

# Aircraft Systems Conceptual Design

An object-oriented approach from <element> to  
<aircraft>

**Ingo Staack**



Division of Fluid and Mechatronic Systems  
Department of Management and Engineering  
Linköping University, SE-581 83 Linköping, Sweden

Linköping 2016

**Cover:**

The cover shows <aircraft id="35"> at <Malmen rwy="19"> and an illustration of the on-board power systems network

Copyright © Ingo Staack, 2016

Aircraft Systems Conceptual Design

An object-oriented approach from <element> to <aircraft>

ISBN 978-91-7685-636-9

ISSN 0345-7524

**Distributed by:**

Division of Fluid and Mechatronic Systems  
Department of Management and Engineering  
Linköping University  
SE-581 83 Linköping, Sweden

Printed in Sweden by LiU-Tryck, Linköping 2016.

*To my parents*

Es gibt nichts Gutes,  
außer man tut es

Erich Kästner 1899-1974



# Abstract

Aircraft Conceptual Design (ACD) is facing new challenges on the way to enhanced fidelity level required in, nowadays complex, system design. The challenge during this early design phase is the use of higher-fidelity methods typically applied in later development stages. Integration of models and simulations to enhance analysis capability while maintaining a streamlined, transparent, and low cost (low effort regarding task time and workforce) work process is therefore required.

In this thesis, the use of object-oriented KBE methods to enable an early integration of simulation models, based on incomplete data, is presented. Before this, careful investigations of modelling and simulation approaches of multi-domain systems are carried out, and their use in the ACD phase is examined regarding the efficiency between effort and result accuracy. A central, parametric information model approach to make this possible is presented. By the means of the extended use of XML, XSD and XSLT, domain-specific architectural models can be translated from this dataset, supporting a direct CAD domain integration and automated model creation.

Modelling systems as graph networks is a simple approach for unified modelling in the conceptual design stage. Based on this theory, the similarity of different modelling approaches like Dependency Structure Matrix (DSM), MDDSM, or Channel-Agency Networks is shown. Using object-oriented programming, all these and more aspects such as e.g. Fault Tree Analysis (FTA) can be handled globally as one graph set.

Based on the outcomes of the theoretical part, the development of a conceptual aircraft design framework is described. Backed by a central XML-based namespace, this framework integrates a complete CAD environment to ensure an appropriate environment for the geometric domain modelling. In addition, the use of KBE for automated simulation

model integration is exemplified by a hydraulic aircraft flight control system (FCS) simulation model. In conclusion, an example of multi-aspect modelling using object-oriented handling of a graph network is shown. For future scenarios, unified modelling and semantic approaches are mentioned.

# Populärvetenskaplig Sammanfattning

Med nya, mer komplexa och mer integrerade flygplansgrundsystem står konceptutvärderingsfasen (ACD) inför nya utmaningar. Multi-disciplinärt, anpassningsförmåga, konstruktionsautomation, förbättrad analysnoggrannhet och modellbaserad integrerad utveckling (MBSE) är några nyckelord.

I denna avhandling diskuterats användning av objektorienterade Knowledge Based Engineering- (KBE) metoder i konceptutvärderingsfasen som möjliggör en tidig integration av simuleringsmodeller baserade på ofullständiga designuppgifter. Denna integration av modeller och simulering ska leda till en ökad analyskapacitet och samtidigt bibehålla en strömlinjeformad, transparent och snabbt arbetsprocess.

I avhandlingens teoretiska del behandlas universella modelleringsmetoder (som DSM och Channel-Agency Nätverk) och simuleringsstrategier. Dessutom analyseras och beskrivs en omsorgsfullt parametriserad informationsmodell samt integrationsstrategier för analysmetoder av olika domäner och noggrannhet.

Baserat på den teoretiska delen beskrivs utveckling av ett ramverk för konceptuell flygplandesign. Uppbyggd runt en systematisk parametriserad, centralt XML-baserad informationsmodell integrerar detta ramverk ett fullständigt 3D CAD verktyg för att säkerställa en lämplig miljö för den geometriska modelleringen. Dessutom visas en användningen av KBE-metoder för integration av en hel flygplanssimuleringsmodell inklusive dess hydrauliska styrsystem. För framtida möjligheter namnges unifierade modelleringsstrategier och semantiska metoder.





