, they are a manager, while to their friends they are merely a friend, and not seen as manager.

3.6. TECHNOLOGIES AS GRAPH TRANSFORMATIONS

Consider the definition of technology presented in Chapter 1, repeated here for convenience:

> A technology is a materialized form of knowledge applied to a given system in order to alter the system's form or behavior to satisfy certain requirements.

Combined with the idea of representing a system with a knowledge graph and the concept of graph transformation rules, it is concluded that a technology may be represented as a graph transformation rule that alters the system graph to reflect a change in the form or behavior. The requirements to be satisfied can either be left implicit, to be determined during simulation of the altered system, or explicated by stating some constraint on a quantity of interest. Simulation then has to prove that the technology indeed satisfies its imposed requirements.

Recall from subsection 2.6.3 that a graph transformation rule consists of a pattern graph, an effect graph and a gluing graph. In the case of a technology, the pattern graph specifies the part of a system that the technology applies to. For example, it could describe a turbine blade, without being specific on the actual shape of the blade. It could, however, be specific on the type of material the blade is made of. Then, the effect graph could introduce holes in the blade surface to introduce film cooling. The gluing graph would in this case be the same as the pattern graph, because no elements of the turbine blade are removed; even though material is removed, the graph description only notices that as a change in the shape attribute of the solid that makes up the turbine blade. Furthermore, the blade retains its behaviors and functions, and only receives more in the effect graph.

At first glance, the idea of a graph transformation rule might seem to limit the scope of which technologies can be modeled. On the contrary, the converse is true. While, as Chapter 1 points out, a baseline system is always considered for good comparison of different technologies, the scope of that baseline system can be as wide as required. Essentially, the baseline system is the largest common divisor of the technologies under investigation. So, even if the technologies are disjoint things from between one must be chosen, there will be a system to which they are applied to fulfill a certain function. On another note, the fact that a graph transformation rule can delete anything from the pattern and add anything in the effect graph, makes them very expressive. The change to a system can be as small as modifying one property's value, to replacing the entire system with a different one.

As we will see in the following chapter, graph transformation rules allow to automate the technology compatibility matrix generation and other relationships between technologies. This results from the fact they are a formal language to specify what a technology is, in contrast to textual or graphical documents as used in common practice. After the application of a technology graph transformation rule, a new system graph is formed. These can be analyzed as any other system graph, as is discussed in Chapter 5. Multiple graph transformations can be applied to the same graph, allowing multiple technologies to be included simultaneously. Therefore, automatically analyzing a large set of technology portfolios becomes possible.

3.7. GRANULARITY

One of the largest unresolved challenges with a proper ontological representation of systems is that of granularity. In this case, granularity refers to the different levels of decomposition of a system. For example, at the highest level, an aircraft is merely an object that transports a certain payload from one place to another, through the air. However, one level lower, it can be broken up into the largest components that make up an airplane: the fuselage, wings, engines and tailplane. Subsequently, these can be broken up again, on and on, to a quantum level if one so desires.

Now, the right level of granularity would depend on where a technology makes its impact. If a technology entails a new coating for a turbine blade, a gas turbine would have to be described in such detail that turbine blades come into the picture. However, not all parts of the gas turbine have to be described at that level of granularity: the compressor and combustion chamber are of no concern for this technology.

The question now arises how one can traverse the various levels of granularity and still conclude the descriptions are about the same object. Concretely, when one describes a technology as a graph transformation rule, the pattern of that rule is to be matched in some knowledge graph describing an engineering system. If the level of granularity in the pattern differs from that in the system graph, how can the pattern matching algorithm detect they are still congruent?

One option is to convert all graphs to the highest level of granularity, i.e. the least informative description. Because this is a mapping from a high-dimensional space to a low-dimensional space, we can always perform the operation, but lose information in the process. To perform the mapping, the mereological part of the ontology is used, i.e. the **part of** relations. All sub-components are disregarded, and their behaviors are combined into a single, all-encompassing behavior. If we were to do this in the turbine blade example, the technology pattern would be simplified to a gas turbine as a black-box component that converts chemical energy into a thrust force (mechanical energy). That would match the high-level-granularity system description, which represents an aircraft broken down into its main components, which are only described in terms of their most top-level function and behavior. Unfortunately, it is now impossible to tell how often and where the technology can be applied to the system, because nowhere the amount of turbine blades is defined.

Therefore, another option is to convert graphs to the lowest level of granularity present among the knowledge graphs considered. Because this operation has to introduce new information, assumptions have to be made. For example, the high-level description of a gas turbine has to be expanded into one where all its sub-components are detailed, up to the turbine blade level. Some default configuration of a gas turbine should be assumed, with prescribed amounts of compressor and turbine blades, combustors, etc.

Alternatively, practitioners could be notified when the expert system detects conflicting levels of granularity. They can subsequently determine the appropriate course of action depending on that specific situation. Otherwise, a measure of the similarity between graphs may be employed that reflects their semantic equivalence. Chapter 4 explores this option in the form of graph edit distance.

3.8. DISCUSSION ON BFO AND PSO

This discussion is an opinion on some of the modeling choices in BFO and PSO. BFO introduces process profiles to specify aspects of a process. It is argued that a process cannot bear qualities. Therefore, to be able to assign measurement data to a process, one has to instantiate universals that specify a certain value for a quality the process affects. The downside of that is a rather convoluted set of process profile universals, for example (from [131]):

- speed profile
  - constant speed profile
    - ◇ 2 mph constant speed profile
    - ◇ 3 mph constant speed profile
- acceleration profile (increasing speed profile)
  - constant acceleration profile
    - ◇ 32 ft/s^2 acceleration profile
    - ◇ 33 ft/s^2 acceleration profile
  - variable acceleration profile
    - ◇ increasing acceleration profile

For each and every possible value of the speed or acceleration, one needs to define a process universal! Furthermore, Smith [131] argue that continuously changing values of qualities must be represented in a discretized manner. This is a result of the finite set of classes one can create for process profiles. This dissertation's author, therefore, argues against this approach. It appears best to avoid creating these process profiles, and rather adopt the PSO approach to use physical behaviours that reflect physical laws. A behaviour is then associated with the physical laws it is based upon, such as Newton's law, the laws of thermodynamics, Navier-Stokes equations, Maxwell's equations, etc.

Let's move on with categorizations in PSO. PSO categorizes energy, field and realizable motion under BFO disposition. Realizable motion seems very specific; many realizable processes could be listed here. Therefore, it is suggested to remove it as a first class universal, or an elucidation is required. Conversely, a field (electric, magnetic, or so) does seem to be a disposition. Energy is used as a quality in this work, rather than as a disposition, as PSO suggests. We even believe energy should be treated as an independent continuant, because of the mass–energy equivalence principle. However, the way energy is used and represented in most engineering conduct makes the use of energy as a quality more logical.

Regarding qualities such as power, one can say that they might not actually exist physically. They are a way to denote the value of how a certain real quality changes over time. Thus, a physical object does not have a velocity or acceleration. Instead, it only has a position. The velocity and acceleration are information entities that describe how the position changes, and how that velocity changes, respectively. A physical behavior then **involves** a set of qualities of the participants of the process it belongs to. The velocity and acceleration may then **be about** that process profile.

3.9. DISCUSSION ON THE USE OF AN ONTOLOGY

An objection to the use of an ontology is that a designer or analyst must have a good understanding of the ontology and how to represent systems and technologies in it. Because different individuals have different viewpoints and conceptions, their description of identical entities may differ. Furthermore, the formal language of an ontology may make it difficult to describe physical entities and phenomena properly, especially for people with less training in its use [138]. This issue is pervasive in ontology research and application, and there seems to be a lack of sound solutions. To alleviate this problem, ontologies should be built up of multiple, general ontologies, such as a mereology, topology, geometry and physical process ontology [138]. The modeling process should follow several steps: defining the components, defining the processes and behavior, and defining the mathematics [60]. It would then be evident what to model at each step and when to transition from one step to another. Another option is to construct a library of systems and processes that a designer can use, rather than defining these themselves.

There may be different ontologies suitable for describing engineering systems and the physics surrounding them. The proposed ontology can be easily replaced by such an other ontology. Only the rules presented in this paper have to be rewritten in terms of those ontologies, for the approach to work.

Rather than having to specify everything by hand, several relations can be determined automatically using more sophisticated software. For example, the **overlaps** relation can be computed using CAD software. Furthermore, the interaction between processes could be determined by simulating them and observing some key qualities. Through principal component analysis and other statistical methods, the behavior of groups of processes can be deduced, which then helps to establish the **inhibiting process of** relationships. This possibility enables the method to extend existing systems engineering practices. However, it requires a rather detailed description of the technologies and systems; something that may not be available in the conceptual design phase.

3.10. CONCLUSION

This chapter aims to find an answer to the research question:

> How to represent engineering systems and technologies consistently and robustly, allowing for knowledge capturing, reuse and sharing?

The answer is an ontology that aims to capture engineering systems as realistically as possible. The basis of this ontology is formed by a combination of three upper ontologies: BFO, IAO and PSO. BFO aims for ontological realism, which means they try to separate what *is* from how that entity is described, modeled or perceived. IAO then enables modeling the information and measurements we have on entities. PSO aims to describe the modeling of a physical system in terms suited for partial differential equation solvers. Thus, it bridges the gap between ontological realism and quantitative analysis.

The proposed ontology introduces some additional classes to BFO and PSO, most notably components, interfaces and some role subclasses. Furthermore, several relations are added that enable the inferences the following chapters build upon. These relations are elucidated and defined to reduce their ambiguity. Apart from the ontology,

a key insight is provided that technologies can be modeled as graph transformations on knowledge graphs that represent engineering systems. This allows technologies to be formally captured and represented.

Because the graphical approach to knowledge representation requires graph matching, granularity becomes an inhibitory factor. Granularity depends on the viewpoint with which a certain entity is perceived. It allows a system to be described very generically, or in extensive detail. However, the resulting knowledge graphs are not isomorphic, but semantically equivalent. Therefore, techniques to infer that equivalence have to be developed.

Ontologies themselves are open to interpretation, as their taxonomy and resulting relations depend on the creator's viewpoints. The classification of energy in PSO, for example, is debatable. Likewise, modeling physical behavior proves challenging and a satisfactory solution that "closes the world"[2] has yet to be found.

Creating the knowledge graphs is a time-intensive task and requires significant knowledge of the ontology. This cannot be expected from any practitioner, and, thus, solutions to this issue have to be found. The easiest solution is to employ knowledge engineers to create higher-level constructs that a practitioner can readily use. Otherwise, intelligent software should aid in the creation of the knowledge graphs. For example, a CAD program could in the background construct a knowledge graph describing the spatial regions and material entities within them, as well as their spatial relationships. That already forms a considerable portion of the knowledge graph of an engineering system.

Despite the challenges presented above, the ontology and graph transformation rules are used in the following chapters to automate parts of the technology evaluation and selection process. Specifically, the next chapter develops techniques to generate technology portfolios from a set of technologies, by constructing a technology compatibility matrix automatically.

[2] This refers to the Closed World Assumption (CWA) as discussed in Chapter 2.

4

TECHNOLOGY PORTFOLIO GENERATION

Parts of this chapter have been published in Journal of Engineering Design **32**, 2 (2021) [139].

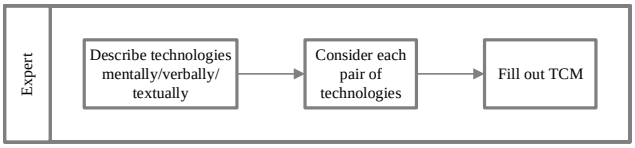


Figure 4.1: Common practice of constructing a TCM, using only expert judgment

In a typical technology evaluation project, different individuals (people or companies) contribute several technologies, which are typically expressed in writing, perhaps with accompanying drawings. From these technology descriptions, a group of experts has to first figure out which combinations of these technologies may be applied to a system of interest. Each valid combination is a technology portfolio and has certain merits and drawbacks on the system performance and other quantities of interest. Then, each portfolio should be quantified in terms of several quantities of interest. Based on this information, the experts can select which portfolios to consider for further development and implementation.

The issue is that the technologies are not formally expressed, and, therefore, each expert may have a different interpretation of each technology. Consequently, understanding how the technologies interact becomes difficult and requires extensive discussion with other experts. Furthermore, only the final result of these discussions are stored in a technology compatibility matrix (TCM). Afterwards, one cannot deduce from the TCM *why* certain technologies are incompatible. The process is illustrated in Figure 4.1

The ontology from Chapter 3 enables machine interpretation of the technologies, such as to automate the construction of a TCM. Automating this task would not only improve the consistency of the technologies' representation, but also cuts back in time spent. To illustrate, a TCM contains $(n^2 - n)/2$ entries, where n is the amount of technologies. Assigning each entry by hand quickly becomes a time-intensive task [32, 85]. Enabling relationships between technologies may also be captured in the TCM, in which case it becomes non-symmetric, and $n^2 - n$ entries have to be considered.

A TCM is not the goal itself, but a means to generate feasible combinations of technologies: technology portfolios. The reason is that the total amount of possible portfolios is 2^n, which poses a challenge when QoIs have to be computed for each. The TCM helps to remove those portfolios that contain incompatible combinations of technologies. Usually, a significant reduction in feasible portfolios is obtained this way. Nonetheless, evaluating all remaining portfolios is usually computationally expensive, or even intractable when high-fidelity analysis models are used. This chapter also considers how to find a representative set of portfolios for which QoIs are computed and then projected back onto the original set.

How technology (in)compatibility may be automatically derived with the ontology from chapter 3 is discussed in section 4.2. Two methods are presented: one based solely on the graph transformation rules, and another which uses FOL rules in addition. Then, section 4.3 considers how technologies enable one another, again through two approaches: only graph transformation rules, or those in combination with FOL. Subsequently, section 4.4 explains how portfolios are generated taking the TCM into consideration. The maximum dissimilarity technique that finds a representative set of portfolios is discussed

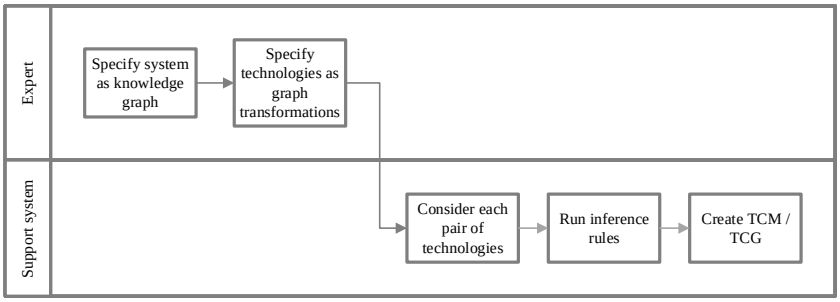


Figure 4.2: Proposed method for constructing a TCM, using a rule-based system relying on an ontology

in section 4.5 and subsection 4.5.1. Finally, three case studies highlight different aspects of the theoretical developments presented in this chapter. section 4.6 presents a case study based on the MANTA project, to illustrate the FOL approach. section 4.7 shows the application of the FOL approach to an industry technology set. This is to strengthen the belief that the method is practically applicable. Finally, section 4.8 discusses application of the graph-transformation-rule-based approach to a game called Factorio. Here, the maximum dissimilarity technique is exemplified.

4.1. METHODOLOGY OVERVIEW

By evaluating what a technology modifies, rules dictate whether that modification is compatible with another technology. That process is illustrated in Figure 4.2. Compare it to Figure 4.1 to see what tasks the support system takes over from experts. Each of these three elements is described in the following sections.

4.2. TECHNOLOGY COMPATIBILITY AND INCOMPATIBILITY

The purpose of this section is to infer whether any two technologies are incompatible with one another. We assume that if no reason is found for the pair to be incompatible (denoted by $\perp$), it is compatible (denoted by $\parallel$). Thus:

$$t_1 \parallel t_2 \Leftrightarrow \neg(t_1 \perp t_2) \tag{4.1}$$

There are two methods to infer compatibility and incompatibility between technologies. The first is purely based on graph transformation rules, while the second implements logic-based rules to infer certain facts. The second method can be implemented as an extension to the first, while the first can act stand-alone.

4.2.1. USING GRAPH TRANSFORMATION RULES

Technology compatibility only requires the transformation rules to be parallel independent (see section 2.6.3). That ensures that neither technology offsets any effect of the other. Note that transformation incompatibility only applies when the technologies are introduced to an overlapping portion of the system graph G. Recall from section 3.6 what a technology transformation rule is.Also keep in mind that the following discussion assumes graph transformations are defined at the same level of granularity, as discussed in

section 3.7. This assumption is necessary for all the graph matching performed throughout this thesis.

Parallel independence is checked as follows. First the maximum common (induced) subgraph (MCS) $K_{L,1,2}$ of L_1 and L_2 is found as:

$$K_{L,1,2} = \text{argmax}_{k_1,k_2} |V_{K_{1,2}}| : K_{L,1,2} \xrightarrow{k_1} L_1, K_{L,1,2} \xrightarrow{k_2} L_2 \tag{4.2}$$

This MCS is independent of where the technologies apply in G, which has to be remedied. In other words, a MCS is sought that applies to specific matches of t_1 and t_2 in G. Suppose that $L_1 \xrightarrow{m_1} G$ and $L_2 \xrightarrow{m_2} G$ are the matches of the technologies in G. To find the MCS of these matches, take $K_{L,1,2}$ and define:

$$K_{1,2} = m_1 \circ k_1(K_{L,1,2}) \cap m_2 \circ k_2(K_{L,1,2}) \tag{4.3}$$

Two technologies are parallel independent, when neither t_1 removes something t_2 uses, or vice versa. That means that their removal graphs may not contain any node or edge that the pattern of the other technology requires. Conversely, if that is the case, the technologies are concluded to be incompatible:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow (L_1 - K_1 \cap K_{1,2} \neq \emptyset) \vee \\ (L_2 - K_2 \cap K_{1,2} \neq \emptyset) \end{aligned} \tag{4.4}$$

Constructing $m_1 : K_{1,2} \rightarrowtail L_1$ and $m_2 : K_{1,2} \rightarrowtail L_2$ in Equation 4.3 only uses equivalence between variables without considering the values of these variables. However, when either a pattern L or effect R contains variables with values, it should be checked that other transformation rules do not interfere with these values.

Values on attributes should also be compatible with one another. Simply checking equality is not sufficient, because rather than single values, a constraint (e.g. upper or lower bound) may be placed on a value. In other cases, a value can be a geometry, and compatibility either means geometrical congruence or similarity. In any of these cases, a value can be represented as a constraint on the domain of values for the attribute. Therefore, suppose there is a variable $x \in \mathbb{X}$. Then a constraint $f : f(x) \rightarrowtail \mathbb{F} \subseteq \mathbb{X}$ limits the possible values for x. When two such constraints are present, compatibility is expressed as:

$$f_1 \parallel f_2 \equiv f_1(x) \cap f_2(x) \neq \emptyset \tag{4.5}$$

With value compatibility defined, technology compatibility resulting from it can be inferred. First, the constraints expressed in L_1 and L_2 should be compatible. If not, the technologies are incompatible:

$$t_1 \perp t_2 \Leftarrow \exists x \in K_{1,2} : f_{L_1}(x) \perp f_{L_2}(x) \tag{4.6}$$

Second, the constraints in the effects of the technologies have to be compatible:

$$t_1 \perp t_2 \Leftarrow \exists x_1 \in R_1, \exists x_2 \in R_2 : x_1 \cong x_2 \wedge f_{R_1}(x_1) \perp f_{R_2}(x_2) \tag{4.7}$$

Two variables are equivalent (i.e. $x_1 \cong x_2$) when they describe the same aspect (either a specifically dependent continuant or information content entity) of the same entity.

So far, negative application conditions (NAC) on the graph transformation rules have not been considered. However, they are a powerful addition to vanilla graph transformations that allows one to specify a morphism $L \overset{l}{\longmapsto} \hat{L}$, such that, with the total graph morphism $L \overset{m}{\longmapsto} G$, there exists no morphism $\hat{L} \overset{n}{\longmapsto} G$ with $n \circ l = m$ [140]. The incompatibility definition can then be updated according to the critical pair analysis by Lambers et al. [141]. Essentially, a negative application condition specifies a pattern in the graph that, if present, prevents the graph transformation rule from being applicable. Therefore, these conditions provide a convenient mechanism to specify when and how technologies are incompatible, on a case-by-case basis. They allow us to circumvent crafting logic rules as presented in subsection 4.2.2 that should be generically applicable and need to balance generality versus specificity. NACs are case-specific, but also more transparent, because they are directly linked to a technology.

Let a NAC be defined as a graph morphism $L \overset{n}{\longmapsto} N$, where $L \subset N$. Thus, the graph N contains the pattern L and a few additional elements, which denote the conditions that may not be present in some host graph G if the rule the NAC belongs to is to be applied.

Then, suppose there are two transformations $t_1 : L_1 \rightarrowtail R_1$ and $t_2 : L_2 \rightarrowtail R_2$, with negative application conditions n_1 and n_2, respectively. Furthermore, $m_1 : L_1 \rightarrowtail G$ and $m_2 : L_2 \rightarrowtail G$ are the matches of the transformation patterns in a host graph G. The transformations t_1 and t_2 are incompatible in G with respect to their NACs n_1 and n_2 if:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & t_1 \circ k_1(K_{L,1,2}) \cap n_2 \circ k_2(K_{L,1,2}) \neq \emptyset \wedge \\ & t_2 \circ k_2(K_{L,1,2}) \cap n_1 \circ k_1(K_{L,1,2}) \neq \emptyset \end{aligned} \tag{4.8}$$

4.2.2. USING ONTOLOGY AND LOGIC

PHYSICAL INCOMPATIBILITY

The main reasoning mechanism for incompatibility of technologies is when their simultaneous application would result in a physically inconsistent situation. The following two rules capture several of such inconsistent situations. A trivial inconsistency arises when two material entities overlap. Thus, if two material entities occupy the same space, they cannot co-exist. Then any two technologies introducing material entities that cannot co-exist are incompatible:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & \exists c_1, c_2, r_1, r_2 : \text{material entity}(c_1) \in (R-K)_1 \wedge \\ & \text{material entity}(c_2) \in (R-K)_2 \wedge \\ & \text{spatial region}(r_1) \wedge \text{spatial region}(r_2) \wedge \\ & c_1 \textbf{ occupies } r_1 \wedge c_2 \textbf{ occupies } r_2 \wedge \\ & r_1 \textbf{ overlaps } r_2 \end{aligned} \tag{4.9}$$

Introducing a process that inhibits the process introduced by another technology leads to incompatibility:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow\ & \exists p_1, p_2 : \text{process}(p_1) \in (R-K)_1 \wedge \\ & \text{process}(p_2) \in (R-K)_2 \wedge \\ & (p_1 \textbf{ inhibits } p_2 \vee p_2 \textbf{ inhibits } p_1) \end{aligned} \tag{4.10}$$

FUNCTIONAL INCOMPATIBILITY

Often, it is undesirable to have two technologies introduce the same functionality in the same region. Therefore, one might wish to define incompatibility between such technologies. This is the case when they introduce processes that have an equivalent effect.

For this, a form of equivalence between processes has to be defined. For now, the equivalence is based on the effect of the process. This is inferred when two processes realize the same type of disposition:

$$\begin{aligned} p_1 \textbf{ has equivalent effect } p_2 \Leftarrow \forall p_1, p_2 : \text{process}(p_1) \wedge \text{process}(p_2) \wedge \\ \exists d_1, d_2 : \text{disposition}(d_1) \wedge \text{disposition}(d_2) \wedge \\ p_1 \textbf{ realizes } d_1 \wedge p_2 \textbf{ realizes } d_2 \wedge \\ d_1 \subseteq d_2 \end{aligned} \tag{4.11}$$

According to this definition, even when a process has multiple dispositions, only one of which is equivalent with one from another process, the processes have an equivalent effect. The equivalent effect is, therefore, only a subset of the full effect of the process. To obtain a stronger notion of equivalence between the processes, the existential quantifier on the dispositions should be replaced with a universal quantifier. Depending on one's viewpoint, that may be the more correct way forward.

The technology incompatibility statement that when the technologies introduce processes that have an equivalent effect, then reads:

$$\begin{aligned} t_1 \perp t_2 \Leftarrow \exists p_1, p_2, r_1, r_2 : \\ \text{process}(p_1) \in (R-K)_1 \wedge \text{process}(p_2) \in (R-K)_2 \wedge \\ p_1 \textbf{ has equivalent effect } p_2 \wedge \\ \text{spatial region}(r_1) \wedge \text{spatial region}(r_2) \wedge \\ p_1 \textbf{ occupies } r_1 \wedge p_2 \textbf{ occupies } r_2 \wedge \\ r_1 \textbf{ overlaps } r_2 \end{aligned} \tag{4.12}$$

Only when the processes overlap in space are they considered incompatible, because in theory, the one technology could introduce a process on the moon and the other on Earth, which obviously have nothing to do with one another. This rule does not capture the situation where the regions r_1 and r_2 are situated such that the processes have the same effect in the same region nonetheless. This could happen, for example, when the processes are flames on two sides of a metal plate. Clearly, they occur in distinct regions, but their effect — heating the plate — is equivalent.

4.3. TECHNOLOGY ENABLING

Besides incompatibility, statements regarding technologies enabling one another have to be inferred. When technology t_1 enables t_2, this is written as $t_1 \prec t_2$. Technology enabling occurs through two mechanisms: the graph transformation rules and physics-based rules in the ontology. Each of these rules is overruled if any incompatibility statement fires for the pair of technologies. This means that:

$$t_1 \prec t_2 \Rightarrow t_1 \parallel t_2 \tag{4.13}$$

and, despite what any of the following rules might suggest:

$$t_1 \perp t_2 \Rightarrow t_1 \not\prec t_2 \wedge t_2 \not\prec t_1 \tag{4.14}$$

4.3.1. USING GRAPH TRANSFORMATION RULES

Technology transformation rules may be sequentially dependent (see section 2.6.3). In that case, the technology that sequentially depends on another is enabled by the latter. Nonetheless, they still have to be parallel independent. When applied to a system graph G, the pattern L_2 should be a subgraph of G after application of t_1. Meanwhile, L_2 should not appear in G, without application of t_1:

$$t_1 \prec t_2 \Leftarrow L_2 \nsubseteq G \wedge G \xrightarrow{t_1} H \wedge L_2 \subseteq H \tag{4.15}$$

This rule fires when, for example, a technology is the addition of a component to the system, and the second technology only works for such a component. The last rule also works when only part of a system is modified, say for example a material is changed from metal to composite, and the other technology requires a composite material to work.

Equation 4.15 only considers a single technology enabling another single technology. However, it may happen that a set of technologies actually enables a single technology. When a technology is enabled by a set of other technologies, a more elaborate algorithm is required to find such sets.

In order to find the sets of enabling technologies, we define a set of inapplicable technologies $\mathbb{T}_N$ in a certain host graph G as $t_i \in \mathbb{T}_N : L_i \nsubseteq G$. Then, for each of the technologies in this set we need to find a collection of enabling technology sets (i.e. a set of sets), because there may be multiple different combinations of technologies that enable a given technology. A valid enabling technology set for t_i is denoted as $\mathbb{T}_E \prec t_i$. Whenever two sets $\mathbb{T}_{E,x} \prec t_i$ and $\mathbb{T}_{E,y} \prec t_i$, and $\mathbb{T}_{E,x} \subset \mathbb{T}_{E,y}$, then $\mathbb{T}_{E,y}$ is regarded as redundant and is not a solution to the problem.

The algorithm to find the enabling technology sets for t_i is implemented as a tree search over the space of technology portfolios $2^{\mathbb{T}\setminus\mathbb{T}_N}$. The tree has one of the applicable technologies as root and splits into whether it is included in the enabling set or not. At each level in the tree, another applicable technology is considered. Several checks enable the algorithm to skip certain solutions, reducing its run-time, although in the worst case it runs in $\mathcal{O}(2^{|\mathbb{T}\setminus\mathbb{T}_N|})$. Pseudo-code of the algorithm is given in Appendix D.

4.3.2. USING ONTOLOGY AND LOGIC

PHYSICAL ENABLING

A technology enables another when it removes an interface or process or disposition that inhibits one that the other encounters in the system. The converse case, when a technology adds a process or interface that another requires, has to be handled as well.

Consider the flame in a jet engine combustor. For it to work, the airflow into the combustor must be relatively laminar. So a technology that laminarizes the inflow of air enables the combustion process. In other words, the enabling technology removes the turbulence process that inhibited the flame. The second case can be illustrated by a technology that adds a pumping process to force the fuel into the injector nozzle.

The challenge is that when this type of enabling occurs, there may be other aspects that still prevent the enabled technology to be fully applicable. For example, this happens when there are two processes that the technology relies on, which are both inhibited by some existing process in the system. If another technology only resolves the inhibition of one of these processes, the other process still prevents application of the technology in question. Therefore, we need to check if all inhibiting aspects are resolved by the enabling technology.

The above description is captured in the following rule. After applying $t_1 : G \mapsto H_1$ and $t_2 : H_1 \mapsto H_2$, this rule infers physical enabling through removal of inhibiting processes or blocking dispositions:

$$\begin{aligned} t_1 \prec t_2 \Leftarrow \forall\ \text{disposition}(d)\ \in (R-K)_2 : \{\nexists\ \text{disposition}(d_G) \in H_1 : \\ d_G\ \textbf{blocking disposition of}\ d\} \wedge \\ \forall\ \text{process}(p)\ \in (R-K)_2 : \{\nexists\ \text{process}(p_G) \in H_1 : p_G\ \textbf{inhibits}\ p\} \wedge \\ ([\exists\ \text{disposition}(d)\ \in (R-K)_2 : \{\exists\ \text{disposition}(d_G) \in G : \\ d_G\ \textbf{blocking disposition of}\ d\}] \vee \\ [\exists\ \text{process}(p)\ \in (R-K)_2 : \{\exists\ \text{process}(p_G) \in G : p_G\ \textbf{inhibits}\ p\}]) \end{aligned} \tag{4.16}$$

The other mechanism by which two technologies may enable one another is when they resolve dependencies imposed on each other's interfaces. This is simply the case when, for example, two electrical conductors are connected, such that a current may flow through. (A battery alone does not produce current, nor an electromotor by itself. If the two are connected, however, a current flows and the function of the electromotor is realized. The battery already fulfills its function by storing chemical energy. Depending on your point of view, it also has a function to convert chemical energy to electric energy and/or to provide it to some other device. In that case, the battery has these functions realized by the complementing interface.)

For this mechanism we require the notion of complementary and collective dispositions [136]. The idea is that an object aggregate C can have a disposition D that is formed by the mereological[1] sum of its constituent dispositions d_i. Thus, $C = \sum c_i$, and $\forall c_i : c_i$ **has disposition** d_i, such that $D = \sum d_i$. The latter sum describes parthood between dispositions, which is not clearly defined. Instead, the process aggregate P that realizes D can be described as the sum of its constituents: $P = \sum p_i : P$ **realizes** D. Then, for each of these constituent processes, there must exist a part of C that manifests it: $\forall p_i (\exists c_i \in C : c_i$ **has disposition** d_i **realized by** $p_i)$. This works, because parthood of processes is a better defined concept.

To make this work in practice, inferences are required that establish how interfaces enable one another's functions. This can be done for two interfaces with the following

[1] Mereology is the study of parts and the wholes they form in philosophy and mathematical logic.

rule:

$$\begin{aligned} I_1 \textbf{ complements } I_2 \Leftarrow\ & \text{interface}(I_1) \wedge \text{interface}(I_2) \wedge \\ & \forall p_0 : I_2 \textbf{ has function } f \textbf{ realized by } p_0 \\ & [(\exists p_2 : p_0 \textbf{ has part } p_2 \wedge \\ & I_2 \textbf{ has disposition } d_2 \textbf{ realized by } p_2) \wedge \\ & \forall p_1 : p_0 \textbf{ has part } p_1 \wedge \\ & \neg(I_2 \textbf{ has disposition } d_2 \textbf{ realized by } p_1) \\ & (\exists d_1 : I_1 \textbf{ has disposition } d_1 \textbf{ realized by } p_1)] \end{aligned} \tag{4.17}$$

This rule states that for each function of interface I_2 it performs at least one of the sub-processes involved in fulfilling that function. Furthermore, for each part of p_0 that is not manifested by interface I_2, there should be a disposition in interface I_1 that does manifest it.

Looking back at the definition from Goldfain et al. [136], it should be clear that p_0 in Equation 4.17 is the collective process P. Furthermore, the interfaces I_1 and I_2 are the parts c_i that manifest the sub-processes p_i (as p_1 and p_2). This rule also works when $p_0 = p_1 = p_2$, because **has part** is both transitive and reflexive (see Table 3.5).

Now it has been established that these two interfaces form a collective that realizes the function intended by one of them, we can bring it back to technology level and infer technology enabling:

$$\begin{aligned} t_1 \prec t_2 \Leftarrow \exists\, \text{interface}(I_2) \in (R-K)_2\ [\exists\, \text{interface}(I_1)\ \in (R-K)_1 : \\ I_1 \textbf{ complements } I_2\big] \end{aligned} \tag{4.18}$$

Thus, a technology enables another when it complements one or more interfaces of the other. To explain why this is taken as the enabling condition, rather than requiring each interface to be complemented by the other technology, consider a pump. A pump has two interfaces: inflow of non-pressurized fluid and outflow of pressurized fluid. A pump is only fully enabled when both ports are connected. However, no single object would simultaneously satisfy both these interfaces. Therefore, only one interface has to be complemented by an enabling technology, even though that may mean that the enabled technology remains with unresolved interfaces.

4.4. GENERATING PORTFOLIOS

When there are no exclusions, the set of portfolios $\mathbb{P}$ is the power set of the technologies $\mathbb{T}$, i.e. $\mathbb{P} = 2^{\mathbb{T}}$. Let a portfolio $P \subseteq \mathbb{T}$ be represented by a vector of length $|\mathbb{T}|$, with binary values indicating that when $P_i = 1$, $t_i \in P$. Using the TCM, one can easily remove those portfolios where: $P_i = P_j = 1$ and $TCM_{ij} = 0$, i.e. $t_i \perp t_j$, $\forall i, j \in [1, |\mathbb{T}|]$.

If a portfolio contains a technology that has to be enabled by other technologies, the portfolio is only valid if it contains all of those other technologies. The matter is complicated by the possibility of multiple valid sets of enabling technologies for any given inapplicable technology. Thus, let $\mathbb{T}_N \subset \mathbb{T}$ be the set of technologies that are not applicable and require enabling technologies. Then, when $P \cap \mathbb{T}_N \neq \emptyset$, P is valid only if $\forall\, t_i \in P \cap \mathbb{T}_N, \exists \mathbb{T}_E \subset P : \mathbb{T}_E \prec t_i$, where $\mathbb{T}_E$ is a set of enabling technologies. The algorithm is described in Appendix F.

4.5. MAXIMUM DISSIMILARITY

It often occurs that it is computationally intractable to analyze an entire set of entities for a given computational budget. For example, a conceptual aircraft analysis method easily takes several minutes to execute. If one has ten technologies, there are a total of $2^{10} = 1024$ portfolios. For each, the method is executed, which, if the method takes five minutes, amounts to 85 hours of computations. Ten technologies is not a large set, nor is five minutes an excessive amount of computational time for a single run.

As a solution, there are some techniques to replace the entire set with a subset that captures as much information about the full set as possible. Only the subset is analyzed and either is used to construct some surrogate model, or the results from the subset are mapped back onto the full set. Typically, this is a design space exploration issue, which could be solved using Latin Hypercube sampling or other random sample generators. However, we can question whether simply selecting an evenly spaced out sample set in the multi-dimensional binary space provides a proper representation of the variability in performance of the portfolios. That is to say, how close is portfolio 100 to portfolio 101 or 011?

Maximum dissimilarity is another technique that creates a representative set of elements from a given high-dimensional space. The subset is created by selecting those elements that are maximally dissimilar to one another, under the assumption that this gives the most diverse, representative subset of entities. Dissimilarity is scored using some distance measure. Commonly, the entities are numeric vectors, and the distance measure is either Manhattan or Euclidean distance. That does not work for technology portfolios, for reasons explained in the following subsection. This is where the graph description of technologies comes into play. Each portfolio results in a different system graph. The similarity between these graphs could be an indication of how similar the actual systems are, and, therefore, how similar their behavior is. Fortunately, different techniques exist to compute similarity over graphs, such as graph edit distance (GED). The edit distance is the cost associated with transforming a graph into another, by adding or removing or substituting nodes and edges. These costs can be specified in any way, to emphasize certain aspects or properties of the graphs. For example, if only structure is important, the cost of adding or removing edges should be high. Conversely, if node types are important, but structure is not, only the cost of adding or removing nodes is high.

Regardless of the distance measure, maximum dissimilarity starts by selecting an item P_0 from a set of portfolios $\mathbb{P}$. Then, it iterates through the remaining portfolios $P_i \in \mathbb{P} \setminus P_0$. It selects the following portfolio as $\text{argmax}_{P_i} : d(P_0, P_i)$. The subset of maximally dissimilar portfolios now contains two elements, so to select a third, the MinMax strategy is employed. For each $P_i \in \mathbb{P} \setminus \mathbb{P}_D$, compute $d_{P_i,\text{min}} = \min(d(P_i, P_j)\ \forall P_j \in \mathbb{P}_D)$, where $\mathbb{P}_D$ is the subset of dissimilar portfolios. Then, the next maximally dissimilar portfolio is chosen as $\text{argmax}_{P_i} : d_{P_i,\text{min}}$. This selection process is repeated up to a user-specified K number of times: the size of the maximal dissimilar subset $\mathbb{P}_D$. The maximum dissimilarity algorithm is provided in Appendix E.

Once $\mathbb{P}_D$ is determined, each of these representative portfolios P_j can be analyzed and a quantity q_j is computed for each. These results have to somehow be mapped back to the original set of portfolios $\mathbb{P}$. By viewing the portfolios in $\mathbb{P}_D$ as representative classes

for $\mathbb{P}$, it becomes a matter of computing the probability that $P_i \in \mathbb{P}$ belongs to $P_j \in \mathbb{P}_D$. For this, the softmax equation is commonly used in machine learning:

$$P(P_i \equiv P_j) = \frac{e^{-d_{ij}}}{\sum_{P_k \in \mathbb{P}_D} e^{-d_{ik}}} \quad , \tag{4.19}$$

where we use the notation $d_{ij} = d(P_i, P_j)$. Softmax ensures that that class probabilities add up to 1. Therefore, we can estimate the quantity $\bar{q}$ for the portfolios not in $\mathbb{P}_D$ as:

$$\bar{q}_i = \sum_{P_j \in \mathbb{P}_D} P(P_i \equiv P_j) \cdot q_j \quad . \tag{4.20}$$

4.5.1. GRAPH EDIT DISTANCE

The maximum dissimilarity approach works for the technology portfolios, provided we can conjure a consistent parameterization that captures all the portfolios. When each portfolio is represented using the same variables, it can be represented as a vector and this approach works well with Euclidean distance. However, portfolios may lead to widely varying systems, with no single parameterization valid for all of them. One might wonder: what if we take the union of all parameter vectors and then just set certain variables to zero or some other value that effectively disables them during simulation? If the value is set to zero, this might lead to falsely estimated distances, because the measure cannot distinguish between a valid value of zero, or one that is meant to indicate absence of that parameter for a portfolio. Otherwise, a NaN or Inf value obviously prevents the calculation of a distance. Finally, this solution makes the distance computation dependent on the simulation models, as these determine the parameterization.

With the graph representation of portfolios, however, graph edit distance (GED) can be employed as the dissimilarity measure. The idea is that similar systems have similar behaviors and their graphs will only differ by few nodes, edges and/or attributes. This is precisely what the GED measures. In fact, we could specify the so-called edit costs in such a way that dissimilar edges have a larger impact on the distance than dissimilar nodes,for example. Furthermore, we could specify that replacing certain types of nodes results in more dissimilarity than replacing other types of nodes. For now, these options are not investigated, but they could significantly improve the accuracy, and, therefore, the applicability of the maximum dissimilarity approach.

FAST GED IMPLEMENTATION

How the GED algorithm works is explained in Appendix C, but suffice it to say it is computationally expensive, even for small to moderate size graphs. Fortunately, the graph transformation rules offer a solution. Each portfolio is a set of technologies, which are graph transformations applied to a certain system graph G. Therefore, a portfolio graph G_{P_i} is different from G by the additions, deletions and substitutions of the graph transformations contained in P_i. The graph edit distance between G_{P_i} and G_{P_j} then does not have to be computed directly. Instead, let $C(t_k)$ be the total cost (implied distance) of the transformation t_k. Then the distance d_i from G to G_{P_i} is computed as:

$$d_i = \sum_{t_k \in P_i} C(t_k) \cdot |m_{t_k}| \quad . \tag{4.21}$$

Here, m_{t_k} denotes the matches of technology t_k in the graph G.

The transformation cost $C(t_k)$ is computed as the sum of the additions, deletions and attribute value substitutions in t_k:

$$C(t_k) = \sum_{n \in (L-K)_k} C_D(n) + \sum_{n \in (R-K)_k} C_A(n) + \sum_{(n_L, n_R) \in A(K_k)} C_S(n_L, n_R) \tag{4.22}$$

where C_D is deletion cost, C_A addition cost and C_S substitution cost. $A(K_k)$ are the attributes in the gluing graph K_k. The cost of replacing n_L with n_R is zero when the value for that attribute is equal in L and R, and some user-specified cost otherwise.

To compute the distance from one portfolio to another, one might simply assume it is equal to $d_i + d_j$. However, that is not entirely the case, because both portfolios may add or remove equivalent graph elements, or assign an equal value to the same attribute. Thus, the distance should account for these cases:

$$\begin{aligned} d_{ij} = & d_i + d_j - C_S\left(A(K_i) \cap A(K_j)\right) - \\ & C_D\left((L-K)_i \cap (L-K)_j\right) - \\ & C_A\left((R-K)_i \overset{\cap}{\equiv} (R-K)_j\right) \end{aligned} \tag{4.23}$$

Each of these terms is addressed in section C.1, because it does not add to the present discussion to describe them here in detail.

4.6. CASE STUDY: AIRCRAFT TECHNOLOGIES

In this section, several technologies are modeled to showcase the inferences that the presented method enables and how these can be leveraged to automatically construct a technology compatibility graph (TCG), which is a generalization of the TCM.

4.6.1. TEST CASE DESCRIPTION

The first case study (Sections 4.6.2 and 4.6.3) revolves around an aerodynamic surface, such as an aircraft wing. The following technologies are considered: a vortex generator, a plasma actuator, natural laminar flow, conformal antennas and conductive structure. The plasma actuator, natural laminar flow and conformal antennas have the function to reduce the friction drag of the wing. The vortex generator aims to prevent flow separation. The conductive structure is meant to remove the need for cables and can provide electricity distributed over a surface. In current practice, experts would now discuss these technologies and establish a TCM by hand. That is, they draw up a matrix and fill out each cell by discussing whether that pair of technologies is incompatible or has any other dependency.

Instead, the proposed approach is applied here. The graph transformation rules (representing technologies) are shown as knowledge graphs with the addition and removal subgraphs. As such, knowledge about what a technology constitutes is captured. Then, we show what inferences the computer algorithm would make in what order to illustrate the approach. Rather than doing so for each pair of technologies, one of the compatibility rules from Section 4.2 or one of the enabling rules from Section 4.3 is illustrated with one pair of the five technologies.

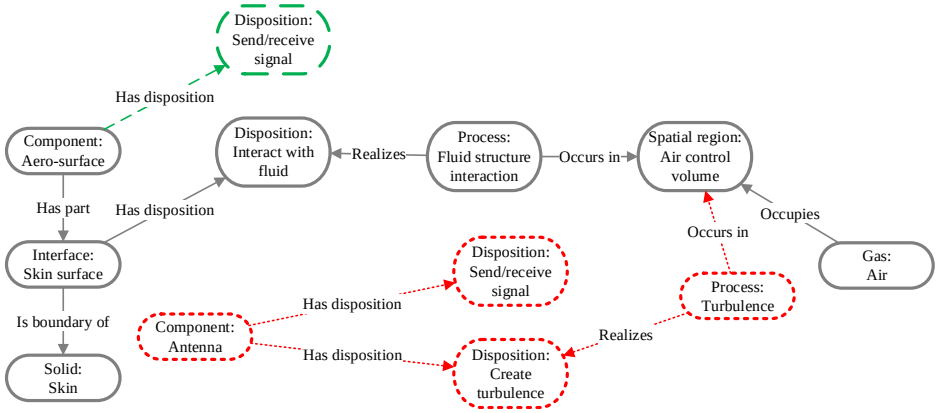


Figure 4.3: Graph transformation of conformal antenna. Grey represents the pattern, green (dashed) the added nodes and edges, and red (dotted) the removed nodes and edges.

4.6.2. COMPATIBILITY REASONING

Two of the incompatibility rules are illustrated in this subsection: inhibitory processes (Section 4.6.2) and functional equivalence of processes (Section 4.6.2).

PHYSICAL INCOMPATIBILITY: INHIBITING PROCESS

A body moving through air experiences friction from the air particles colliding with the object's surface. When the airflow around the body is laminar, the air particles move in layers that hardly interact. Therefore, the particles close to the surface move tangentially to it, which reduces the friction drag, because less collisions occur. Conversely, when the flow is turbulent, the air moves chaotically, with more friction as a result. The portion of air that experiences friction from the body's surface is called the boundary layer. Air outside the boundary layer is usually regarded as laminar, while the boundary layer itself starts of as laminar flow, but quickly transitions into turbulent flow.

On any aerodynamic surface, protrusions introduce turbulence, which inhibits laminar flow. Thus, when a technology introduces natural laminar flow[2], while another introduces turbulence, they are incompatible, according to Equation 4.10. When taking the pattern from Figure 4.3 (i.e. the grey and red parts of the graph) and the effect of Figure 4.4 (i.e. the grey and green parts of the graph), both a turbulent and laminar flow process are present.

To infer incompatibility between these two technologies, several rules are used. First, Equation 3.5 is used to define that turbulence inhibits laminar flow. Then, Equation 3.4 establishes that the laminar flow process in Figure 4.4 occurs in the boundary layer spatial region. This enables Equation 3.6 to infer that the turbulence process in Figure 4.3 inhibits the laminar flow process. Finally, that is the information needed for Equation 4.10 to deduce that the antenna is incompatible with the natural laminar flow technology.

[2] Natural laminar flow occurs when a body is shaped such that the flow around it remains laminar.

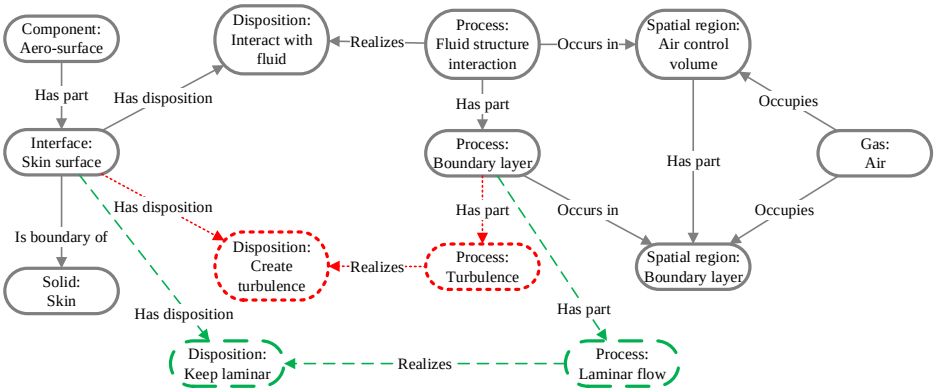


Figure 4.4: Graph transformation of laminar flow. Grey represents the pattern, green (dashed) the added nodes and edges, and red (dotted) the removed nodes and edges.