# Zusammenfassung

Mit steigenden Anforderungen durch immer mehr optimierte, leistungsfähigere, zuverlässigere und langlebigere technische Produkte nimmt die Modellierung und Simulation einen immer größeren Stellenwert ein. Um das Zusammenspiel von Systemen bereits im Flugzeugvorentwurf analysieren zu können, bedarf es einer geschickten Systemmodellierung und geeigneter Arbeitsprozesse, die die Erstellung von Simulationsmodellen auf Basis unvollständiger Daten- und Informationslage ermöglichen. Vor Allem, um das volle Optimierungspotential moderner, integrierter elektrischer Systemarchitekturen ausschöpfen zu können, ist eine Einbeziehung dieser in das Gesamtkonzept innerhalb des Flugzeugvorentwurfs notwendig.

In dieser Arbeit wird ein wissensbasierter Arbeitsprozess (englisch: Knowledge-Based Engineering) für den Flugzeugvorentwurf präsentiert, welcher die Zusammenführung unterschiedlicher Informationen – wie z.B. Domänen- und Produktspezifischer Daten ermöglicht. Dies soll die Einbeziehung der Bordsystemarchitekturen dienen, um die gestiegenen Genauigkeitsanforderungen im Flugzeugvorentwurf bewältigen zu können. Dem vorausgehend werden verschiedene Modellierungsgrundsätze erörtert und ihre Anwendungsmöglichkeit im Hinblick auf die mögliche Implementierung und Anwenderfreundlichkeit im Flugzeugvorentwurf diskutiert.

Ausgehend von den theoretischen Überlegungen wird die Entwicklung eines XML-basierten Flugzeugvorentwurfsprogramms beschrieben, das die vollständige Integration eines kommerziellen CAD-Werkzeuges ermöglicht. Des Weiteren werden die Möglichkeiten aufgezeigt, welche sich durch den Einsatz eines universalen Modells im XML-Format ergeben. Durch Interpretation der Produktdaten in Form eines Graphennetzwerks werden verschiedene Modellierungs- und Analysemöglichkeiten wie

Beispielsweise DSM und C-A Net Modell erörtert und die Integration von Teilaspekten wie der Systemzuverlässigkeit aufgezeigt.

Ziel ist ein universeller Modellierungsansatz, der eine plausible, verständliche und anwenderfreundliche Integration der verschiedenen Teilaspekte des Flugzeugvorentwurfs ermöglicht sowie die Einbindung domäne-spezifischer Programme (wie z.B. CAD) mit Hilfe einer parametrischen, auf XML basierenden Datenbank erlaubt.

# Acknowledgements

The work presented in this dissertation has been carried out as a Ph.D. study at the Division of Fluid- and Mechatronic Systems (Flumes) at Linköping University together with Saab Aeronautics as industrial partner.

The research was mainly funded by the Swedish Governmental Agency, VINNOVA's National Aviation Engineering Research Programms NFFP5 2010-1251 "Konceptmetodik", NFFP6 2014-0927 "CADLAB" and Clean Sky SFWA, the Swedish Aeronautical Development and Demonstration Programm.

There are several people I would like to thank for their support and advice during all these years.

First I want to thank my supervisor Prof. Petter Krus for his guidance and support. Thank you, Petter, for the fruitful working environment and the freedom you gave me which made this research so interesting and challenging. Thank you for your patience and trust in my work!

Secondly, thanks to all my colleges at LiU for all the unforgettable geek-coffee brake discussions. Thanks also to all Flumes flight group members Raghu Munjuruly, Tomas Melin and David Lundström for all the interesting meetings, debates, and collaborations!

I also want to thank the people at Saab who were involved in this research; Mats Bergman and Pål Sarwe for their patience in guiding me through my first research project and the members of the Conceptual Aircraft Design Department, namely Kristian Amadori, Patrick Berry, Peter Furenbäck and Christopher Jouannet for their close collaboration. It was a pleasure to work together with all of you!