4

FUNCTIONAL INCOMPATIBILITY: PROCESS EQUIVALENCE

Although a laminar boundary layer produces less friction than a turbulent one, it is also more prone to a process called separation. Separation is the detachment of the boundary layer from the surface, as a result of the flow reversing direction close to the surface. This results in a lot of additional aerodynamic drag, and, therefore, is highly undesired. Several technologies exist that aim to turn the boundary layer turbulent in a controlled fashion, such that it will not separate. One such technology is a plasma actuator, which exploits ionization of air to turn it into plasma, locally. This has the effect of creating small vortices, making the boundary layer turbulent. Similarly, a vortex generator has the same effect, but is simply a small vane.

A plasma actuator is modeled as shown in Figure 4.5. It adds the plasma actuator component (in green) to an existing aerodynamic surface (in grey). As a second technology, consider the vortex generator, which would have a similar graph, except for the disposition and process of creating plasma, and without the electric interface. The incompatibility of these two technologies follows directly from Equation 4.12. Because the technologies are defined similarly, the fact that p_1 **has equivalent effect** p_2 follows from Equation 4.11.

It should be noted that in practice, these two technologies do not have to be incompatible. For example, if the vortex generator is placed behind the plasma actuator, it may become more efficient due to the turbulent boundary layer. It can then be smaller. These issues are discussed in section 4.9, along with what solution is recommended to avoid them.

4.6.3. ENABLING REASONING

Again, two rules are highlighted to showcase the enabling relationship inferences the ontology enables. First, we look at technologies that remove an inhibiting process to enable another technology in Section 4.6.3. Second, the idea of complementing interfaces is exemplified in Section 4.6.3.

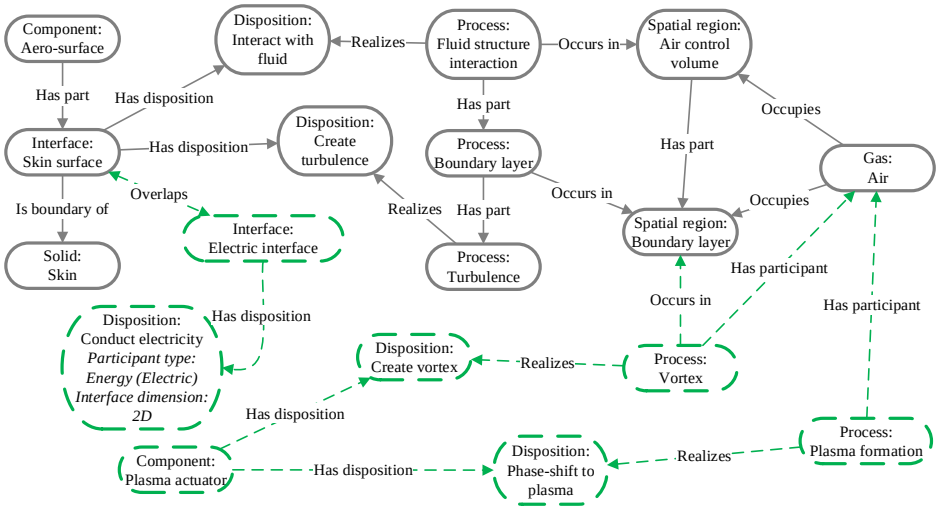


Figure 4.5: Graph transformation of plasma actuator. Grey represents the pattern and green (dashed) represents added nodes and edges.

PHYSICAL ENABLING: REMOVAL OF INHIBITING PROCESS

Consider conformal antennas (see Figure 4.3) and natural laminar flow (see Figure 4.4). A conformal antenna is entirely embedded in a surface, such that the surface has no protrusions. In order to infer that the conformal antenna enables natural laminar flow, Equation 4.16 is used. The second part of the rule, which states that an inhibitory process or blocking disposition should be present when only natural laminar flow is applied, is true, as already examined in section 4.6.2. After application of the conformal antenna rule, the turbulence caused by the antenna is removed. Application of the natural laminar flow rule itself causes the turbulence from the aerodynamic surface to disappear. Now, no process exists that inhibits the laminar flow process, and the first half of Equation 4.16 is also satisfied.

PHYSICAL ENABLING: COMPLEMENTING INTERFACES

As Figure 4.5 shows, the plasma actuator contains an interface that has the disposition to conduct electricity. However, there is no process that realizes it, which prevents Equation 4.17 from firing. If the conductive structure shown in Figure 4.6 is present, however, such a conduction process could be instantiated between the two interfaces introduced by each technology. In order to do that, a separate rule is added to the domain ontology, that creates a transfer process whenever two interfaces overlap that have the disposition

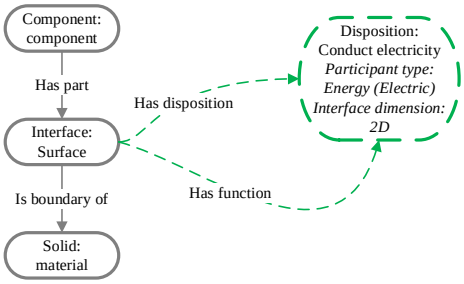


Figure 4.6: Graph transformation of conductive structure. Grey represents the pattern and green (dashed) represents added nodes and edges.

to transmit a certain type of entity. It reads:

$$
\begin{aligned}
&\exists i_1, i_2, d_1, d_2, t, id: \\
&\text{interface}(i_1) \wedge \text{interface}(i_2) \wedge i_1 \textbf{ overlaps } i_2 \wedge \\
&i_1 \textbf{ has disposition } d_1 \wedge i_2 \textbf{ has disposition } d_2 \wedge \\
&\text{disposition}(d_1) \in \text{ToTransmit} \wedge \text{disposition}(d_2) \in \text{ToTransmit} \wedge \\
&d_1 \textbf{ participant type } t \wedge d_2 \textbf{ participant type } t \wedge \\
&d_1 \textbf{ interface dimension } id \wedge d_2 \textbf{ interface dimension } id \\
&\qquad \Rightarrow \exists p_0 : \text{Transfer}(p_0) \wedge p_0 \textbf{ realizes } d_1 \wedge p_0 \textbf{ realizes } d_2
\end{aligned}
\tag{4.24}
$$

Conduction is such a transfer process, when the type of entity to transmit is electric energy.

With these two technologies, Equation 4.24 instantiates the process that Equation 4.17 requires to infer that the interfaces i_1 (the skin surface) and i_2 (the electric interface) complement each other. That, in turn, causes Equation 4.18 to hold true, which concludes that a conductive structure enables the plasma actuator. Note that the inverse is also true, because Equation 4.17 infers that the two interfaces are complementary in either direction.

4.6.4. THE TECHNOLOGY COMPATIBILITY GRAPH

Instead of a TCM, a technology compatibility graph (TCG) is automatically constructed using the above approach. The TCG is shown in Figure 4.7 for the five technologies investigated in this section. Note that Figure 4.7 omits the **compatible** relation, because that unnecessarily clutters the graph. Thus, when no relation is depicted between two technologies, they are compatible.

4.7. CASE STUDY: INDUSTRY TECHNOLOGY SET

To test the applicability of the method to another practical test case, a set of technologies from a study in aircraft industry is taken and evaluated using the proposed method. The purpose of this demonstration is to verify whether the diverse set of technologies that may be encountered in industry can be modeled using the graph transformation ap-

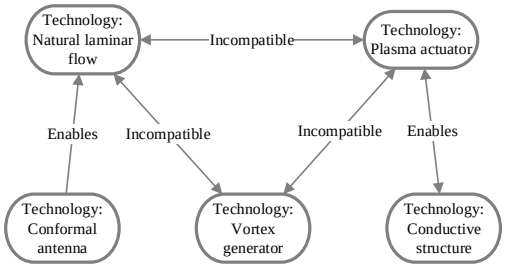


Figure 4.7: The technology compatibility graph (TCG), a generalization of the TCM. Absence of a relation between two technologies indicates compatibility.

proach, and whether the inference rules apply to these cases. The particular set of technologies considered here apply to a complete aircraft. They are distinct from the ones shown in the previous test case. However, due to confidentiality issues, the technologies can not be explicitly displayed, and are replaced by generic identifiers. The differences between the technology compatibility matrices that are generated by experts and by the method are shown in Figure 4.8. As Figure 4.8 shows, there are only few differences between the automated method and the TCM as established by a group of experts. The first

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
T1		-1	-1		-1	-1						
T2			-1			-1						
T3						-1						
T4												
T5						-1						
T6												
T7												
T8												1
T9		1	1									
T10							1					
T11					1					1		
T12												

(a) TCM by experts

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
T1		-1	-1		-1	-1						
T2			-1		-1	-1						
T3					-1	-1						
T4												
T5						-1						
T6												
T7												
T8												1
T9		1	1									
T10							1					
T11					1							
T12												

(b) TCM by method

Figure 4.8: TCM comparison between experts and the current method. A value of -1 denotes incompatibility, while a value of 1 denotes enabling and is a directed relationship (thus, the 1 to the far right reads that T8 enables T12, whereas the 1 to the far left reads that T9 enables T2).

difference is the incompatibility between technologies T2 and T5, and T3 and T5. Both pairs are assessed as incompatible by the current method, due to an inhibiting process, with Equation 4.10. However, the reason the experts do not agree is because of the scale of the two processes. The inhibiting process does occur in the same region as the other process, but is much smaller and does not have a significant effect on it.

The second difference is the enabling relation between T11 and T10. The inability of the current method to detect this, is a result of a lack in modeling capabilities. Concretely, the graph rewriting rule implementation does not allow for optional patterns to be specified, which is required for the correct implementation of T10. Thus, in practice, this limitation of the current method may prevent the correct implementation of

some technologies. Luckily, however, most technologies won't have to rely on optional patterns. Nonetheless, optional patterns should be included in the method, to extend the technology pattern graphs. An optional pattern is matched against the system graph only when it is present in that graph, but is not a requirement for the application of the technology.

4.8. CASE STUDY: FACTORIO

In order to demonstrate the portfolio generation algorithm and maximum dissimilarity principle, another case study is conducted based on the game Factorio. Factorio is a computer game where the player builds items from resources. These items can be used to construct a factory which automates the process of producing items, as shown in Figure 4.9. The ultimate goal of the game is to build a rocket that allows the player to leave the planet they are stranded on. The player starts with only very simple recipes and materials and has to progressively research more advanced recipes and create more intricate components. The world is divided into square cells, and each building (e.g. a conveyor belt, production machine, furnace or robot arm (called inserters)) takes up one or several of these cells. Effectively, a factory constructed this way is a system. Different technologies can be thought up too: faster inserters or conveyor belts, more efficient production machines and electric furnaces instead of fossil fuel based ones. As such, Factorio appears to be an excellent candidate for testing the methodology set out in this chapter, with only graph transformation rules (as there is no actual physics involved in Factorio) and the maximum dissimilarity approach to reduce the technology portfolio space.

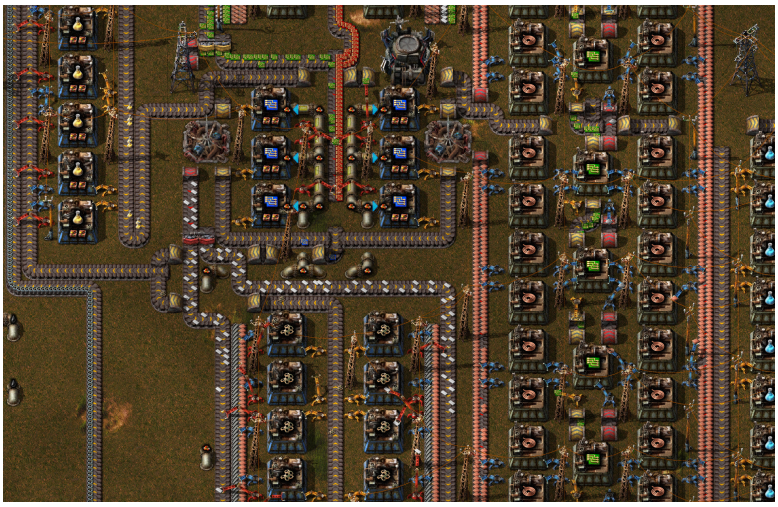


Figure 4.9: A screenshot from the game Factorio

Table 4.1: Component types in the Factorio graph

Component	Symbol	Additional Properties
Chest		Capacity C (500)
Conveyor Belt Straight Conveyor Belt Clockwise Conveyor Belt Counter-clockwise		Capacity C (4) ItemsPerSecond $\dot{m}$ (2)
Inserter		Capacity C (1) RotationsPerSecond ω (1) Reach l (1)
Factory		RecipyId f_{id} (0) Productivity P (0)

4

4.8.1. SYSTEM REPRESENTATION

A Factorio factory can be represented as a graph, where each node is a component, with at least three attributes: the x location, the y location and its rotation r. Both x and y are integer values indicating the index of a grid cell in the world, whereas r can take one of four values: (0) up, (1) right, (2) down, and (3) left.

The different components are presented in Table 4.1. Each component introduces some unique attributes that contains additional information about that component's working. The number in parentheses denotes the default value for that attribute. A chest is a passive component that simply contains items up to its capacity. A conveyor belt transports items from one side of a cell to another side of the cell. An inserter grabs items from an adjacent cell and deposits them on the cell opposite of it. Finally, a factory receives items and consumes them to produce a different item according to some recipy.

After placing components in a grid, an algorithm infers which components connect to other components. For example, two straight conveyor belts are connected if:

- They have identical x coordinate and are adjacent in y and both point either up or down, or
- They have identical y coordinate and are adjacent in x and both point either left or right

Likewise, any type of conveyor belt may be connected to any other type of conveyor belt. Conveyor belts also connect to inserters. Inserters either connect to chests, factories, conveyor belts or other inserters.

The items (or materials) that are present in this simulation of Factorio are presented in Table 4.2. Furthermore, the recipies that factories can use to craft other items are given in Table 4.3. The number in parentheses denotes the amount of inputs or outputs.

Now a factory is specified as in Table 4.4. A graphical depiction of this factory is shown in Figure 4.10. The factory retrieves items from the bottom chest and transports

Component	x	y	r
Chest	1	0	Up
Inserter	1	1	Up
CB S	1	2	Up
CB S	1	3	Up
CB S	1	4	Up
CB S	1	5	Up
Inserter	2	3	Right
Factory	3	3	Right
Inserter	4	3	Right
Inserter	2	4	Right
Factory	3	4	Right
Inserter	4	4	Right
Inserter	2	5	Right
Factory	3	5	Right
Inserter	4	5	Right
CB S	5	3	Up
CB S	5	4	Up
CB S	5	5	Up
CB S	5	6	Up
Inserter	5	7	Up
Chest	5	8	Up

Table 4.4: Example factory as a list of components, with their horizontal x and vertical y positions, and orientation r.

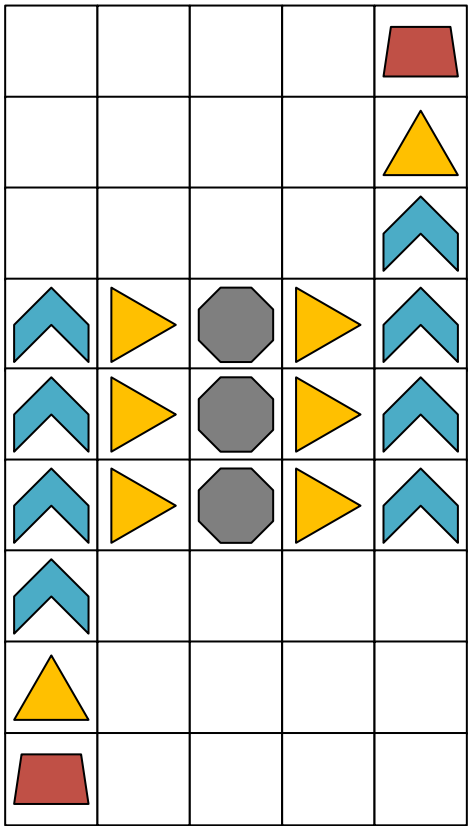


Figure 4.10: Example factory from Figure 4.4 visualized. The symbols are defined in Table 4.1

Table 4.2: Item types in the Factorio graph

Item
Copper Ore
Iron Ore
Copper Plate
Iron Plate
Copper Wire
Green Circuit

Table 4.3: Recipies in the Factorio graph

Recipy	Inputs	Outputs	Crafting time (s)
Copper Plate	Copper Ore (1)	Copper Plate (1)	0.5
Iron Plate	Iron Ore (1)	Iron Plate (1)	0.5
Copper Wire	Iron Plate (1)	Copper Wire (2)	0.5
Green Circuit	Iron Plate (1) Copper Wire (8)	Green Circuit (1)	0.5

them to three factories that operate in parallel. Their outputs are transported to and stored in the upper chest.

4.8.2. TECHNOLOGY SET

A set of five technologies is defined to show the deduction of the TCM, portfolio generation and maximum dissimilarity approach. They are portrayed in Table 4.5. Each technology has an affected component, which is a node in the factory graph that is used as a pattern for the technology graph rewriting rule. The pattern column specifies a constraint on the value of an attribute in that pattern. The same holds for the effect column, which targets the effect graph of the rewriting rule. Finally, the NAC column specifies a constraint over an attribute value that may not be present in the system in order for the technology to be applied.

Table 4.5: Technologies for the Factorio example factory

Technology	Affected component	Pattern	Effect	NAC
Fast Inserter	Inserter	$\omega \leq 1$	$\omega = 2$	$C \geq 2$
Stack Inserter	Inserter	$C \leq 1$	$C = 4$	$\omega \geq 2$
Red Belts	Conveyor Belt	$\dot{m} \leq 2$	$\dot{m} = 3$	
Blue Belts	Conveyor Belt	$\dot{m} \leq 2$	$\dot{m} = 4$	
Factory Productivity	Factory	$P \leq 1$	$P = 1$	

Table 4.6: Technology Incompatibility for the five Factorio technologies. True = incompatible.

	Fast Inserter	Stack Inserter	Red Belts	Blue Belts	Factory Productivity
Fast Inserter	-	True	False	False	False
Stack Inserter		-	False	False	False
Red Belts			-	True	False
Blue Belts				-	False
Factory Productivity					-

4.8.3. QUANTITY OF INTEREST AND ANALYSIS

Suppose the quantity of interest for the factory described above is the time it takes to transform N plates of copper into $2N$ copper wires. To analyze this, the chest at $x = 1, y = 0$ is filled with N copper plates at time $t = 0$. Then the factory is simulated until the chest at $x = 5, y = 8$ contains $2N$ copper wires. Obviously, the recipe in each of the three factories is set to the copper wire recipe from Table 4.3.

4.8.4. TECHNOLOGY COMPATIBILITY MATRIX

Through the use of the graph transformation incompatibility rules, including those for negative application conditions, which were explained in subsection 4.2.1, the compatibility matrix for the five technologies presented in the previous subsection is derived. It is shown in Table 4.6. The result is as expected: the two inserter technologies are incompatible as a result of their NACs. That is because the fast inserter sets ω to 2, whereas the NAC of the stack inserter triggers when $\omega \geq 2$. Similarly, the stack inserter sets C to 4, while the NAC of the fast inserter requires $C < 2$. The belt technologies are incompatible because they result in different values for the ItemsPerSecond attribute $\dot{m}$.

4.8.5. PORTFOLIO GENERATION

There are no inapplicable technologies, and no technologies that enable others. Thus, the TCM is the only information that limits the possible amount of portfolios. The resulting portfolios are shown in Table 4.7. As one can see in Table 4.7, there are only 18 portfolios. However, had the TCM not been there, there would have been $2^5 = 32$ portfolios.

4.8.6. MAXIMUM DISSIMILAR PORTFOLIOS

Before the dissimilar portfolios can be computed, all portfolio distances are computed. Both the full GED algorithm (from Appendix C) and the faster algorithm described in subsection 4.5.1 are used. As expected, they both result in the same distances between the portfolios, which are shown in Table G.1. Note that the distances are symmetric. Obviously, the distance of a portfolio to itself is zero.

From these distances, the maximum dissimilarity algorithm is run for $K = 9$ (i.e. half of the total amount of portfolios). The first portfolio (baseline, without any technolo-

Table 4.7: Technology Portfolios for the five Factorio technologies

Portfolio	Fast Inserter	Stack Inserter	Red Belts	Blue Belts	Factory Productivity
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	1	0	1	0	0
5	0	1	1	0	0
6	0	0	0	1	0
7	1	0	0	1	0
8	0	1	0	1	0
9	0	0	0	0	1
10	1	0	0	0	1
11	0	1	0	0	1
12	0	0	1	0	1
13	1	0	1	0	1
14	0	1	1	0	1
15	0	0	0	1	1
16	1	0	0	1	1
17	0	1	0	1	1

Table 4.8: Maximum dissimilar subsets for the Factorio portfolios

Run 1	0	17	13	7	5	2	12	1	6
Run 2	0	14	16	4	8	3	15	1	11
Run 3	0	13	17	5	7	11	1	15	12

gies) is always included as the first pick. This results in the maximum dissimilar subsets in Table 4.8. The reason there are multiple maximum dissimilar subsets is because sometimes several portfolios tie in distance to the current subset, and a random pick is made between them.

For Run 1, the softmax probabilities are shown in Table G.2. Here, each row is one of the 18 portfolios, and each column is a portfolio in the maximum dissimilar subset. The entry then is the probability of the row portfolio being similar to the representative column portfolio. Thus, each row should sum to 1. Clearly, the portfolios in the maximum dissimilar subset have a probability of 1 of belonging to their representative class of portfolios. The fact that there are non-zero entries listed for these portfolios can be attributed to numerical error.

4.8.7. COMPUTATION OF QOI AND PROJECTION ONTO PORTFOLIO SET

For the three runs of the dissimilarity algorithm in the previous subsection, the softmax probabilities are used to compute estimates of the time the factories take to produce 200 copper wires (i.e. $N = 100$). Table 4.9 summarizes the actual time each portfolio takes,

Table 4.9: Portfolio QoI as computed using softmax for the Factorio example. For each run, the root-mean-squared (RMS) error and accuracy are provided, which measure how well softmax approximates the original results.

Portfolio	Actual time (s)	Run 1	Run 2	Run 3
0	206	205.999	205.999	205.999
1	105	105.000	105.000	105.000
2	106	106.000	106.000	106.000
3	206	205.999	205.999	205.999
4	105	105.000	105.000	105.000
5	77	77.000	73.000	77.000
6	206	205.999	205.999	205.999
7	105	105.000	105.000	105.000
8	77	56.000	77.000	56.000
9	206	205.999	205.999	205.999
10	105	105.000	105.000	105.000
11	106	105.999	105.999	105.999
12	206	205.999	205.999	205.999
13	105	105.000	105.000	105.000
14	73	77.000	73.000	77.000
15	206	205.999	205.999	205.999
16	105	105.000	105.000	105.000
17	56	56.000	77.000	56.000
RMS error	-	21.4	21.4	21.4
RMS % [a]	-	14.3	14.3	14.3
Accuracy[b]	-	89%	89%	89%

[a] RMS as percentage of the range in values in the QoI, which is $206 - 56 = 150$.
[b] Defined as the amount of correctly estimated portfolios divided by the total amount of portfolios.

along with the estimates for each run of the dissimilarity algorithm. As it shows, most portfolios have their factory running time estimated correctly. Only for portfolios 5, 8, 14 and 17 this is not always the case. Run 1 and 3 both contain portfolios 5 and 17 in their subsets, and thus predict those with perfect accuracy. However, portfolio 8 is under-predicted as a result and portfolio slightly over-predicted. Conversely, Run 2 contains portfolios 8 and 14 in its subset, leading to a slight under-prediction for portfolio 5 and an over-prediction for portfolio 17. The reason that the estimates are so dependent on the inclusion of these portfolios in the subset can be attributed to the large discrepancy in distances between several portfolios. As Table G.1 shows, most portfolios have costs larger than or equal to 16 to other portfolios. However, for each portfolio, there is one other with a cost of only 3. This is the portfolio that is identical to it, but with the addition of technology 5: the factory productivity. This technology only incurs an edit cost of 1 for each instance it is applied to. There are only three factories in the graph, so this technology is applied three times. The other technologies have more applications in the graph, and therefore incur higher edit costs.

Even though the results in Table 4.9 are acceptable, they reveal how dependent the result of the maximum dissimilarity approach is on the specification of the graph edit cost that is used to compute the portfolio distances. It is left for further research to investigate how different graph edit cost specifications may lead to better representation of differences between portfolios. Clearly, in this example, technology 5 has a large impact on the objective function, but incurs a small edit cost. In other words, the edit cost does not accurately reflect how influential certain technologies are. For this approach to work reliably, this problem needs to be addressed.

4.9. DISCUSSION

For the presented approach to work well, the graph transformation patterns should correctly match to the system graphs. However, when symmetries are present in either, ambiguities result as a graph matching algorithm identifies multiple matches. Concretely, this may happen when a node in the pattern is considered equivalent with multiple nodes in the system graph. When less information is present, this issue becomes more prevalent. Therefore, equivalence determination is one of the cornerstones of this approach. Further research has to be conducted to establish robust ways to determine equivalence for dispositions and processes.

The implemented approach suffers from the main issue that all rule-based reasoning systems suffer from: the rules are crafted manually and, therefore, extending and maintaining the rule set is labor-intensive. Much rather, we would like a framework that learns the rules automatically. Here, link prediction in graph convolutional networks (GCN) may prove an interesting solution. Each node represents a technology, and edges between them their (in)compatibility to others. Then, a GCN can learn from existing instances the features that make certain technologies incompatible, and others compatible. The big question is to what extent such a method is able to generalize those patterns and truly learn a useful mapping from a technology pair to a compatibility statement. The other option is to let analysts have more control to specify how technologies are allowed to interact with other through the graph transformation rules. Then, (in)compatibility is made explicit in the patterns, effects and (negative) application conditions of the technologies, making the approach more labor intensive, but also more structured and transparent.

Another key insight is that technology compatibility is not a well-defined concept. During the rule construction, it became clear that there is hardly ever a physical reason why technologies are incompatible. Mainly, incompatibility results from design synthesis considerations where redundancy and complexity are to be avoided as much as possible. This corresponds with the functional incompatibility introduced in this chapter. Physical incompatibility in the sense that material entities cannot intersect should be inferred, whereas the process inhibition is only subjective. Only when functions of one technology inhibit functions of the other, the technologies are incompatible. Only transformation incompatibility can be used generically, whenever graph transformation rules are used to represent technologies.

4.10. CONCLUSION

With the ontology from Chapter 3 the technology compatibility matrix (TCM) can be constructed in two ways:

- With graph transformation rules and negative application conditions
- With graph transformation rules and rules in (first-order) logic

The approach with negative application conditions is more flexible, but also requires more rework. The logic rules can be reused across projects. Both approaches allow the TCM to be automatically constructed, which saves practitioners a considerable amount of effort. Because the construction is automated, it is more consistent and traceable.

In combination with additional rules or NACs, the approach also allows to infer which groups of technologies enable other technologies. When a technology is not applicable to the baseline system, this technique can find sets of technologies that are applicable. Together, these enabling relationships and the TCM reduce the size of the portfolio set considerably, which alleviates the evaluation effort. Technology enabling is absent in the technology evaluation literature; hence, can be seen as a novel addition to tackle the combinatorial optimization problem.

When evaluating all portfolios remains intractable, the maximum dissimilarity technique may be employed. A Latin Hypercube Sampling (or similar DOE technique) approach would not work for the combinatorial design space of technology portfolios. The knowledge graphs developed in Chapter 3 again prove useful for the maximum dissimilarity algorithm, because a notion of distance between the technology portfolios is required. While a vector of technology impact factors would provide Euclidean or Manhattan distance measures, it is deemed inappropriate, because not all portfolios could be captured using a single vector. Graph (edit) distance (GED), on the other hand, can be applied to any knowledge graph. Because the costs of the edit operations are an input to the GED algorithm, they can be fine-tuned to approximate the best notion of semantic equivalence. (Note that this would also partially cover the issue of granularity, exposed in section 3.7.)

The approach is demonstrated on a case study with the game Factorio, where a set of 18 portfolios is represented by a maximally dissimilar subset of 9 portfolios. Subsequently, those 9 portfolios were simulated and the results on the quantity of interest projected back on all of the 18 portfolios using the softmax function. Each run of the maximum dissimilarity algorithm resulted in a slightly different subset, all of which misrepresented two portfolios. In terms of accuracy, that equates to 89%. The root-mean-squared error as a percentage of the range in QoI values was 14%, which is significant. Better specification of the GED costs would remedy this error.

A final contribution from this chapter is the implementation of a GED algorithm that specifically focuses on graph transformation rules. It is shown how it substitutes the computation of GED over two knowledge graphs, by only considering the transformation rules that were used to arrive at those two graphs. While the original algorithm is NP-hard, this algorithm runs in polynomial time; as a result, it requires significantly less computation time.

In conclusion, the approach to automating portfolio generation in this chapter suffers from three drawbacks. First, it relies heavily on graph matching, which fails when

different levels of granularity are used in pattern and host graphs. Second, the logic rules are manually crafted and hard to generalize. That is why the approach with NACs is preferred. Third, technology incompatibility is not a clearly defined concept, which ties in with the previous issue in the sense that it depends on the practitioner's viewpoint and thus is better modeled using NACs than FOL.

Regardless of the method that is used to construct the technology compatibility graph, the resulting technology portfolios have to be evaluated on some quantities of interest that underlie the selection decisions down-the-line. The next chapter focuses on how a system with a technology portfolio applied to it can be evaluated for several QoIs semi-automatically. Again, those developments rely on the ontology and graph-based description of systems and technologies.

5

TECHNOLOGY PORTFOLIO EVALUATION

Taking hybrid-electric powertrains for aircraft as an example, there is a vast amount of research papers that develop an analysis framework to investigate several hybrid-electric aircraft concepts. Unfortunately, these frameworks are generally specifically tailored to those hybrid-electric concepts, and are not able to investigate, say, wing-movable technologies. What's more, the frameworks only tackle a certain discipline (e.g. electrical power), and the models they employ do not allow other disciplines (e.g. mass distribution) to be analyzed. As a result, different technologies have to be analyzed using different frameworks with different sets of assumptions and models. Therefore, considerable (re)work is required to be able to evaluate the novel technology concepts. The aim of this chapter is to structure and automate that process.

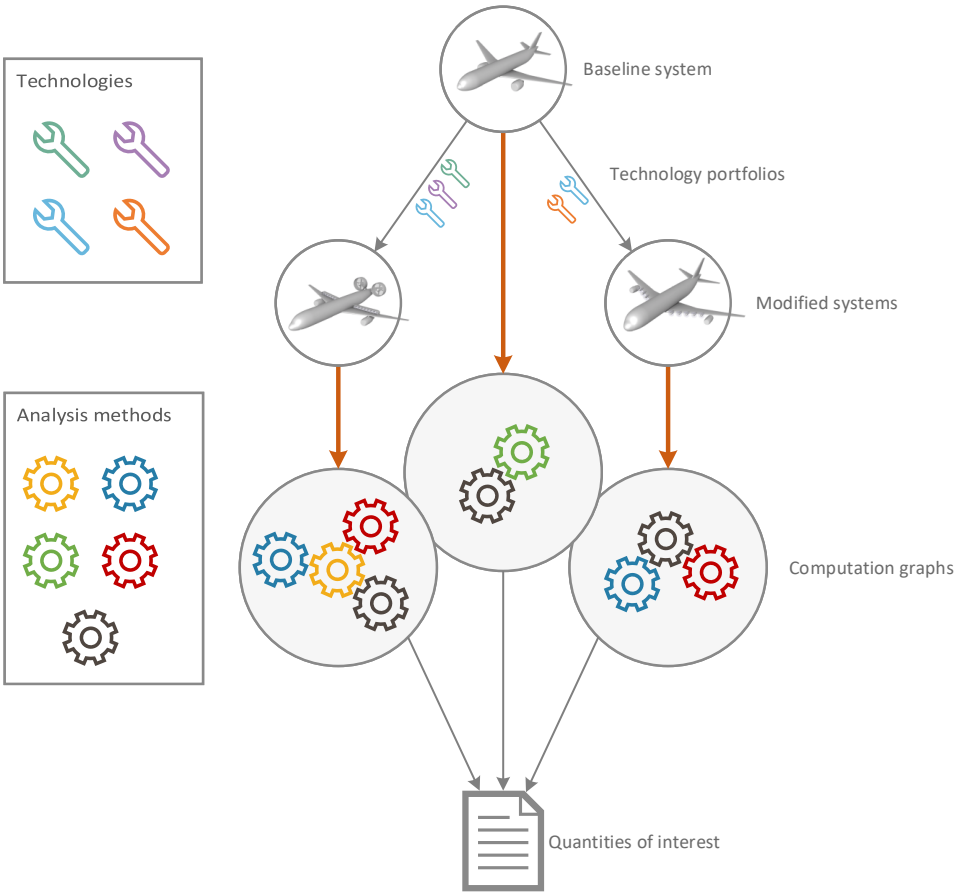


Figure 5.1: Overview of technology portfolio evaluation. Different technologies lead to different systems, which have to be analyzed using some set of available analysis methods. The mapping from system to a computation graph is depicted with the bold, orange arrows, and is the process that this chapter focuses on.

The problem statement is graphically depicted in Figure 5.1. A baseline system leads to different systems through the application of technology portfolios. Each modified

system has to be analyzed to compute the same quantities of interest, but not any system can fit in the same analysis framework. Therefore, a suite of analysis methods is introduced. Each analysis method focuses on a certain part of a system and on a certain discipline. Then, the method introduced in this chapter maps these analysis methods to the modified systems to construct so-called computation graphs. These computation graphs contain the sequence of computations required to quantify the quantities of interest. The framework presented in this chapter then simply executes the analysis methods in the computation graphs in the correct order.

One might wonder how the present method positions itself with respect to the existing practices of model-based systems engineering (MBSE) and multi-disciplinary design optimization (MDO). The short answer: it complements both and can be integrated in existing workflows. The long answer is given in the following two paragraphs for each field separately.

MBSE is systems engineering as before, but aided with a (computer-based) model, that serves as the authoritative source-of-truth, instead of a collection of separate documents (in the form of text and images) as was the case in legacy systems engineering [142]. Commonly, SysML is used as the language to make that model up with. However, other languages are also possible. The systems engineering approach toward product developments consists of three stages: system analysis, system development and system integration. These stages together form the V model, see Figure 5.2. The system analysis phase may be broken up into four parts: requirements engineering, functional design, logical design and physical design. These together make up the RFLP approach, which comprises the descending branch of the V model [143]. Because the present method is more concerned with design space exploration, the R and F parts of the approach are not as important at this stage. However, the L and P parts align with what has to be modeled for the present approach to enable simulation of the system. Our approach enhances the MBSE approach by automating the coupling of analysis methods to the system model. So where conventional MBSE requires those links to be specified manually, the present approach automates that process. Additionally, the ability to investigate multiple technologies at the same time, with no redundancy, is not supported by conventional MBSE methods.

Regarding MDO, several methods have been proposed to construct an MDO workflow from a set of analyses. As Tosserams et al. [144] points out, a decomposition-based approach entails three steps:

1. Specifying the variables and functions of each discipline

2. Specifying the partitioned problem (i.e. the distribution of variables and functions over sub-problems and systems)

3. Coordinating the solution of the partitioned system

According to Tosserams et al. [144], the first two steps have received considerably less attention, which is why they, and for example the method from Alexandrov and Lewis [145, 146] automate step 2. The present method focuses more on the first step. So where the aforementioned methods depart from a set of analysis methods already applied to a

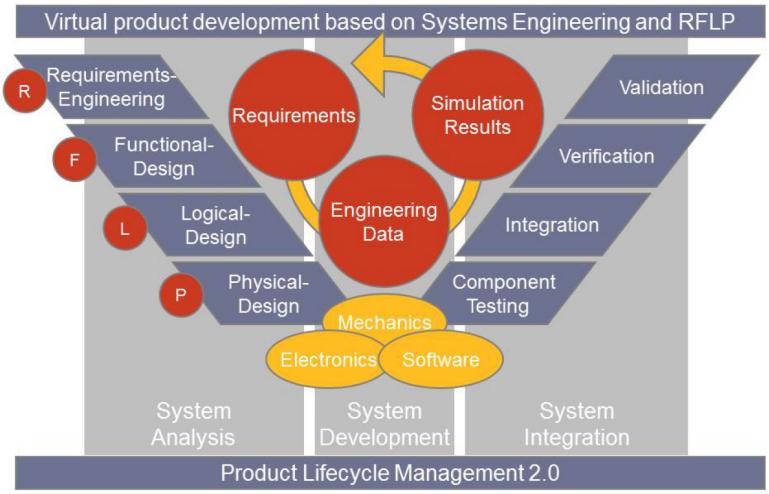


Figure 5.2: The RFLP approach in the V-model as the descending branch. Taken from [143].

given system, the current method automatically sets up that information from the system specification alone.

In the next section, section 5.1, we clarify in more detail how the present approach fits in with both MBSE and MDO. Furthermore, the approach itself is presented in detail. Then, two case studies illustrate how the method could be applied in practice. First, subsection 5.2.1 models a hybrid-electric powertrain for aircraft and demonstrates how a power model and analysis apply to that system. Second, subsection 5.2.2 shows a full aircraft mission analysis. Here, it is demonstrated how a cyclic dependency graph is unrolled in time to compute fuel burn throughout a flight. Furthermore, causal reasoning is used to identify input variable dependencies, for a consecutive probabilistic assessment of the fuel burn. The merits and drawbacks and outlook of the method are discussed in section 5.3 and the chapter is concluded in section 5.4.

5.1. METHODOLOGY

As Figure 5.1 shows, a way to map a unique system to a computation graph is required. One of the research questions from section 1.5 addresses this problem: "How to analyze (novel) aircraft technologies in a consistent, reliable and robust manner, such that their (combined) effects are characterized, with uncertain input metrics/parameters?" There are three parts to the answer:

1. Each individual analysis method has to be mapped to a relevant part of the system under investigation. This process simultaneously parameterizes the system.

2. The mapped analysis methods have to be combined to form a complete computation sequence that results in the quantification of the QoIs.

3. Uncertainty and dependencies regarding the input variables have to be specified

and taken into account in the computation of the QoIs.

The first item may be solved through (sub)graph isomorphism matching, based on the ontology and knowledge graphs developed in Chapter 3. This part is where the present approach complements both MBSE and MDO practices. The use of an ontology allows graphs to be semantically matched, if proper definitions are used in the ontology. PSO is a suitable ontology to describe engineering systems and already contains some ways to include analysis methods as well. However, the mapping from system to analysis method has to be stated manually. Furthermore, the PSO method appears unable to string multiple analysis methods together to form a holistic analysis sequence. Additionally, there does not seem to be an actual implementation of PSO. Therefore, this chapter adds to the PSO ontology only a few relations and presents several algorithms that form a framework around it to solve the problem defined in the previous section.

In MBSE, SysML is typically used as the language to describe systems. Why is SysML not used in this work? The reason is that SysML is not an ontology and therefore lacks strong definitions. In other words, the final user has too much freedom in specifying systems according to their interpretation, which deteriorates sharing and reusing the knowledge captured in SysML. Nonetheless, as Figure 5.3 shows, information captured in PSO terms could be translated into SysML, and vice versa. Effectively, this entails that the concepts that make up SysML should be brought under concepts known in PSO. Through this translation then, the current method can be integrated in existing MBSE workflows, or the other way around.

The work on automated execution of MDO frameworks [53–56, 144–146] address the second point. As such, as soon as the present approach has established a mapping from the system to the analysis methods, the inputs for those works could be generated automatically, as shown in Figure 5.3. There are two slight differences between those works and the current one, though.

1. All of these works view an analysis method as a black-box with a fixed set of (unique) input variables, and a fixed set of output variables. For each method, those variables require a unique name to be distinguished from one another. Based on their name, an input from one method can be matched to the output of another [54, 55]. Tosserams et al. [144], Alexandrov and Lewis [146] even require those links to be specified manually. Ensuring unique names across the system requires the analyst to specify them a priori. The current method automates this and, therefore, allows for a more flexible naming approach. The current method, furthermore, allows an analysis method to have a flexible amount of input or output variables. This is particularly helpful in situations, where, for example, a summation over forces, masses or other quantities is required.

2. All the above works also assume a fixed direction in which an analysis method works. When both the options A->B and B->A would be present, they'd be linked and an infinite loop would result. Otherwise, those methods cannot decide which of the two directions to use. Therefore, the analyst has to specify manually which method has to be used for that particular design problem. The present method is able to automate these decisions. It does this by first constructing the dependency graph, which contains all the possible computation directions (so both A->B, and

B->A) and then prune those directions that are unnecessary for a given problem (defined by the QoIs and input variables).

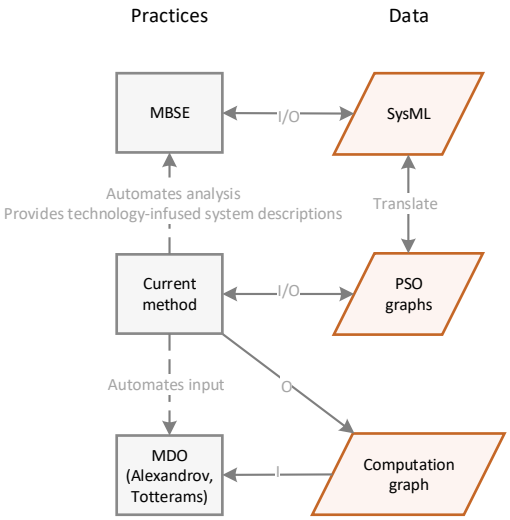


Figure 5.3: Overview of how the present method may be integrated with existing methods for MBSE and MDO.

The XDSM format to display an MDO architecture [147] could be used to display the computation graphs generated by the present framework, if a translator is developed. By extension, the same holds for exporting the computation or dependency graphs as a plain design structure matrix.

As explained in Chapter 2, Monte Carlo Simulation (MCS) can be used to propagate uncertainty distributions through an analysis sequence, to tackle part of item 3. However, when uncertainty quantification and propagation are performed, dependencies between the random input variables should be established. To support an analyst in this, a method is proposed in this chapter, which identifies those input variables that are physically and causally related to one another, and prompts those relationships to the user. The analyst can then decide if there indeed is a dependency, and what that dependency looks like. How this information is used, is discussed extensively in Chapter 6.

The three items discussed above are combined in the method proposed in this chapter. The method allows the automated coupling of (parts of) an engineering system to suitable analysis methods, execute the aggregated analysis, and use the result to further characterize the system. A holistic overview of the proposed method is shown in Figure 5.4. There are two phases that divide the workflow of the present method: the organization phase and the project phase. The organization phase establishes the information that is persistent across different projects. In other words, the mathematical models, analysis methods and domain causality rules identified and specified by the knowledge engineer are problem independent and can be used for different technology evaluation projects. This is why all of them are stored in a database, as indicated by the

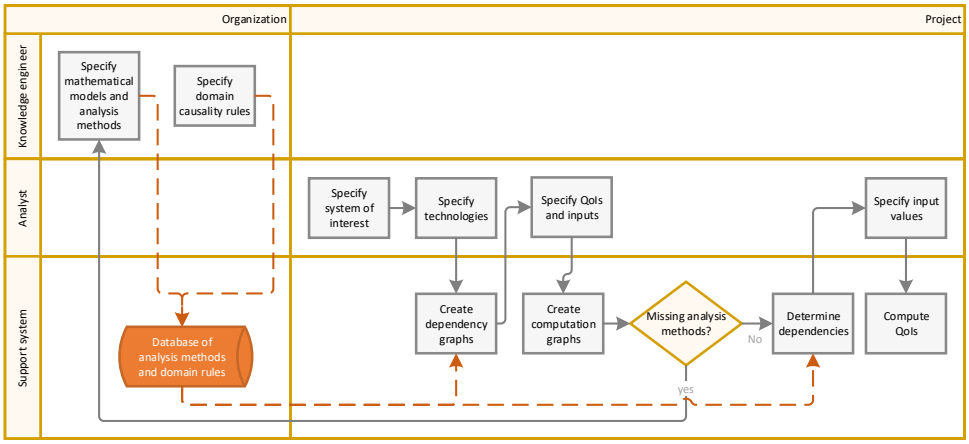


Figure 5.4: Overview of the steps in this method. The orange, dashed arrows indicate flow of information. The grey arrows are the process flow. The organization phase is independent of a technology evaluation project, and, therefore, the database is persistent across projects. The only situation to return to the organization phase is when a project requires analysis methods that are not yet existent in the database.

orange, dashed arrows.

The project phase kicks off with an analyst specifying the system of interest (baseline system) and the technologies they wish to investigate. Both are specified as knowledge graphs, which are then fed into the support system that is the framework this chapter develops. The support system creates a dependency graph for each of the modified systems (i.e. the baseline system plus a technology portfolio), using the analysis methods present in the database. The analyst then has to specify which variables are quantities of interest, and which are inputs. Here, the approach differs from the ones of Alexandrov and Lewis [145, 146] and Tosserams et al. [144], because they specify a single objective function for the optimization problem. In the current method, multiple QoIs can be specified, because only analysis is performed, rather than an optimization. The dependency graph is now pruned such that the QoIs are computed from the known inputs. It might occur that certain intermediate variables or QoIs cannot be computed, because an appropriate analysis method is lacking. In that case, the method will prompt the user that this is the case, and the knowledge engineer has to specify a new analysis method that solves the issue. When a computation graph is constructed, the support system identifies those input variables that physically depend on one another, provided the domain causality rules established in the organization phase. Then, the analyst provides all necessary input values and the quantification of the QoIs can commence.

The subsequent subsections describe how analysis methods are represented and applied to a system graph, to generate the dependency graph and computation graphs.

5.1.1. REPRESENTING ANALYSIS MODELS

The ontology as described in Chapter 3 allows one to specify a physical system. However, no computations can be performed, yet, because there are no variables describing the system numerically, and no analysis methods are present, nor the knowledge of what

these analysis methods can compute. Mapping analysis methods onto the knowledge graph that describes a system should be divided into two steps: creating a mathematical model and mapping that mathematical model to an analysis method. The mathematical model maps the physical entities onto information content entities (i.e. numerical variables). Thus, it provides a parameterization of the physical, real world. For example, a rigid body is modeled as a point mass, with a velocity and acceleration in one point. An analysis method takes a certain model of reality and computes some of the variables from others in the model. There may be multiple analysis methods that work with the same model, e.g. an Euler solver or full Navier-Stokes solver both represent the fluid with the same model (finite volume discretization), but make different assumptions about certain values and relationships between qualities in that fluid. The approach is sketches notionally in Figure 5.5, and is explained in detail in the following two subsections.

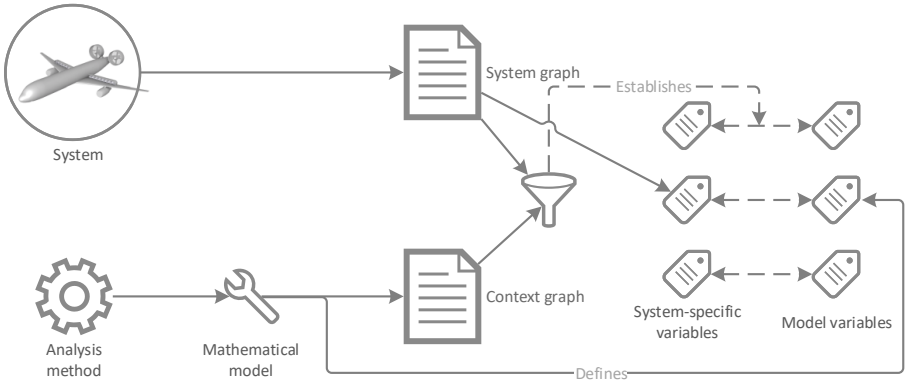


Figure 5.5: Principle of matching analysis methods to a system. The context graph describes a portion of reality that the analysis method processes, and is matched onto the system graph that describes the system of interest. The match is depicted with the filter icon. That match establishes a mapping between the variables describing the system and the model variables, which are specified by the mathematical model accompanying the analysis method.

MATHEMATICAL MODELS

A mathematical model is defined as a graph transformation, that takes a knowledge graph and adds attributes to it. Each of these attributes is labeled with a model context, e.g. "point mass model". Within such a context, each variable is uniquely identified by its name. Across different contexts, therefore, variables with identical names may exist, but it cannot be determined conclusively whether these would be the same variable.

Thus, a mathematical model m is a quadruple (l, H, A, μ), where l is the label, $H = (V_H, E_H)$ is a graph describing the physical system, and $\mu = A \mapsto V_H$ is a surjective function that assigns each attribute A to a vertex in V_H. An attribute A is an ICE (e.g. a PSO Mesh). In Figure 5.5, the context graph is H and the model variables are A.

Taking a physical system graph G, the pattern H may be found using subgraph isomorphism, leading to a set of matches $m_i : H \overset{m_i}{\mapsto} G$. This is depicted with the filter icon in

Figure 5.5. Then for each match the function μ is used to create copies of the attributes A, called A', and add them to G. Those copies are the system-specific variables in Figure 5.5, and the process of creating them is indicated with the dashed arrow annotated with "establishes". So first we generate the mappings $\mu_i = A' \mapsto V_G$, and then create a new graph $G' = (V_G, E_G, A')$. This new graph G' is simply the system graph parameterized with the model variables.

The subgraph isomorphism problem to find the matches m_i is solved using an implementation of a graph edit distance (GED) algorithm by [148], which is based on the well-known A* algorithm and the assignment problem [149]. To make the GED algorithm solve subgraph isomorphism, the cost function as specified by [80] is implemented to make it equivalent to the maximum common subgraph problem. A constraint is superposed that states that the maximum common subgraph be isomorphic to the pattern H.

A separate algorithm infers default attributes and their relationships. These include an object's mass, velocity and acceleration, for example. Additionally, the fact that acceleration is the time derivative of velocity, is added to the graph. The algorithm also identifies which elements of vectors are zero and non-zero. That information is used later on to decompose vector quantities into their constituent elements, to infer that $\boldsymbol{L} = -\boldsymbol{W}$ and $\boldsymbol{T} = -\boldsymbol{D}$, instead of $\boldsymbol{L} + \boldsymbol{W} + \boldsymbol{T} + \boldsymbol{D} = 0$. While the latter equation is also correct, it is less informative than the former two, because it does not recognize that the lift force $\boldsymbol{L}$ only acts in the vertical direction, the weight force $\boldsymbol{W}$ only in the downward direction, the thrust force $\boldsymbol{T}$ in the forward direction, and, finally, the drag force $\boldsymbol{D}$ only in the backward direction [1].

ANALYSIS METHODS

The attributes that a mathematical model generates are related, and an analysis model specifies how. It takes a set of attributes that originate from the same mathematical model and maps them onto an input/output data structure for a specific analysis method / tool. Several modes of an analysis dictate in which directions the computation can be performed, e.g. mass from acceleration and force, or acceleration from mass and force.

An analysis is represented with a tuple (l, v, f_i), where l is the label of the mathematical model the analysis operates on. The surjective function $v : A \mapsto P$ maps the mathematical model attributes A to a set of parameters P of the analysis. Attributes are the representation of physical properties. Parameters are the input and output variables to an analysis method. While in many cases a one-to-one relation exists, making this distinction merely an abstraction, the same parameter could capture multiple attributes. For example, multiple forces acting on the same point mass are all mapped onto the same analysis parameter F. The analysis method could then compute the sum of all attributes mapped to F. Finally, the set of functions $f_i : p_{\text{out}} \in P = f_i(p_{\text{in}} \in P), p_{\text{out}} \notin p_{\text{in}}$ are the different computations the analysis enables. Each f_i computes a parameter p_{out} from a set of input parameters p_{in}. For Newton's second law, for example, there are three f_i: one that computes mass, one that computes acceleration and one that computes one of the forces (which is valid for any of the forces).

[1] This force-breakdown is a simplification of the actual forces on an airplane, but is a reasonable first assumption and used in many undergraduate courses on aircraft aerodynamics. It suffices for the demonstration of the method in this chapter.

5.1.2. DEPENDENCY GRAPH

Once all matches of mathematical models are found in the graph G and the attributed graph G' is constructed, all analyses that have been applied to G are collected. For each analysis, the computation modes f_i are taken. An input $I(f_i) = p_{\text{in}} \in P$ specifies the input parameters of the computation and an output $O(f_i) = p_{\text{out}} \in P$ the output parameters. Taking the set of analysis parameters P, the algorithm determines the attributes in G' that correspond to them by applying all mapping functions in order: $A_I = m_i \circ \mu \circ \nu \circ I$ for the input attributes and $A_O = m_i \circ \mu \circ \nu \circ O$ for the output attributes.

The dependency graph D (this is similar to the Fundamental Problem Graph [56]) is created by including all attributes $A_I \cup A_O$ from G' obtained by applying the two aforementioned input/output mapping functions. In addition, analysis nodes C are included which represent a triple (a, f_i, m_i). Here, a is the analysis and f_i its computation mode, and m_i the mathematical model instance to which the analysis is applied. The two disparate sets of nodes are connected by directed edges from A_I to C and from C to A_O. Therefore, a bipartite graph results that displays the input/output information flow of all analyses applicable to G. The full algorithm is provided in Appendix H.

5

ATTRIBUTE RELATIONS

The graph G' may also include attribute relations, that indicate an attribute is equal to another, or is the (time) derivative of another. These relations are captured in the ontology with the **is derivative of** and **is equal to** relations between ICEs. The first relation is always attributed with a reference to the ICE that the derivative is with respect to, usually one that represents time. The second relation is self-explanatory: it dictates that the values of two ICEs are equal for any given time.

Axioms defined in a domain-specific ontology can define how attribute relations are inferred. For example:

$$a \textbf{ is derivative of } v \Leftarrow \forall a, v, p \ \wedge \ \text{ICE:acceleration}(a) \ \wedge \ \text{ICE:velocity}(v) \ \wedge \ \text{quality:position}(p) \ \wedge \ v \textbf{ is about } p \ \wedge \ a \textbf{ is about } p \tag{5.1}$$

Similarly, (written in natural language for clarity) for any two material entities with the disposition for electrical conductance, that are connected (both have the same relational quality PSO:connection), and each has an ICE power that **is about** the electric energy quality each entity possesses, then those powers **are equal to** another.

Furthermore, when an analysis can be applied element-wise to a vector quantity, a decomposition of the input variables and a combination of the output variables is added as an analysis node. For example, a force is a three-dimensional quantity. Therefore, the sum of forces can be decomposed into the three Cartesian dimensions x, y and z. Each force gets decomposed by an analysis node into its three components (which are added to D as attribute nodes). Then for each dimension, an analysis node performs the addition of the force components. Finally, the summed components are composed back into a total force variable, see Figure 5.6.

Also included is a zero-mask ICE that is about any vector quantity and forces certain elements to zero. For example, in a 2D coordinate system, the weight force always points downwards, so the direction is [0, -1]. This means the x component of the force vector will always be zero. A zero-mask in the form of [0, 1] informs us that the x component

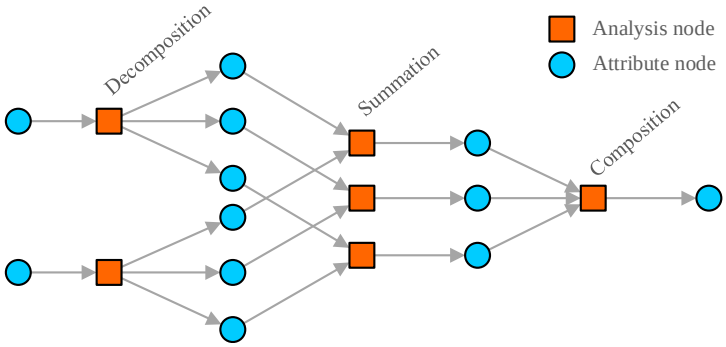


Figure 5.6: Decomposition and composition of vector quantities

will be zero, while the y component can assume any value. Zero masks are used in vector decomposition to prevent zero entries from being added to D as attribute nodes.

5.1.3. COMPUTATION GRAPH

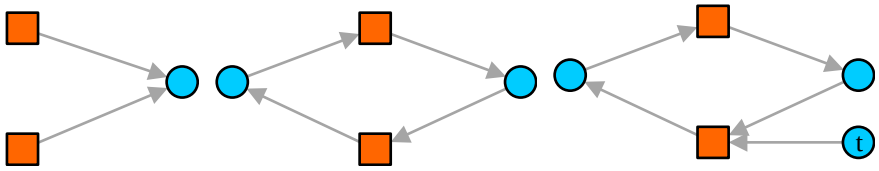
The dependency graph contains all the possible directions in which information may flow, and, therefore, it is problem independent. A problem is defined as a set of QoIs and known variables, specified by the analyst, as shown in Figure 5.4 at the start of this section. Therefore, each attribute has an associated role. The role is either "known", "QoI" or "unspecified". By default each attribute has the unspecified role. After the system is parametrized, the practitioner has to establish which attributes are QoIs and for which ones the values are known. Additionally, each attribute is either constant or time-varying.

The computation graph is a subgraph of the dependency graph that specifically computes the indicated QoIs from the indicated knowns. Two preprocessing steps are performed before the computation graphs are created. The first step involves removing all incoming edges of known variables. Because their value is predetermined, there is no need to compute these (which is what an incoming edge implies). The second step is to remove all equality analysis nodes from the graph. All attributes that reflect the same quantity are replaced with a single attribute. The roles are assigned appropriately to these replacement attributes. For example, when x is a known and equal to y, which is unspecified, then x' is known. The same is true for the QoI role. However, when x is known and y is a QoI, while they should be equal, an error results, because these roles are conflicting.

To construct a computation graph, an algorithm starts by creating a single computation which contains all the QoIs and no analyses. It proceeds by extending this graph by prepending an analysis that computes one of the QoIs, along with the input attributes to that analysis. When multiple analyses are present in the dependency graph to compute a single QoI, an additional computation graph is generated for each such an analysis. The process repeats for the remaining QoIs and the newly added input attributes. A computation graph is complete when all the input attributes (leaf nodes) are knowns and each QoI is computed (i.e. has a predecessor). This process may generate multiple computa-

tion graphs, which use different analyses at different points.

Any computation graph has to be transformable into a directed acyclic graph (DAG) and no attribute node may have more than one predecessor (see Figure 5.7(a)). The above algorithm ensures the latter condition. When the graph does contain cycles (see Figure 5.7(b)), these have to include a derivation or integration term, such that the cycle is broken along the temporal dimension (see Figure 5.7(c)). (Concretely, $x^{(t+1)}$ is computed from $x^{(t)}$ and Δt.) Then for a time-step t the graph is acyclic.



(a) Invalid (b) Invalid (c) Valid

Figure 5.7: Valid and invalid structures in a computation graph

Note that the choice to only allow cycles that have a temporal break in them is made to ensure the computation graph is determinate. However, it is possible to allow for cycles, as long as they represent a converging loop of calculations. This is impossible to ensure for black-box analysis methods. Nonetheless, if that is desired, the constraint on a derivation or integration term in a cycle should be dropped. In that case, there should be no cycles where different modes of the same analysis are called on the same set of attributes. Such a condition would lead to an infinite loop where a is computed from b and vice versa. The full algorithm is provided in Appendix I.

5.1.4. CAUSAL REASONING FOR DEPENDENCIES

From the above approach, multiple computation graphs may result, each of which is equally valid. The designer chooses which computation graph to use, depending on the analysis methods used in that graph. For a deterministic (single-point) computation the designer specifies the values of the known variables and the computation graph can be executed in order to obtain the QoIs. However, when performing a probabilistic computation, joint dependency structures should be imposed on the input variables, or at least on those which have a dependency [88]. Determining which variables have a dependency can be difficult, especially for practitioners non-versed in probabilistic computations. To alleviate this effort, a reasoning mechanism is introduced that determines which variables are physically related. Those pairs of variables are then presented to the designer, who has to determine what dependency exists (which may be absent, nonetheless) and specify a value for it.

Specifically, the relation **causally influences** was introduced in section 3.5 as a subrelation of PSO **physically related to**, to specify that an entity causes a change in another entity, without specifying what kind of change. As an example of a rule, consider aerodynamics, where the geometry of an object submersed in a gas affects the pressure in the

gas. It is written as follows:

$$\begin{aligned} g \textbf{ causally influences } p \Leftarrow & \text{ geometry}(g) \wedge \text{ pressure}(p) \wedge \\ & \text{solid}(s) \wedge \text{ spatial region}(sr) \wedge \text{ gas}(f) \wedge \\ & s \textbf{ located in } sr \wedge f \textbf{ occupies } sr \wedge \\ & p \textbf{ characterizes } f \wedge g \textbf{ characterizes } s \end{aligned} \tag{5.2}$$

One rule that is domain independent is based on a reasoning mechanism in Bayesian networks: V-structures. The idea is that when two variables causally influence a third variable, information cannot directly flow from the one parent to the other. However, when the child variable is known, or constrained, a causal relationship results between the parent variables. Thus, the rule reads:

$$\begin{aligned} q_1 \textbf{ causally influences } q_2 \Leftarrow & \text{ quality}(q_1) \wedge \text{ quality}(q_2) \wedge \text{ quality}(q_3) \wedge \\ & q_1 \textbf{ causally influences } q_3 \wedge q_2 \textbf{ causally influences } q_3 \wedge \\ & \exists q_4 : q_3 \textbf{ is constrained by } q_4 \end{aligned} \tag{5.3}$$

A reasoning mechanism fires this set of rules on the attributed graph G' until no more inferences are made. Then, when a computation graph is chosen, the leaf nodes (those with no predecessor) are collected, and for each pair the algorithm checks if there exists a **causally influences** relationship between them. For each pair with that relationship, the designer is prompted to specify a dependency structure.

5.1.5. QOI COMPUTATION

For a deterministic computation, the designer is requested to provide values for all the input variables. For constant variables, only one value should be specified, while for time-varying variables the value at every time-step has to be specified by the analyst, as shown in Figure 5.4. The computation is performed step-wise by marching through time. For each time-step, which is specified by the analyst, the computation graph is executed. The computation graph computes at least one variable in the next time-step, which kick-starts the following iteration. Obviously, when there is no temporal dependency, the computation graph is only executed once.

When a probabilistic simulation is to be performed, the dependency data in the form of copulas and probability distributions on random variables have to be provided in addition to the input data for a deterministic computation. This is where the reasoning mechanism of the previous section is employed. It indicates the pairs of input variables, for which it is likely a dependency exists. The analyst can then either specify a full joint distribution (an example is shown later in Figure 5.18), or merely a correlation coefficient. The method then creates a user-specified amount of samples from the distributions and runs a deterministic computation of the graph for each sample, to consecutively build a Cumulative Distribution Function (CDF) on the QoIs. This is identical to the standard Monte Carlo Simulation process.

5.2. APPLICATION

Two case studies show how the method may be applied in practice. The first focuses on the analysis of hybrid-electric powertrain architectures for aircraft. It is shown how (a part of) the physical system graph G is constructed. Then, a small mathematical model and analysis are discussed that analyze the power throughput of a generic hybrid-electric powertrain architecture. Consecutively, it is shown the dependency graph and one of the possible computation graphs, that is time-independent.

The second case study looks at a full aircraft mission analysis. It considers a fuel-based aircraft, for which the fuel burn depends on the mass of the airplane, which decreases as fuel is burnt. Therefore, a time-dependent computation graph results. Furthermore, this case study shows how force decomposition is employed to construct the dependency graph. Finally, it is shown how a probabilistic calculation may be performed on a flap technology that alters the aerodynamic behaviour of the aircraft throughout the mission and show what the effect is on the fuel burn.

5.2.1. HYBRID POWERTRAIN

As de Vries et al. [46] demonstrate, a set of hybrid-electric powertrain architectures can be analyzed for conceptual design through a generalized model. This model is the Serial-Parallel Partial Hybrid (SPPH) architecture, shown in Figure 5.8. By setting certain parameters to zero and modifying the signs of the power input and output, nine different hybrid-electric architectures are obtained. The disadvantage of this approach is that it is only able to capture those nine architectures. Furthermore, if you assume the airplane is propelled by more than one propulsor (e.g. four propellors) they would need to be grouped into one item in the model of Figure 5.8.

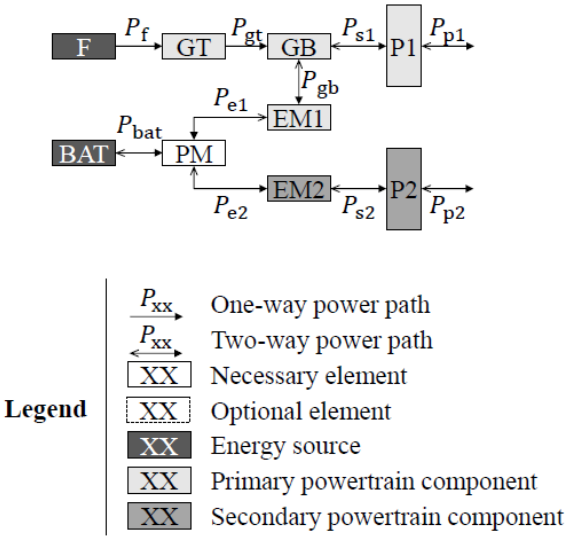


Figure 5.8: Schematic of the serial-parallel partial hybrid architecture (from [46])

In the subsequent sections, the steps from Figure 5.4 are detailed for the evaluation of hybrid-electric powertrain architectures. To start, an electromotor and battery are modeled to form a system, and then the mathematical models and analyses are applied to that. The sections continue by deriving the dependency graph and computation graph and analyzing the power throughput for the full SPPH architecture.

SPECIFY SYSTEM OF INTEREST

The knowledge graph describing an electromotor and battery is depicted in Figure 5.9. It shows how the electromotor has a process that transforms energy. That process operates on the electric input and shaft output fiat object parts. These contain energy and have the function to transmit energy. Finally, roles specify what these fiat object parts are with respect to the electromotor's behavior. The same is true for the battery, although

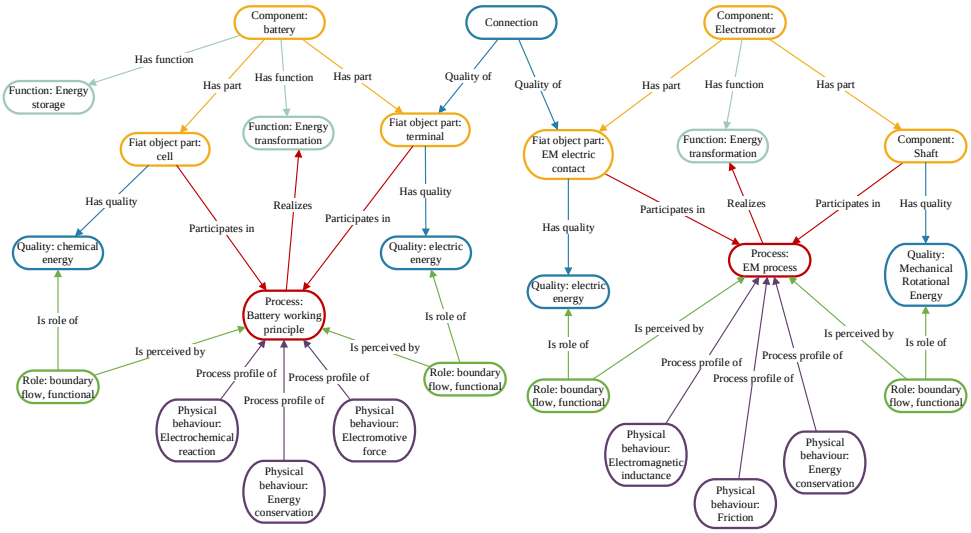


Figure 5.9: Knowledge graph depiction of an electromotor connected to a battery. The color coding only serves to more easily distinguish between the graph elements.

the physical behaviours differ and, obviously, it contains chemical energy and no mechanical energy. The knowledge graph shows a generalized representation of these two components.

While only the battery and electromotor are shown in this section, the same process can be repeated for the other components in Figure 5.8, to construct the entire powertrain architecture. As the resulting graph is very large, it is omitted here for the sake of brevity.

SPECIFY MODEL AND ANALYSIS

The mathematical model for conversion of energy in any single component is depicted in Figure 5.10. Observe that the energy qualities do not specify what type of energy is concerned; indeed, this is irrelevant to the model. Furthermore, the energy conversion process can take any form as long as it can be captured by the energy conservation

Table 5.1: The function μ for the power model

Attribute (label)	Vertex	Quantity type	Unit	Context
Power in (P_{in})	Energy 1	Power	W	Power model
Power out (P_{out})	Energy 2	Power	W	Power model
Efficiency (η)	Component	Non-dimensional	%	Power model

law. The colored portions of the graph indicate subgraphs that may be repeated multiple times in the source graph. The model then expands to include all those subgraphs. This way, multiple input and/or output energy flows can be considered using this one model.

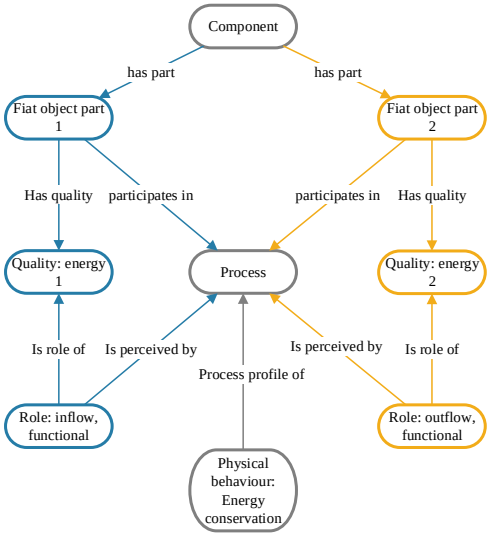


Figure 5.10: Model for energy transformation systems. Blue and yellow subgraphs may be repeated in the engineering system's graph.

Attributing a model instance following the function μ is shown in Table 5.1. Based on the role assigned to an energy quality[2], the appropriate power-in attribute or power-out attribute is added. The context of each attribute is the same.

Now, the analysis takes an instance of a power model (thus, l = power model) and applies an identity mapping through ν. That is, the parameters P correspond directly to

[2]It may be confusing why power is used to describe energy, rather than energy content. We view energy as a first-class entity that has a certain content. Conservation of energy is then expressed as the total energy content in a system remaining constant when there is no net influx or efflux of energy. However, when there is an influx or efflux, it is easier to express them using the time-derivative of energy — power —, instead of an amount of energy accumulated over time. Hence, the energy conservation law is expressed herein as equating the power influx and efflux.

the model attributes A. Then the following three modes f_i are available:

$$\begin{aligned} P_{\text{in},i} &= \sum P_{\text{out}} / \eta - \sum_{\setminus i} P_{\text{in}} \\ P_{\text{out},i} &= \sum P_{\text{in}} \cdot \eta - \sum_{\setminus i} P_{\text{out}} \\ \eta &= \sum P_{\text{out}} / \sum P_{\text{in}} \end{aligned} \tag{5.4}$$

Another model takes care of recognizing the connection instance in Figure 5.9 and an equality analysis applies to the involved energy qualities, equating their power attributes. For example, the battery terminal power is equal to the electric power input to the electromotor.

CREATE DEPENDENCY GRAPH

Before a model can be generated from the system knowledge graph, the direction of each energy flow has to be determined. As de Vries et al. [46] describe, there are nine feasible combinations of directions. Appendix J elaborates on how that is automated in this dissertation. However, it suffices to say here that we can figure out those different operating configurations of the powertrain architecture, and randomly pick one to continue the demonstration.

To each of the component instances, the power analysis equations can be applied, resulting in the dependency graph in Figure K.1. As can be observed, this graph is very cluttered; all possible directions of computations between variables are included. Recall that this happens, because the quantities of interest and inputs are not yet specified at this point. Figure K.1 also shows orphan variables nodes (the blue nodes without any edges connecting them). These are variables that are not used or computed by any analysis method. They naturally appear for any system, and it is not necessarily a problem, unless any of them has to become an input or QoI, evidently.

Perhaps, the construction of a dependency graph could be skipped to generate a computation graph directly when the QoIs and inputs are known. Nonetheless, the dependency graph is computationally expensive to generate, but has to only be done once. Generating the computation graph from the dependency graph is a cheap operation, and has to be done every time a different set of inputs or quantities of interest is evaluated. Thus, it is better to only once construct the dependency graph and then generate multiple computation graphs from it, than to generate those computation graphs directly, which would increase the total computational cost.

CREATE COMPUTATION GRAPH

While the dependency graph shows all the different things that could be computed, a computation graph is required to show how to compute a certain quantity of interest (QoI) from several knowns. Before such a computation graph can be generated, however, the analyst should define which variables are known, and which are the QoIs. Taking that as a starting point, the procedure described in subsection 5.1.3 generates all possible computation graphs.

In this particular study, suppose the QoI is P_{bat}, while the knowns are P_f, P_{p1} and P_{p2}. This problem leads to the computation graph in Figure 5.11. As one might expect,

it shows how P_{gt} is computed from P_f, P_{s1} from P_{p1} and then P_{gb} from P_{gt} and P_{s1}, and so forth. Note that there is only one computation graph for this problem. This is not necessarily true for all problems, because there may be different ways to compute the same variable. Traversing the computation graph and providing values for the input variables P_f, P_{p1}, P_{p2} and η_i leads to a value of the QoI P_{bat}.

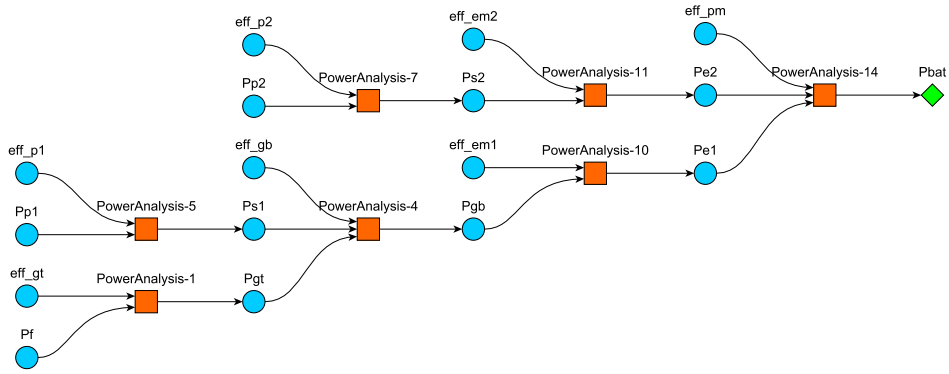


Figure 5.11: Computation graph for SPPH powertrain architecture

COMPUTATION EXAMPLE

To show that the presented method can perform the powertrain analysis, Table 5.2 contains the results of a calculation using the appropriate computation graph for each of the nine SPPH configurations. The input variables — P_{p1}, P_{p2}, P_{f} and the component efficiencies — are generated randomly for each architecture. The computation graph then is used to compute the intermediate power variables and finally produces the value for the quantity of interest: P_{bat}.

5.2.2. AIRCRAFT MISSION ANALYSIS

The second application is a full mission analysis of a subsonic transport aircraft with the infusion of a novel technology: a second-degree-of-freedom flap. The idea behind this technology is that it can be used to optimize the aerodynamic efficiency of the airplane throughout the mission. It shows the method's capabilities to construct a complete computation graph for an aircraft mission analysis, including time marching and force decomposition. Furthermore, it shows how this computation is modified as a result of technology inclusion. Finally, it is shown how dependencies between some technology variables are determined and perform both a deterministic and probabilistic computation of the fuel mass. Again, the subsequent sections follow the structure of Figure 5.4.

SPECIFY SYSTEM OF INTEREST

The full aircraft description is shown in Figure 5.12. In summary, it is constructed as a component with three sub-components: a wing, an engine and a fuel tank. While this does not constitute an entire aircraft, it suffices for the purpose of demonstrating the method.

Table 5.2: Computations of P_{bat} for each of the nine configurations from Table J.1. The input variables (above the first horizontal line) are generated randomly. The others are computed using a computation graph as in Figure 5.11.

Variable	S1	S2	S3	S4	S5	S6	S7	S8	S9
P_{p1}	1	1	1	4	4	2	2	2	1
P_{p2}	3	1	2	1	3	3	4	1	1
P_{f}	2	5	3	5	3	5	3	3	3
η_{gt}	0.8	0.9	0.7	0.6	0.8	0.7	0.9	0.8	0.9
η_{gb}	0.8	0.9	0.8	0.8	0.7	0.8	0.6	0.6	0.7
η_{p1}	0.9	0.8	0.7	0.9	0.8	0.7	0.6	0.8	0.7
η_{p2}	0.9	0.9	0.7	0.7	0.6	0.8	0.7	0.9	0.7
η_{em1}	0.8	0.7	0.8	0.6	0.7	0.8	0.9	0.8	0.7
η_{em2}	0.6	0.8	0.7	0.8	0.7	0.6	0.9	0.7	0.9
η_{pm}	0.9	0.9	0.7	0.9	0.8	0.7	0.6	0.9	0.6
P_{gt}	1.6	4.5	2.1	3	2.4	3.5	2.7	2.4	2.7
P_{gb}	0.169	2.8	0.251	2.556	4.743	0.071	2.34	2.4	2.38
P_{s1}	1.111	1.25	1.429	4.444	5	2.857	1.2	1.6	0.7
P_{s2}	3.333	1.111	1.4	1.429	1.8	2.4	5.714	1.111	0.7
P_{e1}	0.135	1.96	0.201	4.259	6.776	0.089	2.106	1.92	1.666
P_{e2}	5.555	1.389	0.98	1.786	1.26	1.44	6.349	1.587	0.63
P_{bat}	6.038	0.375	0.827	6.717	7.209	0.919	8.476	0.141	1.378

SPECIFY MODELS AND ANALYSES

Showing all the model pattern graphs and maps is omitted for brevity. Nonetheless, the pattern graphs of the chemical conversion and lift–drag polar analysis models are shown in Figure 5.13(a) and Figure 5.13(b), respectively. Table 5.3 shows the set of models and analyses used to construct the dependency graph in the following subsection. In addition, particular analysis methods deal with hierarchies of objects. To elaborate: the mass of a component is equal to the sum of its constituent masses. Likewise, component forces add up to a resultant force. Finally, for a rigid body, considered as a point mass, the velocity and acceleration of each sub-component is equal to the system's velocity and acceleration.

Table 5.3: Methods and analyses for mission analysis

Model/analysis	Parameters	Equations
Storage	m_{tot}, m_{fixed}, m_{f}, $\dot{m}_{\text{f}}$	$m_{\text{tot}} = m_{\text{fixed}} + m_{\text{f}}$, $\dot{m}_{\text{f}} = \frac{\partial}{\partial t} m_{\text{f}}$
Newton	F_i, m, a	$\sum_i F_i = m \cdot a$
Aero normalization	F, C_F, ρ, V, S	$F = C_F \cdot S \cdot \frac{1}{2} \rho V^2$
Chemical conversion	SE, $\dot{m}_{\text{f}}$, P	$P = SE \cdot \dot{m}_{\text{f}}$
Force power	F, P, V	$P = F \cdot V$
Lift-drag polar	C_L, C_D, A, e, C_{D_0}	$C_D = C_{D_0} + \frac{C_L^2}{\pi A e}$
Gravity	m, g, W	$W_x = 0, W_y = 0, W_z = -m \cdot g$

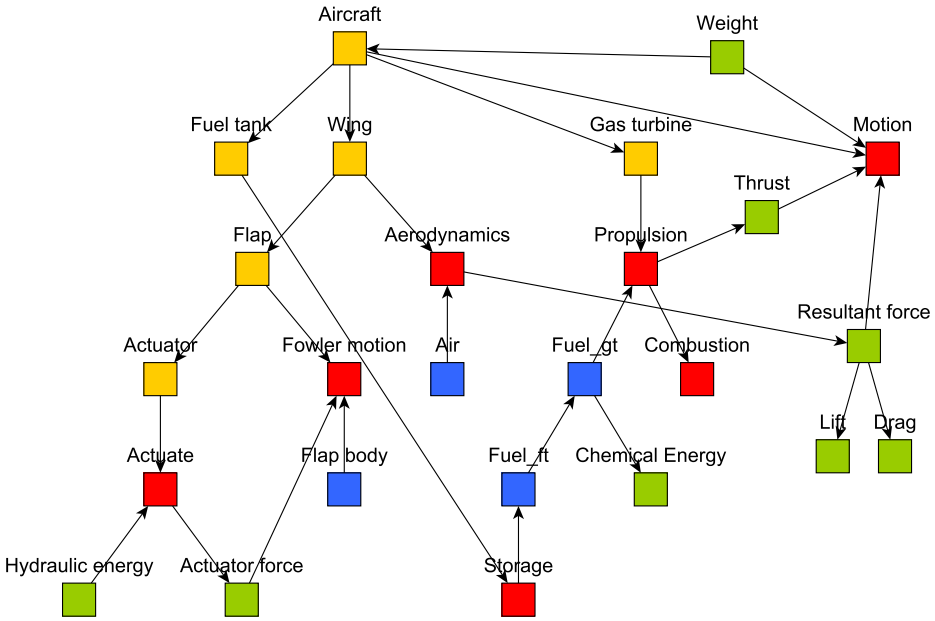


Figure 5.12: System graph for simplified conventional aircraft. Yellow: components, Red: behaviours, Blue: material entities, Green: energy and force.

CREATE DEPENDENCY GRAPH

The full dependency graph for the aircraft mission analysis including the flap technology is depicted in Figure K.2. This section focuses on a part of its construction: force decomposition. In a two-dimensional mission analysis, an aircraft is considered as a point mass, and a force balance exists during flight: $\boldsymbol{L} = -\boldsymbol{W}$ and $\boldsymbol{T} = -\boldsymbol{D}$, i.e. lift is equal to (in magnitude) the weight and thrust is equal to drag. Therefore, the vertical and horizontal acceleration are zero during steady flight.

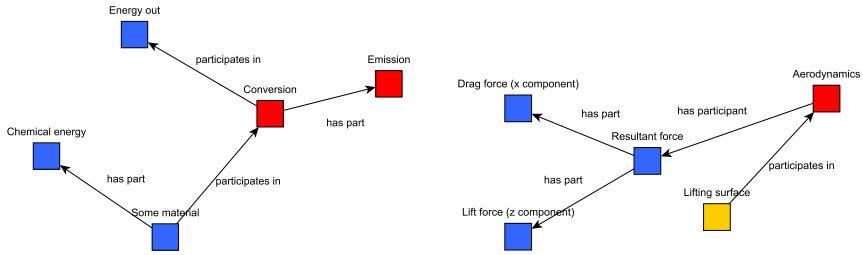
Naively, applying Newton's second law to the aircraft as a whole leads to the equation:

$$\boldsymbol{L} + \boldsymbol{W} + \boldsymbol{T} + \boldsymbol{D} = m \cdot \boldsymbol{a} \quad , \tag{5.5}$$

where, even if we know that $\boldsymbol{a} = 0$, three of the four forces need to be known for the last to be computed. This is illustrated in Figure 5.14(a). However, only the weight force is known; the others have to be computed. As humans, we intuitively recognize that lift and weight act vertically, while thrust and drag act horizontally. Then, the above equation can be split up and lift can be computed from weight. The lift-drag polar analysis[3] is used to compute drag, which in turn gives thrust.

This is where the zero-masks from subsection 5.1.2 come into the picture. Each of the four forces is attributed with a zero-mask, providing information about the direction

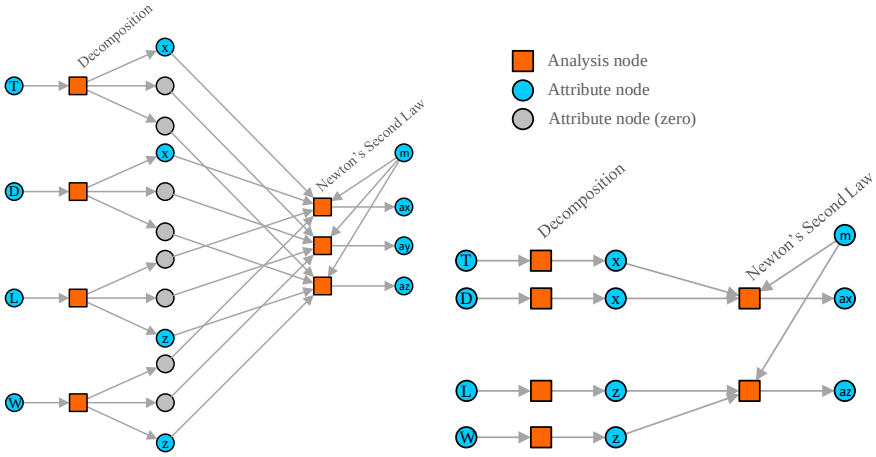
[3] From physics, we know that the drag of a body is dependent on the lift as well as the profile drag of that body. We capture this by means of a so-called drag polar, a second-degree polynomial function that relates the non-dimensional drag coefficient (C_D) to the non-dimensional lift coefficient (C_L): $C_D = C_{D,0} + kC_L^2$.



(a) Chemical conversion (b) Lift–drag polar

Figure 5.13: Pattern graphs for two analysis methods

of the forces. The grey attribute nodes in Figure 5.14(a) illustrate this. Simply removing



(a) Full graph (b) Reduced graph

Figure 5.14: Dependency graph for Newton's Second law before and after applying zero-masks

these nodes and their edges from the dependency graph results in Figure 5.14(b). Here, the force balances in x and z direction are decoupled, resulting in the intuition that $L = -W$ and $T = -D$!

CREATE COMPUTATION GRAPH

Ultimately, fuel burn is the quantity of interest for an aircraft mission analysis. Therefore, either the fuel mass contained in the fuel tank is of interest, or the mass flow of fuel out of the tank. The former is chosen. The known variables are shown in Table 5.4. Given those specifications, the computation graph is automatically constructed, resulting in Figure 5.15. The QoI is shown in the bottom-right corner, as a node with a diamond shape. The overall flow of computation is clockwise. Circles are attribute nodes, while

Table 5.4: The attribute roles for a mission analysis

Attribute	Role	Constant
m_{f}	QoI	False
m_{ft}	Known	True
m_{gt}	Known	True
m_{w}	Known	True
C_{D_0}	Known	True
A	Known	True
e	Known	True
ρ	Known	False
S	Known	True
a_{ac}	Known	False
V_{ac}	Known	False
SE	Known	True
t	Known	False
δ_{f}	Known	False
k_e	Known	False

squares are analysis nodes. Any attribute node without predecessors is a leaf node, and hence an input variable, from Table 5.4.

Notice how the computation graph is cyclic. Therefore, the QoI depends on itself. Luckily, there is a time derivative in the cycle; the QoI is the integral value of the fuel mass flow, which is computed from the thrust force at a certain time-step. Thus, the computation graph can be unrolled in time, and is, therefore, computable.

Additionally, observe that Figure 5.15 contains the technology variables δ_{f} (flap_rot) and k_e (2dofflap_k_e) as input variables, which modify through an analysis method the C_{D_0} and e of the wing. This is a simple example, where the technology effect is modeled as a k-factor, but in a more elaborate example, the whole lift–drag polar analysis could have been replaced with another analysis method as a result of introducing the flap.

The reason to represent the flap technology as in Figure 5.15, is that it is implemented as a flap schedule that changes the camber of the wing during the entire flight. As a result, the lift-to-drag ratio is optimized for each point in the mission, which is required due to the changing air density ρ and aircraft mass m_{ac}.

COMPUTE QOIS

The computation graph shown in Figure 5.15 can be executed in a deterministic setting, where each variable has one value, and if it is time-varying, has one value per time-step. The QoI — fuel mass — is computed at all points in time, as shown in Figure 5.16(a). Additionally, the values of all other variables in the computation are also stored, such that, for example, the lift coefficient is also accessible, as Figure 5.16(b) shows. The full details of the computations performed in this subsection are presented in Chapter 6.

COMPUTE QOIS IN PROBABILISTIC SETTING

The novel flap technology illustrates how dependencies are deduced. It has two attributes: the flap rotation δ_{f} and an Oswald span-efficiency-factor impact k_e. Rules are

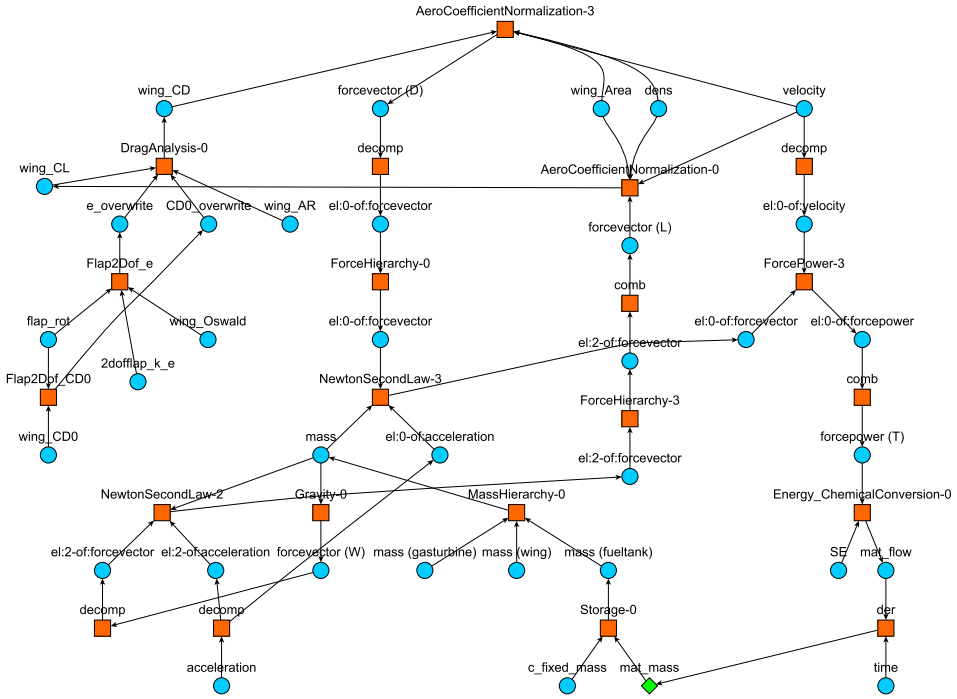
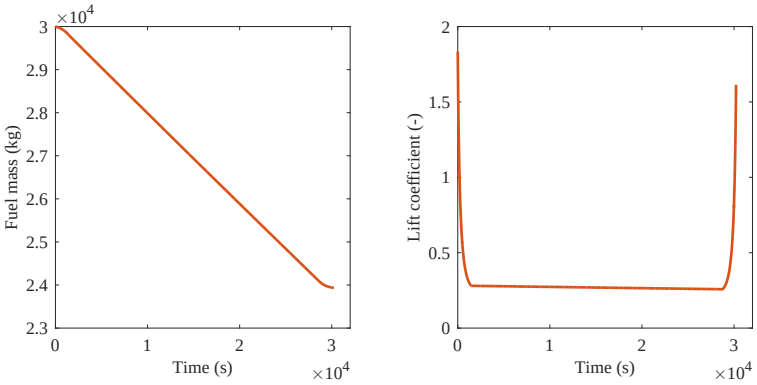


Figure 5.15: Computation graph for mission analysis of fuel-based aircraft. The QoI is the diamond shaped node in the bottom–right corner. The general flow of computation runs clockwise.

added that state that the flap rotation influences the flap geometry, which in turn affects the airflow around it. Because that is the same airflow as the one around the wing, the stress in the wing structure is influenced. Because the stress is constrained during design by the material-based stress allowable, the mass of the wing structure is influenced as well. This is an example of a causal relationship between the flap deflection and the wing mass. All the **causally influences** relations deduced by the algorithm are depicted in Figure 5.17.

Without the design constraint on the wing stress, the flap deflection would merely affect the wing stress, and geometry. However, the wing structure volume and mass would remain unaffected.

Both the wing structure mass and the flap rotation are input variables in the computation graph (see Figure 5.15). Therefore, a dependency structure has to be specified between these two variables. An example of the dependency between these two variables is shown in Figure 5.18. Each dot in this figure represents a possible combination of the variables, and the density of dots represents the probability of those combinations occurring. This dependency indicates that there exists a negative correlation: the higher the maximum flap deflection, the lower the wing mass, due to load alleviation. Furthermore, the higher the maximum flap deflection, the more uncertainty exists regarding the mass reduction; hence, the higher spread.



(a) Fuel burn (b) Lift coefficient

Figure 5.16: Fuel mass and lift coefficient over time from mission analysis

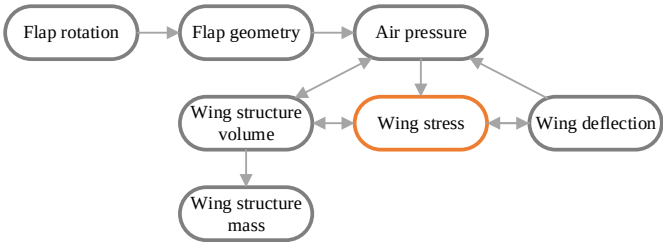


Figure 5.17: Causal influence relations between variables in the mission analysis example. The orange box around wing stress indicates that it is a constrained variable.

Including the flap and the dependency structure from the previous section, whilst also providing probability distributions to (some of) the input variables, enables the computation graph to be executed for each sample. This is a Monte Carlo simulation and results in a probability distribution on the quantity of interest, as shown in Figure 5.19. Note that the fuel burn in Figure 5.19 means the block fuel burn over the entire flight. In other words, it is the total amount of fuel burnt during the mission. The probability distribution collects the frequency of those amounts occurring as a result of uncertainties regarding fuel specific energy, component efficiencies and masses, mission parameters, and the technology variables.

The result for the baseline aircraft, without the flap technology, is also shown in Figure 5.19 as the blue, dashed line. This example shows how the method enables a probabilistic computation of some quantity of interest, which can subsequently be used for technology evaluation and comparison. In this particular case, the flap technology concentrates the fuel burn around a slightly lower median value than the baseline. Furthermore, it drops below the baseline for higher values of fuel burn; thus, it has a lower tail dependency. Therefore, the technology reduces fuel burn overall.

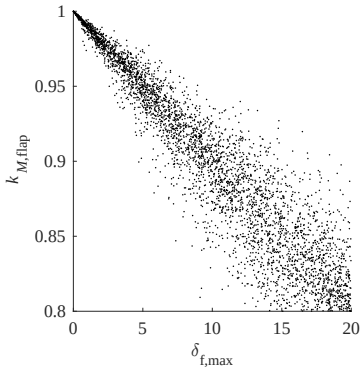


Figure 5.18: Dependency structure between maximum flap deflection $\delta_{\text{f,max}}$ and mass reduction factor $k_{M,\text{flap}}$.