Warm thanks also to the whole open-minded CEAS Technical Committee Aircraft Design (TCAD) society, especially Prof. D. Scholz and the guys from DLR, namely Björn Nagel and Daniel Böhnke; it was a pleasure to meet you all over the world for a beer or two and talk Hamburger-Bavarian language: Moin moin und Grüß Gott!

Thanks also to my brother for all the sports adventures and challenges; we finally survived;-)

My greatest gratitude, finally goes go to my parents, who supported and encouraged me in any matter, sports career, hobbies, and education. Thank you so much for your endless love and for believing in me!

Linköping, October 2016,

A handwritten signature in black ink, reading "Ingo Staack". The script is cursive and fluid, with the first name "Ingo" and last name "Staack" clearly distinguishable.

Ingo Staack

# Abbreviations

6DOF	Six Degrees of Freedom
ACD	Aircraft Conceptual Design
AOG	Aircraft on Ground
ASCD	Aircraft Systems Conceptual Design
ATA 100	Air Transport Association Specification 100
BLT	Block Lower Triangular
BPR	Bypass Ratio
C-A net	Channel-Agency Net
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CPACS	Common Parametric Aircraft Configuration Schema
DAL	Design Assurance Level
DMM	Domain Mapping Matrix
DOC	Direct Operating Costs
DOM	Document Object Model
DSE	Degree of Subsystem Electrification
DSM	Dependency Structure Matrix
ECS	Environmental Control System
EMC	Electro-Magnetic Compatibility
EMI	Electro-Magnetic Interference
EPDS	Electric Power Distribution System
ETOPS	Extended Operation

FCS	Flight Control System
FMEA	Failure Mode and Effects Analysis
FMI	Functional Mockup Interface
FTA	Fault Tree Analysis
GA	Generic Algorithm
GeXF	Graph Exchange XML Format
GUI	Graphical User Interface
HTML	HyperText Markup Language
IC	Integrated Circuit
IDE	Integrated Development Environment
KB	Knowledge Base
KBC	Knowledge Base Component
KBE	Knowledge-Based Engineering
KBS	Knowledge Base System
MBCA	Model-Based Component Acquisition
MBSE	Model-Based Systems Engineering
MCE	Motor Control(ler) Electronics
MDDSM	Multi-Domain Dependency Structure Matrix
MDM	Multiple Domain Matrix
MDO	Multidisciplinary Design Optimisation
MEA	More Electric Aircraft
MTBF	Mean Time Between Failure
OML	Outer Mold Line
OOP	Object-Oriented Programming
OWL	Ontology Web Language
PCS	Power Consumption System
PDS	Power Distribution System
PFCS	Primary Flight Control System
PGS	Power Generation System
R&D	Research and Development

RDF	Resource Description Framework
SE	Systems Engineering
SFC	Specific Fuel Consumption
SHC	Spare Holding Costs
SoS	Systems of Systems
SVD	Singular Value Decomposition
SVG	Scalable Vector Graphics
TCM	Technology Capability Matrix
TLM	Transition Line Model
TMS	Thermal Management System
TRL	Technology Readiness Level
UAV	Unmanned Aerial Vehicle
VLM	Vortex Lattice Method
XHTML	XML conformal HTML
XML	Extensible Markup Language
XSD	Extensible Markup Language (XML) Schema Definition
XSLT	Extensible Stylesheet Language Transformation