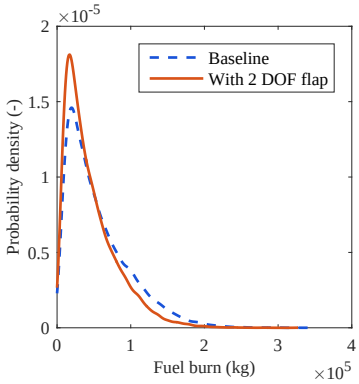


Figure 5.19: Probability density function of block fuel burn when a probabilistic computation is performed

5

5.3. DISCUSSION

The main contribution of this chapter comes from the way in which analysis modules are put into the context of the ontology. Specifying a context graph to which an analysis applies is more flexible than the static mapping from ontological terms to analysis terms as employed by former approaches [61]. The context graph is matched onto a knowledge graph describing any (novel) system, such that the analysis can be applied to that portion of the graph. Because the context graph can be specified such that multiple pattern extensions may be present (e.g. multiple input and/or output powers), this approach is more flexible than common input/output matching approaches, e.g. the one used by Van Gent et al. [54].

The SPPH application shows how the presented methodology is used to describe a system such as a hybrid-electric powertrain for aircraft and how a computation may be performed automatically, with minimal user interaction. Compare the approach to the original work by de Vries et al. [46] and notice how the same is achieved, but the very specific method in that work is replaced by a very generic method that requires, as a cost, a more thorough specification of the topic under investigation — the powertrain architecture. de Vries et al. [46] obtains representative powertrains through human reasoning, something which the current approach does not require (although would support). Furthermore, to deal with different sets of input and output variables of the computation, as well as different directions of power flow throughout the powertrains, they specify a separate matrix (Equation 18 in Ref. [46]). With the present approach, all of that is performed automatically, i.e. an algorithm can figure out those power flow directions (shown in Equation 5.2.1), and generate the appropriate computation graph for each case.

The second contribution is the qualitative physics reasoning for determining dependency structures between variables. It is believed that this is the first attempt at structuring and automating this process in a formal manner. Probabilistic assessment of novel technologies and systems is gaining more foothold in present research, although its in-

fancy in the design field causes the use of mostly basic approaches. Even though dependencies between random variables are essential for a proper assessment of posterior probability distributions, they are often overlooked or omitted. It is believed that this is a result of a lack of knowledge and understanding of how to incorporate dependencies between variables. Therefore, the presented method assists a practitioner in specifying these dependencies. The qualitative causal reasoning mechanism prompts which pairs of variables the practitioner should consider to specify a dependency for.

A primary drawback of the implemented approach is the specification of the knowledge graphs. As Figure 5.9 demonstrates, these graphs become convoluted quickly, and are therefore not easy to construct for designers or engineers non-versed in PSO or ontologies in general. Therefore, it is proposed to build a graphical user interface that allows the user to draw up a system (e.g. using Modelica or Simulink), while the knowledge graph is built automatically by the software based on the user inputs.

Another drawback of the current causal reasoning approach is that it relies heavily on rules that are crafted by a domain ontology designer. However, there is research that focuses on machine learning of causal structures. For example, schema networks [150, 151], interaction networks [152] or relational deep reinforcement learning [153] might provide clues as to how to let a system learn causal patterns by itself, rather than relying on a human expert to specify them.

Similarly, mathematical models and analyses are now specified by humans beforehand. The software only performs graph matching to find the places where the analyses may be applied. However, research into state representation learning [154] aims to automatically infer factors to describe an environment. Perhaps such an approach can learn to automatically construct a mathematical model for systems described with the knowledge graphs. The benefit would be that models can be ascribed to a system even more flexibly than with the current approach, thus allowing for slight variations in how a system is described. Graph convolutional networks might prove useful for learning the context graphs that are used to map analyses onto the knowledge graphs for the same reasons.

As a final remark, it should be noted that the computation graphs resulting from the presented approach do not synthesize efficient, optimized computations. Moreover, the approach uses graph matching extensively, which is a computationally expensive routine. As such, the presented method is not computationally efficient or optimized. Technology evaluation does not have to be, though. It is usually conducted in the conceptual and preliminary (design) phases of a project, where high-fidelity analysis methods are usually avoided. Additionally, the computational inefficiency is not expected to induce so much overhead that generating thousands of samples for a Monte Carlo Simulation becomes intractable, when an optimized code would be tractable.

5.4. CONCLUSION

This chapter presents a modular framework for conceptual design analysis that relies on knowledge graph descriptions of the system of interest and modular analyses that are automatically applied to such a graph. From the system knowledge graphs and a set of analyses, a computation sequence is consecutively generated and the user is queried for input data. In addition, qualitative physics reasoning is employed to infer input variables

that are mutually dependent, for which a joint probability distribution has to be set up. Then, either a deterministic or probabilistic simulation of the system can be executed.

While the method incurs a significant modeling effort, its merits reside in the capability to capture the context of analysis models and automatically apply them to any given system. This is done through the use of the PSO ontology with the extensions mentioned in Chapter 3. PSO itself does not provide the means to capture the context of analysis methods, and neither was any other method found that does so. The fact that the PSO and BFO ontologies are used with little modification shows that the method can easily be integrated to existing methods based on those upper ontologies. Another contribution of the presented method is the flexibility of the analysis models. Rather than that they are black-boxes with fixed input/output, the graph-based description allows the same analysis method to be automatically applied to different situations, as was demonstrated in the hybrid-electric powertrain case study. Additionally, technologies are easily included: they modify the system knowledge graph and a new computation graph is built subsequently. The final contribution resides in the qualitative reasoning for dependencies on random input variables in probabilistic computations. Although the physics-based rules that underlie this capability are crafted by human experts, they alleviate a practitioner's effort when setting up a probabilistic assessment. Therefore, this process becomes more consistent and less prone to error.

The current approach relies on rule-based algorithms, which in the future should be replaced with more flexible learning mechanism from machine learning literature. Finally, specifying the knowledge graphs and analysis models properly may prove challenging for practitioners. Solutions that simplify these procedures should be found.

With an evaluation of the QoIs for each technology portfolio, enough information is available to support a selection decision among them. However, in some cases, a selection is not easily made, because uncertainty on the QoIs may prevent a clear distinction. In such situations, insight is desired into what level of impact certain technologies have to have for a given improvement in a QoI. This is an inverse design problem, and is the focus of the next chapter. Thus, a method is presented that takes the probability distributions on the QoIs that the method from this chapter can produce. These probability distributions are fed into an algorithm that both enables selections between technologies, as well as the inverse design queries.

6

PROBABILISTIC INVERSION

Parts of this chapter have been published in Journal of Aircraft (2021) [155].

During conceptual design, decisions are made regarding which configurations and technologies to include in an engineering system. For example, questions arise on which wing movables to employ and where to position them, what material and structural layout is optimal for the wing box, whether to use a hydraulic or an electric actuator system, etcetera. In this dissertation, such questions are called inverse (design) queries, because they involve working from a computed quantity of interest back to the input variables, and figuring out how the latter should change to satisfy certain requirements. Typically, such design decisions are supported by deterministic investigations; the different options are represented in some simulation model and quantities of interest (QoIs) are computed for a specific, or small set of designs or missions. Such an approach provides poor generalization of the conclusions, because these are specific to the design or mission chosen. Therefore, very little can be concluded about the performance of a certain technology or configuration across-the-board.

Many design queries can be answered with optimization approaches. When including uncertainty, one can use either robust design optimization or reliability-based design optimization [156–158]. These approaches usually convert a probability distribution into scalar measures like mean and variance. Moreover, they tend to focus on specific designs and/or missions. When the precise designs and missions are still unknown, design space exploration may be employed. The high-dimensional design space can, for example, be visualized with Cobweb plots. These display the joint distribution of the variables and can subsequently be used to single out regions (i.e. value ranges of variables) of interest [159]. Cobweb plots can only be used to analyze the current design space, and show which ranges of variables influence other variables. Figuring out how the design space should change to achieve certain goals is not possible, however, and requires other methods.

This chapter investigates probabilistic inversion (PI) to support technology development and selection. The hypothesis is that this approach circumvents the drawbacks associated with optimization and design space exploration approaches and is a useful means to rank technologies when their specification and effects are uncertain. Forward uncertainty propagation is performed through sampling and PI is employed to answer inverse (design) queries. Probabilistic inversion is explained in section 6.1. The application consisting of the model, the input variables and their distributions are discussed in section 6.2. Two test cases are performed on this application. Firstly, a single technology is analyzed with both forward uncertain quantification (UQ) and PI in section 6.3. Secondly, a set of three technologies is assessed and PI is used to infer which technologies are most suitable to meet user-imposed requirements in section 6.4. The merits and drawbacks of the method are discussed in section 6.5, and the work is concluded in section 6.6 by establishing that probabilistic inversion is a powerful method for technology prioritization for selection during conceptual design and offers possibilities that conventional optimization or design space exploration techniques do not.

6.1. METHODOLOGY

Probabilistic inversion (PI) is employed in this work to solve inverse queries. The aim is to find the values for some input values that achieve certain goals defined on the output variables. In this section, the principles behind PI are discussed and the algorithms to

solve PI problems are presented.

Before diving into PI itself, though, an overview of how PI fits in with uncertainty quantification and propagation is provided in Figure 6.1. The first part of the method,

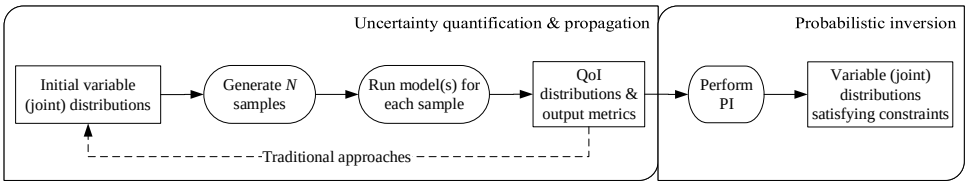


Figure 6.1: Overview of the method in three steps: uncertainty quantification & propagation and probabilistic inversion. The information from PI can be used for subsequent technology selection.

uncertainty quantification and propagation, is common to other approaches [12, 13, 47, 87, 160]. This part entails setting up the input variables and their (joint) distributions. Not all existing approaches employ joint distributions, but it is not unique to PI. In this work, the joint distributions are specified using copulas, which are explained in Section 2.8.3. All the variable distributions are then sampled N times, and each sample is propagated through the analysis models that compute the quantities of interest (QoIs). The sampling procedure is indifferent to the type of model, so any black-box input/output model works. Therefore, the models can be anything from physics-based simulations to design codes to surrogate models. The output samples are used to construct cumulative density functions (CDFs) on the QoIs. Further metrics may be derived from these CDFs to support decision making. When the output does not satisfy certain goals, traditional approaches require an iterative procedure, indicated by the dashed line in Figure 6.1. Rather simply, the input is changed iteratively to obtain the desired output.

This is where PI comes in. Instead of adopting an iterative procedure, PI is one additional step that is conducted after the uncertainty propagation. Constraints are imposed on any of the variables (input and output) and PI figures out how the distributions of the other variables change to satisfy those constraints. The updated distributions are used to answer design queries or technology prioritization queries.

6.1.1. PROBABILISTIC INVERSION

Probabilistic design, in the most general form, requires finding input random vector $\boldsymbol{X} \in \mathbb{R}^N$ (with some properties) to a particular model $G : \mathbb{R}^N \mapsto \mathbb{R}^M$ that yields output vector $\boldsymbol{Y} \in \mathbb{R}^M$ with a pre-specified distribution. In other words, we want to find $\boldsymbol{X}$ such that $G(\boldsymbol{X}) \sim \boldsymbol{Y}$ (where $\sim$ denotes having the same distribution); hence, invert the function G over random variables. This process is called probabilistic inversion (PI) [161].

For some simpler types of models, PI can be carried out with Bayesian methods. In many applications, however, the function G is very complicated, so the distribution of $G(\boldsymbol{X})$ is found via simulations. In such cases PI can be performed via sample re-weighting [162], which avoids inverting the function G, by re-weighting the samples of inputs and outputs of the model to satisfy specified constraints. The sample re-weighting process is as follows. For a set of samples $(x_1^{(i)}, ..., x_N^{(i)})$ generated by sampling from $\boldsymbol{X}$, a

set of output samples is computed as:

$$(y_1^{(i)}, ..., y_M^{(i)}) = G(x_1^{(i)}, ..., x_N^{(i)})$$

The sample file $s_i = (x_1^{(i)}, ..., x_N^{(i)}, y_1^{(i)}, ..., y_M^{(i)})$, $i = 1, ...K$ is obtained (shown in Figure 6.1), where each sample has the same probability $p(s_i) = 1/K$. The idea is to find a different than uniform distribution over the samples. That is, the weights w_i have to be found, such that re-sampling with these weights leads to a new set of samples that satisfies the requirements on $\boldsymbol{Y}$ and $\boldsymbol{X}$.

To make this problem computationally tractable the requirements on $\boldsymbol{Y}$ and $\boldsymbol{X}$ are in the form of quantiles or percentiles of variables in $\boldsymbol{Y}$ and $\boldsymbol{X}$ or functions of these variables. In this simplified form, PI via sample re-weighting can be viewed as an optimization problem to find weights $(w_1, ..., w_K)$ which minimize relative information with respect to the uniform distribution subject to linear constraints corresponding to percentiles of variables in $\boldsymbol{Y}$ and $\boldsymbol{X}$ or their functions. In theory, there are many distributions that could satisfy the imposed constraints, which is why the minimum relative information gives the distribution closest to the starting distribution, i.e. the uniform distribution. If we were only interested in computing the mean, then updating the sample weights and computing $E(X) = \sum w_i \cdot x_i$ would suffice. However, we are interested in the full distributions, which requires re-sampling with the new weights to construct them.

In principle, this problem can be solved with help of optimization software. In this dissertation, however, the PI problem via sample re-weighting is solved with the Iterative Proportional Fitting (IPF) algorithm, introduced in Ref. [163]. This algorithm is much faster than optimization approaches. IPF starts with a uniform distribution over the samples, and then iteratively re-weights them to satisfy the imposed constraints, one constraint at the time. If the optimization problem is feasible, IPF converges and provides a minimally informative solution to the problem with respect to the starting distribution [164]. IPF will not converge when the problem does not have a solution. In this case algorithms that minimize 'infeasibility' are proposed, such as PARFUM [165] and PREJUDICE [166, 167].

ITERATIVE PROPORTIONAL FITTING

Iterative Proportional Fitting (IPF) iteratively re-weights samples to satisfy the imposed constraints. Those constraints are provided as percentiles for a set of variables. Even if the complete sample set contains all variables $\boldsymbol{X}$, IPF only requires the constrained variables $\boldsymbol{Z} \subseteq \boldsymbol{X} \cup \boldsymbol{Y} \cup H(\boldsymbol{X}, \boldsymbol{Y})$. Thus, $\boldsymbol{Z}$ is a subset of $\boldsymbol{X}$, $\boldsymbol{Y}$ and functions of these two. The samples of $\boldsymbol{Z}$ are presented as a matrix:

$$\boldsymbol{Z} = \begin{bmatrix} z_{11} & \cdots & z_{1M} \\ \vdots & \ddots & \vdots \\ z_{K1} & \cdots & z_{KM} \end{bmatrix} \tag{6.1}$$

where K is the number of samples and M the number of variables in $\boldsymbol{Z}$.

For each variable Z_m a vector of percentiles is specified. From these percentiles the inter-percentiles q_m are derived. For example, the 5%, 50% and 95% percentiles are

specified for Z_m, then $q_m = [0.05, 0.45, 0.45, 0.05]$. All vectors q_m are stored in the set $\mathscr{Q} = \{q_m\}$. It is defined as a set of vectors rather than a matrix, because this allows for different amounts of percentiles to be specified for different variables.

For each percentile, the value of the corresponding variable has to be specified. Thus, let $Q_m = |q_m|$ be the amount of quantile constraints q_m on the variable Z_m. Then r_m has length $|r_m| = Q_m - 1$, where each entry r_{mj} is computed as:

$$\forall\, j = 1, ..., Q_m - 1 : r_{mj} \equiv P(Z_m \leq r_{mj}) = \sum_{i=1}^{j} q_i \tag{6.2}$$

For all variables, the vectors r_m are collected into a set $\mathscr{R} = \{r_m\}$.

The matrix Z, the inter-percentile set $\mathscr{C}$ and the constraint values $\mathscr{R}$ together are the inputs to the IPF algorithm. The algorithm starts by creating a set of indicator matrices, which specify which samples belong to which inter-percentiles. Therefore, for each variable there is a matrix $A^m \in \mathbb{R}^{K \times Q_m}$ given by:

$$A^m_{ij} = \begin{cases} Z_{im} \in (-\infty, r_{m,1}] & \text{if } j = 1 \\ Z_{im} \in (r_{m,Q_m-1}, \infty) & \text{if } j = Q_m \\ Z_{im} \in (r_{m,j-1}, r_{m,j}) & \text{otherwise} \end{cases} \tag{6.3}$$

Thus, each column corresponds to an inter-percentile $q_{m,j}$ and each row evaluates whether the value of that sample in $\boldsymbol{Z}$ is within the range specified by r_m for that inter-percentile. The set $\mathscr{A} = \{A^m\}$ is the last piece of information IPF requires.

IPF starts with an initial vector of sample probabilities $\boldsymbol{p}$, which is the uniform distribution over the samples. Thus, each values $p_i = p(s_i) = 1/K$. Then an outer loop runs through a prescribed number of iterations. During each iteration, the vector $\boldsymbol{p}$ is updated for each variable separately. Therefore, an inner loop runs over the variables Z_m and performs the following operation:

$$p'_i = \sum_{j=1}^{Q_m} A^m_{ij} \frac{p_i q_j}{\sum_{i \in A^m_{\cdot j}} p_i} \tag{6.4}$$

This process is shown in Figure 6.2. After each inner loop iteration, p_i is set to p'_i. As such, IPF updates $\boldsymbol{p}$ to satisfy each quantile constraint on each variable one at-a-time, while possibly violating the constraints over a previous variable. However, as the iterations proceed (and if the problem has a solution) the constraints for each variable will be satisfied [161, 164].

ERROR OF IPF

The outer loop of IPF runs over a prescribed number of iterations, but it could happen that IPF reaches satisfactory convergence before that limit is reached, or instead does not converge on time. With the constraints and the vector $\boldsymbol{p}$, the achieved percentiles can be computed and compared to the specified percentiles. This provides a measure of the error of IPF at any given iteration.

The constraints, specified by the combination of the quantiles $\mathscr{Q}$ and their values $\mathscr{R}$, can be written as a linear combination of the sample probability vector $\boldsymbol{p}$. They form

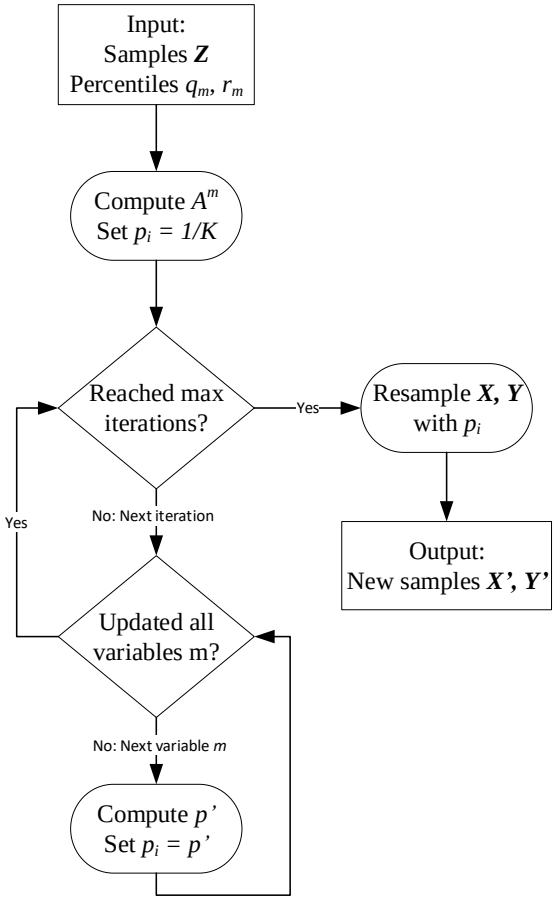


Figure 6.2: Flowchart of the IPF algorithm. The outer loop runs over a prescribed maximum number of iterations, while the inner loop iterates over all variables m.

the constraint set $\mathscr{C}$, which contains for all $m = 1,...,M$ and $j = 1,...,Q_m$ the following equality:

$$C_{j,m} : \sum_{i=1}^{K} p_i A_{ij}^m = q_{j,m} \tag{6.5}$$

Essentially, each constraint counts the number of samples that would fall within a certain inter-percentile, given the new probabilities p_i. Thus, there are $|\mathscr{C}| = \sum_{m=1}^{M} Q_m$ constraints; one for each combination of percentile and variable in $\boldsymbol{Z}$. The set $\mathscr{C}$ can now be written as a set of linear equations $\boldsymbol{C} \cdot \boldsymbol{p} = \boldsymbol{q}$. $\boldsymbol{C}$ is a $|\mathscr{C}| \times K$ matrix, with each entry $C_{(j,m),i} = A_{ij}^m$. The vector $\boldsymbol{p}$ of length K contains p_i and $\boldsymbol{q}$ are the percentiles, reshaped into a vector of length $|\mathscr{C}|$. The vector $\boldsymbol{q}$ should have the same values as the corresponding entries in $\mathscr{Q}$. As a measure of the error, the maximum absolute value of $\boldsymbol{q} - \boldsymbol{C} \cdot \boldsymbol{p}$ is taken.

PARFUM

IPF may not always converge, especially when the provided constraints make the problem infeasible. In such cases an approximate algorithm may be used. PARFUM is such an algorithm [165], which works similar to IPF. Unlike IPF, however, PARFUM updates each variable separately on each iteration (identical to Equation 6.4):

$$p_i^m = \sum_{j=1}^{Q_m} A_{ij}^m \frac{p_i q_j}{\sum_{i \in A_{:j}^m} p_i} \tag{6.6}$$

after which a mean is taken of the weights for each sample. In this case, the geometric mean is used:

$$p_i = \left(\prod_{m=1}^{M} p_i^m \right)^{\frac{1}{M}} \tag{6.7}$$

When the problem is feasible, this gives the same result as IPF. When the problem is infeasible, PARFUM provides a minimally informative solution that minimizes the distance to the constraints [161].

GROUPED SAMPLE RE-WEIGHTING

For reasons that will become apparent in section 6.4, it is sometimes desired to only re-weight groups of samples simultaneously, instead of individually for each sample as IPF and PARFUM do. This way, the distributions over the variables in a group of samples are left unaltered. In order to achieve this, the sample set is divided into P disjoint groups. After PI is performed on the full vector $\boldsymbol{p}$, each sample has its own weight. However, now each sample in a group should receive the same weight. This is done by simply taking the mean of the sample weights for each group and assigning that weight to each sample in the group:

$$\forall s_i \in \mathscr{P}_j, j = 1, ..., P : p(s_i) = \frac{\sum_{s_k \in \mathscr{P}_j} p(s_k)}{|\mathscr{P}_j|} \tag{6.8}$$

Afterwards, $\boldsymbol{p}$ is scaled to sum to one. Note that, because solving PI problems with grouped sample re-weighting often is infeasible, the PARFUM algorithm should be used instead of IPF. However, it has not yet been proven that PARFUM with this additional step converges.

6.1.2. EXAMPLE OF APPLYING PI

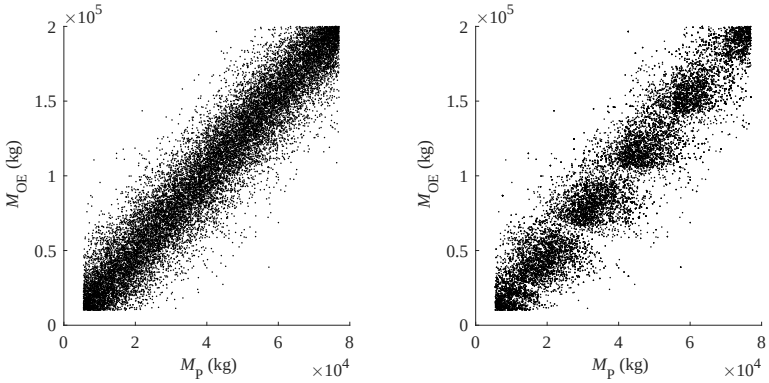
Before continuing with a more realistic problem setting, a simple example is discussed to show how probabilistic inversion works in practice. Consider the mass breakdown of the maximum zero fuel mass M_{ZF} computed as follows:

$$M_{ZF} = M_{OE} \cdot (1 - k_{OEM}) + M_P \tag{6.9}$$

where M_{OE} is the operating empty mass, k_{OEM} is a percentage mass reduction and M_P is the passenger or payload mass. This mass breakdown is not representative of the real world, and only is constructed for a simple demonstration of PI.

It is assumed that k_{OEM} is uniform on the interval $[0, 0.3]$ and is independent of M_{OE} and M_P. The margins of M_{OE} and M_P are assumed uniform on [1e4, 2e5] and [5.5e3, 7.7e4],

respectively. These values are purely notional. In a real case, they would have to be estimated from data or expert elicitation. Furthermore, from engineering insight it is logical that M_{OE} and M_P are correlated, because a higher payload mass leads to higher structural mass. Conversely, no more structural mass is present than strictly necessary for a given payload mass. This dependency is modeled with a Frank copula with $\alpha = 18.1915$ corresponding to Kendall's τ of 0.8. The Frank copula is chosen because it is symmetric and has no tail dependency. The value of 0.8 is chosen arbitrarily, and should in real applications be estimated from data or with an expert solicitation procedure. The resulting scatter plot of samples from the joint distribution of M_{OE} and M_P is shown in Figure 6.3(a).



(a) Original distribution (b) Distribution after PI

Figure 6.3: Joint distribution between M_{OE} and M_P, modeled using a Frank copula, before and after PI

Now, a given set of aircraft is considered for which M_{OE} and M_P are as specified above and the percentage mass reduction is k_{OEM}. Suppose that for this range of aircraft some new technology can be considered that allows to alter the distribution of k_{OEM}. The question is how much percentage M_{OE} reduction is required for a given reduction in M_{ZF}.

The requirements on the distribution of M_{ZF} are specified in the form of 5^{th}, 10^{th}, 30^{th}, 50^{th}, 70^{th}, 90^{th}, 95^{th} and 99^{th} percentiles, which are equal to 0.2428, 0.3353, 0.7440, 1.1605, 1.5740, 1.9721, 2.1083 and 2.2932 (all $\times 10^5$), respectively. These values are 11% smaller than the original distribution of M_{ZF}.

PI requires the problem to be translated into a set of constraints. Recall that constraints take the form of the quantiles $\mathscr{Q}$ and their values $\mathscr{R}$. Essentially, these are points on the CDFs that the practitioner desires to obtain after performing PI.

Two different functions for constraints are distinguished: either a constraint fixes a variable's distribution so PI cannot alter it, or it reflects a new distribution that is desired by the designer. For the first function, the percentiles of the original distributions can be specified as constraints. However, that does not mean that the distributions are not changed by PI at all. In case of a joint distribution, all linear combinations of the variables would have to be constrained. Thus, a distribution can never be fully constrained

in a way that keeps it completely unaltered during PI. Nonetheless, these constraints do limit the extent to which a distribution is affected and are therefore sufficient. Examples of how and why certain variables or their joint distributions are constrained can be found later in sections 6.4.1 and 6.4.2. The second constraint function requires the percentiles of a new distribution. These percentiles may be obtained from data or from expert elicitation or policy makers using existing techniques. Naturally, they may be assumed to get an indication of the resulting responses.

For this example, a fixed set of aircraft is considered. Thus, M_{OE} and M_P need to be kept relatively unchanged and only the distribution of k_{OEM} should be adjusted to the requirements on the distribution of M_{ZF}. To that end, M_{OE} and M_P are constrained to keep their 5^{th}, 10^{th}, 30^{th}, 50^{th}, 70^{th}, 90^{th}, 95^{th} and 99^{th} percentiles close to the original. Thus, for each of these variables, each of these percentiles is specified with the value of the original uniform distribution. Furthermore, to keep the dependency between M_{OE} and M_P similar, the distribution of the sum of M_{OE} and M_P is also constrained on the above eight percentiles. Therefore, the dependency imposed by Figure 6.3(a) should remain identical. Dropping the constraints on the margins or dependence of M_{OE} and M_P would permit PI to alter their distributions, which would consequently reflect a different set of aircraft.

With the aforementioned constraints, the IPF algorithm is run and a new sample set is generated. The CDFs for the new samples are shown in Figure 6.4 (as dashed, red lines), with the original CDFs for reference (as blue, solid lines). Notice that the CDFs of M_{OE} and M_P hardly have changed due to the constraints, while k_{OEM} becomes skewed to the right as expected.

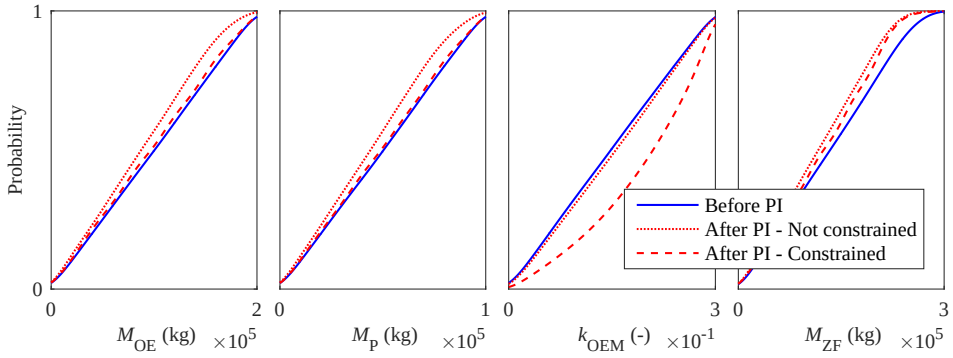


Figure 6.4: Cumulative density functions before and after PI.

In order to better illustrate what PI does, the probability density functions (PDFs) corresponding to the CDFs are shown in Figure 6.5. Again, the red, dashed lines show the PDFs after PI, while the blue, solid lines are the original PDFs. The distribution over k_{OEM} has shifted all the way to its largest value. Observe the oscillatory nature of the PDFs for M_{ZF}, M_{OE} and M_P. This is an important effect of using percentiles rather than complete distributions as constraints; thus, it is inherent to IPF. PI picks samples within each inter-percentile and re-weights those equally to meet a certain percentile

constraint. Therefore, the distributions only match at the specified percentiles, and may vary in-between.

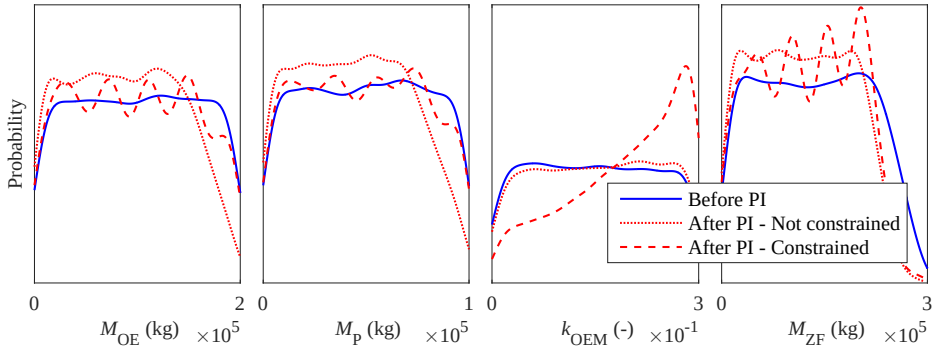


Figure 6.5: Probability density functions before and after PI.

A similar observation can be made for the bivariate distribution of M_{OE} and M_P in Figure 6.3. After PI (Figure 6.3(b)) the scatter plot is different than the original one. Clear discontinuities can be seen in the samples corresponding to the oscillations of the PDFs shown in Figure 6.5. Nonetheless, the dependency between M_{OE} and M_P, measured as the Pearson coefficient, remains almost identical to the initial value of 0.95, due to the constraint on the sum of these two variables.

If the constraints on M_{OE}, M_P and their sum are not included, the results of PI look very different. These are shown in Figure 6.4 and Figure 6.5 as dotted, red lines. From both figures it becomes clear that k_{OEM} is hardly affected, while the distributions of M_{OE} and M_P show a larger deviation form the original. This is easily understood as PI picks the most influential variables to achieve the constraints. In this case, obviously, M_{ZF} is most easily reduced by reducing either M_{OE} or M_P or both. From Figure 6.5 we can furthermore observe that the PDFs are smoother as PI has more freedom with a constraint only on M_{ZF}. Thus, weights are redistributed more evenly over samples and the resulting PDFs and CDFs after re-sampling are smoother. Finally, the correlation between M_{OE} and M_P also changes from 0.95 to somewhat below 0.94. That is not a big difference, but shows that for more complex problems and/or more constraints PI alters the dependency between variables.

6.2. APPLICATION

The mission analysis method that is used to showcase PI in an aircraft technology evaluation and selection setting is discussed in this section. Consecutively, the inputs to this method are presented together with their probability distributions. Finally, the technologies that are under investigation and how they are modeled is detailed.

6.2.1. MISSION ANALYSIS METHOD

A simplified mission is simulated, which starts at zero altitude and take-off speed V_0 as well as an assumed fuel mass M_{F_0}, then climbs to cruise altitude h_{cr} and acceler-

ates to cruise speed V_{cr}. The range flown during climb is now computed, but the range flown during cruise or descent are unknown. Therefore, descent is first computed backwards (since the final point is known), such that the range flown during descent becomes known. The final point is at zero altitude and V_0, with zero fuel mass. When both climb and descent have been computed, the range flown in cruise is known, which is assumed to be flown at constant altitude and speed. In its entirety, the mission analysis function can be described as:

$$(M_F, R) = \text{MA}(M_{F_0}, \boldsymbol{x}) \tag{6.10}$$

where $\boldsymbol{x}$ is a sample that holds a value for each of the input variables (presented in Section 6.2.3) other than the initial fuel mass estimate M_{F_0}. The function MA returns both the consumed fuel mass M_F and range flown R. At any point, the assumed initial fuel mass M_{F_0} may turn out to be insufficient ($R < R_{req}$) . Conversely, when the entire mission is flown ($R \geq R_{req}$), residual fuel may remain ($M_{F_0} > M_F$). Therefore, an outer iteration aims to find the particular value for M_{F_0} that is adequate to fly the specified range. That value gives the quantity of interest — fuel burn. The outer iteration is implemented using a minimization-within-bounds algorithm, which minimizes the error in fuel mass and range

$$\epsilon(M_{F_0}, R) = \text{abs}(M_F / M_{F_0} - 1) + \text{abs}(R / R_{req} - 1) \tag{6.11}$$

by adjusting M_{F_0} as:

$$\text{argmin}_{M_{F_0}} \, \epsilon(M_{F_0}, R) \tag{6.12}$$

For a fair comparison between aircraft and technologies, the block fuel burn M_F is not a suitable metric: heavier, long-range aircraft consume more fuel, even if they are more efficient than lighter, regional aircraft. Therefore, the payload-range energy efficiency (PREE) [46] is used. However, this metric is based on energy consumption and is defined as $R \cdot M_P \cdot E^{-1}$, where E is the energy consumed during an entire mission. Thus, this metric measures efficiency, which has to be increased. However, in this work the QoI has to be minimized, so the inverse of PREE is taken, and the energy consumption is replaced with block fuel burn, resulting in the PRE^{-1} metric. Thus, PRE^{-1} is the block fuel normalized with range and payload mass, i.e. $M_F \cdot (R \cdot M_P)^{-1}$.

The mission analysis computations assume steady flight. Furthermore, thrust during climb is assumed to be a constant fraction T_{cl} of the maximum thrust T_{max}, and during descent a constant fraction T_{des} is assumed. Finally, a speed–altitude profile is assumed, given by:

$$\frac{dV}{dh} = \frac{V_{cr} - V}{h_{cr} - h} \tag{6.13}$$

during climb, and the negative of that during descent. This profile is not necessarily realistic, but suffices for the current application, where only differences between different inputs matter.

6.2.2. COST COMPUTATION

No detailed cost module is present in the current method. However, a cost metric is included in order to have a second objective that conflicts with PRE^{-1} . It can be interpreted as any kind of cost, e.g. recurring cost, non-recurring cost and/or direct operating

cost. The baseline aircraft is considered to have a normally distributed cost with mean $\mu = 1$ and standard variance $\sigma = 0.05$. Each technology adds some cost measure to this baseline distribution. Depending on these technology costs, the final cost for each portfolio may be smaller or larger than the baseline.

6.2.3. INPUT VARIABLES

In this section the distributions and dependencies for the input variables of the mission analysis are set up. Most researchers that include uncertainty do not focus on how the probability distributions should be obtained [88]. Characterizing those distributions for subsequent analysis is a challenging task, which often relies on expert judgment. Even though that is labor-intensive and subjective, there is no viable alternative presently, and Figure 6.1 shows it is adopted here as well. However, an alleviating remark is made by Cook and Jarrett [157] who address the question: "How important is the choice of how to represent input uncertainties mathematically in robust airfoil optimization?" as an example of what effect the specific choice of uncertainty distribution has on the outcome. The answer is that the difference between probabilistic results (i.e. different probability distributions) is insignificant, albeit their difference with the deterministic cases was large. In one case, even, the probabilistic optimum was Pareto-dominant with respect to the deterministic one. Thus, including uncertainty is important. However, not just any distribution works, and especially dependencies between variables have to be taken into account.

Table 6.1 shows all the inputs to the mission analysis method ($\boldsymbol{x}$ in Equation 6.10). Most of these variables are assigned a uniform distribution, to reflect a wide range of aircraft and missions. TSEC and SE are assigned a triangle distribution, because their mean is derived from data and the triangle distribution allocates more probability mass around this value, while having finite bounds. Finally, some variables are represented with scalar values, because they are aircraft independent and this way the design space is reduced.

Table 6.1: Input variables specification

Variable Name	Symbol	Units	Distribution	Dependency
Cruise altitude	h_{cr}	m	Uniform	Independent
Cruise speed	V_{cr}	m/s	Uniform	Independent
Range	R	m	Uniform	Independent
Payload mass	M_P	kg	Uniform	Correlated with M_{OE}
Take-off & Landing speed	V_0	m/s	Scalar (45)	Independent
Wing loading	W/S	N/m^2	Uniform	Determines wing area S
Wing aspect ratio	A	-	Uniform	Determines wing span b
Thrust-to-weight ratio	T/W	-	Uniform	Independent

Continued on next page

Table 6.1 – continued from previous page

Variable Name	Symbol	Units	Distribution	Dependency
Climb throttle	T_{cl}	-	Scalar (0.85)	Independent
Descent throttle	T_{des}	-	Scalar (0.05)	Independent
Operating empty mass	M_{OE}	kg	Uniform	Correlated with M_{P}
Clean zero-lift drag	C_{D_0}	-	Uniform (0.01, 0.02)	Independent
Thrust specific energy consumption	TSEC	J/(N s)	Triangle (600,750,900)	Independent
Oswald factor	e	-	Scalar (0.7)	Independent
Fuel Specific Energy	SE	J/kg	Triangle ($45{\cdot}10^6$, $46{\cdot}10^6$, $47{\cdot}10^6$)	Independent
Time step	Δt	s	Scalar (30)	N/A

6.2.4. TECHNOLOGIES

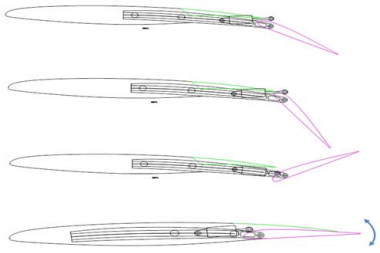


Figure 6.6: Side view of the 2nd DOF flap, with different extreme positions

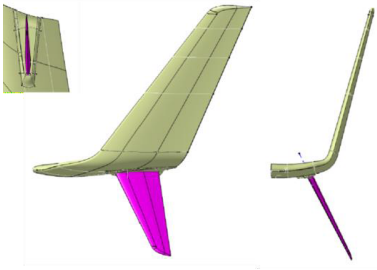


Figure 6.7: Illustration of the rotating winglet downer, which can rotate around it's length-axis to alter the airflow around the winglet

In the ensuing case studies, three technologies are used to showcase PI. The first, which is also investigated individually, is a second-degree-of-freedom (2nd DOF) flap, see Figure 6.6. This flap has two actuators that independently control the extension and rotation of the flap, to allow it to provide maneuver load alleviation and camber optimization. Therefore, it is modeled using three variables: $\delta_{\text{f,max}}$, $k_{M,\text{flap}}$ and k_e. The first is the maximum deflection of the flap, either upward or downward. The second is the effect of maneuver load alleviation on the structural weight, measured as a percentage of M_{OE}. (A lighter wing box can be designed when MLA effectively reduces the maximum load factor. [31, 168, 169]) The third models the effect of camber optimization [170] by being added to the aircraft's Oswald factor. Its distribution is estimated from Ref. [171]. Both $\delta_{\text{f,max}}$ and k_e influence the aircraft lift–drag polar as follows:

$$C_D = C_{D_0} + k_0 \cdot \delta_{\text{f}} + \frac{C_L^2}{\pi A(e + k_e \cdot \delta_{\text{f}})} \tag{6.14}$$

In this equation, δ_f is measured in degrees, thus k_e is measured in 1/deg. The factor k_0 is also measured in 1/deg, and has a value of 5×10^{-5}. That value was arbitrarily set to obtain an interesting test case. The effect of δ_f for a fixed value of k_e is shown in Figure 6.8. Essentially, an increasing flap deflection shifts the polar to the right, while reducing induced drag; thus, making the polar less steep.

In the mission analysis, the L/D ratio is optimized at each point during the mission by varying the flap deflection angle δ_f in the range $[0, \delta_{f,max}]$. During climb and cruise, L/D is maximized by the flap for a given C_L, as follows:

$$\text{argmin}_{\delta_f \in [0,\delta_{f,max}]} \; C_D \tag{6.15}$$

while during descent it is minimized for the steepest path:

$$\text{argmax}_{\delta_f \in [0,\delta_{f,max}]} \; C_D \tag{6.16}$$

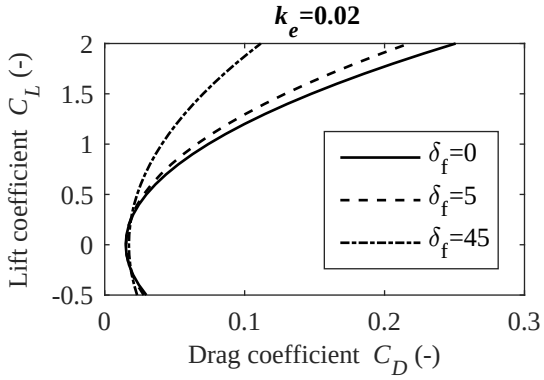


Figure 6.8: Lift-to-drag polars affected by flap deflection for notional 2nd DOF flap.

All three technology-defining variables, and their distributions are shown in Table 6.2. It also shows the dependency imposed on $\delta_{f,max}$ and $k_{M,flap}$. This follows the insight that the higher the flap deflection is, the more maneuver load alleviation is achieved. However, for increasingly large deflections, more uncertainty is present in the mass reduction that is achieved. To model this effect, a rotated Clayton copula is used, with $\alpha = 8$, computed from a Kendall's τ of 0.8. This copula is shown in Figure 6.9 in the leftmost plot.

The second technology is a rotating winglet downer, which is a rotating element originating from the winglet's base and pointing downward, as shown in Figure 6.7. It deflects to alter the lift distribution around the wing tip and consequently offers maneuver and gust load alleviation [172]. It is modeled using a single variable, $k_{M,downer}$, which also is a percentage of M_{OE}. When both this and the flap technology are present, the combined effect is not simply the multiplication of both mass reduction factors. Instead, $k_{M,downer}$ is expected to be closer to 1 when $k_{M,flap}$ is low, and vice versa. Therefore, a negative dependency has to be imposed on these two variables. Furthermore, it is assumed that it is impossible to reach either variable's minimum when both are present.

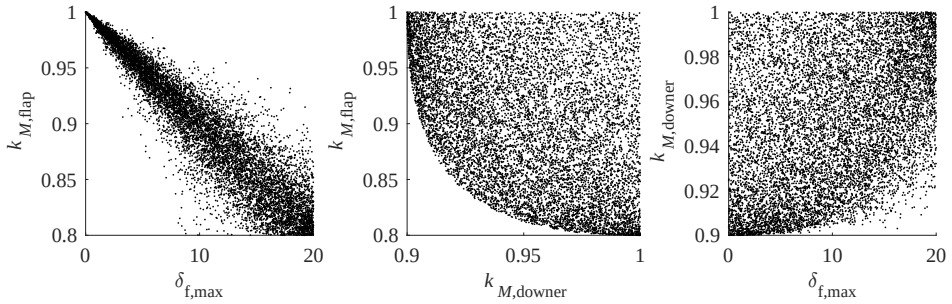


Figure 6.9: Dependencies between the three variables $\delta_{\text{f,max}}$, $k_{M,\text{flap}}$ and $k_{M,\text{downer}}$ characterizing the 2nd DOF flap and winglet downer.

For these reasons, this dependency is modeled using a Clayton copula with $\alpha = -0.75$, computed from a Kendall's τ of -0.3. The resulting joint distribution is shown in the center plot in Figure 6.9. The rightmost plot shows the dependency between $\delta_{\text{f,max}}$ and $k_{M,\text{downer}}$, resulting from the two previously mentioned dependencies. The combined effect of the two technologies $k_{\text{tot}} = k_{M,\text{flap}} \cdot k_{M,\text{downer}}$. Thus, drawing from the center distribution in Figure 6.9, and applying this multiplication to obtain k_{tot}, the distributions in Figure 6.10 are obtained. Observe that k_{tot} (in some cases) achieves more reduction than either $k_{M,\text{flap}}$ or $k_{M,\text{downer}}$ alone, although not as much as the multiplication of their minima (i.e. $0.8 \cdot 0.9 = 0.72$). Instead, the minimum of k_{tot} lies around 0.77. In summary,

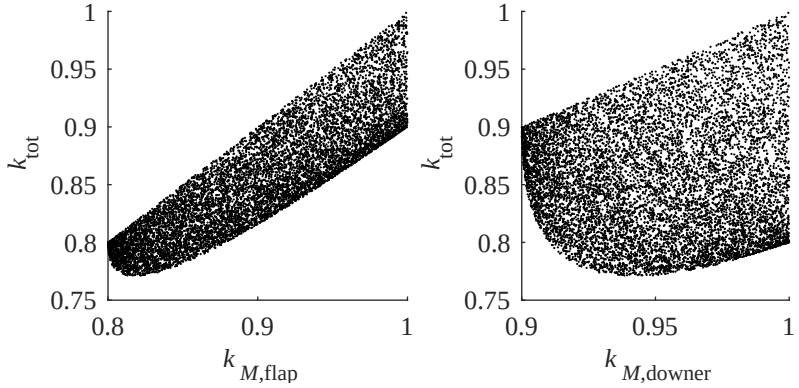


Figure 6.10: Dependencies between the 2nd DOF flap and winglet downer mass impacts and their combined effect k_{tot}. These plots are the result of combining those from Figure 6.9 with the fact that $k_{\text{tot}} = k_{M,\text{flap}} \cdot k_{M,\text{downer}}$.

the 2nd DOF flap and winglet downer offer a mass reduction as follows:

$$\Delta M = \begin{cases} k_{M,\text{flap}} \cdot M_{\text{OE}} & \text{if only flap} \\ k_{M,\text{downer}} \cdot M_{\text{OE}} & \text{if only winglet downer} \\ k_{\text{tot}} \cdot M_{\text{OE}} & \text{if both flap and winglet downer} \end{cases} \tag{6.17}$$

The third technology is some engine improvement (e.g. better turbine blade cooling, to increase turbine inlet temperature) that improves the thrust-specific energy consumption (TSEC) of the aircraft. It is modeled with k_{TSEC}, also shown in Table 6.2, and is independent from the other two technologies.

Table 6.2: Technology variables specification

Variable Name	Symbol	Units	Distribution	Dependency
Maximum flap deflection	$\delta_{\text{f,max}}$	deg	Uniform (0, 30)	Correlated with $k_{M,\text{flap}}$
2nd DOF flap mass impact	$k_{M,\text{flap}}$	-	Uniform (0.8, 1)	Correlated with $\delta_{\text{f,max}}$
2nd DOF flap Oswald efficiency impact	k_e	deg^{-1}	Uniform (0.012, 0.036)	Independent
Rotating downer mass impact	$k_{M,\text{downer}}$	-	Uniform (0.9, 1)	Independent
Engine TSEC impact	k_{TSEC}	-	Uniform (0.5, 0.99)	Independent

For each technology the cost is determined through a deterministic function of its characterizing variables, or a random distribution, or both. These functions are crafted to give a spread in total cost, such that a trade-off between fuel burn and cost results. The cost impacts, and total cost are computed as:

$$
\begin{aligned}
C_0 &= \mathcal{N}(\mu = 1, \sigma = 0.05) \quad , \\
\Delta_{C,\text{flap}} &= \delta_{\text{f,max}}/100 \quad , \\
\Delta_{C,\text{downer}} &= \mathcal{N}(\mu = 0.1, \sigma = 0.02) \quad , \\
\Delta_{C,\text{engine}} &= (1 - k_{\text{TSEC}})/5 + \mathcal{N}(\mu = 0, \sigma = 0.05) \quad , \text{and} \\
C &= C_0 + \Delta_{C,\text{flap}} + \Delta_{C,\text{downer}} + \Delta_{C,\text{engine}} \quad .
\end{aligned}
$$

Here, $\mathcal{N}$ is a Gaussian distribution. Obviously, when a certain technology is not included, its cost is not added to the total cost.

6.2.5. VERIFICATION

The mission analysis method is examined in this section. Firstly, the response of the model for a single aircraft and design point is investigated. Secondly, a global sensitivity analysis is conducted to study the response of the model, in order to support the conclusions in the subsequent test cases.

MISSION ANALYSIS FOR SINGLE AIRCRAFT

An Airbus A320 aircraft is notionally represented using the available input variables to study the various responses of the MA model. In Figure 6.11, the results of the mission

analysis for this input are depicted. Important to observe is that, during climb and descent, the flap deflection δ_f is always at its maximum value $\delta_{f,max}$ and otherwise zero. This is a deficit in the lift–drag polar model (explained in section 6.2.4): the polars for zero and maximum deflection cross in very small region. Therefore, C_L is either above or below it, and the one with least drag in that region is chosen. During cruise, the C_L is within the transition range from zero to maximum deflection, and hence a gradual variation results. In a more realistic setting, adverse effects of flap deflection, such as separation, have to be modeled and will influence (i.e. limit) the flap deflection.

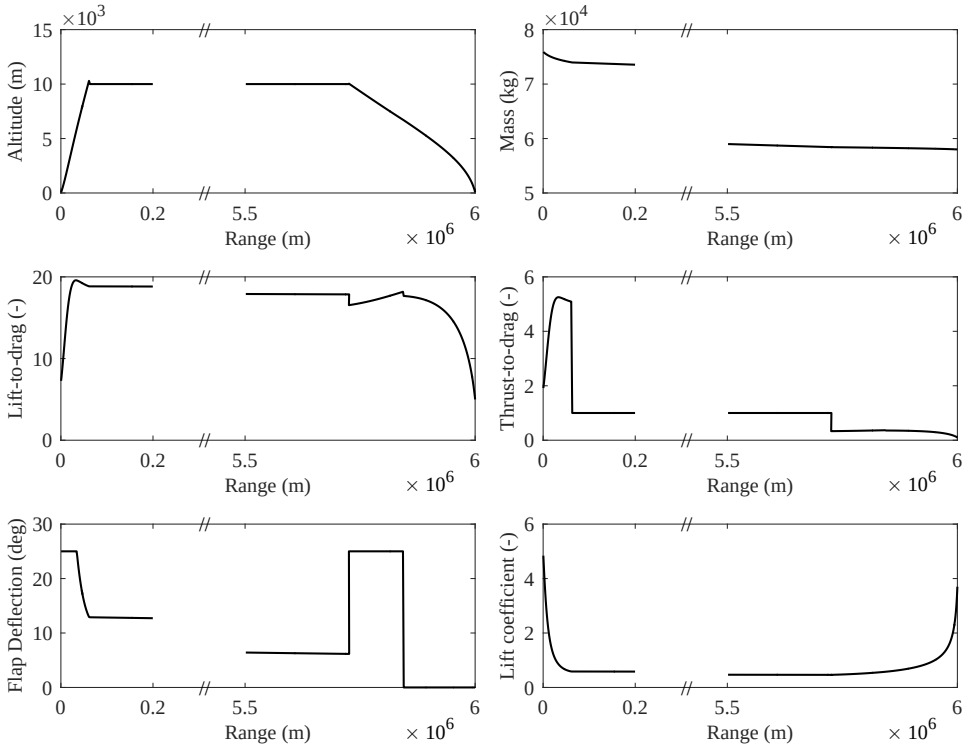


Figure 6.11: Mission analysis results showing variation of altitude, mass, L/D, T/D, δ_f and C_L with range flown.

For the rest, the mission analysis computation behaves as expected, with realistic values for all parameters(even though C_L almost reaches 5 as a result of the assumption for $\frac{dV}{dh}$). This conclusion is not enough to support the case studies, though. To make conclusions about the effect of technologies, or do probabilistic inversion, the sensitivity of this analysis method has to be measured against the variables of interest.

GLOBAL SENSITIVITY ANALYSIS

A sensitivity analysis (SA) is performed to study the response of the model with respect to changes in its input. This information supports the conclusions that are drawn from PI in the various test cases in the following sections. For example, when PI modifies the distribution of one variable significantly, while another is hardly affected, there are two

possible explanations. The first explanation is that the former variable simply is more effective in reaching the goal that PI is set out to achieve, even though both variables have a measurable influence on that goal. In contrast, the second explanation is that the model used to generate the samples is insensitive to the latter variable; hence, PI only changes the former, because it is the only variable explaining the variance in the goal.

Commonly, sensitivity analyses are conducted around a particular point in the design space (local SA). This type of analysis is justified when investigating a specific design, rather than a wide range of designs. For what is currently of interest, a wide range of aircraft designs, global sensitivity analysis is a more appropriate tool. It is performed as follows.

There are two quantities of interest in this study: PRE^{-1} and cost. A sensitivity analysis is only conducted on the first, as for cost there are linear equations with the technology impact variables, and as such it is already known that the sensitivities are nonzero. For PRE^{-1} , the product moment correlation, rank correlation and correlation ratio are computed. The first is a measure of the strength of a linear association of two variables, on a scale from -1 to 1. Rank correlation is similar, except that it measures how well the two variables follow a monotonic function. Finally, correlation ratio provides a sense of the extent to which a variable's variance explains another variable's variance. It is measured on a scale from 0 to 1. Therefore, unlike the other two, correlation ratio does not give a direction of influence.

6

To study the sensitivity of the mission analysis method, the sensitivity of PRE^{-1} is computed with respect to the variables in Table 6.1, without any of the technologies. The results are shown in Figure 6.12, where the sensitivities are sorted from highest to lowest, in an absolute sense. It becomes clear that C_{D_0} and TSEC are most influential. That M_F

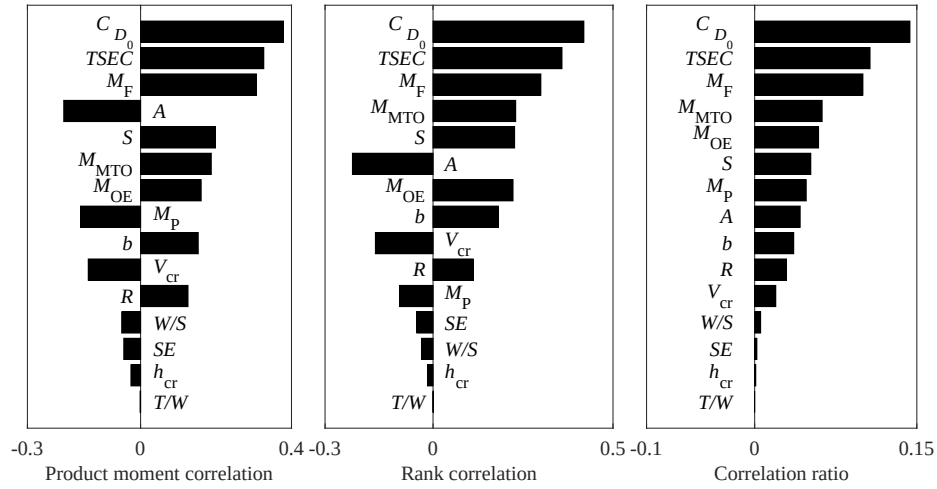


Figure 6.12: Sensitivity measures of PRE^{-1} with respect to the variables in Table 6.1.

is in third place is surprising, as it is the variable that is measured, except that it is not yet normalized with R and M_P. Therefore, it may be concluded that there are other variables

explaining the variance in M_F other than R and M_P. Each variable's direction of influence agrees with engineering instinct. Therefore, it is concluded that the mission analysis behaves as expected and shows good sensitivity with respect to the input variables.

A similar sensitivity analysis only includes the technology impact variables from Table 6.2, in the data set where all three technologies are included. This gives the sensitivities in Figure 6.13, which are as expected, except for the sign of $k_{M,\text{downer}}$. Thus,

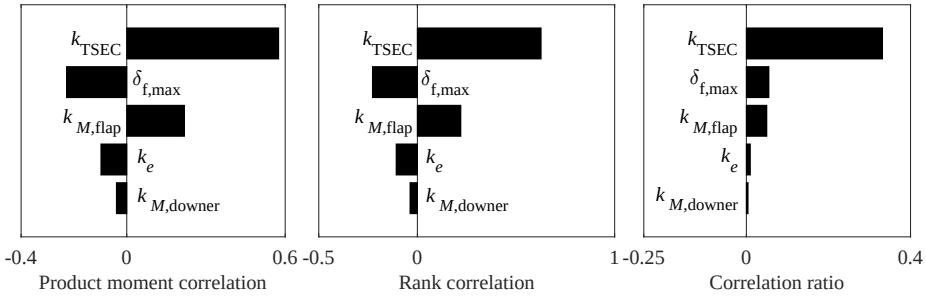


Figure 6.13: Sensitivity measures of PRE^{-1} with respect to the variables in Table 6.2.

6

$k_{M,\text{downer}}$ has a negative correlation, while the same sign as $k_{M,\text{flap}}$ is expected, because the two are effectively the same. To check correct operation of the mission analysis, the sensitivity study was repeated for when only the downer technology is included, which indeed shows the correct sign for $k_{M,\text{downer}}$. With this confirmation, the incorrect sign is deemed insignificant, because the absolute value of the correlation is small.

The sensitivity analyses show that the QoI PRE^{-1} is sufficiently affected by the technologies and the other input variables. Thus, PI should have little to no bias towards certain variables as a result from model insensitivities.

6.3. PI FOR A SINGLE TECHNOLOGY

Even though PI shows most potential over conventional approaches in the case of multiple objectives, this section starts with a single technology and a single objective. This is because relevant design queries can be answered using this approach and it is a suitable stepping-stone for the more involved multi-objective technology prioritization in the next section.

For a single technology queries of the form "what technology impacts are required for a given reduction of x% in QoI y? " might be of interest. Such questions are easily posed in PI, by constraining y and observing the changes in the technology parameters. This section shows how PI enables answering such queries, specifically for the 2nd DOF flap technology, introduced in subsection 6.2.4. Next, the use of PI to study technology maturation is presented.

6.3.1. INVERSE QUERY: TECHNOLOGY STATE REQUIRED FOR PRESCRIBED BENEFIT

Studying what values the technology variables should attain for a given change in one or more QoI is done using PI. The technology variables ($\delta_{\text{f,max}}$, $k_{M,\text{flap}}$ and k_e) are allowed to move freely, while the other input variables are held fixed. A distribution is specified on PRE^{-1} that reduces it by some amount for each percentile. The same bivariate distributions between M_{OE} and M_{P} and between $\delta_{\text{f,max}}$ and $k_{M,\text{flap}}$ are constrained as well.

The results in Figure 6.14 follow intuition: the median PRE^{-1} has reduced by 15%, as

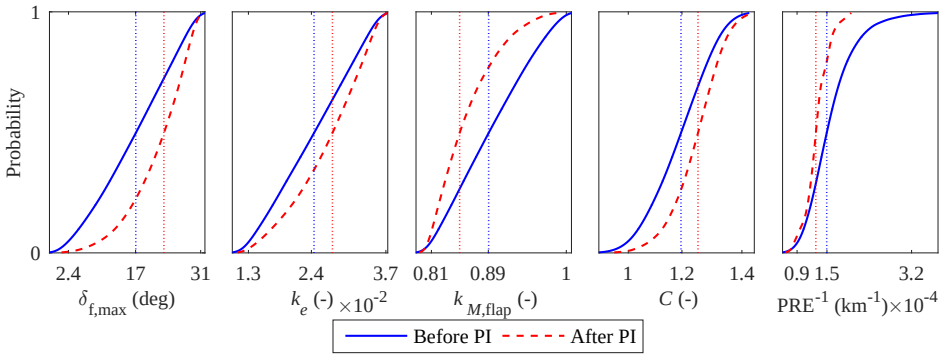


Figure 6.14: Cumulative density functions for 2nd DOF flap technology with original (blue) distributions and for a 15% PRE^{-1} reduction (red, dashed). The vertical lines represent the medians of the two curves.

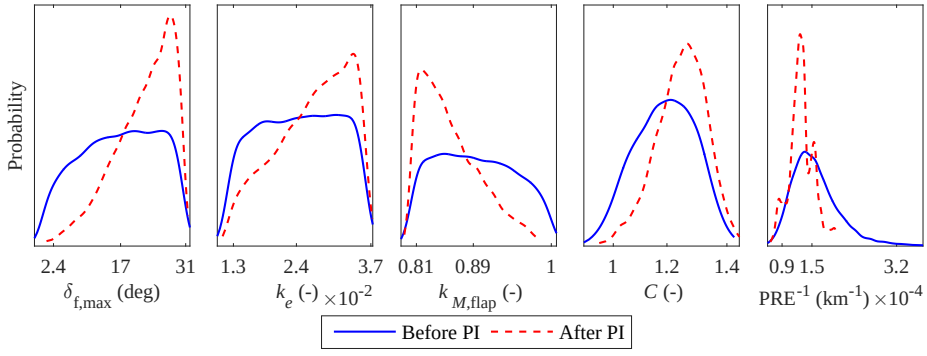


Figure 6.15: Probability density functions for 2nd DOF flap technology with original (blue) distributions and for a 15% PRE^{-1} reduction (red, dashed).

specified. Correspondingly, the maximum flap deflection has shifted to the right, as has k_e. The deflection's median went from 17 deg to 23 deg: an increase of 35%. At the same time, its variance reduced by 40%. The median of k_e increased by 13%, from 0.0245 to 0.0277, with a variance reduction of 14%. As expected, the distribution for $k_{M,\text{flap}}$ shifts to the left, as a result of the increase in $\delta_{\text{f,max}}$. Likewise, cost has increased. The fact that

variance is reduced is attributed to the constraint on PRE^{-1} having lower variance than the original distribution. The corresponding PDFs are shown in Figure 6.15.

While traditional approaches may be able to obtain statistical measures such as the mean and variance, PI provides a full distribution for the variables of interest, to fulfill certain goals. With pure forward propagation approaches, that is only attainable through iteratively updating the input distributions and propagating these through the models.

6.3.2. ALTERNATIVE USE OF PI: INVESTIGATING TECHNOLOGY MATURATION

Technology maturation can be modeled by updating the input distributions of the technology variables [87]. However, running the analysis models again to generate thousands of samples is time-consuming. With PI, the same result can be achieved, provided the updated distributions are within the range of the original distributions. Where the traditional approach requires propagating N samples through the analysis models twice, the PI approach only requires a single forward propagation pass.

For the 2nd DOF flap, the distributions of $\delta_{\text{f,max}}$ and k_e are adjusted. The flap deflection now follows a triangle distribution from 12 deg to 30 deg, with its peak at 20 deg. The Oswald efficiency impact is also provided with a triangle distribution from 0.02 to 0.036, with a peak at 0.03. In both cases, the range of values is less than the original, uniform distribution. Furthermore, more probability mass is located around the expected values: 20 deg and 0.03, respectively. Finally, since the dependency of $\delta_{\text{f,max}}$ with $k_{M,\text{flap}}$ (recall Figure 6.9, leftmost plot) has to be maintained, $k_{M,\text{flap}}$ is given an updated uniform distribution corresponding with the range of values for $\delta_{\text{f,max}}$.

In order to set up the PI problem, the new distributions on $\delta_{\text{f,max}}$, $k_{M,\text{flap}}$ and k_e are imposed as constraints. Furthermore, the joint distributions of M_{OE} and M_{P}, and $\delta_{\text{f,max}}$ and $k_{M,\text{flap}}$ are constrained, as well as all input variables that should remain unchanged.

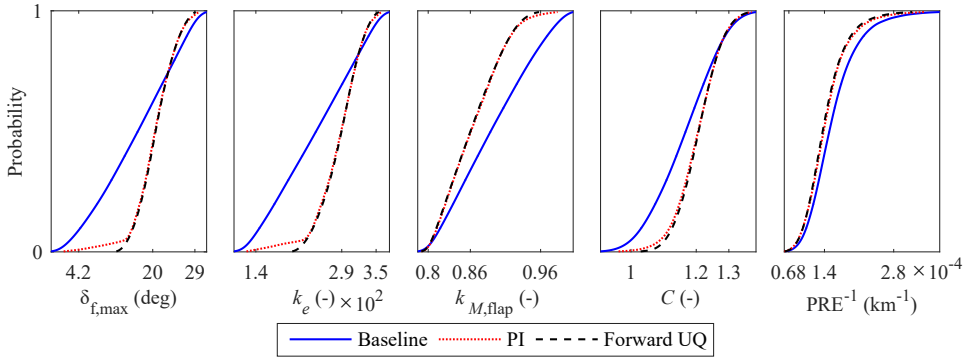


Figure 6.16: Cumulative density functions for 2nd DOF flap technology with original (blue) distributions for $\delta_{\text{f,max}}$ and k_e. After technology maturation the CDFs are computed with PI (red, dotted) and with forward uncertainty propagation (black, dashed). The horizontal axis tick-marks are the 5, 50 and 99 percentiles, in that order.

As Figure 6.16 illustrates, the results from forward uncertainty propagation and PI are virtually identical. The only noticeable difference lies in the lower percentiles of the $\delta_{\text{f,max}}$ and k_e CDFs. This is simply a result of not constraining PI at even lower percentiles.

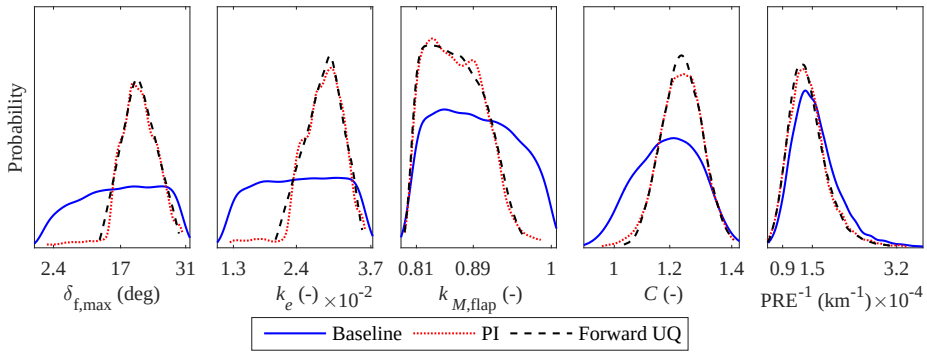


Figure 6.17: Probability density functions for 2nd DOF flap technology with original (blue) distributions for $\delta_{f,max}$ and k_e. After technology maturation the PDFs are computed with PI (red, dotted) and with forward uncertainty propagation (black, dashed). The horizontal axis tick-marks are the 5, 50 and 99 percentiles, in that order.

Regardless, the updated CDFs of cost and fuel burn show the same effect. The corresponding PDFs are shown in Figure 6.17.

Comparing the median (50-percentiles) of these two QoIs, a 3% cost increase and an 8% fuel burn (per passenger kilometer) are measured. Furthermore, the variance of cost decreased by 51% and for fuel burn by 20%. For the forward propagation, a 3% cost increase and 9% fuel burn reduction are computed. Thus, as Figure 6.16 already shows, the two approaches give an almost identical result. However, the variance reduction using forward propagation is more pronounced: 59% as opposed to 30% for PI. This is attributed to the discrepancy in lower percentiles, mentioned earlier.

It should be re-emphasized that the above approach only works when the matured technology distributions lie within the originally sampled distributions. That is because PI can only re-weight existing samples; therefore, samples outside the original set cannot be created or inferred. However, the modified distributions should not necessarily have the same shape as the original ones.

6.4. PI FOR TECHNOLOGY AND PORTFOLIO SELECTION

In contrast to the previous test case, all three technologies are investigated here. Rather than inspecting a single technology, the differences between all portfolios resulting from multiple technologies can be studied. A portfolio is simply a set of included technologies, represented as a vector, where each entry corresponds to a technology and is 1 when it is included and 0 otherwise. Thus, there are 2^n portfolios in total, where n is the amount of technologies. Additionally, the strength of PI is showcased: dealing with multiple objectives. Rather than only having PRE^{-1} as goal, cost is a goal as well.

For the present study, the resulting CDFs for PRE^{-1} and cost are depicted in Figure 6.18 for each portfolio. It becomes clear that in terms of PRE^{-1} , portfolios with technology 3 (third digit is 1) perform better than the ones without it. The portfolio with all technologies included (111) performs best (it is furthest to the left), which can be

expected since none of the technologies were defined to have negative effects on fuel burn. Second best is the portfolio 101, which combines technologies 1 and 3. The worst portfolio is the baseline (000). Thus, already from that figure, if PRE^{-1} is the only requirement, an ordering and selection of the portfolios can be made. However, with cost

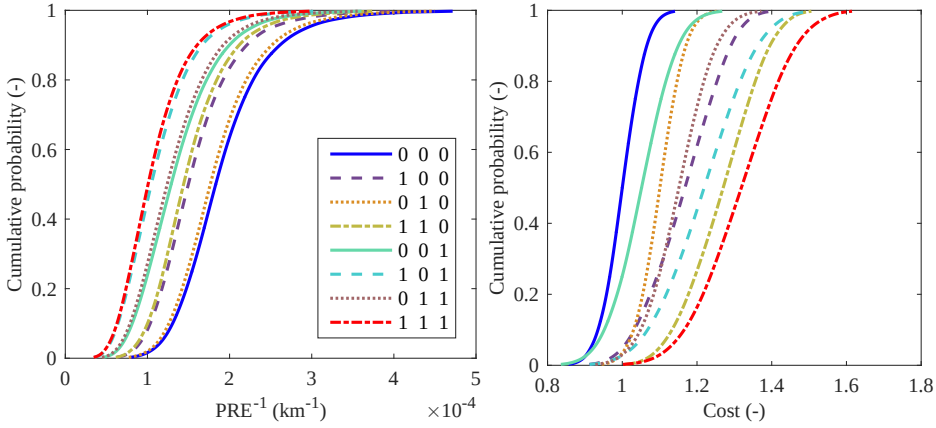


Figure 6.18: CDFs of PRE^{-1} and cost for multiple portfolios.

(see Figure 6.18) there is an opposite trend: the more technologies included in a portfolio, the higher the cost. So here, portfolio 111 performs worst (furthest to the right) and the baseline performs best. There is a different spread in the CDFs for cost, so when constraints are imposed on fuel burn and cost simultaneously, there is no direct way of telling which portfolio performs best. Consequently, concluding which technologies are most promising is not directly observable anymore.

6.4.1. TECHNOLOGY PRIORITIZATION WITH MULTIPLE OBJECTIVES

The strength of PI lies in its ability to deal with multiple, possibly conflicting, goals. However, the way in which the constraints are set up affects what query is posed to PI and provides different results. For this study, the multi-objective constraint should guide the obtained samples towards the Pareto front of fuel burn and cost. That way, we learn which distribution of portfolios is closest to the Pareto front. Thus, this distribution gives the best trade-off between the objectives.

As pointed out by Binois et al. [173], there is an analogy between Pareto fronts and the level curves of a copula[1]. The zero-level curve of a copula corresponds to the Pareto front of the multivariate distribution that the copula represents. In order to redistribute the samples closer to the Pareto front, new margins have to be specified for the objective variables. However, specifying new margins alters the copula as well, because the copula is computed from the margins as explained in subsection 2.8.3. This is not desired, because the dependency captured by this copula is a result of the model and should be

[1] A level curve of a multivariate function f is formed by all solutions $\{\boldsymbol{x}\}$ where $f(\boldsymbol{x}) = c$, i.e. where the function f has a given value c.

treated as a physical fact. Therefore, the copula itself has to be constrained. Such a constraint can be implemented as the sum on the margins in copula space (the variables U_i in subsection 2.8.3). Furthermore, this sum has to be updated every iteration of IPF, because IPF updates the margins on every iteration. The quantiles of the sum are kept constant, though, because that will fix the copula.

To demonstrate this procedure, both a PRE^{-1} target and cost target are specified. The PRE^{-1} target are the percentiles of the 111 portfolio CDF. The cost target are the percentiles of the baseline CDF (000 portfolio). These are the extremes of the sample space, as shown in Figure 6.18.

IPF performs exactly as intended with the constraints on the margins of PRE^{-1} and cost and a constraint on their copula. The margins are satisfied and the copula is hardly affected, as Figure 6.19 and Figure 6.20 show. The obtained CDFs (see Figure 6.19) match

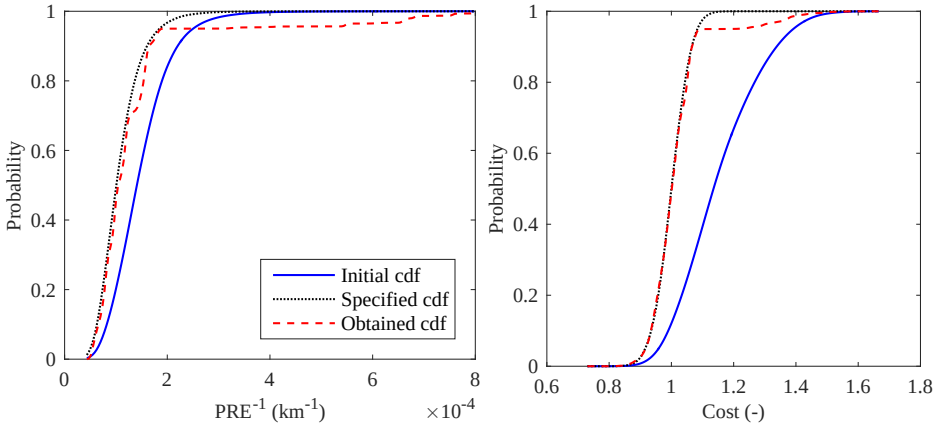


Figure 6.19: Comparison of initial, specified and obtained CDFs for PRE^{-1} and cost, using PI with a multi-objective constraint.

the specified CDF perfectly at the constrained percentiles. From the 95^{th} percentile on, the CDFs do not match, which is to be expected due to the discrete nature of PI. The level curves after PI (red, dashed lines in Figure 6.20) are in the same location as the initial ones (black, solid), although they wiggle around these somewhat. That is a result of the discrete constraints, in combination with only a relatively small subset of samples receiving most of the weight after PI. It can be shown that the copulas before and after PI are very similar by computing the Pearson correlation coefficients. In copula space, the correlation initially was -0.3956, while after PI it is -0.3871: a difference of only 2%. When the constraints are less stringent, even better agreement is obtained.

The previous results may be further explained by focusing on what happens in sample space. The bivariate distribution of PRE^{-1} and cost is shown in Figure 6.21 on the left, with the level curves of the 2-dimensional CDF. These level curves are similar to the ones of the copula in Figure 6.20. The center plot in Figure 6.21 shows how these level curves shift after PI, due to the updated margins. All except the 95^{th} percentile lines have moved towards the lower left corner; thus, reducing both PRE^{-1} and cost. Furthermore,

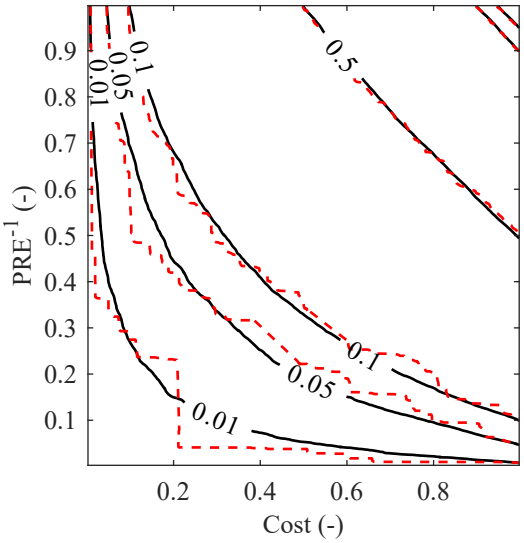


Figure 6.20: Comparison of copula level curves of PRE^{-1} and cost, before and after PI with a multi-objective constraint.

the lines are closer together: the updated 90th percentile is on the initial 50th percentile, and the updated 50th percentile on the initial 10th percentile. Therefore, the reweighted samples are squeezed into a region closer to the Pareto front. In the right plot in Figure 6.21 the updated bivariate distribution is shown. Clearly, only few samples remain after reweighting, which explains the non-smooth results observed in Figure 6.19 and Figure 6.20.

As a result of the multi-objective constraint the portfolio and technologies frequencies in the re-sampled set have changed as well. This is the result of main interest, and it is shown in Figure 6.22. These frequencies are simply the amount of samples that contain a certain portfolio or technology, with respect to the total number of samples. The change with respect to the initial sample set is shown, because not every portfolio or technology is equally represented in the initial set (i.e. the initial frequencies are not uniformly distributed).

Mainly portfolios 001 and 101 have received more weight, leaving technology 3 to see an increase in frequency as well. Technology 1 only suffers a small decrease, while technology 2 clearly does not satisfy the imposed constraint. When selecting a portfolio, it should be 101 as it is most featured in the re-sampled set, while technology 3 should receive most development resources.

6.4.2. TECHNOLOGY PRIORITIZATION USING GROUPED SAMPLE RE-WEIGHTING

The previous results were obtained with individual sample re-weighting. However, the grouped sample re-weighting approach should be considered as well. The difference between the individual sample re-weighting and grouped sample re-weighting approaches

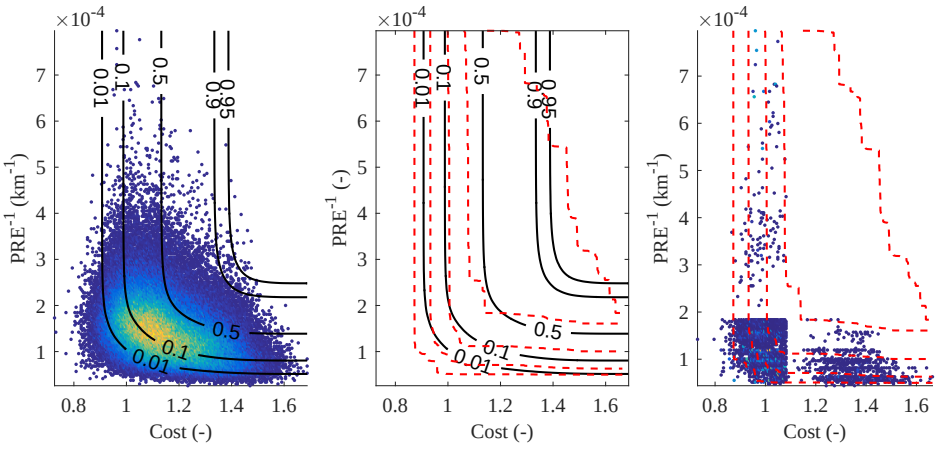


Figure 6.21: Comparison of bivariate distribution of PRE^{-1} and cost, before and after PI with a multi-objective constraint.

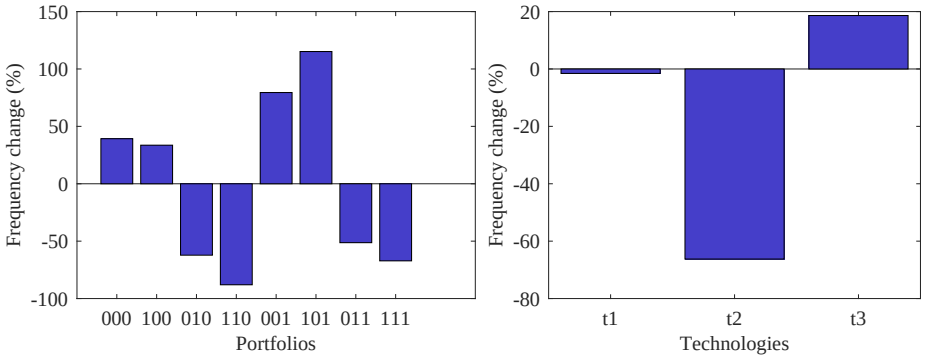


Figure 6.22: Portfolio and technology frequency shifts using PI with a multi-objective constraint.

can be considered by observing Figure 6.23. On the left, the bivariate distributions for each portfolio of PRE^{-1} and cost are shown. This is the entire sample space that PI works with. When constraints are imposed and PI is performed, only a subset of these samples remains. The center plot in Figure 6.23 shows this for individual sample re-weighting. It becomes clear that a lot of samples receive near zero weight to move the overall bivariate distribution to the bottom-left corner. Moreover, the shapes of the colored bivariate distributions (each corresponding to one portfolio) change as a result. That means the underlying distributions of the variables for each portfolio have shifted (similar to the single technology case in Section 6.3). This effect is desired when one is interested in which portfolio has the most potential to satisfy the imposed goals, when it's initial distributions are not set in stone. On the contrary, when the distributions for each portfolio are well defined and should not be allowed to change, the grouped approach is needed. It's result is shown in the rightmost plot, where the bivariate distributions of two portfo-

lios remain, while all others have zero probability mass. The shapes of these bivariates have not changed with respect to the leftmost plot.

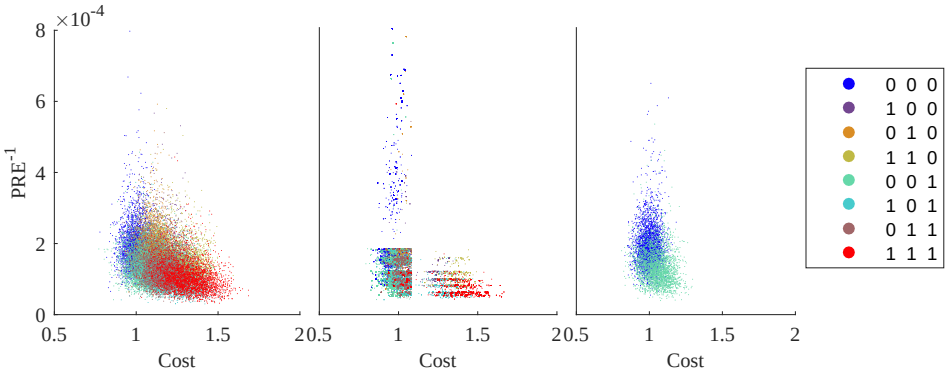


Figure 6.23: Bivariate distribution of PRE^{-1} and cost when: left) no constraints are imposed, i.e. before PI, center) the margins of cost and PRE^{-1} are constrained and with individual sample re-weighting, and right) the margins of cost and PRE^{-1} are constrained with grouped sample re-weighting.

The grouped sample re-weighting has to be performed using PARFUM, because IPF is very likely not to converge. That is because there is too little room to play when only the weight of entire portfolios may be changed. For this reason, PARFUM will also not achieve the specified margins exactly, but will get as close as possible. The distance between the initial, obtained and specified CDFs can be computed using the metric presented by Cook et al. [174]. This shows for PRE^{-1} that the initial distance to the specified constraint is $6.1 \cdot 10^{-5}$ and after PI it is $5.1 \cdot 10^{-5}$. Similarly, for cost the initial distance is 0.24 and after PI it is 0.055. Because the distance after PI is not zero, the constraints are not satisfied exactly. Nonetheless, the distance clearly reduces, which shows that PARFUM moves towards those constraints. It is the direction in which the portfolio weights change that we are interested in. These frequency shifts are shown in Figure 6.24. Technology 3 is clearly the best choice, as it is also the only portfolio that receives increased weight, alongside the baseline portfolio. This is not too different from the individual sample re-weighting result, except that technology 1 has completely dropped from the re-sampled set.

6.5. DISCUSSION

In the foregoing, a probabilistic assessment of technologies is presented, and how to select those technologies based on their probabilistic outcomes. Conventional approaches to this problem do not employ uncertainty. Instead, deterministic points are picked at which the technologies are evaluated and selected on. In this section, the merits of the current probabilistic approach are discussed, compared to the conventional approach. Furthermore, some drawbacks are identified as well.

Using probability distributions instead of deterministic values not only reflects uncertainty during the conceptual design phase, but also allows inclusion of difficult-to-quantify QoIs. Take the cost in this study, for example, which is only reflected using

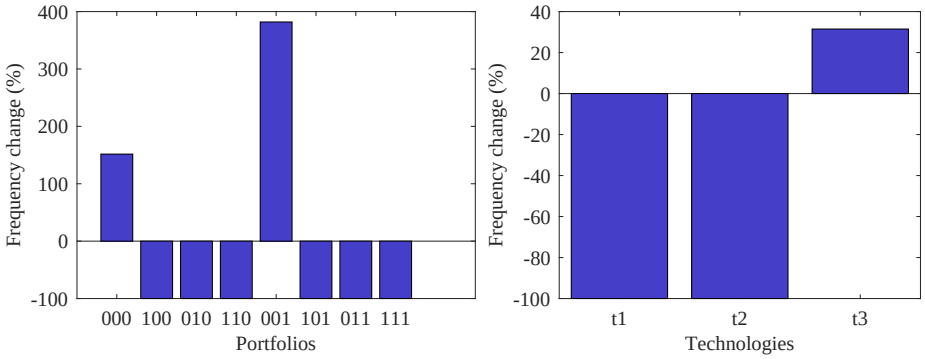


Figure 6.24: Portfolio and technology frequency shifts using PI with a multi-objective constraint, grouped sample re-weighting and run with PARFUM.

engineering insight to increase with higher technology performance. A Gaussian noise is added to introduce uncertainty in that assumption. The actual value of the cost metric does not matter, as only the relative change is of importance. Similar approaches can be taken to include strategic preferences, or metrics such as reliability or aesthetics. The fact that a distribution is used rather than a deterministic, fixed point reduces the bias of assumptions made during this process.

Another argument favoring the use of a probabilistic approach is when the effect of technologies are highly non-linear in the QoIs. In the present work the effect of each technology is fairly linear, which shows in Figure 6.18 as parallel CDFs. As soon as the CDFs cross each other more, the benefit of a technology over another becomes more ambiguous. Consequently, picking a deterministic point at which to evaluate them becomes more difficult and arbitrary.

IPF requires a set of samples, which is why sampling is used as the uncertainty propagation technique. The obvious disadvantage of sampling is the computational cost. Thousands of samples are no exception (we used 10 to 20 thousand samples in this study), and for each sample, the analysis method has to be run. This incurs significant computational time, especially for more advanced simulations (our simple mission analysis takes a couple of seconds to run, so for all samples several hours are required). In future studies, therefore, strategies to reduce the amount of required samples should be researched. One idea is to not compute all portfolios, but only a subset that covers the design space efficiently. Nonetheless, this issue is common to conventional sample-based approaches as well. Conversely, PI itself is very fast, and incurs no significant time expenditure.

However, because PI only relies on a sample file, any analysis method can be used. Therefore, black-box simulations and complex function can be employed easily. The only restriction is, again, that it is fast enough to generate thousands of samples.

Another effect of PI only relying on a sample file is that it can never exceed the boundaries defined by those samples. Thus, PI is confined to the design space covered by the initial sample set. If, however, distributions after PI would skew towards the limits of

their range, that would indicate that the design space is not large enough and a more optimal solution may be found outside of the existing samples.

As the previous paragraph suggests, PI may be used to iteratively re-define the starting distributions and dependencies for the analysis. In some cases, PI incurs a dependency between variables that should not be there, requiring a statement of independence. Otherwise, PI may generate a dependency that was not modeled, but should be present from a physical or model perspective. Then, this dependency can be included in subsequent runs of the analysis routine.

With respect to the accuracy of the results, two things can be said. First, PI is devised to provide an approximation of the inverse of a stochastic function. Therefore, besides the inputs and outputs being random variables, the re-distribution of these variables due to PI is approximate. That is a result of the discretization of the constraints, and the re-sampling rather than inverting of the function. Second, after re-weighting, the original samples are re-sampled. Whenever sampling occurs, there is an associated sampling bias, due to the random nature of a sampling process. This is expected to have very little influence on the CDFs after PI, but it could show some shift in the portfolio and technology frequency computations. Thus, it is advised to perform re-sampling multiple times and take an average of the results, or show an error bar. Nonetheless, the deviations due to sampling bias are expected to be small and when a large difference in frequency is observed after PI, this likely outweighs any possible sampling bias.

Optimization approaches are an alternative to perform function inversion. When the metrics used for optimization are in the form of probability distributions, only some scalar measures of those distributions (e.g. mean and variance) are actually used as objective functions. PI uses the entire distribution directly; thus, keeps more information. Furthermore, PI easily handles multiple objectives, without requiring some weighted combination of the objectives as optimization approaches do. Finally, while optimization approaches aim to find a minimum objective value, PI only targets a specified value. It therefore is less inclined to push the boundaries of the design space. However, which of the two methods is more appropriate depends on the problem at hand. For technology prioritization, however, we advocate the use of PI.

PI also differs from design exploration methods in that it defines a new design space that satisfies the specified requirements, rather than only selecting subsets of the initial design space that meet those requirements. In other words, design space exploration can only indicate the direction to move towards, while PI specifies the path to take.

6.6. CONCLUSION

The merits of a probabilistic approach towards technology evaluation and selection are exposed in this chapter. Rather than evaluating a deterministic, fixed point design (e.g. only one specific aircraft and mission), a technology is evaluated for an entire space of aircraft and missions. Using probabilistic inversion, target distributions on QoIs are set, which consequently result in different distributions on the input variables. This shows how the technology variables (and possibly others) need to change in order to achieve certain requirements. Similarly, when multiple technologies are combined into portfolios, the combined effects are quantified, and probabilistic inversion shows how the frequency of the technologies should change to achieve certain goals. Multi-objective goals

can be imposed on PI, without having to be converted to a single-objective function, as usually is done in design optimization. Moreover, PI can be used with any analysis method, because it only relies on a set of samples obtained from the analyses. However, that limits the complexity of the model, due to computational time constraints. Thus, PI is a powerful tool during conceptual design to explore the technology design space and prioritize technologies for selection.

This chapter concludes the technical developments of this dissertation. It completes the process flow of technology representation, to portfolio generation, portfolio evaluation and, finally, portfolio selection. The following chapter takes these developments and synthesizes them into an overview that is compared to the state-of-the-art. Furthermore, it discusses how the method differs from actual practice in the MANTA project, where this research originated from.

7

SYNTHESIS

In this chapter, the state-of-the-art approach for technology evaluation and selection is compared with the framework proposed in this dissertation. Both approaches are summarized graphically and then compared in section 7.1. Subsequently, section 7.2 discusses the MANTA project. First, the technologies in MANTA are described, followed by a description of the process that was implemented to reach the selection of two final technologies. Then, a description is given of how the methodology proposed within this dissertation may be applied, in section 7.3.

7.1. METHOD OVERVIEW AND COMPARISON WITH STATE-OF-THE-ART

Let us combine the theoretic elements developed in the previous chapters to synthesize a holistic methodology towards technology evaluation and selection. First, however, the state-of-the-art method is shown, to establish a baseline on which the approach that is presented in this dissertation improves.

7.1.1. STATE-OF-THE-ART METHODOLOGY

An overview of the state-of-the-art method is presented in Figure 2.4, which is an elaboration of Figure 1.5(a). The method takes in a set of technologies and a system of interest, which are combined into technology portfolios. These are subsequently described with impact factors, which are chosen among a set defined by the available analysis tool. The analysis tool then computes several quantities of interest, which are combined with qualitative selection criteria, if there are any. These criteria are then scored and weighted by a group of experts to produce the final technology ranking.

Notice how all the steps in this methodology are performed by the users of the method, except for the computation of the quantities of interest. Furthermore, the technologies, system of interest and technology portfolios are all represented in a textual format. Only the impact factors and quantities of interest form numeric data. Also notice how the state-of-the-art method does not contain any steps that are problem-independent. Indeed, all steps have to be fully repeated for each technology evaluation and selection problem.

7.1.2. DISSERTATION METHODOLOGY

The developments from Chapters 3 through 6 are shown together in Figure 7.1. This figure is an elaboration of Figure 1.5(b). The method starts with the ontology developed in Chapter 3. Additionally, graphs, knowledge graphs and graph transformations form the basis of the rest of the approach. The graph theory and graph transformations are outlined by dashed lines to indicate they are not developed as part of the method, but rather included as a starting point. Based on the ontology, FOL rules, parameters and analysis methods are defined. From that point on, the system of interest and technologies are defined, which are combined into technology portfolios. These are parameterized using the analysis methods. Then, the user is queried for the quantities of interest, input variables and their (joint) distributions. This data is combined into the computation graphs, which enable the evaluation of the quantities of interest, either probabilistically or deterministically. Assuming a probabilistic approach is adopted, the method results

in distributions on all variables, which can be fed into the probabilistic inversion (PI) method from Chapter 6 to produce a final ranking of the technology portfolios.

The proposed method clearly shows three separate layers: a problem-independent layer, a unique layer for a problem, and a variable layer for a problem. A problem in this context is a set of technologies and one system of interest, to be fed into the technology evaluation and selection process. Thus, everything up to the analysis methods is problem-independent. That is a result from the context that the ontology provides to how parameters and analysis methods are specified. The ontology also forms the building blocks for the knowledge graphs that describe the system of interest and technologies. The unique layer is a set of elements of which there are only one in a problem. Obviously, the system of interest and set of technologies belong here, as these together define the problem. The portfolios and resulting system graphs are a direct result of the SoI and technologies, and, in turn, the dependency graphs as well. Variable per problem are then the queries that are posed on the technology set. For each query, different quantities of interest and/or input variables and/or distributions apply, which result in different computation graphs and outcomes.