# Tools, Software, and Programming Languages

Amesim	Multi-domain Systems Simulation Software
ASTRID	Aircraft On-Board Systems Sizing and Trade-Off Analysis in Initial Design Tool (Matlab, Politecnico di Torino)
BeX	Aircraft Conceptual Design Sizing Tool (Excel)
CADLab	Conceptual Aircraft Design Laboratory (tool suite)
CATIA®	3D CAD Environment, Dassault Systems
Doxygen	Documentation Tool (C++)
Eclipse	Integrated Development Environment (Java)
Excel	Electronic Spreadsheet Program
FreeMind	Mind-mapping Software (Java, open-source, based on XML)
Gephi	Graph/Network Visualisation and Analysis Platform (Java, open source)
Hopsan	Simulation Environment for Fluid and Mechatronic Systems (C++)
Matlab®	Matrix Laboratory (Mathworks)
Modelica	Multi-Domain Modelling Language (open source)
openCASCADE	3D Modelling Kernel (C++, open-source)
openVSP	Vehicle Sketch Pad (C++, NASA, open-source)
oXygen	XML/XSLT Editor, Debugger and Profiler (Java)

Python	Dynamic Object-Oriented Programming Language (open source)
RAPID	Robust Aircraft Parametric Interactive Design (based on CATIA <sup>®</sup> )
Simulink	Graphical Programming Environment (integrate with Matlab <sup>®</sup> )
SUMO	Open Source Surface Modeller
SysML	Systems Modelling Language
Tango	Aircraft Systems Conceptual Design Tool (Matlab <sup>®</sup> )
Tornado	Vortex Lattice Method (VLM) Tool (Matlab <sup>®</sup> , open source)
UML	Unified Modelling Language
VAMPzero	Aircraft Conceptual Sizing Tool (Java, open source, DLR)
VB	Visual Basic: Event-Driven Programming Language (Microsoft <sup>®</sup> )
VBA	Visual Basic Application (in Microsoft <sup>®</sup> Excel)

# Contents

- 1 Introduction 1**
  - 1.1 Dissertation Outline . . . . . 2
    - 1.1.1 Detailed Outline Description . . . . . 3
  - 1.2 Thesis Domain . . . . . 5
    - 1.2.1 Delimitations . . . . . 6
  - 1.3 Research Method . . . . . 7
    - 1.3.1 Research Environment and Related Projects . . . . 7
    - 1.3.2 Research Questions . . . . . 8
    - 1.3.3 Research Approach . . . . . 9
  - 1.4 Contribution . . . . . 9
    - 1.4.1 Publications . . . . . 10
  - 1.5 Remarks . . . . . 11
- 2 Design Influence on Aircraft Performance 13**
  - 2.1 Aircraft Performance and Efficiency . . . . . 13
    - 2.1.1 Historic Trends and Future Estimates of Aircraft Fuel Efficiency . . . . . 13
    - 2.1.2 Fuel Economics – Aircraft Design and Operations 15
  - 2.2 Performance Contribution of On-Board Systems . . . . . 16
    - 2.2.1 Power Off-Takes Efficiency . . . . . 18
    - 2.2.2 Growth Factor . . . . . 19
- 3 The Conceptual Aircraft Design Process 21**
  - 3.1 Process Characteristics . . . . . 21
  - 3.2 Information Model . . . . . 23
    - 3.2.1 Parametric Design . . . . . 24
    - 3.2.2 Namespace Hierarchy and Data Structure . . . . . 24
  - 3.3 Domain-Specific Tool Implementation . . . . . 26