Notice that not all steps are executed by the users; instead, many steps in the unique and variable layers are performed by the algorithms developed in the foregoing chapters. Additionally, no information is stored in a textual format, but only in the form of (knowledge) graphs or numerical data.

7.1.3. COMPARISON

It should be clear that the methods shown in Figure 2.4 and Figure 7.1 partially overlap: they are effectively analogous in the unique and variable layers. In other words, from the definition of the system of interest and set of technologies, portfolios are generated, represented as numerical quantities, evaluated with some analysis method and then selected based on a set of quantities of interest and/or selection criteria. However, there are important distinctions in terms of who/what performs the steps, how the data is stored and represented, and which steps are taken precisely.

First, the proposed method features significantly more automation than the state-of-the-art approach. While in the latter only the evaluation of the quantities of interest is performed automatically, the proposed methodology is automatic after the definition of the SoI and technologies. It only requires user interaction to define the quantities of interest, known inputs and their (joint) distributions.

Second, textual data is hard to parse, open to interpretation and does not allow for analysis. Because of that, information is difficult to retrieve, share and reuse. Furthermore, textual data inhibits consistent and reliable decision making. In the proposed approach, all data is either stored in a knowledge graph, or is numeric. This makes it machine-interpretable and removes all the disadvantages that textual data has. There are two objections to the use of knowledge graphs to store the data, however. First, generating the knowledge graphs is time-consuming and requires considerable knowledge of the employed ontology. Second, for the rest of the approach to work automatically, all quantities of interest must be numerical. That is also why Figure 7.1 does not feature the selection criteria like Figure 2.4 does. These selection criteria may be subjective and rep-

resented on an ordinal[1] scale, whereas the proposed method only works with objective, quantifiable criteria, measured on either interval[2] or ratio[3] scales. Of course, any subjective criterion may be represented on such a scale with a numerical value. Otherwise, one could incorporate subjective criteria in the final decision making process, in order to circumvent this drawback.

Third, the proposed method involves more steps than the original approach, although most of them are automated. Most notably, the extra steps involve the creation of the dependency and computation graphs. Because these are unique for each technology portfolio, the approach is more flexible and extensible than the state-of-the-art, where typically one or few existing analysis tools are employed. This also removes the dependency of the impact factors on the analysis tool. Sure, the parameterization of each portfolio depends on the analysis methods that are applied to it, but each portfolio can have a different set of parameters. Furthermore, the dependency graphs are created using a pre-defined suite of analysis methods that are automatically mapped onto the system graphs, as is elaborated in Chapter 5. This fact makes the approach modular and enables novel analysis methods to be included, such that novel technologies can be readily analyzed. As such, the gap between levels 1 and 3 in Figure 1.4 is reduced.

[1] An ordinal scale offers an ordering of its elements, where a larger value is better than a lower value. However, the difference between two values has no meaning, i.e. the difference between 3 and 10 is interpreted as equal to the difference between 3 and 7.

[2] An interval scale orders its values just like an ordinal scale does. However, in contrast to an ordinal scale, the interval scale stipulates that the differences between values have a meaning. Thus, in an interval scale, the difference between 10 and 20 is the same as the difference between 30 and 40.

[3] A ratio scale builds on interval scales by incorporating a notion of the ratio between values. That is to say, 20 is two times as much as 10, for example.

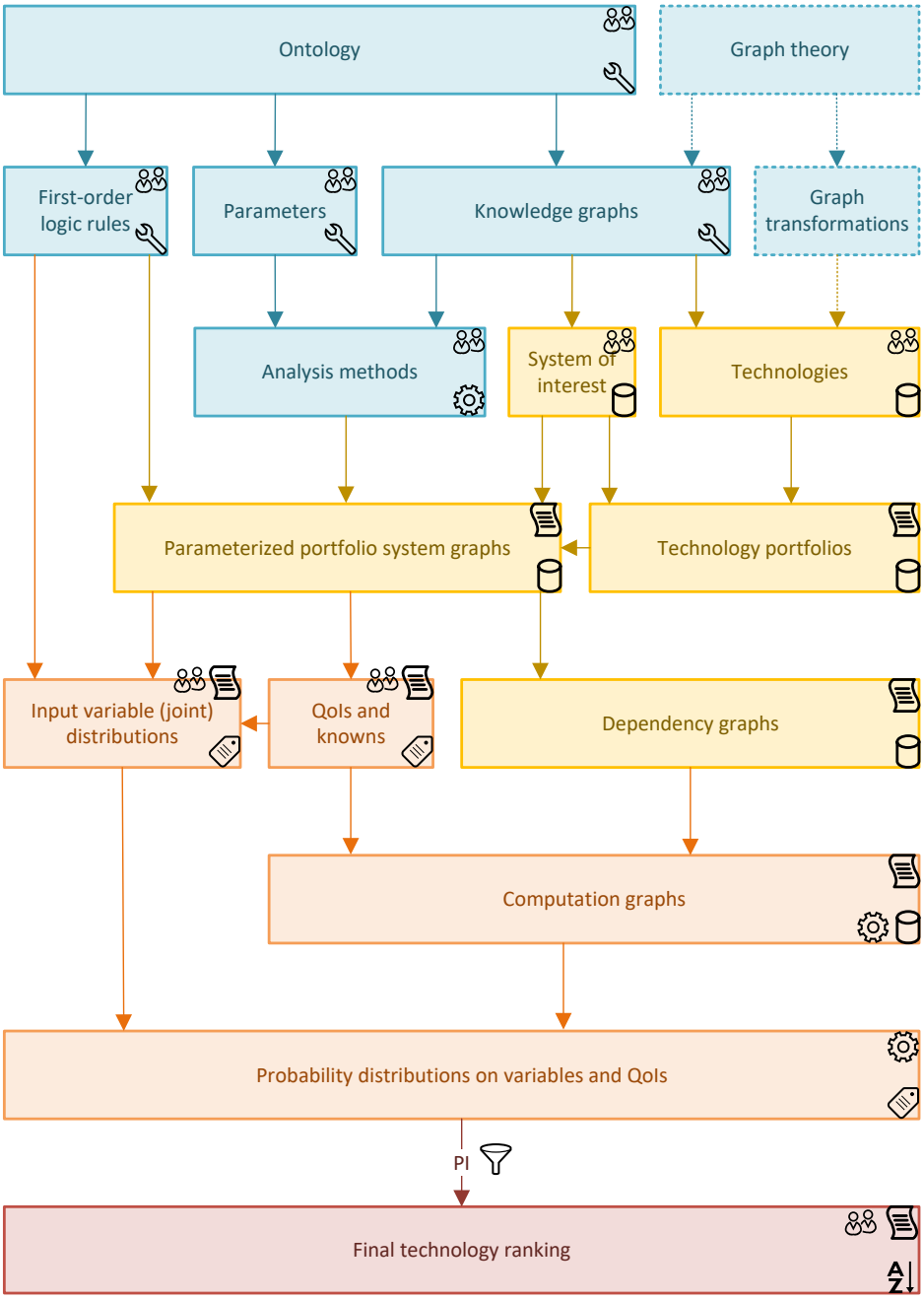


Figure 7.1: Overview of proposed method for technology evaluation and selection. Refer to Figure 2.4 for a legend of the icons and colors. The dashed lines indicate the component is used in the method, but not part of it.

7.2. THE MANTA PROJECT

This thesis research has its roots in the MANTA project, which was introduced in section 1.3. MANTA stands for Movables for Next Generation Aircraft and was started to develop and demonstrate innovative multifunctional movables that increase airframe efficiency over the complete flight envelope of business jets (BJ) and large passenger aircraft (LPA) as contribution to the societal challenge for this topic to reduce 3 to 5% CO_2. Therefore, one of the key quantities of interest for MANTA is aircraft fuel burn. In MANTA, an initial set of 29 novel movable technologies was devised, to be analyzed and down-selected to 2 technologies. One for a high-speed business jet (HSBJ) and one for a large passenger aircraft (LPA).

Because of temporal misalignment, the methodology presented within this thesis could not be applied to the MANTA project. However, MANTA provided interesting technologies for ideation, and a showcase of the state-of-practice in technology selection in the aerospace industry. This section elaborates on that last point and as such, illustrates some of the shortcomings with current practices that were introduced in Chapter 2.

In the following subsections, the technology evaluation and selection process in MANTA is scrutinized in more detail than the previous discussion of Figure 7.2. The original 29 technologies are presented, after which the down-selection to the six technologies in Figure 7.2 is explained. Then, the aircraft-level performance analysis, used to compute the fuel burn, is discussed. Subsequently, the final technology ranking and selection procedure is elaborated upon. Finally, a reflection on the adopted process is provided, which shows where and how the deviations from the state-of-the-art method may have affected the outcome of the MANTA project.

7.2.1. PROJECT OVERVIEW

The core task of the TUD during the first two stages of MANTA comprised the evaluation of the proposed innovative movable technologies on aircraft level. In other words, the quantities of interest (mainly fuel burn) were to be estimated for each technology, on both a notional high-speed business jet (HSBJ) and a large passenger aircraft. Therefore, the decision problem entailed a set of technologies, of which only two could be selected for further research and development. One technology was to be selected for the business jet platform, while the other would be selected for the LPA.

Within the section Flight Performance and Propulsion (FPP) of the Faculty of Aerospace Engineering at TUD, the aircraft-level assessment of the MANTA technologies was conducted by the author of this dissertation. At FPP, there is a conceptual aircraft design tool called the Aircraft Design Initiator (*Initiator* for short). While its purpose is to produce consistent airplane designs at conceptual level, it was used within MANTA to investigate the impact of the movable technologies on conventional aircraft architectures — the BJ and LPA. The Initiator consists of multiple modules that perform sizing and analysis of various aspects of an aircraft, such as the mass distribution, longitudinal stability, structural layout, cabin layout, engines, aerodynamics, etc. It starts a sizing routine from a set of top-level requirements, which are specified by the user. Several settings and parameters can be tweaked to guide the program to certain solutions. However, the Initiator lacked the ability to receive a fixed aircraft design and consecutively introduce technologies and measure their effect on certain quantities of interest.

For MANTA, an intermediate solution was implemented. The technology evaluation and selection process adopted in MANTA largely followed the state-of-the-art approach, as can be seen by comparing Figure 7.2 to Figure 2.4. The Initiator was modified to keep a certain fixed aircraft baseline and the technologies were introduced as changes to existing variables in the modules of the Initiator. That is, they were modeled as k-factors [31]. However, as shown in Chapter 7, many of the MANTA technologies are intricate, small changes relative to the entire aircraft, and, as such, the Initiator lacked sensitivity to those changes. Furthermore, many of the intricacies of the MANTA technologies could not easily be captured in the existing variables and parameters present in the Initiator. This demonstrates the need for a more systematic approach to technology assessment at aircraft level.

The technology evaluation and selection approach adopted within MANTA is extensively discussed in the following subsections, but Figure 7.2 provides an overview of the entire process. There are four distinctions between the state-of-the-art approach and the one that was conducted in MANTA. First, there are two systems of interest: the high-speed business jet and the large passenger aircraft. This distinction is not impactful, because the two were kept clearly separated within MANTA, and can, therefore, be seen as two distinct technology selection processes. Second, the adaptive secondary air intake technology was selected for further development based solely on the fact that it is an excellent morphing technology demonstrator. This choice technically disregards MANTA's project objective and disregards the state-of-the-art approach for technology selection. Third, the five technologies that were evaluated in terms of their impact on aircraft fuel burn, had to be represented with only three technologies, because the available impact factors disallowed a distinction between two pairs of them. Fourth, and finally, the five technologies were only ranked based on a set of qualitative criteria, while disregarding the quantified effects on aircraft fuel burn. In fact, the TRL was also disregarded in the final technology ranking, while both TRL and fuel burn form MANTA's project goals.

7.2.2. TECHNOLOGIES

At the start of the MANTA project, the consortium came up with 29 technologies, listed in Table 7.1. The technologies can be categorized into four target applications:

1. Adaptive outer wing and winglet (LPA)
2. Multi-functional flap and adaptive trailing edge (HSBJ)
3. Morphing air inlet (LPA)
4. Morphing spoiler (LPA), formerly the Affordable leading-edge slat

Each target application has one of the two aircraft platforms associated with it. Several of the technologies are illustrated in Figure 7.3. An example of the second degree-of-freedom flap is shown in Figure 6.6, and the rotating winglet downer in Figure 6.7.

Note that the morphing air inlet target application only has one technology in it: the adaptive secondary air intake. After some time, Airbus decided it was no longer interested in the morphing spoiler and leading-edge slat target application. Additionally,

Table 7.1: Innovative movable technologies proposed in the MANTA project. The six technologies in bold were selected in a first trade-off.

Technology	Platform	Score 1	Score 2
Adaptive winglet	LPA	-1	3
Winglet with conventional tab	LPA	-1	3
Morphing trailing edge for a winglet	LPA	0	8
Rotating winglet downer	LPA	-1	3
Folding wing-tip featuring load alleviation	LPA	2	10
Morphing trailing edge for the wing	LPA	2	8
Adaptive shock bump	HSBJ	-1	-1
Shock bump combined with adaptive flap	HSBJ	0	4
Loads management with adaptive flap and ailerons	HSBJ	0	8
Segmented ailerons	HSBJ	1	5
Micro-geometric spoiler	HSBJ	-1	-1
Morphing spoiler	HSBJ	2	10
Full-span segmented-flap system	HSBJ	-1	-1
Morphing flap with eccentric rod	HSBJ	1	3
Second degree-of-freedom flap (inside track)	HSBJ	2	10
Second degree-of-freedom flap (outside track)	HSBJ	4	12
Load-bearing flap-support fairing	HSBJ	2	6
Internal kinematic solution for flap deployment	HSBJ	1	9
Thermoplastic welded multi-spar flap box	HSBJ	3	2
Flutter suppression flap	HSBJ	-2	-6
Gust alleviation using pressure adaptive material	HSBJ	3	8
Multi-functional / Reduced track leading edge slat	LPA	4	12
Smart leading edge droop nose	LPA	1	7
Multi-bar linkage deployment slat	LPA	2	4
Corrosion resistant steel slat-track	LPA	1	1
Silent multi-functional Krueger flap	LPA	1	3
Adaptive secondary air intake	LPA	4	12
Double hinged rudder	LPA	2	2
Relaxed static stability	-	5	13

the morphing air inlet proved to be an excellent proving ground for morphing technology, and was determined to be included in the final set of technologies henceforth. As a result, the two technologies to be selected would come from either the first or second target application.

Before any quantitative analysis was conducted, the consortium decided to reduce the available set of technologies even further, to make the evaluation tractable. This is precisely the type of decision the method from this dissertation aims to prevent. A set of metrics was defined to score and rank the technologies qualitatively. These metrics are depicted in Figure 7.4. A technology impact matrix was constructed, of which an excerpt is shown in Figure 7.5. Particularly, the six remaining technologies after this initial selection round are shown. For these technologies, a quantitative analysis has been performed to select one for each of the two target applications (or platforms). Note that the adaptive air intake would be selected no matter what (as a third technology). That decision resulted from the realization that the adaptive air intake would be the most effective demonstrator of morphing technology. However, the decision actually clashes with the project statement, and the decision bypasses the whole technology evaluation and selection process. Ideally, that should never happen.

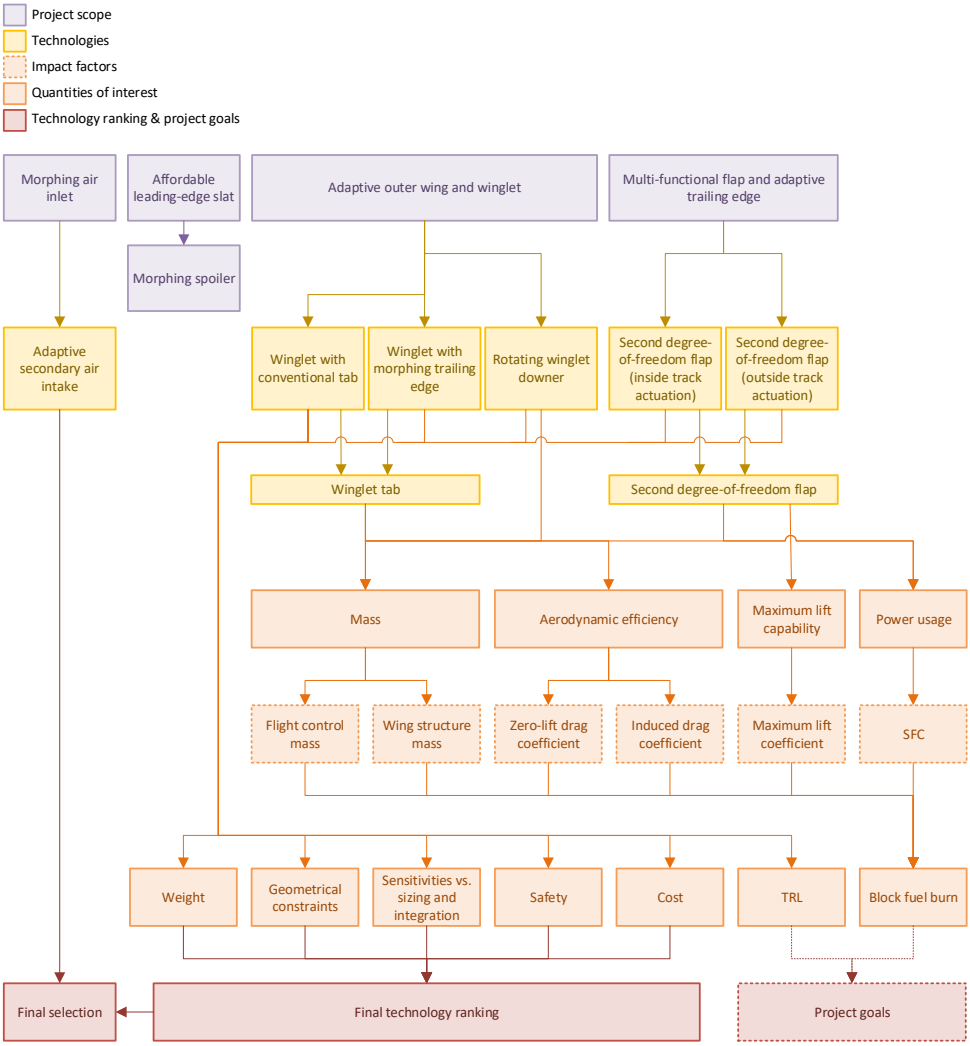
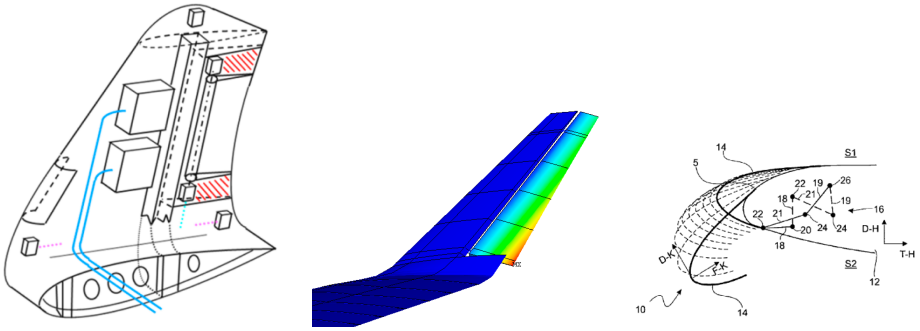


Figure 7.2: Overview of MANTA process for technology evaluation and selection



(a) Winglet tab (b) Morphing winglet (c) Morphing slat

Figure 7.3: Illustrations of several of the MANTA technologies

Metric	Level	Significant negative impact	Small negative impact	No / Negligible Impact	Small positive impact	Significant positive impact
Mass (Component Level)	Component	X<-15%	-15%<X<-5%	-5%<X<5%	5%<X<15%	X>15%
Mass (Aircrat Level)	Aircraft	X<-0.15%	-0.15%<X<0.05%	-0.05%<X<0.05%	0.05%<X<0.15%	X>0.15%
Maintenance Cost	Component	X<-7%	-7%<X<-2%	-2%<X<2%	2%<X<7%	X>7%
Manufacturing/Recurring Cost	Component	X<-15%	-15%<X<-5%	-5%<X<5%	5%<X<15%	X>15%
NRC	Aircraft	X<-15%	-15%<X<-7%	-7%<X<7%	7%<X<15%	X>15%
Aerodynamic Efficiency	Aircraft	X<-0.5%	-0.5%<X<-0.2%	-0.2%<X<0.2%	0.2%<X<0.5%	X>0.5%
Noise	Aircraft	X<-3%	-3%<X<-1%	-1%<X<1%	1%<X<3%	X>3%
C_{Lmax}	Aircraft	X<-5%	-5%<X<-2%	-2%<X<2%	2%<X<5%	X>5%
Power	Component	X<-10%	-10%<X<-2%	-2%<X<2%	2%<X<10%	X>10%
Actuation	Component	N/A	N/A	N/A	N/A	N/A
Activity	Component	N/A	N/A	N/A	N/A	N/A
Volume Requirement	Component	N/A	N/A	N/A	N/A	N/A
Reliability / Certification	Component	N/A	N/A	N/A	N/A	N/A
TRL	Component	N/A	N/A	N/A	N/A	N/A
Assumptions	Component	N/A	N/A	N/A	N/A	N/A

Figure 7.4: Impact metrics for MANTA with color coding, identifying a percentage change required for a certain qualitative impact.

Target Application	ID	Solution	Mass (component)	Mass (aircraft Level)	DOC	Manufacturing/Recurring Cost (Component)	NRC	Aerodynamic Efficiency	C	Reliability / Certification	TRL
Outer wing and winglet	4.02	Conventional tab for winglet	Winglet mass increase.	Reduction through spanwise load optimization & gust load alleviation	Increase in maintenance	Increase	N/A	Reduction of induced in cruise, TO, climb & loiter	N/A	Complexity increase. Actuator failure	2-3
Outer wing and winglet	4.03	Morphing trailing edge for winglet	Increase in winglet mass	Reduction through spanwise load optimization & gust load alleviation	Increase in maintenance	Increase	Increase	Reduction of (induced) drag (compared to tab - 4.02)	N/A	Actuator failure	2
Outer wing and winglet	4.27	Rotating winglet downer	Downer mass increase.	Reduction (of wing structure mass) through active & passive load control.	Increase in maintenance	Increase	Increase	N/A	N/A	N/A	
Multifunctional Flap + TE	4.14	Second degree of freedom flap with actuation inside track	Flap track omission, Wing box protrusion, but replacement of spoilers, possibly aileron	N/A	Increase in maintenance	Dependent on platform /trade-off (reduction possibly)	No effect	Increase L/D in take-off & climb. Optimized off-design cruise performance	Increase	Typical certification issues. Fuel tank penetration. Flap flutter issues	3
Multifunctional Flap + TE	4.15	2nd DOF flap using hinge (external actuation)	Reduction through load alleviation	N/A	Increase in maintenance	Increase	No effect	Reduction of drag in high speed flight	Increase	Low speed control, steep-descent certification	3
Adaptive Air Intake	4.26	Variable stiffness, electrically actuated laminate	Reduction in component mass due to fewer parts	N/A	Reduced maintenance cost	No (small) effect	Increase	Reduction of pressure losses	N/A	Reliability increase, less maintenance	3-4

Figure 7.5: Impact matrix for MANTA

7.2.3. AIRCRAFT-LEVEL PERFORMANCE ANALYSIS

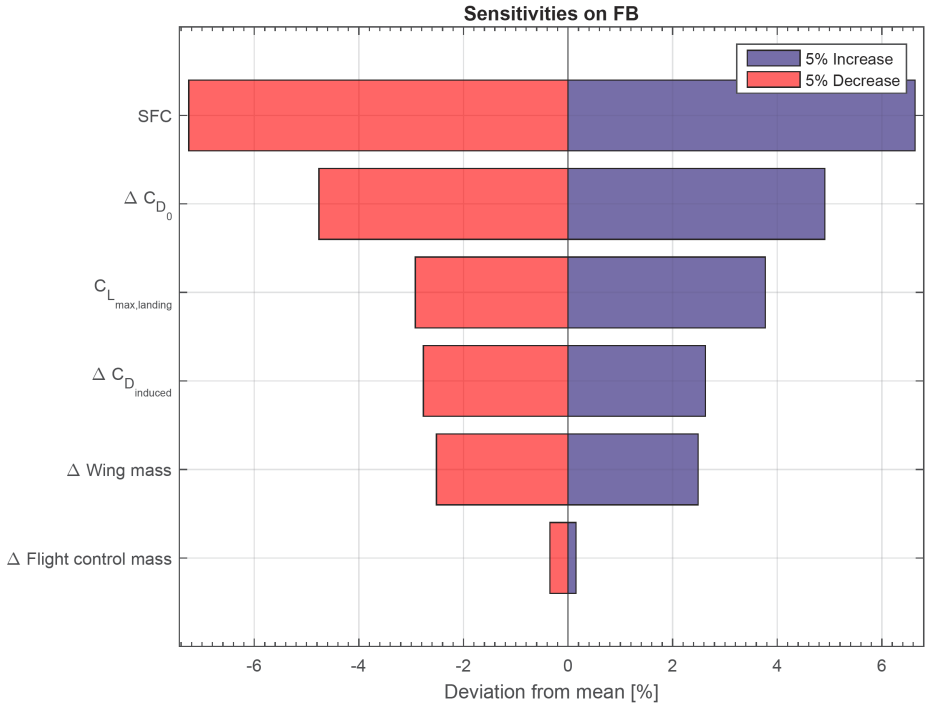
For the evaluation of the impact of each technology on aircraft fuel burn, the Aircraft Design Initiator [175, 176] — or Initiator, for short — was employed. The Initiator consists of a series of disciplinary analysis and sizing modules that are combined in an efficient framework. The individual analysis modules are continuously updated with improved analysis methods to enhance the reliability or flexibility of the Initiator.

The Initiator typically generates a conceptual sizing of an aircraft design from top-level requirements. However, in this case, the two target platforms were to be modeled and kept fixed, whilst only changing parts that reflect the implementation of the MANTA technologies. Both the LPA and HSBJ were modeled such that the Initiator would produce a design with key characteristics similar to the specifications of the reference aircraft.

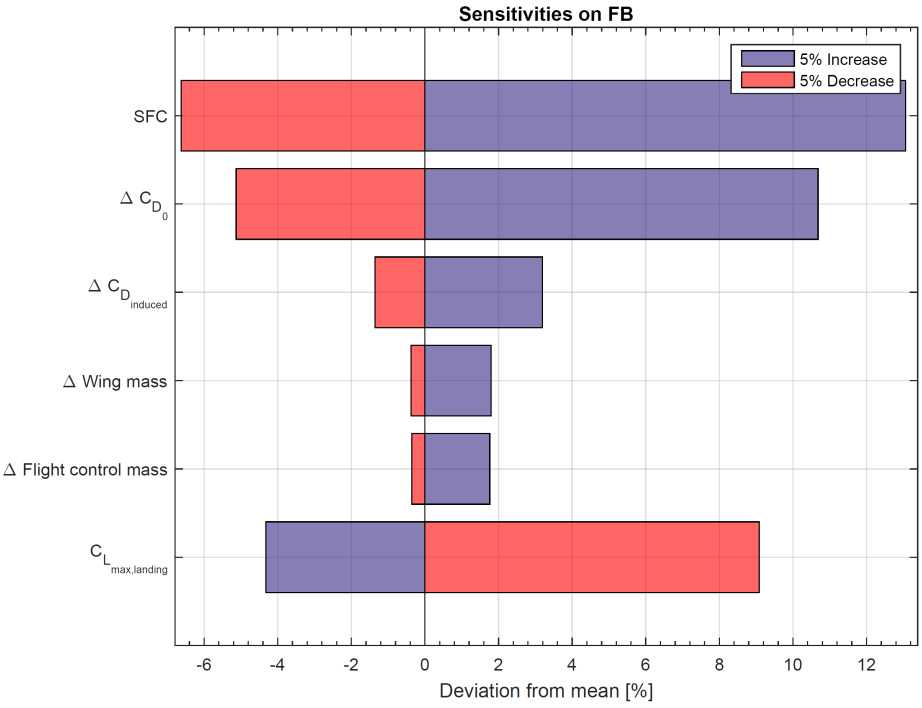
Even though the Initiator consists of a plethora of analysis modules, and we even included a newly developed maneuver load alleviation analysis method, it lacked the capability to model all intricate details of the MANTA technologies. For example, the Initiator does not support enough detail to model the various actuators and kinetic mechanisms required for the different movable technologies. Furthermore, the aerodynamic model does not support the various movables. Therefore, the common approach of k-factors (see section 2.1) was adopted to model and quantify the technologies. The k-factors already exist as input variables to the Initiator modules and coarsely capture the effects of the technologies. Six k-factors made up the analysis: $C_{L_{\text{max,landing}}}$, SFC, Δ wing mass, ΔC_{D_0}, $\Delta C_{D_{\text{induced}}}$ and Δ flight control mass. These k-factors were subsequently fed into the analysis methods of the Initiator to estimate the aircraft fuel burn for a given mission.

Another problem presented itself during the MANTA project. The consortium partners were unable to provide estimates for these six k-factors. Because of that, it was chosen to perform a sensitivity study where each k-factor was increased and decreased by 5% with respect to the baseline value, and observe the effect on the quantity of interest — fuel burn. This would provide a guideline of what would be required for any amount of fuel burn reduction.

The sensitivity of fuel burn with respect to the six k-factors is shown in Figure 7.6(a) for the LPA and in Figure 7.6(b) for the HSBJ. Notice how the specific fuel consumption (SFC) is the most influential factor, for both platforms. The zero-lift drag C_{D_0} is in second place for both. Beyond that, the ranking of the k-factors differs per platform, which should translate into different technologies performing differently.



(a) LPA



(b) HSBJ

Figure 7.6: Sensitivities of the two MANTA reference platforms to six k-factors

7.2.4. FINAL SELECTION

The final selection criteria do not include the impact of the technologies on the aircraft fuel burn. Even though MANTA's objective is to reduce fuel burn by 3-5%. The following reasoning can be found in the selection report:

> "Although, the aim of MANTA is to increase the potential of movables, we decided not to include aircraft performance into the criteria list. The reason is that flap concepts would outrank winglet concepts on this criterion and therefore a split is applied between flap concepts and winglet concepts. In addition, the concepts are ranked by weight. Aircraft performance is largely dictated by component weight and aerodynamic shape. The aerodynamic shape is a criterion, even so the weight. If we would also add aircraft performance to the list, we would count it twice. Therefore, this criterion is not, as such, in the list."

Ironically, this also happened to another of the key technology metrics:

> "One of the goals of MANTA is to developing promising concepts to TRL4-5. This is translated as feasible within MANTA. However, this is a yes or no rating and through this difficult to rate. Therefore, this criterion was not included in the list."

The final selection criteria that were used to rank the technologies can be found in Table 7.2. For the remaining five technologies (the sixth, adaptive air intake, technology would be further developed regardless) these criteria were evaluated loosely based on available quantitative data, but mostly based on expert judgment. The final scores and ranking are shown in Table 7.3. Some interesting observations can be made here. For example, the "Sensitivities vs. sizing and integration" and "Safety" criteria proved indecisive, as they provide equal scores to all concepts within that target application. Because these are qualitative criteria, it is very likely the experts were unsure how exactly the technologies differed in these aspects, and conservatively gave them identical scores. Another observation is that the two chosen best technologies are also best when both target applications would not have been separated. This shows the decision was invariable to the target application division.

7.2.5. REFLECTION

The assessment of the criteria and scores in Table 7.3 may have been biased. Before the selection was made, the consortium already had preferred technologies and the decision matrices were filled out to favor the desired outcomes. This observation is seconded by Peerlings [177, p. 133], who investigated the project's plan, deliverables and minutes of meetings. A discrepancy between the stated objective of a fuel burn reduction combined with elevating technologies from TRL 1-3 to 4-5 and the consortium's industrial interests was noted. Thus, it is clear that there were implicit considerations and objectives when selecting among the available technologies. Although there is no hard scientific evidence for this, it appears that this occurs frequently in decision making with qualitative criteria. Decision makers should at least include a "preference" criterion. While this does

Table 7.2: Final selection criteria for MANTA technologies

Subject	Criterion	Sub-criteria	Weight
Geometrical constraints	Internal constraints	Fuel tank volume	5
		Wing box height	4
	External constraint	Ground clearance	2
Sensitivities vs. sizing and integration	Dimensional stability	Aerodynamic shape	4
	Positioning accuracy	Angular setting	5
Safety	Failure probability	Actuator jam	3
		Incorrect actuator movement	3
		Fire hazard	5
Weight	Structure	Part count / degree of integration	3
		Resistance against impact	3
	Actuation system	Actuator amount and size	4
		Total length of lines	3
	Integration in wing structure	Structural reinforcements and modifications	3
		Fairings and aerodynamic seals	3
Cost	Manufacturability	Use of proven production methods	3
		Enabled reliable recurring production	3
		Enabled high rate production	3
	Direct operating cost	Inspectability	5
		Maintainability	5
		Repairability	3

Table 7.3: Final decision matrix for five final technologies

Selection Criteria	Winglet Concepts			Flap Concepts	
	Morphing tab	Conventional tab	Rotating downer	Outside track	Inside track
Geometrical constraints	41	33	25	29	15
Sensitivities vs. sizing and integration	9	9	9	27	27
Safety	33	33	33	27	17
Weight	57	71	43	71	77
Cost	48	54	66	66	44
Score	188	200	176	220	180
Rank	2	1	3	1	2
Selected	no	yes	no	yes	no

not remedy the bias, it makes the consideration explicit, rather than factor it into other criteria implicitly. This improves the transparency and repeatability of the decision.

Another questionable decision during MANTA was when the initial set of technologies was reduced to the six technologies for which a quantitative analysis was performed. Given the qualitatively scored impact matrix in Figure 7.5, and scores on the impact metrics, the total scores in Table 7.1 are obtained. The weighting scheme is provided in Table 7.4 Even though the weights of the impact metrics have been defined a posteriori, there is some mismatch between the selected concepts and the highest scoring technologies in this list. Most notably, the multi-functional (reduced track) slat concept in the Affordable Slat target application performs as well as the adaptive air intake, while that whole target application was dropped. Furthermore, the morphing spoiler performs equally well to the second DOF flap with inside track actuation, and received its own target application that replaced the slat target application. Nonetheless, the morphing spoiler was excluded from the final technology selection. The outer wing and winglet target application shows the most discrepancies: the conventional tab for winglet and rotating winglet downer have significantly lower scores than the morphing trailing edge for winlget, folding wing-tip and morphing trailing edge. The last two were not included for the final selection, while they are among the top three scoring technologies for this target application. Finally, relaxed static stability appears to be a very interesting technology for the MANTA objective, but was not considered, because it is outside of the expertise of the consortium.

7.3. APPLICATION OF METHODOLOGY TO MANTA

This dissertation focuses on formalizing and automating the technology evaluation and selection process. The ability to formally capture knowledge about technologies is a ben-

Table 7.4: Two weighting schemes for Table 7.1

Impact metric	Score 1	Score 2
Mass (component)	1	1
Mass (aircraft)	1	5
Aerodynamic Efficiency	1	5
$C_{L,\max}$	1	3
Power	1	5

Table 7.5: Functions of the movable technologies in MANTA

Function	Sub-functions
Mass reduction	Maneuver load alleviation Gust load alleviation Smaller component Lighter material
Drag reduction	Zero-lift drag Wave drag Induced drag Trim drag Laminar flow
High-lift performance increase	$C_{L,\max}$ increase

efit, and the resulting automation reduces repetitive human effort and increases consistency. A framework is developed herein that should achieve these benefits, but it has not been applied in practice. How that may be done is illustrated in this section for the MANTA project. It directly constitutes an interesting direction for future work: how may the herein developed method be applied in practice and what benefits are realized, and what limitations present themselves?

For each of the original 29 technologies, several functions were assigned, which indicate what each technology was supposed to do. Among these functions severe redundancy was present. Therefore, a representative set of only three functions could be identified. Some of these may be broken up into different causes that attribute to them. These functions are shown in Table 7.5.

These functions correspond well to the k-factors with which the sensitivity study in subsection 7.2.3 was performed. However, the specific fuel consumption (SFC), whilst being the most influential, is not represented in these functions. Some of the technologies feature different amounts or types of actuators, and/or require more or less activation of control surfaces throughout a flight. Therefore, it seems logical that the power requirement, along with the time of activity, should be included in the evaluation. Effectively, the hydraulic (or electric) power required by the new technologies extract that power from the engines at the time instants that they are used; thus, temporarily increasing the SFC. This increases the fuel burn, and so this effect is not a function, but is

important to take into account, nonetheless.

Knowing these are the functions that the technologies have, and knowing the effects that they have, the baseline system can be modeled. At the very least, we need to model an aircraft consisting of a fuselage, wings, horizontal tail, vertical tail, engines and actuation system. The wings should have flaps, ailerons, winglets, slats and an actuation system as parts. The fuselage has an auxiliary air intake.

The case study in Chapter 6 already showed how the second-degree-of-freedom flap technology can be integrated into a mission analysis method for estimation of the fuel burn at aircraft level, including uncertainty. The flap deflection was coupled to the wing mass reduction (as an effect of MLA) through a bivariate probability distribution. Furthermore, a bivariate distribution between the flap deflection and drag coefficient was established. Together, these distributions would affect the input variables to the mission analysis, which, combined with a flap deflection strategy throughout the flight simulation, would result in a fuel burn impact. However, this approach does not enable us to distinguish between the two flap concepts proposed within MANTA: one with the actuation inside the track, and another with actuation outside of the track. The main differences are these:

- Inside track actuation removes the need for flap track fairings. Therefore, the wetted area and consequently zero-lift drag of the aircraft are reduced.
- Outside track actuation does not penetrate the rear spar. This saves weight.
- Outside track actuation does not intersect the fuel tank, thus does not require dry bays. A larger usable fuel volume results.
- Inside track actuation has lower power requirement (for the specific concept presented in MANTA).
- Outside track actuation has lower system weight.

When modeling these two alternative flap concepts, we require a few more components in the baseline system description: the wing rear spar, fuel tank and flap track fairings. Then, the inside track actuation concept is modeled with the following alterations the system: removal of the flap track fairing components, change of the spar geometry (and volume and mass), introduction of the actuation system with associated weights and power requirements, and reduction of the fuel tank volume. The outside track actuation concept only introduces its actuation system with associated weights and power requirements, and changes the flap track fairing geometry (and wetted area). The components only have to be modeled with a mass and power requirement at first, but could be fully analyzed with proper analysis methods when these are available and computationally tractable. The flexibility and extensibility of the presented method allows all of the components to be taken into account, without any further work required by the analyst.

All the other MANTA technologies have to be modeled analogously to the flap technologies as described in the foregoing. Now, the method requires the ability to analyze the quantities of interest using the knowledge graphs and transformation rules. For that, the analysis methods have to be specified. Fortunately, not too many methods are required for MANTA, although many of them involve complex physics. Table 7.6 shows

Table 7.6: Required analysis methods for holistic aircraft performance assessment within MANTA

Analysis method	Computes	Sensitive to
Engine performance	SFC	Flight condition, Drag, Power requirements
Aerodynamics	Lift and drag coefficients	Flight condition, geometry, control surface deflections
Aerodynamics delta	Change in lift and drag coefficients	Flight condition, surface imperfections, laminar flow
Actuation	Power requirements	Flight conditions, control surface deflection rate, hinge kinematics
Mass	Mass	Component hierarchy, geometry, material, loads
Longitudinal stability	Required lift coefficient, trim angle	Flight conditions, mass balance, geometry
Geometry	Volume, component mass	Geometry, material

the analysis methods that would be required to perform a medium-fidelity assessment of the MANTA technologies. While the required level of fidelity is not made explicit, the required sensitivities rule out empirical models in most cases. Therefore, mostly physics-based models, which rely on geometrical input are required.

The rest of the methodology proceeds similar to the test case in Chapter 6, section 6.4 for the mission analysis of an aircraft with the second degree-of-freedom flap. To summarize, the computation graph for each portfolio (or technology in this case) is created following the process from Chapter 5, subsection 5.1.3. Subsequently, the probability distributions and joint distributions for the input variables should be specified as explained in Chapter 6, subsection 6.2.3. Finally, following a Monte Carlo simulation of the computation graph, probabilistic inversion can be utilized to answer queries about the quantities of interest. In case of MANTA, a target on the fuel burn should be specified that reduces it by 3-5% over the entire range in the CDF and then the frequency shift of the technologies indicates which technologies are most likely to realize that benefit.

7.3.1. DISCUSSION OF METHOD APPLICATION

The majority of the workload induced by the proposed methodology resides in the construction of knowledge graphs for the various systems and technologies that have to be evaluated. In this section, the discussion is focused on the amount of detail that is required to construct these knowledge graphs, and what the drawbacks and benefits are.

HOW MUCH DETAIL?

Normally, the technologies have to be modeled in as much detail as needed. The baseline system should reflect that level of detail as a consequence. However, within MANTA,

combinations of technologies are not to be considered. Because of that, the compatibility matrix or technology enabling dependencies do not have to inferred, and we are only interested in analyzing the performance of the system with each technology implemented. Another reason neither the system nor the technologies need to be described in much detail is the lack of sophisticated analysis methods. Although some of the technologies represent small changes to the aircraft and influence aerodynamic properties very locally (e.g. the shock control bumps), most technologies present larger changes to either mass or aerodynamic efficiency, or both. In those cases, a low-fidelity analysis method suffices and the k-factor approach can be adopted.

There is only one objection to that approach. It reduces the method to a k-factor analysis, where the knowledge about the technologies is lost. So, while for the analysis of the technologies nothing more is required, one should still model the technologies in sufficient detail to capture form, function and behavior. This becomes especially evident when modeling the flap and winglet technologies from Table 7.3. In terms of the k-factors, the inside and outside flap track concepts are indistinguishable. The same holds for the morphing and conventional tab for the winglet.

One may now wonder why to go through the trouble of describing these technologies in detail in knowledge graphs. Why not just use the k-factor approach as that is what it boils down to? The argument against this viewpoint is that the k-factor approach loses information and does not allow for subsequent, more detailed investigations. Instead, when capturing the technologies properly, the methodology in Chapter 5 allows them to be analyzed using the k-factor approach, but also with more sophisticated analysis methods, such as finite-element or finite-volume methods. In fact, the MANTA consortium partners used such advanced analysis methods to provide estimates for the k-factors that could be fed into the holistic aircraft analysis tool. The proposed method would allow those advanced analysis methods to be incorporated directly into the overall aircraft evaluation framework, which reduces the amount of assumptions and simplifications. That, in turn, is expected to yield a more accurate quantification of the QoIs.

Apart from losing information about the technology, the k-factor approach has two more potential drawbacks. In some cases, no k-factors are available to capture a certain technology effect. For example, the Initiator (see Section 7.2.3) is unable to take into account power requirements of hydraulic actuators. Now, we know that down the line, those power requirements impact the specific fuel consumption (SFC). Thus, we could employ a k-factor towards the SFC, but it becomes difficult to estimate the effect an actuator has on SFC. Furthermore, if multiple technologies impact SFC in such a way, we have to establish dependencies between all of their impacts. That becomes tedious very quickly.

The other potential problem is that a technology effect cannot be effectively modeled by a k-factor. For example, the second degree-of-freedom flap is meant to have a flap schedule that makes it deploy continuously and differently during flight, to have the aircraft fly at its optimal lift-to-drag ratio at any combination of speed and altitude. Capturing this effect in a k-factor would average the impact of such a time-dependent process into a single number. Estimating that single number might prove difficult, and result in a loss of technology information.

BENEFITS OF EXTRA DETAIL

Consider the second degree-of-freedom flap concepts. Several geometrical changes are made to the wing rear spar and the flap track fairings, for example. For now, we simply gave the changes in mass, volume and wetted area as inputs, making the approach not much different from a k-factor approach. However, thanks to the knowledge graph approach, where all the components and changes are explicated, there are two important distinctions to the k-factor approach.

1. The system description can be indefinitely refined, and the flexibility of analysis methods would allow the mass, power and other hierarchical properties to be updated properly. Therefore, k-factors are effectively introduced on-the-fly where and when they are needed, rather than requiring some overarching one that captures them all. For higher fidelity analysis methods, such a refined specification of the mass distribution, for example, may actually be required. This ties in directly with the second distinction:

2. More advanced analysis methods can more easily be applied to the evaluation of the technologies. Instead of specifying the changes in mass, volume and area, a CAD program could calculate these based on geometry. Because a change in geometry is defined in the knowledge graph (and transformation rule), only an actual value needs to be given to that geometry attribute. A CAD program could be automatically linked to that attribute and the changes in derived properties is computed on-the-fly.

A NOTE ON ANALYSIS METHODS

In most technology evaluation approaches, new analysis methods are developed to be able to quantify new QoIs and represent the new technologies. This is still required with the proposed method. The only difference is that the present method requires the additional specification of the context graph for each analysis method. This graph ensures the analysis method can be automatically applied, and ensures the analysis method is easily reused in future studies. The modularity of the presented framework poses some requirements on newly developed analysis methods: they should be as independent as possible from other analysis methods and perform no overlapping computations. This is already often the case in existing software tools, so should not pose a problem.

A NOTE ON ROBUSTNESS, CONSISTENCY AND TRACEABILITY

As section 7.2 exposes, the technology evaluation process executed within MANTA deviated from a structured approach in that some technologies were discarded outside of the selection process, new technologies were introduced mid-way and some technologies got selected without a reason that reflects the project objective.

Applying the present method to MANTA would solve these issues. The formal representation of the technologies reduces ambiguity and misinterpretation. Several meetings have been held to discuss what actually was and was not meant and included with certain technologies. Formally representing the technologies would remove the need for such meetings, while making the knowledge explicit, such that it may be retrieved, viewed and modified at any time by any of the involved parties. This greatly enhances traceability of the evaluation and selection process.

At the same time, because a lot of the intensive work is automated, no technologies would have to be dropped before an actual evaluation has taken place. This makes the selection process more detailed and, therefore, increases the trustworthiness of the resulting decisions.

A central database could be constructed to serve as the authoritative source of truth, making all analyses consistent, as the same assumptions and inputs are used. Furthermore, the automatic construction of an analysis sequence would remove the need for a considerable amount of manual work. Additionally, unambiguous communication between various analysis methods would be baked-in to the the method. What happened in MANTA is that detailed calculations were performed to estimate values of k-factors. Then, these were communicated to another partner who performed the impact analysis at vehicle level. Thus, additional communication and data sharing was required to get the numbers from the one software to another. The automated analysis framework would remove this problem.

7.4. CONCLUSION

This chapter synthesizes the developments from the foregoing chapters into one overview, which is compared to the state-of-the-art. Furthermore, the MANTA project is discussed, and how it deviated from the state-of-the-art. An example is provided of how the methodology from this dissertation can be applied to the MANTA project. From this, the required level of detail for the knowledge graphs is discussed. Finally, decision quality is discussed, as it is relevant to the methodology, but has not been treated previously.

A comparison is made between the state-of-the-art method and the herein developed method towards technology evaluation and selection. Three key differences between these methods have been identified. First, the proposed method features significantly more automation than the state-of-the-art approach. Second, in the proposed approach, all data is either stored in a knowledge graph, or is numeric. This makes it machine-interpretable and removes all the disadvantages that textual data has in the state-of-the-art method. Third, the approach is more flexible and extensible than the state-of-the-art, because the dependency and computation graphs are generated on-the-fly and uniquely for each technology portfolio. This also removes the dependency of the impact factors on the analysis tool.