3.3.1	Geometry Domain Integration . . . . .	27
3.3.2	Causal versus Acausal Implementation . . . . .	28
<b>4</b>	<b>Cascaded Systems Design</b>	<b>31</b>
4.1	System Nomenclature . . . . .	31
4.1.1	Inter-System Relationships . . . . .	32
4.1.2	System Properties . . . . .	33
4.2	System Design Drivers . . . . .	34
4.2.1	System Performance and Efficiency . . . . .	34
4.2.2	System Faults, Safety and Reliability . . . . .	36
4.3	Technology Classification . . . . .	37
4.3.1	Natural Order of System Technologies . . . . .	38
4.3.2	Technology Scalability . . . . .	40
4.4	The Power and Signal Component Concept . . . . .	41
4.4.1	Power Components . . . . .	41
4.4.1.a	Energy Transformations . . . . .	41
4.4.1.b	Power Control/Energy Adaption . . . . .	43
4.4.1.c	Exergy . . . . .	46
4.4.2	Signal Components . . . . .	46
4.5	Cyber-physical Systems . . . . .	48
<b>5</b>	<b>Model-Based System Design</b>	<b>49</b>
5.1	The Use of Models . . . . .	49
5.1.1	Model Abstraction . . . . .	49
5.1.2	Early Adopters . . . . .	50
5.2	Model Types . . . . .	50
5.2.1	Physical Models and Simulation . . . . .	52
5.2.2	Computational Simulation Models . . . . .	55
5.2.2.a	Model Fidelity and Computational Efforts	56
5.2.2.b	ACD Tool Fidelity . . . . .	57
5.2.2.c	Uncertainty Handling . . . . .	59
5.3	Engineering System Design: Process Integration . . . . .	60
5.3.1	Model Semantics and Ontology . . . . .	61
5.3.1.a	The Semantic Web Approach . . . . .	62
5.3.1.b	XML as a Universal Exchange Format . . . . .	62
5.3.2	System Knowledge in Knowledge-Based Engineering	64
5.3.3	Visual Model Representations . . . . .	66
5.3.3.a	Human Cognitive Aesthetic . . . . .	66
5.3.3.b	Graphic Tool Implementation . . . . .	67

<b>6</b>	<b>System Modelling</b>	<b>69</b>
6.1	KBE Model Translation . . . . .	69
6.2	Dependency Structure Matrix (DSM) . . . . .	71
6.2.1	Nomenclature and Notation . . . . .	72
6.2.2	Sorting and Clustering . . . . .	73
6.2.2.a	Sequencing/Partitioning . . . . .	76
6.2.2.b	Clustering . . . . .	76
6.2.2.c	Algorithm Limitations . . . . .	79
6.3	Multi-Domain Dependency Structure Matrices (MDDSM)	82
6.4	Graph Theory and Representation . . . . .	83
6.5	Channel Agency Networks . . . . .	85
6.5.1	Modelling with C-A Nets . . . . .	86
6.5.1.a	C-A Net Channel Definition . . . . .	87
6.5.1.b	C-A Net Agency Definition . . . . .	88
6.5.2	Extending the C-A Net model . . . . .	89
<b>7</b>	<b>Example Model Implementations</b>	<b>91</b>
7.1	Usecase1: Conceptual Aircraft Design Framework . . . . .	93
7.1.1	CADlab Framework Overview . . . . .	93
7.1.1.a	Framework Design . . . . .	94
7.1.1.b	Communication and Database . . . . .	94
7.1.2	Tool Implementation . . . . .	95
7.1.2.a	Tango Implementation . . . . .	95
7.1.2.b	RAPID Implementation . . . . .	96
7.1.3	The CADLab Database Namespace . . . . .	98
7.1.4	Usecase Conclusions . . . . .	98
7.2	Usecase2: KBE-Based Simulation Model Integration . . .	100
7.2.1	Aircraft Hydraulic System Topology . . . . .	100
7.2.2	Statistical Analysis of Hydraulic FCSs . . . . .	101
7.2.3	KBE Hydraulic System Implementation . . . . .	103
7.2.4	Cascaded Systems Simulation Metamodel Implementation . . . . .	105
7.2.5	Implementation of the Hopsan Metamodel . . . . .	107
7.2.6	KBE Process on the Metamodel . . . . .	108
7.2.6.a	Hydraulic Actuator Model Definition . . .	109
7.2.6.b	Simulation Model Instantiation . . . . .	110
7.2.7	Usecase Conclusions . . . . .	112
7.3	Usecase3: Graph-Based Modelling . . . . .	115
7.3.1	DSM Modelling Process . . . . .	117

7.3.1.a	Implicit and Explicit Model Formulation	117
7.3.1.b	Model Representations . . . . .	118
7.3.2	System Partitioning and Component Placement . .	119
7.3.2.a	GUI-Based Graph Sorting and Clustering	120
7.3.3	FTA Modelling . . . . .	121
7.3.4	Graphical C-A Net Modelling Approach . . . . .	122
7.3.4.a	GUI-Based C-A Net Implementation . . .	123
7.3.4.b	XML-Based C-A Net Implementation . .	123
7.3.5	Usecase Conclusions . . . . .	124
<b>8</b>	<b>Discussion and Conclusion</b>	<b>125</b>
8.1	Discussion . . . . .	125
8.2	Detailed Discussion . . . . .	126
8.3	Answers to the Research Questions . . . . .	132
8.4	Conclusion . . . . .	134
8.5	Future Work . . . . .	135
	<b>Bibliography</b>	<b>137</b>
	<b>Index</b>	<b>153</b>