The graph-theoretic description of the technologies allows users to gradually increase the level of detail. In turn, the analysis methods employed to analyze the technologies can be of higher fidelity. Thus, as a project such as MANTA matures, the technology assessment becomes more detailed and epistemic uncertainty is reduced, ideally leading to a better trade-off.

In the following, and final, chapter, the developments from this dissertation and the observations from the present chapter are summarized and the most important and relevant conclusions are presented. In addition, recommendations for improvements and future research are provided.

8

CONCLUSION

This dissertation set out to develop a structured, repeatable and traceable method for the evaluation and selection of novel technologies in complex engineering systems. Three components of the methodology were identified: technology representation and portfolio generation, technology (portfolio) evaluation and technology (portfolio) selection. Before answering the main research question, this chapter answers the four sub-questions into which the main research question was divided. Subsequently, the main research question is answered as well. Based on the answers, recommendations for improving on and extending the work are provided thereafter. Finally, an outlook to future research and possible extensions of the framework is given.

8.1. CONCLUSIONS

Let us consider each of the four research sub-questions, and treat them in detail.

How to represent aircraft systems and technologies consistently and robustly, allowing for accurate analysis during conceptual and preliminary design? A formal knowledge representation is achieved through an ontology that aims to capture engineering systems as realistically as possible. The basis of this ontology is formed by a combination of three upper ontologies: Basic Formal Ontology (BFO), Information Artifact Ontology (IAO) and Physics-based Simulation Ontology (PSO). With this ontology at the foundation, engineering systems are captured in knowledge graphs. Knowledge graphs are flexible data structures and, therefore, support the flexibility, extensibility and applicability of the method.

Technology is defined in this dissertation as the application of knowledge to a system in order to alter the system's form or behavior to satisfy certain requirements. The key insight following from this definition is to model aircraft systems as knowledge graphs and technologies as graph transformation rules. Because the knowledge graphs describe systems, the transformation describes the change to that system.

How to define dependencies between technologies, and how to characterize these dependencies based on the physical behavior of the technologies? The dependencies between technologies take two forms in this dissertation. First, when combining technologies into technology portfolios, technology incompatibility and technology enablers are considered. Secondly, for the analysis of the technology (portfolio) performance, the dependencies between the variables that model the technologies are captured. Technology incompatibility and technology enablers are inferred through dependencies between the graph transformation rules that represent the technologies. A first-order-logic-based rule set is created that enables inference of dependent variables. It works by stating generalized causal influences between properties of physical entities. A practitioner can view which variables are considered as dependent by the rules, and specify joint dependency structures for those.

How to analyze (novel) aircraft technologies in a consistent, reliable and robust manner, such that their (combined) effects are characterized, with uncertain input metrics/parameters? The key insight to answer this question is that any framework for

technology evaluation has to be modular and extensible. Modularity is achieved by specifying separate analysis models for different aspects of a system. A knowledge graph specifies the pattern in a system that the analysis model applies to. A parameterization algorithm is introduced that matches the patterns of the available analysis models in the system. For each such a match, the required parameters are then introduced to the system graph. This approach allows the system description to be completely independent from the analysis models. Successively, a dependency graph is generated. The dependency graph contains all possible computation directions between the variables in the system. From this graph, a computation graph is distilled given the quantities of interest and known input variables. The computation graph contains one specific possible computation sequence to compute the QoI from the inputs. When the inputs are uncertain, a Monte Carlo Simulation is performed to propagate the uncertainty to the QoIs.

How should a ranking of a set of technologies or technology portfolios be obtained for multiple, conflicting, uncertain quantities of interest? The probabilistic inversion (PI) technique is proposed to solve this problem. The technology selection problem is framed as an inverse problem, which is solved with PI through sample re-weighting. A desired distribution on the quantities of interest is specified and PI redistributes the technology portfolios to come closest to that desired goal. The shift in frequency of the technology portfolios offers a ranking of them. Because PI is implemented as a sample-based approach, it works with any analysis method, as long as it computationally tractable to generate thousands of samples. Additionally, it can solve a variety of other queries that are relevant to technology evaluation.

Now, the main research question **"How is an aircraft technology portfolio selected in a robust, consistent and traceable manner to minimize an objective function, given uncertain technology metrics, while including technology dependencies?"** is considered. While there is no single correct answer to this question, the method developed within this dissertation is one possibility. Thus, the technologies, aircraft system and analysis methods have to be represented with knowledge graphs adhering to a physics-based ontology. A set of algorithms ensures a robust, consistent and traceable process to evaluate the objective function. Probabilistic inversion is employed to obtain a ranking of the technology portfolios, based on requirements on the objective function. This approach offers several benefits over the state-of-the-art approaches towards technology evaluation and selection:

- The formal knowledge representation removes ambiguity and enables a single source of truth.
- Automatic portfolio generation, including technology compatibility and enablers avoids the repetitive, labor-intensive and error-prone task of creating a technology compatibility matrix manually. This improves the consistency of the generated portfolios.
- Analysis methods are represented with context graphs, allowing them to be automatically applied to any given system. Furthermore, this explicates what the analysis method models and quantifies, while supporting reuse and modularity.

- Inverse technology design and selection queries no longer have to be computed using an iterative approach, but can be answered directly with probabilistic inversion.

8.2. RECOMMENDATIONS

The work done in this dissertation is only a start and several points for improvements and further research are left open. The first point is the ontology for engineering systems. The present ontology does not define common concepts, such as gas turbines or wings. For the approach to be more easily used by industry experts, an ontology that defines the concepts in their domain of discourse allows them to more easily create the required knowledge graphs. Currently, each institution that employs a form of model-based systems engineering may have their own standard for describing systems. However, no generalized framework can be built around that, without requiring those institutions to fill a lot of gaps. All inference rules would have to be hand-crafted, similarity is different for each of them and when the standard is not well defined, automation becomes nearly impossible.

The most important considerations to arrive at a suitable ontology are how to represent physical behavior and how to deal with granularity [1]. These remain questions yet to be answered. Nonetheless, the answer to these questions involves a fixed basis of atomic universals that enable the description of any higher-level concept (e.g. an engineering system's behavior). Physical behavior may be classified using process profiles, that reflect the physical laws as we know them: Newton's law, the laws of thermodynamics, Navier-Stokes equations, Maxwell's equations, etc. The teleological view that assigns functions to components and systems may be included in the ontology through a functional basis, of which several have been developed in literature. Unfortunately, many such developments are debatable and open to interpretation — something which should be avoided; indeed, the ontology has precisely the goal to remove ambiguity by providing sound definitions. Whether it is possible to arrive at an exhaustive basis of physical behaviors is unknown. However, it is assumed to be possible, and that thought is shared by Forbus [77]:

> Qualitative Process Theory concerns the structure of qualitative dynamics. We can view it as specifying a language in which certain commonsense physical models can be written. Can this language be extended to form a full language of behavior for physical systems? Although I have not yet done so, I will argue that the answer is yes, and that several advantages would result from the extension.

The graph matching processes that occur frequently throughout the proposed methodology — to find patterns in host graphs — are based on (sub)graph isomorphism. Instead, they should be based on semantic graph matching, which is a form of inexact graph matching techniques. The idea is to match graphs based on semantic equivalence, rather than structure and corresponding labels. Inexact graph matching is still an open area of research, so it is unlikely that a satisfactory solution already exists for complex

[1] Granularity refers to the different levels of decomposition of a system, see section 3.7

knowledge graphs such as the ones encountered in this dissertation. Nonetheless, the graph edit cost approach might already work sufficiently well provided the edit costs are learned based on a database of (synthetically generated) valid graphs. The downside of that approach is that the graphs in that database have to be labeled by a human expert.

In several places, first-order logic rules are used to infer dependencies between technologies or variables. Generally, these rules are crafted to represent physical causality, such that the system can reason qualitatively about physics. That suggests that a learning algorithm could infer these rules by itself, given observations of some physical domain. Research into that has already been conducted, e.g. schema networks [150, 151], interaction networks [152] or relational deep reinforcement learning [153].

The final recommendation is to create some mechanism to support the generation of knowledge graphs. This could either be the integration of a background process into existing CAD software, that automatically writes a knowledge graph from the drawing a user makes. Otherwise, a question-answer system could guide the practitioner in constructing a knowledge graph by firing a set of questions based on the responses that explicate the modeler's intent. Finally, suggestions to automatically complete knowledge graphs can be learned from previous ones, much like how Google Search completes a query.

8.3. OUTLOOK

Provided the recommendations are implemented, the methodology presented in this dissertation can be extended to achieve more complex and interesting tasks. For example, the method provides a notion of similarity between technologies and systems. Then, that similarity extends to the variables describing these entities. Given similarity between variables, values for them can be extrapolated from previously observed values for similar variables. Thus, the computer system can take over expert judgment. A value in this case would be a probability distribution. The degree of similarity additionally introduces some uncertainty.

The methodology can also be combined with experimental testing to reduce epistemic uncertainty. Suppose, for example, a set of technologies is evaluated and no selection can be made, because the resulting probability distributions overlap, ruling out any form of stochastic dominance. In many cases, this situation can be improved by reducing the (epistemic) uncertainty on the input variables, which typically describe aspects of the technologies. With the PI technique, those variables that contribute most to the uncertainty can be singled out. A dedicated test campaign can then be conducted to estimate the distributions for these variables with less uncertainty. Then, the technology evaluation can be performed again, likely resulting in a clearer ranking than before.

ACKNOWLEDGEMENTS

I want to thank my promotor Leo Veldhuis and supervisor Roelof Vos for their support and feedback throughout my journey as a Master student and PhD candidate. Furthermore, I'd like to thank Kristian Amadori, Christopher Jouannet and Dorota Kurowicka for their help and insight. Likewise, I'd like to thank all committee members for taking the time to review my work and your questions during the defense. All colleagues and friends from the FPP department also receive my sincere gratitude for their support, jokes and good times.

Of course, life would be boring without all my family and friends, so here's a big thank you to all of you. Especially mom and dad, for always being there for me, and Vincent and Sandra, for being sweet, and Dixie, for always making me smile.

Finally, I am very grateful to you, Julieta, for being a fabulous unicorn, and, for the beautiful cover of this thesis, of course.

REFERENCES

[1] W. Edwards, *The theory of decision making.* Psychological bulletin **51**:**4**, p. 380 (1954).

[2] P. Slovic, S. Lichtenstein, and B. Fischhoff, *Decision making,* (Wiley, 1988).

[3] M. Peterson, *An Introduction to Decision Theory*, Cambridge Introductions to Philosophy (Cambridge University Press, 2009).

[4] J. W. Pratt, H. Raiffa, R. Schlaifer, et al., *Introduction to statistical decision theory* (MIT press, 1995).

[5] A. Hanea and G. Nane, *Calibrating experts' probabilistic assessments for improved probabilistic predictions,* Safety Science **118**:pp. 763–771 (2019).

[6] R. Cooke, M. Mendel, and W. Thijs, *Calibration and information in expert resolution; a classical approach,* Automatica **24**:**1**, pp. 87–93 (1988).

[7] S. Monti and G. Carenini, *Dealing with the expert inconsistency in probability elicitation,* IEEE Transactions on Knowledge and Data Engineering **12**:**4**, pp. 499–508 (2000).

[8] H. Jimenez and D. N. Mavris, *Characterization of Technology Integration Based on Technology Readiness Levels,* Journal of Aircraft **51**:**1**, pp. 291–302 (2014).

[9] M. R. Kirby and D. N. Mavris, *Forecasting Technology Uncertainty in Preliminary Aircraft Design,* in *World Aviation Conference* (AIAA, San Francisco, 1999) p. 14.

[10] M. A. Cartagena, J. E. Rosario, and D. N. Mavris, *A Method for Technology Identification , Evaluation , and Selection of Aircraft Propulsion Systems,* in *36th AIAA/ASME/SAE/ASEE joint Propulsion Conference*, July (AIAA, Huntsville, 2000).

[11] B. Roth, B. German, D. Mavris, and N. Macsotai, *Adaptive selection of engine technology solution sets from a large combinatorial space,* in *37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, July (AIAA, Salt Lake City, 2001).

[12] K. N. Gatian and D. N. Mavris, *Enabling Technology Portfolio Selection through Quantitative Uncertainty Analysis,* in *15th AIAA Aviation Technology, Integration and Operations Conference*, June (AIAA, Dallas, 2015) pp. 1–28.

[13] D. S. Soban and D. N. Mavris, *Assessing the Impact of Technology on Aircraft Systems Using Technology Impact Forecasting,* Journal of Aircraft **50**:**5**, pp. 1380–1393 (2013).

[14] I. Chakraborty and D. N. Mavris, *Assessing Impact of Epistemic and Technological Uncertainty on Aircraft Subsystem Architectures,* Journal of Aircraft **54**:**4**, pp. 1388–1406 (2017).

[15] I. Chakraborty and D. N. Mavris, *Integrated Assessment of Aircraft and Novel Subsystem Architectures in Early Design,* Journal of Aircraft **54**:**4**, pp. 1268–1282 (2017).

[16] Z. Lu, E.-S. Yang, D. DeLaurentis, and D. Mavris, *Formulation and test of an object-oriented approach to aircraft sizing,* in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September (AIAA, Albany, 2004) pp. 1–14.

[17] S. Dufresne, C. Johnson, and D. N. Mavris, *Variable Fidelity Conceptual Design Environment for Revolutionary Unmanned Aerial Vehicles,* Journal of Aircraft **45**:**4**, pp. 1405–1418 (2008).

[18] L. U. Iqbal, *Utilization of Geometry Definition Tools in Aircraft Design: Need for Paradigm Shift,* in *55th AIAA Aerospace Sciences Meeting*, January (AIAA, Grapevine, 2017) pp. 1–12.

[19] C. Pornet, C. Gologan, P. C. Vratny, A. Seitz, O. Schmitz, A. T. Isikveren, and M. Hornung, *Methodology for Sizing and Performance Assessment of Hybrid Energy Aircraft,* Journal of Aircraft **52**:**1**, pp. 1–12 (2014).

[20] G. Bucsan, *Generalized Methodology for Sizing Unconventional Propulsion and Configuration Aircraft,* in *55th AIAA Aerospace Sciences Meeting*, January (AIAA, Grapevine, 2017) pp. 1–32.

[21] T. Nam, *A generalized sizing method for revolutionary concepts under probabilistic design constraints*, Ph.D. thesis, Georgia Institute of Technology (2007).

[22] I. Chakraborty, *Subsystem Architecture Sizing and Analysis for Aircraft Conceptual Design Subsystem Architecture Sizing and Analysis*, Ph.D. thesis, Georgia Institute of Technology (2015).

[23] S. Dufresne, *A hierarchical modeling methodology for the definition and selection of requirements*, Ph.D. thesis, Georgia Institute of Technology (2008).

[24] M. R. Kirby, T. Nam, H. Ran, S. Dufresne, G. Burdette, W. Sung, and D. N. Mavris, *Advanced Vehicles Modeling for the Next Generation Air Transportation System (NextGen Vehicle Integration NRA),* in *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, September (AIAA, Head, 2009).

[25] P. Heinemann, P. Panagiotou, P. Vratny, S. Kaiser, M. Hornung, and K. Yakinthos, *Advanced Tube and Wing Aircraft for Year 2050 Timeframe,* in *55th AIAA Aerospace Sciences Meeting*, January (AIAA, Grapevine, 2017).

[26] L. Marx, *Technology: The emergence of a hazardous concept,* Social Research **64**:**3**, pp. 965–988 (1997).

[27] C. Bush, *Taking hold of technology: Topic guide for 1981-1983,* American Association of University Women (1981).

[28] R. S. Merrill, *The study of technology,* International Encyclopedia of the Social Sciences **15**:pp. 576–589 (1968).

[29] J. K. Feibleman, *Pure science, applied science, technology, engineering: an attempt at definitions,* Technology and Culture **2**:**4**, pp. 305–317 (1961).

[30] C. Mitcham and E. Schatzberg, *Defining technology and the engineering sciences,* in *Philosophy of technology and engineering sciences* (Elsevier, 2009) pp. 27–63.

[31] A. Mancini and R. Vos, *The effect of maneuver load alleviation strategies on aircraft performance indicators,* in *AIAA Aviation 2019 Forum*, June 17-21 (Dallas, Texas, 2019).

[32] K. Amadori, E. Bäckström, and C. Jouannet, *Selection of Future Technologies during Aircraft Conceptual Design,* in *55th AIAA Aerospace Sciences Meeting*, January (AIAA, 2017) pp. 1–11.

[33] S. T. Piantadosi, H. Tily, and E. Gibson, *The communicative function of ambiguity in language,* Cognition **122**:**3**, pp. 280–291 (2012).

[34] P. J. Couturier, C. Tribes, and J.-Y. Trépanier, *Framework for the Robust Design Optimization of an Airframe and its Engines,* Journal of Aerospace Engineering **28**:**2**, pp. 1–10 (2015).

[35] P. Hantos, *Systems Engineering Perspectives on Technology Readiness Assessments in Software-Intensive System Development,* Journal of Aircraft **48**:**3**, pp. 738–748 (2011).

[36] F. Burgaud, J.-G. Durand, and D. N. Mavris, *A Decision-Support Methodology to Make Enterprise-Level Risk/Value Trade-Offs,* in *19th AIAA Non-Deterministic Approaches Conference*, January (AIAA, Grapevine, 2017).

[37] A. Utturwar, S. Rallabhandi, D. DeLaurentis, and D. Mavris, *A bi-level optimization approach for technology selection (for aircraft design),* in *9th AIAA/ISSMO Symposium and Exhibit on Multidisciplinary Analysis and Optimization*, September (AIAA, Atlanta, 2002).

[38] C. B. Patel, M. R. Kirby, and D. N. Mavris, *Niched-Pareto Genetic Algorithm for Aircraft Technology Selection Process,* in *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September (AIAA, Portsmouth, 2006) pp. 1–11.

[39] H. Jimenez and D. Mavris, *Pareto-optimal aircraft technology study for environmental benefits with multi-objective optimization,* Journal of Aircraft **54**:**5**, pp. 1860–1876 (2017).

[40] S. Van Haver and R. Vos, *A Practical Method for Uncertainty Analysis in the Aircraft Conceptual Design Phase,* in *53rd AIAA Aerospace Sciences Meeting*, January (AIAA, Kissimmee, 2015).

[41] D. E. Veley, M. Blair, J. V. Zweber, A. Force, A. V. Directorate, S. Division, and W.-p. A. F. Base, *Aerospace Technology Assessment System,* in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September (AIAA, St. Lious, 1998).

[42] R. Raffaeli and M. Germani, *Knowledge-based approach to flexible part design,* Journal of Engineering Design **21**:**1**, pp. 7–29 (2010).

[43] A. Olewnik and K. Lewis, *A decision support framework for flexible system design,* Journal of Engineering Design **17**:**1**, pp. 75–97 (2006).

[44] P. F. Albuquerque, P. V. Gamboa, and M. A. Silvestre, *Mission-based multidisciplinary aircraft design optimization methodology tailored for adaptive technologies,* Journal of Aircraft **55**:**2**, pp. 755–770 (2018).

[45] A. Antonakis, T. Nikolaidis, and P. Pilidis, *Effects of propulsion system operation on military aircraft survivability,* Journal of Aircraft **56**:**6**, pp. 2131–2143 (2019).

[46] R. de Vries, M. Brown, and R. Vos, *Preliminary sizing method for hybrid-electric distributed-propulsion aircraft,* Journal of Aircraft **Articles in Advance**:**0**, pp. 1–17 (2019).

[47] K. N. Gatian and D. N. Mavris, *Planning technology development experimentation through quantitative uncertainty analysis,* in *54th AIAA Aerospace Sciences Meeting*, January 4-8 (San Diego, California, 2016).

[48] C. Jouannet, K. Amadori, E. Bäckström, and D. Bianchi, *Uncertainty management in technologies prioritization for future aircraft program,* in *AIAA Scitech 2020 Forum*, 6-10 January (Orlando, FL, 2020).

[49] A. V. D. Laan and M. V. Tooren, *Incorporating cost analysis in a multi-disciplinary design environment for aircraft movables,* Journal of Engineering Design **19**:**2**, pp. 131–144 (2008).

[50] C. Jouannet and P. Krus, *Direct simulation-based optimization for aircraft conceptual design,* in *7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences,17th LTA Systems Tech Conf; followed by 2nd TEOS Forum*, 18-20 September (Belfast, Northern Ireland, 2007) https://arc.aiaa.org/doi/pdf/10.2514/6.2007-7827 .

[51] M. V. Raghu Chaitanya, *Knowledge Based Integrated Multidisciplinary Aircraft Conceptual Design*, Ph.D. thesis, Linköping University (2014).

[52] W. Anemaat, B. Kaushik, R. Hale, and N. Ramabadran, *Aaaraven: Knowledge-based aircraft conceptual and preliminary design,* in *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 23-26 April (Honolulu, Hawaii, 2007).

[53] M. A. Y. Ramakers, *Accelerating Aircraft Design Using Automated Process Generation*, M.sc. thesis, Delft University of Technology (2015).

[54] I. Van Gent, P. D. Ciampa, B. Aigner, J. Jepsen, G. La Rocca, and E. J. Schut, *Knowledge Architecture Supporting Collaborative MDO in the AGILE Paradigm,* in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, June 5-9 (Denver, Colorado, 2017).

[55] I. Van Gent, R. Lombardi, G. La Rocca, and R. d'Ippolito, *A Fully Automated Chain from MDAO Problem Formulation to Workflow Execution,* in *EUROGEN 2017*, September 13-15 (Madrid, Spain, 2017).

[56] D. J. Pate, J. Gray, and B. J. German, *A Graph Theoretic Approach to Problem Formulation for Multidisciplinary Design Analysis and Optimization,* Structural Multidisciplinary Optimization **49**:**5**, pp. 743–760 (2014).

[57] P. Borst, A. Pos, J. Top, and J. Akkermans, *Physical systems ontology,* in *Working Papers European Conference on Artificial Intelligence ECAI 1994 Workshop on Implemented Ontologies* (Amsterdam, 1994) pp. 4–80.

[58] P. Borst, J. Akkermans, A. Pos, and J. Top, *The physsys ontology for physical systems,* in *Working Papers of the Ninth International Workshop on Qualitative Reasoning* (University of Amsterdam, Amsterdam) pp. 11–21.

[59] T. Gruber and G. Olsen, *An ontology for engineering mathematics,* in *Principles of Knowledge Representation and Reasoning* (San Mateo, CA, 1994) pp. 258–269.

[60] P. Borst, H. Akkermans, and J. Top, *Engineering ontologies,* International Journal of Human-Computer Studies **46**:**2-3**, pp. 365–406 (1997).

[61] H. Cheong and A. Butscher, *Physics-based simulation ontology: an ontology to support modelling and reuse of data for physics-based simulation,* Journal of Engineering Design **30**:**10-12**, pp. 655–687 (2019).

[62] M. Štorga, M. M. Andreasen, and D. Marjanović, *The design ontology: foundation for the design knowledge exchange and management,* Journal of Engineering Design **21**:**4**, pp. 427–454 (2010).

[63] Y. Kitamura and R. Mizoguchi, *Ontology-based systematization of functional knowledge,* Journal of Engineering design **15**:**4**, pp. 327–351 (2004).

[64] J. Hirtz, R. B. Stone, D. A. Mcadams, S. Szykman, and K. L. Wood, *A functional basis for engineering design : Reconciling and evolving previous efforts,* Research in Engineering Design **13**:pp. 65–82 (2002).

[65] D. M. Judt and C. Lawson, *Development of an Automated Aircraft Subsystem Architecture Generation and Analysis Tool,* Engineering Computations **33**:**5**, pp. 1327–1352 (2016).

[66] L. Yuan, Y. Liu, Z. Sun, Y. Cao, and A. Qamar, *A hybrid approach for the automation of functional decomposition in conceptual design,* Journal of Engineering Design **27**:**4-6**, pp. 333–360 (2016).

[67] C. Sen, J. D. Summers, and G. M. Mocko, *A protocol to formalise function verbs to support conservation-based model checking,* Journal of Engineering Design **22**:**11-12**, pp. 765–788 (2011).

[68] C. Sen, J. D. Summers, and G. M. Mocko, *Physics-Based Reasoning in Conceptual Design Using a Formal Representation of Function Structure Graphs,* Journal of Computing and Information Science in Engineering **13**:pp. 1–12 (2013).

[69] T. Wilschut, L. Etman, J. Rooda, and J. A. Vogel, *Generation of a function-component-parameter multi-domain matrix from structured textual function specifications,* Research in Engineering Design pp. 1–16 (February 2018).

[70] D. M. Judt and C. P. Lawson, *Application of an automated aircraft architecture generation and analysis tool to unmanned aerial vehicle subsystem design,* Journal of Aerospace Engineering **229**:**9**, pp. 1690–1708 (2015).

[71] M. D. Guenov, A. Molina-Cristobal, V. Voloshin, A. Riaz, and A. S. J. Van Heerden, *Aircraft Systems Architecting – a Functional - Logical Domain Perspective,* in *16th AIAA Aviation Technology, Integration, and Operations Conference*, June (AIAA, Washington, 2016).

[72] J. Castet, M. Rozek, M. Ingham, N. Rouquette, and S. Chung, *Ontology and Modeling Patterns for State-Based Behavior Representation,* in *2018 AIAA Aerospace Sciences Meeting*, January (AIAA, Kissimmee, Florida, 2015).

[73] J. Kaderka, M. Rozek, J. Arballo, D. Wagner, and M. Ingham, *The Behavior, Constraint and Scenario (BeCoS) Tool: A Web-Based Software Application for Modeling Behaviors and Scenarios,* in *2018 AIAA Aerospace Sciences Meeting*, January (AIAA, Kissimmee, Florida, 2018).

[74] C. Sen, J. D. Summers, and G. M. Mocko, *A Formal Representation of Function Structure Graphs for Physics-Based Reasoning,* Journal of Computing and Information Science in Engineering **13**:pp. 1–13 (2013).

[75] J. De Kleer and J. S. Brown, *A qualitative physics based on confluences,* Artificial Intelligence **24**:**1-3**, pp. 7–83 (1984).

[76] J. De Kleer, *How circuits work,* Artificial Intelligence **24**:**1-3**, pp. 205–280 (1984).

[77] K. D. Forbus, *Qualitative process theory,* Artificial Intelligence **24**:pp. 85–168 (1984).

[78] R. Rosenberg and D. Karnopp, *Introduction to physical system dynamics* (McGraw-Hill, Inc., 1983).

[79] D. Karnopp, D. Margolis, and R. Rosenberg, *System dynamics: A unified approach*, 2nd ed. (John Wiley & Sons, 1990).

[80] H. Bunke, *On a relation between graph edit distance and maximum common subgraph,* Pattern Recognition Letters **18**:**8**, pp. 689–694 (1997).

[81] B. Helms, K. Shea, and F. Hoisl, *A Framework for Computational Design Synthesis based on Graph-Grammars and Function-Behavior-Structure,* in *International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, August-September (ASME, San Diego, 2009).

[82] M. Irani and S. Rudolph, *Design Grammars for Conceptual Designs of Space Stations,* in *54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics and the International Institute of Space Law*, September-October (AIAA, Bremen, 2003).

[83] V. Stavrev, *A Shape Grammar for Space Architecture - Part II. 3D Graph Grammar - An Introduction,* in *41st International Conference on Environmental Systems*, July (AIAA, Portland, 2011).

[84] H. Ehrig, C. Ermel, U. Golas, and F. Hermann, *Graph and Model Transformation*, 1st ed. (Springer, 2015) p. 472.

[85] K. Amadori, E. Bäckström, and C. Jouannet, *Future Technologies Prioritization for Aircraft Conceptual Design,* in *2018 AIAA Aerospace Sciences Meeting*, January (AIAA, Kissimmee, Florida, 2018).

[86] I. Chakraborty and D. N. Mavris, *Assessing impact of epistemic and technological uncertainty on aircraft subsystem architectures,* Journal of Aircraft **54**:**4**, pp. 1388–1406 (2017).

[87] K. N. Gatian and D. N. Mavris, *Facilitating Technology Development Progression through Quantitative Uncertainty Assessments,* in *AIAA Aviation Forum*, June (AIAA, Atlanta, 2014).

[88] T. Zaidi, H. Jimenez, and D. Mavris, *Copulas Theory for Probabilistic Assessment: Overview with Application to Airplane Performance Analysis,* Journal of Aircraft **52**:**6**, pp. 1802–1820 (2015).

[89] W. Yao, X. Chen, W. Luo, M. Van Tooren, and J. Guo, *Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles,* Progress in Aerospace Sciences **47**:**6**, pp. 450–479 (2011).

[90] W. L. Oberkampf, J. C. Helton, and K. Sentz, *Mathematical representation of uncertainty,* in *Non-Deterministic Approaches Forum*, April (AIAA, Seattle, 2001) pp. 1–23.

[91] W. L. Oberkampf, S. M. DeLand, B. M. Rutherford, K. V. Diegert, and K. F. Alvin, *Error and uncertainty in modeling and simulation,* Reliability Engineering & System Safety **75**:**3**, pp. 333–357 (2002).

[92] W. L. Oberkampf, J. C. Helton, C. A. Joslyn, S. F. Wojtkiewicz, and S. Ferson, *Challenge problems: Uncertainty in system response given uncertain parameters,* Reliability Engineering and System Safety **85**:**1-3**, pp. 11–19 (2004).

[93] G. J. Klir and R. M. Smith, *On measuring uncertainty and uncertainty-based information: Recent developments,* Annals of Mathematics and Artificial Intelligence **32**:**1-4**, pp. 5–33 (2001).

[94] G. Choquet, *Theory of Capacitites,* Annales de l'institute Fourier **5**:pp. 11–14 (1954).

[95] H. Agarwal, J. E. Renaud, E. L. Preston, and D. Padmanabhan, *Uncertainty quantification using evidence theory in multidisciplinary design optimization,* Reliability Engineering and System Safety **85**:**1-3**, pp. 281–294 (2004).

[96] O. Schwabe, E. Shehab, and J. Erkoyuncu, *Uncertainty quantification metrics for whole product life cycle cost estimates in aerospace innovation,* Progress in Aerospace Sciences **77**:pp. 1–24 (2015).

[97] J. S. Wilson, *Uncertainty Quantification With Mitigation Actions for Aircraft Conceptual Design*, Ph.D. thesis, Georgia Institute of Technology (2015).

[98] P. Hajela and S. Vittal, *Optimal Design in the Presence of Modeling Uncertainties,* Journal of Aerospace Engineering **19**:**4**, pp. 204–216 (2006).

[99] V. Pandey, *Uncertainty Modeling using Mixture Distributions for Decision-Based Design,* in *19th AIAA Non-Deterministic Approaches Conference*, January (AIAA, Grapevine, 2017) pp. 1–8.

[100] B. Olshausen, *Bayesian probability theory,* The Redwood Center for Theoretical Neuroscience pp. 1–6 (2004).

[101] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. (Pearson Education, 2008) p. 1132.

[102] M. Beynon and B. Curry, *The Dempster-Shafer theory of evidence: An alternative approach to multicriteria decision modelling,* Omega **28**:pp. 37–50 (2000).

[103] P. Soundappan, E. Nikolaidis, R. T. Haftka, R. Grandhi, and R. Canfield, *Comparison of evidence theory and Bayesian theory for uncertainty modeling,* Reliability Engineering and System Safety **85**:**1-3**, pp. 295–311 (2004).

[104] G. Elidan, *Copula bayesian networks,* in *Advances in neural information processing systems* (2010) pp. 559–567.

[105] A. Sklar, *Random variables, joint distribution functions, and copulas,* Kybernetika **9**:**6**, pp. 449–460 (1973).

[106] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques* (MIT press, 2009).

[107] T. A. Zang, M. J. Hemsch, M. W. Hilburger, S. P. Kenny, J. M. Luckring, P. Maghami, S. L. Padula, and W. J. Stroud, *Needs and Opportunities for Uncertainty-Based Multidisciplinary Design Methods for Aerospace Vehicles*, Tech. Rep. July (NASA, Hampton, 2002).

[108] C. J. Roy and W. L. Oberkampf, *A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing,* Computer Methods in Applied Mechanics and Engineering **200**:**25-28**, pp. 2131–2144 (2011).

[109] J. J. Alonso, M. S. Eldred, P. Constantine, K. Duraisamy, C. Farhat, G. Iaccarino, and J. Jakeman, *Scalable Environment for Quantification of Uncertainty and Optimization in Industrial Applications (SEQUOIA),* in *19th AIAA Non-Deterministic Approaches Conference*, January (AIAA, 2017) pp. 1–19.

[110] B. Peherstorfer, K. Willcox, and M. Gunzburger, *ACDL TR16-1*, Tech. Rep. (2016).

[111] X. Hu, X. Chen, G. T. Parks, and W. Yao, *Review of improved Monte Carlo methods in uncertainty-based design optimization for aerospace vehicles,* Progress in Aerospace Sciences **86**:pp. 20–27 (2016).

[112] S. Wu and E. Livne, *Probabilistic Aeroservoelastic Reliability Assessment Considering Control System Component Uncertainty,* AIAA Journal **54**:**8**, pp. 2507–2520 (2016).

[113] W. Chen, R. Jin, and A. Sudjianto, *Analytical Variance-Based Global Sensitivity Analysis in Simluation-Based Design under Uncertainty,* Journal of Mechanical Design **127**:**September**, pp. 875–886 (2005).

[114] A. Elham and M. J. L. Van Tooren, *Aerodynamic Shape Optimization Using Symbolic Sensitivity Analysis,* in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, January (AIAA, Grapevine, 2017).

[115] A. Griewank, *On Automatic Differentiation*, Tech. Rep. November (Mathematics and Computer Science Division, Argonne, 1988).

[116] M. Bartholomew-Biggs, S. Brown, B. Christianson, and L. Dixon, *Automatic differentiation of algorithms,* Journal of Computational and Applied Mathematics **124**:**1-2**, pp. 171–190 (2000), arXiv:arXiv:1011.1669v3 .

[117] T. Beck, *Automatic differentiation of iterative processes,* Journal of Computational and Applied Mathematics **50**:**1-3**, pp. 109–118 (1994).

[118] T. Beck and H. Fischer, *The if-problem in automatic differentiation,* Journal of Computational and Applied Mathematics **50**:**1-3**, pp. 119–131 (1994).

[119] S. Chinchalkar, *The Application of Automatic Differentiation to Problems in Engineering Analysis,* Computer Methods in Applied Mechanics and Engineering **118**:pp. 197–207 (1994).

[120] J. Su and J. E. Renaud, *Automatic Differentiation in Robust Optimization,* AIAA Journal **35**:**6**, pp. 1072–1079 (1997).

[121] S. Bae, N. H. Kim, and C. Park, *Confidence Interval of Bayesian Network and Global Sensitivity Analysis,* in *19th AIAA Non-Deterministic Approaches Conference*, January (AIAA, Grapevine, 2017) pp. 1–16.

[122] Z. Jiang, W. Chen, and B. J. German, *Multidiciplinary Statistical Sensitivity Analysis Considering both Aleatory and Epistemic Uncertainties,* AIAA Journal **54**:**4**, pp. 1326–1338 (2016).

[123] A. Saltelli, S. Tarantola, and K. Chan, *A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output,* Technometrics **41**:**1**, pp. 39–56 (1999).

[124] I. M. Sobol, *Global Sensitivity Indices for Nonlinear Mathematical Models,* Mathematics and Computers in Simulation **55**:pp. 271–280 (2001).

[125] E. C. Decarlo and S. Mahadevan, *Efficient Global Sensitivity Analysis for Time - Dependent , Multidisciplinary Models,* in *19th AIAA Non-Deterministic Approaches Conference*, January (AIAA, Grapevine, 2017) pp. 1–10.

[126] H. Liu, W. Chen, and A. Sudjianto, *Probabilistic Sensitivity Analysis Methods for Design Under Uncertainty,* in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September (AIAA, Albany, 2004) pp. 1–15.

[127] M. M. Opgenoord and K. E. Willcox, *Sensitivity analysis methods for uncertainty budgeting in system design,* AIAA Journal **54**:**October**, pp. 3134–3148 (2016).

[128] K. Tang, P. M. Congedo, and R. Abgrall, *Adaptive surrogate modeling by ANOVA and sparse polynomial dimensional decomposition for global sensitivity analysis in fluid simulation,* Journal of Computational Physics **314**:pp. 557–589 (2016).

[129] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with basic formal ontology* (Mit Press, 2015).

[130] B. Smith, *Basic formal ontology 2.0 draft specification and user manual*, Tech. Rep. (2015).

[131] B. Smith, *Classifying processes: an essay in applied ontology,* Ratio **25**:**4**, pp. 463–488 (2012).

[132] B. Smith, *On classifying material entities in basic formal ontology,* in *Interdisciplinary Ontology: Proceedings of the Third Interdisciplinary Ontology Meeting* (Keio University Press, 2012) pp. 1–13.

[133] W. Ceusters and B. Smith, *Aboutness: Towards foundations for the information artifact ontology,* in *Proceedings of the Sixth International Conference on Biomedical Ontology (ICBO)* (CEUR vol. 1515, 2015) pp. 1–5.

[134] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, *Sweetening ontologies with dolce,* in *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, edited by A. Gómez-Pérez and V. R. Benjamins (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002) pp. 166–181.

[135] M. C. Rea, *The problem of material constitution,* Philosophical Review **104**:**4**, pp. 525–552 (1995).

[136] A. Goldfain, B. Smith, and L. G. Cowell, *Dispositions and the infectious disease ontology,* in *Formal Ontology in Information Systems*, Frontiers in Artificial Intelligence and Applications (IOS Press, 2010) pp. 400–413.

[137] A. Goldfain, B. Smith, and L. G. Cowell, *Towards an ontological representation of resistance: The case of mrsa,* Journal of Biomedical Informatics **44**:**1**, pp. 35 – 41 (2011), ontologies for Clinical and Translational Research.

[138] J. Benjamin, P. Borst, H. Akkermans, and B. Wielinga, *Ontology construction for technical domains,* in *International Conference on Knowledge Engineering and Knowledge Management* (Springer, 1996) pp. 98–114.

[139] M. N. Roelofs and R. Vos, *Automatically inferring technology compatibility with an ontology and graph rewriting rules,* Journal of Engineering Design **32**:**2**, pp. 90–114 (2021).

[140] A. Habel, R. Heckel, and G. Taentzer, *Graph grammars with negative application conditions,* Fundamenta Informaticae **26**:**3, 4**, pp. 287–313 (1996).

[141] L. Lambers, H. Ehrig, and F. Orejas, *Conflict detection for graph transformation with negative application conditions,* in *International Conference on Graph Transformation* (Springer, 2006) pp. 61–76.

[142] J. A. Estefan, *Survey of model-based systems engineering (MBSE) methodologies*, Tech. Rep. (2008).

[143] S. Kleiner and C. Kramer, *Model based design with systems engineering based on rflp using v6,* in *Smart Product Engineering*, edited by M. Abramovici and R. Stark (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013) pp. 93–102.

[144] S. Tosserams, A. Hofkamp, L. Etman, and J. Rooda, *A specification language for problem partitioning in decomposition-based design optimization,* Structural and Multidisciplinary Optimization **42**:**5**, pp. 707–723 (2010).

[145] N. Alexandrov and R. Lewis, *Reconfigurability in mdo problem synthesis, part 1,* in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (AIAA, 2004) https://arc.aiaa.org/doi/pdf/10.2514/6.2004-4307 .

[146] N. Alexandrov and R. Lewis, *Reconfigurability in mdo problem synthesis, part 2,* in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (AIAA, 2004) https://arc.aiaa.org/doi/pdf/10.2514/6.2004-4308 .

[147] A. B. Lambe and J. R. R. A. Martins, *Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes,* Structural and Multidisciplinary Optimization **46**:**2**, pp. 273–284 (2012).

[148] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau, *An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems,* in *4th International Conference on Pattern Recognition Applications and Methods 2015* (Lisbon, Portugal, 2015).

[149] J. Munkres, *Algorithms for the assignment and transportation problems,* Journal of the Society for Industrial and Applied Mathematics **5**:**1**, pp. 32–38 (1957).

[150] A. Gorodetskiy, A. Shlychkova, and A. r. I. Panov, *Delta schema network in model-based reinforcement learning,* arXiv e-prints p. arXiv:2006.09950 (2020).

[151] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, *Schema networks: Zero-shot transfer with a generative causal model of intuitive physics,* in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, Sydney, NSW, Australia, 2017) p. 1809–1818.

[152] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, *Interaction networks for learning about objects, relations and physics,* in *Advances in neural information processing systems* (2016) pp. 4502–4510.

[153] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, et al., *Relational deep reinforcement learning,* arXiv preprint arXiv:1806.01830 (2018).

[154] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm, *Unsupervised state representation learning in atari,* arXiv preprint arXiv:1906.08226 (2019).

[155] M. N. Roelofs, D. Kurowicka, and R. Vos, *Selecting technologies in aircraft conceptual design using probabilistic inversion,* Journal of Aircraft **58**:**2**, pp. 347–362 (2021).

[156] B. Ozturk and A. Saab, *Optimal aircraft design decisions under uncertainty via robust signomial programming,* in *AIAA Aviation 2019 Forum*, June 17-21 (Dallas, Texas, 2019).

[157] L. W. Cook and J. P. Jarrett, *Robust airfoil optimization and the importance of appropriately representing uncertainty,* AIAA Journal **55**:**11**, pp. 3925–3939 (2017).

[158] D. Bianchi, T. H. Orra, C. Silva, and F. J. Silvestre, *Optimization Under Uncertainty: Rethinking Margins at the Conceptual Design,* in *2018 Multidisciplinary Analysis and Optimization Conference*, June 25-29 (Atlanta, Georgia, 2018).

[159] R. M. Cooke, T. A. Zang, D. N. Mavris, and J. Tai, *Sculpting : A Fast , Interactive Method for Probabilistic Design Space Exploration and Margin Allocation,* in *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, June (AIAA, Dallas, 2015) pp. 1–13.

[160] Y. Huang and D. S. Soban, *Technology impact forecasting as a framework for assessment of multi-functional composites,* in *2018 Aviation Technology, Integration, and Operations Conference*, June 25-29 (Atlanta, Georgia, 2018).

[161] C. Du, D. Kurowicka, and R. Cooke, *Techniques for generic probabilistic inversion,* Computational Statistics & Data Analysis **50**:**5**, pp. 1164 – 1187 (2006).

[162] R. Cooke and D. Kurowicka, *Uncertainty Analysis With High Dimensional Dependence Modelling* (Wiley Series in Probability and Statistics, 2006) p. 284.

[163] J. Kruithof, *Telefoonverkeersrekening,* De Ingenieur **52**:pp. 15–25 (1937).

[164] I. Csiszár, *I-divergence geometry of probability distributions and minimization problems,* The Annals of Probability pp. 146–158 (1975).

[165] R. M. Cooke, *Parameter fitting for uncertain models: modelling uncertainty in small models,* Reliability Engineering & System Safety **44**:**1**, pp. 89–102 (1994).

[166] B. Kraan, *Probabilistic inversion in uncertainty analysis: and related topics,* Ph.D. thesis, Delft University of Technology (2002).

[167] B. Kraan and T. Bedford, *Probabilistic inversion of expert judgments in the quantification of model uncertainty,* Management Science **51**:**6**, pp. 995–1006 (2005).

[168] R. J. White, *Improving the airplane efficiency by use of wing maneuver load alleviation,* Journal of Aircraft **8**:**10**, pp. 769–775 (1971).

[169] J. Xu and I. Kroo, *Aircraft design with maneuver and gust load alleviation,* in *29th AIAA Applied Aerodynamics Conference,* June 27-30 (Honolulu, Hawaii, 2011).

[170] E. Ting, D. Chaparro, N. Nguyen, and G. E. C. Fujiwara, *Optimization of variable-camber continuous trailing-edge flap configuration for drag reduction,* Journal of Aircraft **55**:**6**, pp. 2217–2239 (2018).

[171] E. Obert, *Aerodynamic design of transport aircraft* (IOS press, 2009).

[172] F. Fonte, F. Toffol, and S. Ricci, *Design of a wing tip device for active maneuver and gust load alleviation,* in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference,* January 8-12 (Kissimmee, Florida, 2018).

[173] M. Binois, D. Rullière, and O. Roustant, *On the estimation of pareto fronts from the point of view of copula theory,* Information Sciences **324**:pp. 270 – 285 (2015).

[174] L. W. Cook and J. P. Jarrett, *Horsetail Matching for Optimization Under Probabilistic , Interval and Mixed Uncertainties,* in *19th AIAA Non-Deterministic Approaches Conference,* January (AIAA, Grapevine, 2017) pp. 1–18.

[175] R. Vos and J. van Dommelen, *A conceptual design and optimization method for blended-wing-body aircraft,* in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference* (AIAA) https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1756 .

[176] R. Elmendorp, R. Vos, and G. La Rocca, *A conceptual design and analysis method for conventional and unconventional airplanes,* in *ICAS 2014: Proceedings of the 29th Congress of the International Council of the Aeronautical Sciences,* 7-12 September, International Council of Aeronautical Sciences (ICAS, St. Petersburg, Russia, 2014).

[177] B. Peerlings, *Holistically improving screening decisions under uncertainty in aircraft conceptual design and technology assessment*, M.sc. thesis, Delft University of Technology (2019).

[178] K. Riesen and H. Bunke, *Approximate graph edit distance computation by means of bipartite graph matching*, Image and Vision computing **27**:**7**, pp. 950–959 (2009).

A

CONSTRAINT SATISFACTION PROBLEM

A constraint satisfaction problem (CSP) is the question of assigning values to a set of variables, that satisfy a set of constraints. Many problems can be framed as a CSP, such as map coloring, Sudoku puzzles, or the subgraph isomorphism problem.

Formally, a CSP consists of three elements: a set of variables $\mathscr{X} = \{X_1, ..., X_n\}$, a set of domains $\mathscr{D} = \{D_1, ..., D_n\}$ for each variable and a set of constraints $\mathscr{C} = \{C_1, ..., C_m\}$. A domain D_i holds the allowable values $\{v_1, ..., v_k\}$ for the variable X_i. A constraint C_i is a tuple $(\boldsymbol{X}, r_i)$, where $\boldsymbol{X} \subseteq \mathscr{X}$ and $r_i(\boldsymbol{X} = \boldsymbol{v})$ a constraint function defined on the variable assignment that returns *true* when the constraint is satisfied, and *false* otherwise.

A CSP can be solved using backtracking search, which is a form of depth-first search that chooses values for a variable one at a time and backtracks to a previous point when no valid assignments remain. There are other algorithms to solve CSPs, and extensions or simplifications of the backtracking search that apply in certain specific forms of a CSP. Algorithm 1 is plain backtracking search with some additional features:

- Either return the first valid solution, or return all possible solutions. The first option is relevant if we're only interested in whether there is a solution. The second option when we actually want to retrieve the solutions.
- Assign a score to a solution and return the highest scoring solution.
- Provide a partial (but consistent) solution, which is completed by the algorithm. Suppose we already found a subgraph isomorphism, but now have a supergraph of the subgraph and want to figure out if that supergraph is also a subgraph of the host graph, then we use this feature.

The algorithms make use of some publicly known variables (i.e. across the procedures):

- $\mathscr{S}$: the set of found solutions

- S: the current solution in the backtracking search
- GetFirst: a boolean specifying whether only the first solution of interest
- GetBest: a boolean specifying if the solutions should be scored and only those with maximum scores are kept
- $q_{\max}$: the maximum score of solutions found so far

Algorithm 1 Pseudo-code for Constraint Satisfaction Problem algorithm

Require: $\mathscr{X} \neq \emptyset$
Require: $\forall D_i \in \mathscr{D} : D_i \neq \emptyset$
1: **procedure** CONSTRAINTSATISFACTIONPROBLEM($\mathscr{X}$, $\mathscr{D}$, $\mathscr{C}$, S_0 (optional))
2: **if** $S_0 \neq$ NULL **then**
3: $S \leftarrow S_0$
4: **else**
5: $S \leftarrow \emptyset$
6: **end if**
7: $q_{\max} \leftarrow 0$
8: $\mathscr{S} \leftarrow \emptyset$
9: $\mathscr{X}_R \leftarrow \mathscr{X}$
10: $\mathscr{D}_R \leftarrow \mathscr{D}$
11: **for all** $(X_i, v_i) \in S$ **do**
12: $(\mathscr{X}_R, \mathscr{D}_R) \leftarrow$ FORWARDCHECKING(X_i, v_i, $\mathscr{X}_R$, $\mathscr{D}_R$, $\mathscr{C}$)
13: **end for**
14: BACKTRACK($\mathscr{X}_R$, $\mathscr{D}_R$, $\mathscr{C}$)
15: **return** $\mathscr{S}$
16: **end procedure**

The SCORE function in Algorithm 3 has to be supplied whenever GetBest is set to true. It assigns a score $q \geq 0$ to a complete solution of the CSP. The maximum score $q_{\max}$ is set to zero at the start of the CSP in Algorithm 1.

The forward checking procedure in Algorithm 4 only checks binary constraints. It reduces the domains for the remaining variables to the only remaining valid values, given the current solution.

In theory, solving a CSP works regardless of the order in which variables are assigned. However, a good heuristic that orders the variables can significantly speed up the algorithm. The heuristic in Algorithm 5 selects the first variable that has the least amount of valid values left to assign. The idea is that when a variable only has one valid value, all further solutions should have that value. Thus, by first assigning it and then propagating the constraints, we prune the remaining domains and fewer possible solutions remain.

Algorithm 2 Pseudo-code for Backtracking procedure

```
1: procedure BACKTRACK(X_R, D_R, C)
2:     if GetFirst and not GetBest and S ≠ ∅ then
3:         return
4:     end if
5:     if X_R = ∅ then
6:         KEEPBESTSOLUTION
7:         return
8:     end if
9:     if ∃D_i ∈ D_R : D_i = ∅ then
10:        return
11:    end if
12:    X_j ← SELECTVARIABLEMRV(X_R, D_R)
13:    for all v_j ∈ D_j do
14:        S ← S ∪ (X_j, v_j)
15:        (X'_R, D'_R) ← FORWARDCHECKING(X_j, v_j, X_R, D_R, C)
16:        BACKTRACK(X'_R, D'_R, C)
17:        S ← S \ (X_j, v_j)
18:    end for
19: end procedure
```

Algorithm 3 Pseudo-code for Solution Checking procedure

```
1: procedure KEEPBESTSOLUTION
2:     if GetBest then
3:         q ← SCORE(S)
4:         if q > q_max then
5:             S ← ∅
6:             S ← S ∪ S
7:             q_max ← q
8:         else if q = q_max and not (GetFirst and S ≠ ∅) then
9:             S ← S ∪ S
10:        end if
11:    else
12:        S ← S ∪ S
13:    end if
14: end procedure
```

Algorithm 4 Pseudo-code for Forward Checking procedure

Require: $\forall D_i \in \mathscr{D}_R : D_i \neq \emptyset$

procedure FORWARDCHECKING(X_i, v_i, $\mathscr{X}_R$, $\mathscr{D}_R$, $\mathscr{C}$)
  $\mathscr{X}'_R \leftarrow \mathscr{X}_R \setminus X_i$
  $\mathscr{D}'_R \leftarrow \emptyset$
  for all $X_j \in \mathscr{X}_R \setminus X_i$ **do**
    $D_j \leftarrow \emptyset$
    $C_{ij} \leftarrow \forall (\boldsymbol{X}_k, r_k) \in \mathscr{C} : X_i \in \boldsymbol{X}_k \wedge X_j \in \boldsymbol{X}_k$
    for all $v_j \in D_j$ **do**
      if $\nexists C \in C_{ij} : C(X_i, v_i, X_j, v_j) \mapsto \mathit{false}$ **then**
        $D_j \leftarrow D_j \cup v_j$
      end if
    end for
    $\mathscr{D}'_R \leftarrow \mathscr{D}'_R \cup D_j$
  end for
  return $(\mathscr{X}'_R, \mathscr{D}'_R)$
end procedure

Algorithm 5 Pseudo-code for Variable Selection heuristic (minimum remaining values)

Require: $\mathscr{X}_R \neq \emptyset$

Require: $\forall D_i \in \mathscr{D} : D_i \neq \emptyset$

procedure SELECTVARIABLEMRV($\mathscr{X}_R$, $\mathscr{D}_R$)
  $d_{\min} \leftarrow \min(|D_i| \forall D_i \in \mathscr{D}_R)$
  $X_{textmin} \leftarrow X_i : |D_i| = d_{\min}$
  return $X_{textmin}$
end procedure

B

SUBGRAPH ISOMORPHISM

There are multiple solutions to the subgraph isomorphism problem. However, the algorithm in Algorithm 6 is based on the constraint satisfaction problem (CSP), see Appendix A. Essentially, it first finds the mapping between the vertex sets of the two input graphs G and L. Then, for each such solution, the corresponding edge maps are found. Finally, attributes are mapped to one another. The algorithm could be significantly simplified if the two following conditions are met:

1. Between every pair of nodes, there exists at most one edge.
2. For each node or edge, all attributes are unique, i.e. are distinguishable under the employed equivalence definition.

In the case that these conditions are met, only the first CSP has to be solved. The edges and attribute maps then follow directly from the node maps.

In any case, Algorithm 6 attempts to find the matches of L in G, if there are any. It accepts a few optional parameters:

1. m_0: a partial solution, i.e. $L_0 \subset L$ such that $m_0 : L_0 \mapsto G$.
2. OnlyFirst: a boolean specifying whether only the first solution is of interest (the algorithm terminates after a solution is found, without attempting to find more).
3. AttrValEquiv: a boolean specifying whether attributes are to be matched based on name only (false) or on commensurate values as well (true).

Algorithm 6 Pseudo-code for Subgraph Isomorphism algorithm

procedure SUBGRAPHISOMORPHISM(G, L, m_0 (optional), OnlyFirst (optional), AttrValEquiv (optional)) $\mathscr{X}$, $\mathscr{D}$, $\mathscr{C}$, S_0 (optional)
  $\mathscr{M} \leftarrow \emptyset$
  $\mathscr{X}_N \leftarrow V(L)$
  $\mathscr{D}_{N,i} \leftarrow \forall v_G \in V(G) : X_{N,i} \equiv v_G$
  $\mathscr{C}_N \leftarrow \emptyset$
  for all $e_L \in E(L)$ **do**
    $\mathscr{C}_N \leftarrow \mathscr{C}_N \cup \exists e_G \in E(G) : e_L \equiv e_G$
  end for
  $\mathscr{M}_N \leftarrow$ CONSTRAINTSATISFACTIONPROBLEM($\mathscr{X}_N$, $\mathscr{D}_N$, $\mathscr{C}_N$, m_0, OnlyFirst)
  for all $m_N \in \mathscr{M}_N$ **do**
    $\mathscr{X}_E \leftarrow E(L)$
    $\mathscr{D}_{E,i} \leftarrow \forall e_G \in E(G) : X_{E,i} \equiv e_G$
    $\mathscr{M}_E \leftarrow$ CONSTRAINTSATISFACTIONPROBLEM($\mathscr{X}_E$, $\mathscr{D}_E$, $\emptyset$, OnlyFirst)
    for all $m_E \in \mathscr{M}_E$ **do**
      $\mathscr{X}_A \leftarrow A(L)$
      if AttrValEquiv = true **then**
        $\mathscr{D}_{A,i} \leftarrow \forall a_G \in A(G) : X_{A,i} \equiv a_G \wedge v(X_{A,i}) \equiv v(a_G)$
      else
        $\mathscr{D}_{A,i} \leftarrow \forall a_G \in A(G) : X_{A,i} \equiv a_G$
      end if
      $\mathscr{M}_A \leftarrow$ CONSTRAINTSATISFACTIONPROBLEM($\mathscr{X}_A$, $\mathscr{D}_A$, $\emptyset$, OnlyFirst)
      for all $m_A \in \mathscr{M}_A$ **do**
        $\mathscr{M} \leftarrow \mathscr{M} \cup (m_N, m_E, m_A)$
      end for
    end for
  end for
  return $\mathscr{M}$
end procedure

C

GRAPH EDIT DISTANCE

The Graph Edit Distance (GED) algorithm from Refs. [148, 149, 178] is presented in this appendix. The algorithm is an adaptation of the well-known A* algorithm, which is supplemented with the assignment problem (Hungarian algorithm) to solve for the optimal edit costs.

Rather than explicating the full algorithm, this appendix elaborates on some specific implementation details, that are not clearly explained in the original papers. For the main depth-first GED algorithm, see Algorithm 2 by Abu-Aisheh et al. [148]. In line 2, they state to generate C_v and C_e. These are the cost matrices for the nodes and edges, respectively. These can be generated following the structure in that same paper. In the implementation for this dissertation, a cost matrix C_a is added as well, which is the cost of substituting and/or inserting or removing attributes. The vertices V_1 are then sorted in order of ascending costs.

The tricky part of the algorithm is in estimating the upper bound UB of the graph edit cost. The Munkres algorithm as described by Riesen and Bunke [178] is used for this. This algorithm returns assignments for each vertex in g_1 that pairs it with a vertex in g_2 if it is substituted, or none if it is deleted. Additionally, the edit cost of those assignments is returned. In case the cost matrices are not square, the Munkres algorithm only evaluates the $\min(|V_1|, |V_2|)$ node substitutions. Therefore, the maximum cost of deleting and inserting the remaining elements has to be added. This is repeated for the edges and attributes as well, as shown in Algorithm 7. Finally, the UB is increased by one, because of the inequality operator in line 27 of the algorithm. If the first found upper bound were to be the lower bound as well, the algorithm would never terminate. Note that replacing the inequality operator with a $\leq$ operator would also solve the problem, but makes the algorithm less efficient, because it starts investigating solutions that are just as good as the currently best solution.

The BESTCHILD method simply returns the edit path for which $g(p) + lb(p)$ is smallest. The lower bound $lb(p)$ is computed using Algorithm 7. Rather than considering all elements of g_1 and g_2, though, only the remaining elements are considered (i.e. those that are not already present in the current edit path). However, neither Abu-Aisheh et al.

Algorithm 7 Pseudo-code for computing upper bound UB in GED algorithm

$UB, N_S \leftarrow \text{MUNKRES}(C_V)$
$UB \leftarrow UB + \max(0, |V_1| - N_S) \cdot C_D$
$UB \leftarrow UB + \max(0, |V_2| - N_S) \cdot C_A$
$UB, N_S \leftarrow UB + \text{MUNKRES}(C_E)$
$UB \leftarrow UB + \max(0, |E_1| - N_S) \cdot C_D$
$UB \leftarrow UB + \max(0, |E_2| - N_S) \cdot C_A$
$UB, N_S \leftarrow UB + \text{MUNKRES}(C_A)$
$UB \leftarrow UB + \max(0, |A_1| - N_S) \cdot C_D$
$UB \leftarrow UB + \max(0, |A_2| - N_S) \cdot C_A$
$UB \leftarrow UB + 1$

[148] nor Riesen and Bunke [178] explain very well how $g(p)$ is computed. When only vertices are considered, $g(p)$ is simply the sum of the entries of the cost matrix for each edit in the current edit path. However, to consider edges all previous vertex assignments have to be considered. When $u_1 \mapsto v_2$ and $u_3 \mapsto v_4$ any edge in g_1 between u_1 and u_3 can be substituted with edges in g_2 between v_2 and v_4. Edges in g_1 that do not respect the established vertex edits have to be deleted, and, conversely, such edges in g_2 have to be inserted.

Attributes complicate the computation of $g(p)$ even further, because both vertices and edges can contain attributes. Therefore, for each substitution of a vertex or edge, the costs of substituting, inserting and deleting the associated attributes has to be considered. When a vertex or edge is deleted, all of its attributes also have to be deleted. Similarly, when a vertex or edge is inserted, all of its attributes are inserted as well.

C.1. FAST GED IMPLEMENTATION DETAILS FOR GRAPH TRANSFORMATION RULES

As presented in subsection 4.5.1, the edit costs that occur after application of several graph transformation rules can be computed directly from the graph transformation rules, without employing the computationally expensive GED algorithm as described above. This section dives into detail of how the terms in Equation 4.23 can be computed.

First consider the substitution cost of attributes. Each technology t_k might change the value of an attribute. That attribute retains its identity, thus is present in the gluing graph K_k. Therefore, when applying m_{t_k} to $A(K_k)$ one obtains the attributes in G that are affected by t_k. Given two portfolios P_i and P_j, and performing this operation one obtains the multiset $A(G)_{ij}$. A multiset is a tuple $M = (S, n)$, with S the support of the multiset (obtained as $\text{Supp}(M)$), i.e. the set of which the multiset contains multiple repetitions. The function $n : S \mapsto \mathbb{N}$ associates an integer multiplicity to each element in S, i.e. the amount of times the multiset contains that element. Then, we can compute the overlapping substitution cost as:

$$C_S\left(A(K_i) \cap A(K_j)\right) = \sum_{a \in \text{Supp}(A(G)_{ij})} [n(a) - 1] \cdot \left[C_S(a, a_{R_i}) + C_S(a, a_{R_j})\right] \quad . \tag{C.1}$$

For the overlapping deletion cost, we simply apply m_{t_k} to $(L-K)_k$. This gives all deleted elements in G. For transformations t_k and t_l, we take $D(t_k)$ and $D(t_l)$ as the deletions. Then:

$$C_D\left((L-K)_i \cap (L-K)_j\right) = \sum_{a \in D(t_k) \cap D(t_l)} C_D(a) \quad . \tag{C.2}$$

Finally, the equivalent addition costs need to be discounted. While for added nodes this is trivial — simply determine if they are equivalent, and if so, subtract the addition cost twice — it is more intricate for edges and attributes. For an edge with both vertices in $R-K$ the problem is identical to nodes. However, when an added edge has one or both vertices in K, then it can only be equivalent if the vertices in K, mapped to G through m_{t_k}, are the same vertices. The same applies to attributes and the elements to which they belong. This is the reason the $\overset{\cap}{\cong}$ operator is used instead of $\cap$ in Equation 4.23. Let's define this operator for nodes, edges and attributes, separately. For nodes:

$$n_i \overset{\cap}{\cong} n_j \equiv n_i \cong n_j \tag{C.3}$$

For edges:

$$e_i \overset{\cap}{\cong} e_j \equiv \begin{cases} s(e_i) \cong s(e_j) & \text{if } s(e_i) \in (R-K)_i \wedge s(e_j) \in (R-K)_j \\ m_{t_i} \circ s(e_i) = m_{t_j} \circ s(e_j) & \text{if } s(e_i) \in K_i \wedge s(e_j) \in K_j \\ \mathit{false} & \text{otherwise} \end{cases} \wedge \\ \begin{cases} t(e_i) \cong t(e_j) & \text{if } t(e_i) \in (R-K)_i \wedge t(e_j) \in (R-K)_j \\ m_{t_i} \circ t(e_i) = m_{t_j} \circ t(e_j) & \text{if } t(e_i) \in K_i \wedge t(e_j) \in K_j \\ \mathit{false} & \text{otherwise} \end{cases} \tag{C.4}$$

For attributes:

$$a_i \overset{\cap}{\cong} a_j \equiv a_i \cong a_j \wedge \\ \begin{cases} p(a_i) \overset{\cap}{\cong} p(a_j) & \text{if } p(a_i) \in (R-K)_i \wedge p(a_j) \in (R-K)_j \\ m_{t_i} \circ p(a_i) = m_{t_j} \circ p(a_j) & \text{if } p(a_i) \in K_i \wedge p(a_j) \in K_j \\ \mathit{false} & \text{otherwise} \end{cases} \tag{C.5}$$

While these operations may seem cumbersome, they are computable in polynomial time (provided the morphisms m_{t_k} are already known), rather than in non-polynomial time, as the GED algorithm is.

D

ENABLING TECHNOLOGY SET

Algorithm 8 finds all the sets of technologies that enable the technologies which cannot be applied to G. For this, it makes a call to Algorithm 9, which returns for an inapplicable technology t any possible set of technologies that enables it. Therefore, note that the statement **return** solution $\prec t$ should not be interpreted as returning from the method call, but rather as a yield statement that proceeds as an enumerator. As such, for a given technology t, there may be multiple solutions.

Algorithm 8 Pseudo-code for Enabling Technology Set algorithm

```
1: procedure GetEnablingTechnologySet(G, 𝕋, TCM)
2:     𝕋_N ← t ∈ 𝕋 : L_t ⊈ G
3:     𝕋_P ← 𝕋 \ 𝕋_N
4:     for all t ∈ 𝕋_N do
5:         ExtendSolution(G, t, ∅, 𝕋_P, TCM)
6:     end for
7: end procedure
```

Algorithm 9 Pseudo-code for extending an enabling technology set

Require: $\mathbb{T}_P \neq \emptyset$

```
1: procedure ExtendSolution(G, t, T_E, T_P, TCM)
2:     t_e ← first element of T_P
3:     solution ← T_E ∪ t_e
4:     canApply ← ∄t_i ∈ T_E : TCM_{t_e,t_i} = 1
5:     if canApply and CheckSolution(G, t, solution) then
6:         return solution ≺ t
7:     end if
8:     ExtendSolution(G, t, T_E, T_P \ t_e, TCM)
9:     if canApply then
10:        ExtendSolution(G, t, solution, T_P \ t_e, TCM)
11:    end if
12: end procedure
```

Algorithm 10 Pseudo-code for checking if a set of technologies enables a given technology

```
1: procedure CheckSolution(G, t, T_E)
2:     G_T ← G
3:     for all t_e ∈ T_E do
4:         G_T ← G_T ⟼^{t_e} G'_T
5:     end for
6:     if L_t ⊆ G_T then
7:         return true
8:     else
9:         return false
10:    end if
11: end procedure
```

E

MAXIMUM DISSIMILARITY

The maximum dissimilarity algorithm in Algorithm 11 finds a subset of size K of maximally dissimilar items given a matrix of distances $\boldsymbol{D}$ between these items. One can optionally specify the first item i_0 to be included in the subset.

Algorithm 11 Pseudo-code for Maximum Dissimilarity algorithm

Require: $N \in \mathbb{Z} > 0$
Require: $K \in \mathbb{Z} \leq N$
Require: $\boldsymbol{D} \in \mathbb{R}^N \times \mathbb{R}^N$
Require: $i_0 \in [1, N]$

procedure MAXIMUMDISSIMILARITY(N, K, $\boldsymbol{D}$, i_0 (optional))
  if i_0 = NULL **then**
    $i_0 \leftarrow 1$
  end if
  $R \leftarrow 1, ..., N \setminus i_0$
  $S \leftarrow i_0$
  for $i \leftarrow 1, K-1$ **do**
    $d_{\max} \leftarrow -\infty$
    $i_{\max} \leftarrow -1$
    for all $r \in R$ **do**
      $d_{\min} \leftarrow \infty$
      for all $s \in S$ **do**
        if $\boldsymbol{D}_{r,s} < d_{\min}$ **then**
          $d_{\min} \leftarrow \boldsymbol{D}_{r,s}$
        end if
      end for
      if $d_{\min} > d_{\max}$ **then**
        $d_{\max} \leftarrow d_{\min}$
        $i_{\max} \leftarrow r$
      end if
    end for
    $S \leftarrow S \cup i_{\max}$
    $R \leftarrow R \setminus i_{\max}$
  end for
  return S
end procedure

F

PORTFOLIO GENERATION

This algorithm simply generates the power set of all portfolios given a set of technologies. It consecutively removes those portfolios which either contain incompatible technologies ($\boldsymbol{TCM}_{t_i,j} = 1$) or do not contain an enabling technology set for inapplicable technologies. The last condition is checked using the statement $\nexists \mathbb{T}_E \in p : \mathbb{T}_E \prec t_i$. To find whether $\mathbb{T}_E \prec t_i$, the enabling technology set algorithm from Appendix D is used.

Algorithm 12 Pseudo-code for Portfolio Generation algorithm

Require: $\boldsymbol{TCM} \in [0,1]^{|\mathbb{T}|} \times [0,1]^{|\mathbb{T}|}$

procedure GENERATEPORTFOLIOS(G, $\mathbb{T}$, $\boldsymbol{TCM}$)
  $\mathbb{P} \leftarrow 2^{\mathbb{T}}$
  $\mathbb{P}_V \leftarrow \emptyset$
  for all $p \in \mathbb{P}$ **do**
    isValid $\leftarrow$ true
    for all $t_i \in \mathbb{T}$ **do**
      for all $t_j \in p$ **do**
        if $\boldsymbol{TCM}_{t_i,j} = 1$ **then**
          isValid $\leftarrow$ false
        end if
      end for
      if $L_{t_i} \nsubseteq G$ and $\nexists \mathbb{T}_E \in p : \mathbb{T}_E \prec t_i$ **then**
        isValid $\leftarrow$ false
      end if
    end for
    if isValid = true **then**
      $\mathbb{P}_V \leftarrow \mathbb{P}_V \cup p$
    end if
  end for
  return $\mathbb{P}_V$
end procedure

G

Factorio Results

The full tables of portfolio distances and softmax assignment probabilities are presented in this appendix. They belong to the case study in section 4.8.

Table G.1: Portfolio distances for the Factorio example

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	16	16	16	32	32	16	32	32	3	19	19	19	35	35	19	35	35
1	16	0	32	32	16	48	32	16	48	19	3	35	35	19	51	35	19	51
2	16	32	0	32	48	16	32	48	16	19	35	3	35	51	19	35	51	19
3	16	32	32	0	16	16	16	32	32	19	35	35	3	19	19	19	35	35
4	32	16	48	16	0	32	32	16	48	35	19	51	19	3	35	35	19	51
5	32	48	16	16	32	0	32	48	16	35	51	19	19	35	3	35	51	19
6	16	32	32	16	32	32	0	16	16	19	35	35	19	35	35	3	19	19
7	32	16	48	32	16	48	16	0	32	35	19	51	35	19	51	19	3	35
8	32	48	16	32	48	16	16	32	0	35	51	19	35	51	19	19	35	3
9	3	19	19	19	35	35	19	35	35	0	16	16	16	32	32	16	32	32
10	19	3	35	35	19	51	35	19	51	16	0	32	32	16	48	32	16	48
11	19	35	3	35	51	19	35	51	19	16	32	0	32	48	16	32	48	16
12	19	35	35	3	19	19	19	35	35	16	32	32	0	16	16	16	32	32
13	35	19	51	19	3	35	35	19	51	32	16	48	16	0	32	32	16	48
14	35	51	19	19	35	3	35	51	19	32	48	16	16	32	0	32	48	16
15	19	35	35	19	35	35	3	19	19	16	32	32	16	32	32	0	16	16
16	35	19	51	35	19	51	19	3	35	32	16	48	32	16	48	16	0	32
17	35	51	19	35	51	19	19	35	3	32	48	16	32	48	16	16	32	0

Table G.2: Portfolio similarities as computed using softmax for the Factorio example

	0	17	13	7	5	2	12	1	6
0	1.000E+000	6.305E-016	6.305E-016	1.266E-014	1.266E-014	1.125E-007	5.603E-009	1.125E-007	1.125E-007
1	1.125E-007	7.095E-023	5.603E-009	1.125E-007	1.425E-021	1.266E-014	6.305E-016	1.000E+000	1.266E-014
2	1.125E-007	5.603E-009	7.095E-023	1.425E-021	1.125E-007	1.000E+000	6.305E-016	1.266E-014	1.266E-014
3	2.260E-006	1.266E-014	1.125E-007	2.544E-013	2.260E-006	2.544E-013	1.000E+000	2.544E-013	2.260E-006
4	2.544E-013	1.425E-021	1.000E+000	2.260E-006	2.544E-013	2.863E-020	1.125E-007	2.260E-006	2.544E-013
5	1.266E-014	5.603E-009	6.305E-016	1.425E-021	1.000E+000	1.125E-007	5.603E-009	1.425E-021	1.266E-014
6	1.125E-007	5.603E-009	6.305E-016	1.125E-007	1.266E-014	1.266E-014	5.603E-009	1.266E-014	1.000E+000
7	1.266E-014	6.305E-016	5.603E-009	1.000E+000	1.425E-021	1.425E-021	6.305E-016	1.125E-007	1.125E-007
8	2.544E-013	1.000E+000	1.425E-021	2.544E-013	2.260E-006	2.260E-006	1.266E-014	2.862E-020	2.260E-006
9	1.000E+000	2.544E-013	2.544E-013	1.266E-014	1.266E-014	1.125E-007	2.260E-006	1.125E-007	1.125E-007
10	1.125E-007	2.863E-020	2.260E-006	1.125E-007	1.425E-021	1.266E-014	2.544E-013	1.000E+000	1.266E-014
11	1.125E-007	2.260E-006	2.863E-020	1.425E-021	1.125E-007	1.000E+000	2.544E-013	1.266E-014	1.266E-014
12	5.603E-009	1.266E-014	1.125E-007	6.305E-016	5.603E-009	6.305E-016	1.000E+000	6.305E-016	5.603E-009
13	6.305E-016	1.425E-021	1.000E+000	5.603E-009	6.305E-016	7.095E-023	1.125E-007	5.603E-009	6.305E-016
14	1.266E-014	2.260E-006	2.544E-013	1.425E-021	1.000E+000	1.125E-007	2.260E-006	1.425E-021	1.266E-014
15	1.125E-007	2.260E-006	2.544E-013	1.125E-007	1.266E-014	1.266E-014	2.260E-006	1.266E-014	1.000E+000
16	1.266E-014	2.544E-013	2.260E-006	1.000E+000	1.425E-021	1.425E-021	2.544E-013	1.125E-007	1.125E-007
17	6.305E-016	1.000E+000	1.425E-021	6.305E-016	5.603E-009	5.603E-009	1.266E-014	7.095E-023	5.603E-009

H

DEPENDENCY GRAPH ALGORITHM

The main function that generates the dependency graph for a given system is provided in Algorithm 13. This method only generates a graph with all the attributes from the system graph S, and calls two other methods that include the analysis nodes into the dependency graph.

Algorithm 13 Pseudo-code for constructing dependency graph

```
1: procedure GETDEPENDENCYGRAPH(S)
2:     G ← (A(S), ∅)
3:     EXTENDGRAPH(G)
4:     INCLUDEATTRIBUTERELATIONS(G)
5: end procedure
```

The extension algorithm in Algorithm 14 is the primary workhorse for the dependency graph generation. It loops through all possible analysis, their applications in S and the computation modes. Then it adds an analysis node and the appropriate input edges and output edges to the dependency graph G. In case the analysis can be decomposed into multiple analyses, this is done by Algorithm 15.

The decomposition of analysis methods in Algorithm 15 is broken into three steps: adding decomposition of the input attributes, combination of the output attributes, and the addition of the elementwise operations. Decomposing attributes into their elements and recombining those elements is done in ADDCOMBINEANALYSIS. This algorithm keeps track of whether the attribute has already been decomposed. If not, its elements are added to G and a combination and decomposition analysis are added as well. The decomposition analysis takes the elements of a tensor quantity and copies those to the scalar valued elements. The combination analysis does the inverse: it copies the scalar values and places them back into the tensor quantity.

The final part of the dependency graph constitutes analysis methods that represent equalities and derivatives. These relationships between attributes are present in the sys-

Algorithm 14 Pseudo-code for extending the dependency graph

```
1:  procedure EXTENDGRAPH(G)
2:      for all a_i ∈ 𝔸 do
3:          for all m_j ∈ m(a_i, G) do
4:              for all f_k ∈ f(a_i) do
5:                  n ← (a_i, f_k, m_j)
6:                  V(G) ← V(G) ∪ n
7:                  if a_i is decomposable then
8:                      ADDELEMENTWISEANALYSIS(G, a_i, m_j, f_k)
9:                  else
10:                     for all p_in ∈ I(f_k) do
11:                         e ← (m_j ∘ μ_j ∘ ν_i ∘ p_in, n)
12:                         E(G) ← E(G) ∪ e
13:                     end for
14:                     for all p_out ∈ O(f_k) do
15:                         e ← (n, m_j ∘ μ_j ∘ ν_i ∘ p_out)
16:                         E(G) ← E(G) ∪ e
17:                     end for
18:                 end if
19:             end for
20:         end for
21:     end for
22: end procedure
```

tem graph S in the form of typed edges between attributes. Algorithm 16 adds these two types of analysis methods.

Algorithm 15 Pseudo-code for adding a element-wise decomposable attribute

procedure AddElementwiseAnalysis(G, a_i, m_j, f_k)
  for all $p_{\text{in}} \in I(f_k)$ **do**
    AddCombineAnalysis(G, $m_j \circ \mu_j \circ \nu_i \circ p_{\text{in}}$)
  end for
  for all $p_{\text{out}} \in O(f_k)$ **do**
    AddCombineAnalysis(G, $m_j \circ \mu_j \circ \nu_i \circ p_{\text{out}}$)
  end for
  for $m = 1$ to number of components of a_i **do**
    $m'_j \leftarrow m_j$ where attributes are replaced with m-th component
    $n \leftarrow (a_i, f_k, m'_j)$
    $V(G) \leftarrow V(G) \cup n$
    for all $p_{\text{in}} \in I(f_k)$ **do**
      $e \leftarrow (m'_j \circ \mu_j \circ \nu_i \circ p_{\text{in},m}, n)$
      $E(G) \leftarrow E(G) \cup e$
    end for
    for all $p_{\text{out}} \in O(f_k)$ **do**
      $e \leftarrow (n, m'_j \circ \mu_j \circ \nu_i \circ p_{\text{out},m})$
      $E(G) \leftarrow E(G) \cup e$
    end for
  end for
end procedure

Algorithm 16 Pseudo-code for including attribute relations in dependency graph

```
1:  procedure INCLUDEATTRIBUTERELATIONS(G, S)
2:      for all e ∈ E(S) : type(e)=is derivative of do
3:          n_int ← analysis node for integration
4:          n_der ← analysis node for derivation
5:          V(G) ← V(G) ∪ n_int
6:          E(G) ← E(G) ∪ (s(e), n_int)
7:          E(G) ← E(G) ∪ (n_int, t(e))
8:          V(G) ← V(G) ∪ n_der
9:          E(G) ← E(G) ∪ (t(e), n_der)
10:         E(G) ← E(G) ∪ (n_der, s(e))
11:     end for
12:     for all e ∈ E(S) : type(e)=is equal to do
13:         n_eq1 ← analysis node for equality
14:         n_eq2 ← analysis node for equality
15:         V(G) ← V(G) ∪ n_eq1
16:         E(G) ← E(G) ∪ (s(e), n_eq1)
17:         E(G) ← E(G) ∪ (n_eq1, t(e))
18:         V(G) ← V(G) ∪ n_eq2
19:         E(G) ← E(G) ∪ (t(e), n_eq2)
20:         E(G) ← E(G) ∪ (n_eq2, s(e))
21:     end for
22: end procedure
```

I

COMPUTATION GRAPH ALGORITHM

The computation graphs are generated from a dependency graph G, given a set of quantities of interest $\mathbb{Q}$. Algorithm 17 creates the first partial computation sequence, which only contains the quantities of interest. It then calls Algorithm 18, which extends this computation sequence to a complete and valid one, and possibly creates multiple alternatives along the way.

A computation sequence consists of three elements: the PENDING set are all attributes that still have to be computed in the computation. Thus, analysis nodes have to be added to compute those attributes. The VISITED set are the attributes that are being computed in the computation sequence. Finally, the SEQUENCE set are the analysis nodes that make up the computation.

Algorithm 17 Pseudo-code for computation graph algorithm

```
1: procedure CreateComputationGraphs(G, ℚ)
2:     C ← ∅
3:     for all q ∈ ℚ do
4:         Pending(C) ← q
5:     end for
6:     P_C ← C
7:     while P_C ≠ ∅ do
8:         ProcessComputation(P_C, G)
9:     end while
10: end procedure
```

Whenever a unique, complete computation sequence is found, it is added to a set of solutions. This is done in Algorithm 18. If a sequence is not yet complete, all pending attributes are considered, and new computation sequences are generated that include an analysis node to compute the pending attribute. If a computation sequence is valid, which is checked with Algorithm 20, it is added to the pool of pending solutions P_C.

Algorithm 18 Pseudo-code for extending a computation

```
1:  procedure ProcessComputation(P_C, G)
2:      C ← FirstOf(P_C)
3:      if IsComplete(C, G) and C is a unique solution then
4:          Add C to solutions
5:      else
6:          while Pending(C) ≠ ∅ do
7:              C_P Pending(C)
8:              a ← FirstOf(C_P)
9:              Visited(C) ← a
10:             if a is Known then
11:                 continue
12:             end if
13:             for all n ∈ PredecessorsOf(a, G) \ Sequence(C) do
14:                 C' ← Copy(C)
15:                 Sequence(C') ← n
16:                 for all p ∈ PredecessorsOf(n, G) do
17:                     if p ∉ Visited(C') then
18:                         Pending(C') ← p
19:                     end if
20:                 end for
21:                 if CheckValidity(C'), G then
22:                     P_C ← P_C ∪ C'
23:                 end if
24:             end for
25:         end while
26:     end if
27: end procedure
```

A computation graph (i.e. sequence) is complete when it computes all the quantities of interest, from only known variables. Furthermore, there may be no dangling analysis nodes, i.e. analysis nodes without predecessors or successors. These conditions are checked in Algorithm 19.

A computation sequence has to satisfy several constraints, which were explicated in subsection 5.1.3. Essentially, Algorithm 20 checks if the computation graph contains no cycles which are indeterminate, or cycles without a derivative or integral term.

Algorithm 19 Pseudo-code for checking if a computation solution is complete

```
1: procedure IsComplete(C, G)
2:     V_C ← V(G) \ C
3:     E_C ← E(G) ∈ V_C × V_C
4:     for all v ∈ V_C such that ∄e ∈ E_C : t(e) = v do
5:         if v is a QoI then
6:             return false
7:         end if
8:         if v is an Analysis Node then
9:             return false
10:        end if
11:        if v is not a Known then
12:            return false
13:        end if
14:    end for
15:    return true
16: end procedure
```

Algorithm 20 Pseudo-code for checking if a computation solution is valid

```
1: procedure CheckValidity(C, G)
2:     V_C ← V(G) \ C
3:     E_C ← E(G) ∈ V_C × V_C
4:     K ← (V_C, E_C)
5:     for all cycle ∈ K do
6:         if ∃a_i, a_j ∈ cycle : a_i ≅ a_j and m_i ≅ m_j and f_i = f_j then
7:             return false
8:         end if
9:         if ∄a ∈ cycle : a is an analysis node for derivation/integration then
10:            return false
11:        end if
12:    end for
13:    return true
14: end procedure
```

J

System Flow Directions

The Serial-Parallel Partial Hybrid (SPPH) powertrain architecture from Chapter 5 has nine different operating directions. That is to say, electrical power has a total of nine unique ways in which it may flow through that powertrain architecture. This is because the battery may either supply or store energy, and the electrical machines either operate as motors or as generators. In generic terms, any system may have different situations where the flow of information, matter or energy through the system changes direction. These possible directions are solved for by means of a Constraint Satisfaction Problem (CSP). This appendix illustrates how that works by walking through the SPPH example.

To illustrate how that works, consider Figure 5.9, where the roles of the electric energy and rotational energy are merely described as boundary flows. Conversely, observe Figure 5.10 and notice how an input and an output flow have to be specified. The model requires this, because the direction of a flow relative to a domain determines how the process in that domain behaves. (A simple example is a bucket with a connected hose. If there is an inflow of water through the hose, the bucket fills up with water and eventually overflows. Conversely, when there is an outflow of water through the hose, the bucket eventually empties and air is drawn in through the hose.) Thus, a boundary-flow role is perceived as either an inflow or outflow to a certain domain. Therefore, this particular role can be divided into two sub-categories: inflow and outflow. An algorithm needs to define the directions of flow throughout a system is to assign the appropriate roles to each of the boundary flow entities.

In terms of the CSP, there are as many variables as there are boundary flow entities in the system. For now, it is assumed that each boundary flow only participates in two adjacent processes. In other words, it sits on a boundary of exactly two spatial regions. Then, picking either of these two processes, the boundary flow either flows into it or out of it. As such, each variable has two possible values in the CSP, and it needs to keep track with respect to which process those values are measured. For example, the electric energy of the battery in Figure 5.10 may flow into our out of the battery. That is measured with respect to the battery working principle. Because of the connection with the EM, the battery energy has the opposite role as perceived by the EM process.

Table J.1: Solutions to the CSP obtaining all configurations of the SPPH architecture. For the variable names, refer to Figure 5.8. Inflow is denoted + and outflow as -. The variables are grouped into sequences that need to have the same signs in a valid solution

Variable	Reference	S1	S2	S3	S4	S5	S6	S7	S8	S9
P_{p1}	P1	-	-	-	-	-	-	+	+	+
P_{s1}	GB	-	-	-	-	-	-	+	+	+
P_{gb}	EM1	+	+	+	-	-	-	+	+	+
P_{e1}	PM	+	+	+	-	-	-	+	+	+
P_{bat}	Bat	-	+	+	-	+	-	+	-	+
P_{e2}	PM	-	-	+	-	+	+	-	-	+
P_{s2}	EM2	-	-	+	-	+	+	-	-	+
P_{p2}	P2	-	-	+	-	+	+	-	-	+

A CSP would not be complete without constraints. This particular problem is constrained by the conservation laws. Concretely, when a domain has an inflow, there must be an outflow, or there must be an accumulative quality within the domain. The constraint reads that for each process (which discerns a domain), a full assignment for its participants should contain any one of the three following pairs: a inflow and an outflow, an inflow and an internal flow, or an outflow and an internal flow. Such a formulation assumes that any internal flow has a capacity. However, an internal flow with capacity is not always pragmatic. For example, an electric wire has an internal electric flow, but a very small, or in fact, negligible, capacitance. In the CSP formulation, therefore, the cases where an internal flow is required are dropped, leaving only one option: inflow + outflow must be present simultaneously for any domain. This is the only constraint necessary to solve the problem.

From the CSP, all different assignments of inflow/outflow can be generated. And, indeed, doing this for the SPPH architecture, nine assignments are obtained, as shown in Table J.1. These are the same as the ones obtained by de Vries et al. [46]. For each of these assignments the power model can be applied, separately. Each component in the architecture is attributed with an efficiency variable, and each energy quality with a power variable. For each particular model instance (limited to one component), these power variables are attributed with the input or output role.

K

DEPENDENCY GRAPHS CASE STUDIES

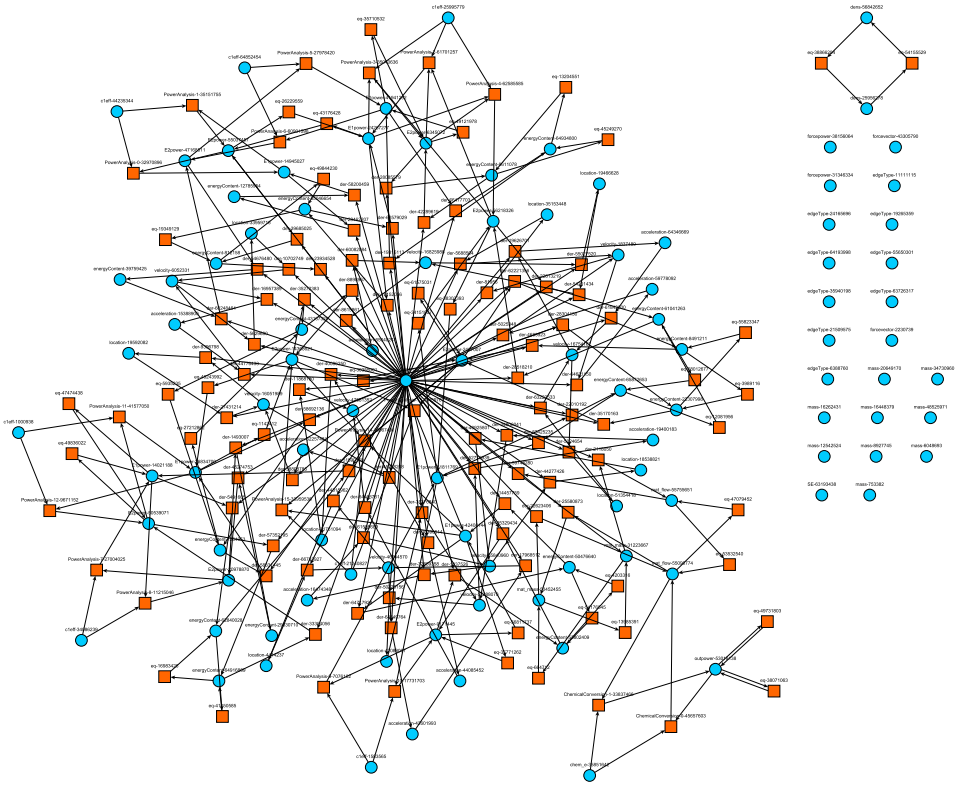


Figure K.1: Full dependency graph for SPPH powertrain architecture

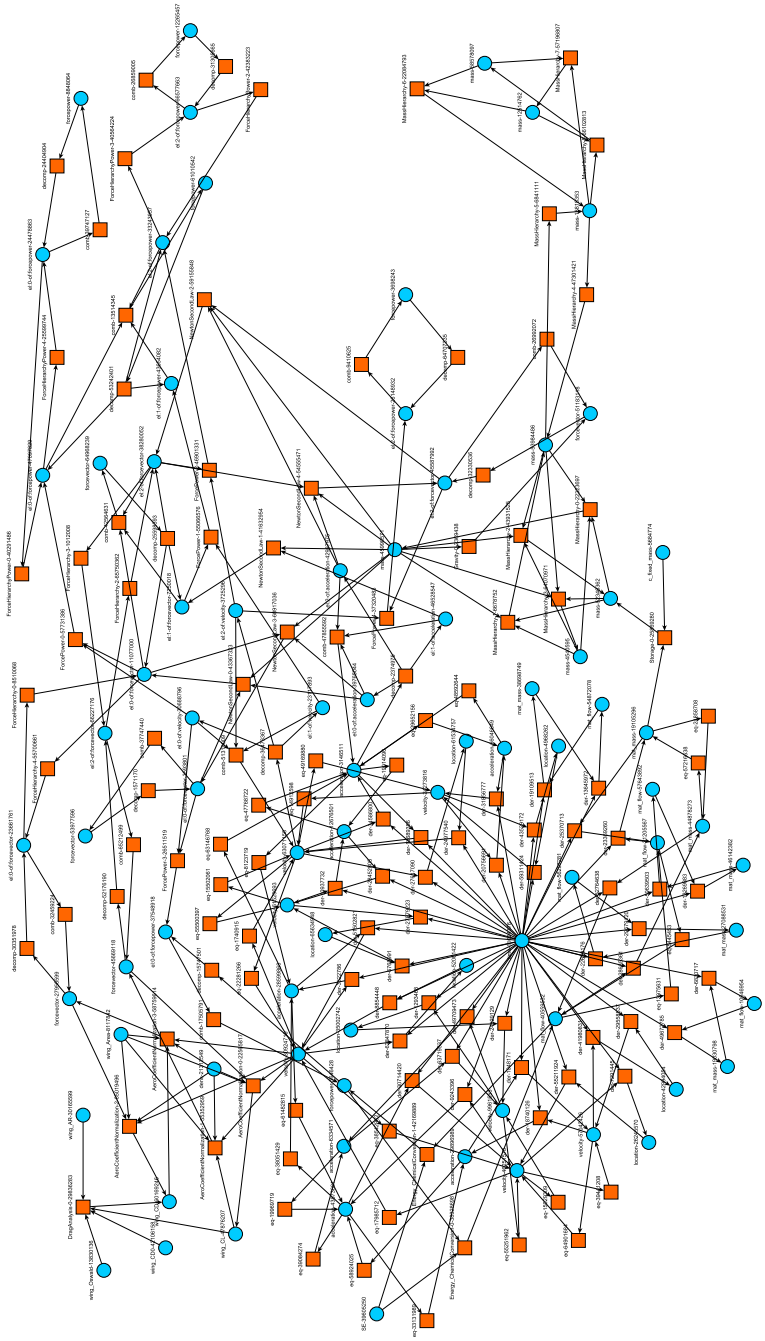


Figure K.2: Dependency graph for mission analysis of conventional aircraft with second DOF flap

CURRICULUM VITÆ

Martijn ROELOFS

10-03-1992 Born in Leidschendam, the Netherlands.

EDUCATION

2017-2020 PhD. Aerospace Engineering
Delft University of Technology, Delft, the Netherlands
Thesis: Graph-based, probabilistic framework for technology evaluation and selection
Promotor: Prof. dr. L.L.M. Veldhuis
Copromotor: Dr. R. Vos

2013–2016 MSc in Aerospace Engineering
Delft University of Technology, Delft, the Netherlands
Thesis: Semi-Analytical Composite Oval Fuselage Mass Estimation
Supervisor: Dr. R. Vos

2010–2013 BSc in Aerospace Engineering
Delft University of Technology, Delft, the Netherlands

2004–2010 Secondary School
Christelijk Gymnasium Sorghvliet, Den Haag, the Netherlands

List of Publications

Journal Publications

2. **M.N. Roelofs, D. Kurowicka, R. Vos**, *Selecting Technologies in Aircraft Conceptual Design Using Probabilistic Inversion*, Journal of Aircraft **58:2**, pp. 347-362 (2021).

1. **M.N. Roelofs, R. Vos**, *Automatically inferring technology compatibility with an ontology and graph rewriting rules*, Journal of Engineering Design **32:2**, pp. 90-114 (2021).

Conference Publications

4. **M.N. Roelofs, R. Vos, K. Amadori, C. Jouannet**, "Automatically Generating the Technology Compatibility Matrix Using Graph Transformations", *AIAA Aviation 2019 Forum*, AIAA Paper 2019-2886, June 2019. doi: 10.2514/6.2019-2886

3. **M.N. Roelofs, R. Vos**, "Formalizing Technology Descriptions for Selection During Conceptual Design", *AIAA SciTech 2019 Forum*, AIAA Paper 2019-0816, January 2019. doi: 10.2514/6.2019-0816

2. **M.N. Roelofs, R. Vos**, "Uncertainty-Based Design Optimization and Technology Evaluation: A Review", *2018 AIAA Aerospace Sciences Meeting*, AIAA Paper 2018-2029, January 2018. doi: 10.2514/6.2018-2029

1. **M.N. Roelofs, R. Vos**, "Semi-Analytical Composite Oval Fuselage Weight Estimation", *55th AIAA Aerospace Sciences Meeting*, AIAA Paper 2017-0466, January 2017. doi: 10.2514/6.2017-0466