# 1

## Introduction

Despite great achievements in research on product development, management, and systems engineering, and the vast increase in computational power, the overwhelming increase in complexity of both civil and military aircraft design undermines the design process, resulting in longer development times and higher development costs.

In the standardized product development process, following the product-V development cycle [EIA632, 2003], a common design process dilemma is the limited knowledge in early development process stages, where important (design) decisions have to be made that influence product and development costs as well as the product's performance and properties [Ullman, 2002], [Weilkiens et al., 2015]. To address this problem, mainly three counteracting processes can be chosen: (a) delay (important) design decisions until more knowledge is available, (b) increase the available product related information in the early design stages, or (c) to enable for efficient or automatic execution with different input setups.

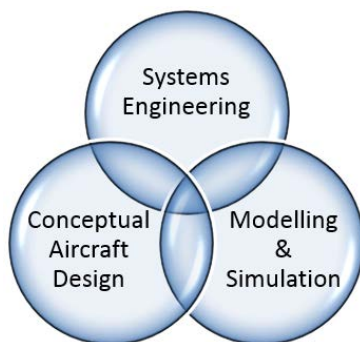
In Aircraft Conceptual Design (ACD), the main issue is the lack of detailed product/aircraft properties which are required as inputs for the analytic tools. On the way to new More Electric Aircraft (MEA) architectures, with greater influence on the power generation and potentially fusion of the propulsion and power generation system in future hybrid electric designs, these systems cannot be neglected in the ACD phase.

Traditional semi-empirical on-board system designs do not comply with the higher analysis fidelity demand and do not fit to the higher-fidelity physics-based analysis methods. Furthermore, new technologies and never before seen system architectures cannot be satisfactorily ad-

dressed by semi-empirical formulas based on traditional system layouts.

Paving the way to include on-board system design within ACD – hereinafter called Aircraft Systems Conceptual Design (ASCD) – requires good ways of modelling to achieve a smooth and efficient model integration with a minimum of effort. For this purpose, Model-Based Systems Engineering (MBSE) and Knowledge-Based Engineering (KBE) methods have been applied to the ACD phase as described in this thesis.

## 1.1 Dissertation Outline



**Figure 1.1** *The three main thesis domains*

The thesis consists of three main parts:

- I The **aircraft conceptual design phase analysis part**, focusing on the specific characteristics and challenges of this design phase (Chapter 3). To justify the conceptual design phase analysis, a short analysis of aircraft performance is given in the preceding Chapter 2.
- II A **theory part, dealing with system design and system modelling** in Chapter 4–6. This section serves as the foundation of model implementation and advanced tool framework design.
- III **Application part**: Three example applications are described in Chapter 7, performed on the basis of the theoretical parts.

A detailed conclusion (Section 8.2 on page 126) including extensive use of cross-references to the related chapters may give a short path to identify relevant parts of this work for the reader.



### 1.1.1 Detailed Outline Description

The ACD analysis of the ACD is based on the aircraft performance measures and technology trends shown in **Design Influence on Aircraft Performance**, Chapter 2. In addition to the usual performance measures such as (fuel consumption) efficiency, particular focus is given to the performance influence of the on-board power systems (see Section 2.2), including the energy supply concepts (power off-takes) of jet-engine driven aircraft.

Chapter 3 **The Conceptual Aircraft Design Process** explores the demands of the ACD phase concerning tool framework integration, implementation strategies and the work process analysis. After the theoretical part, this topic will be picked up again in the Usecase1 example in Section 7.1, showing the implementation of an ACD framework, based on the findings in this and the following chapters.

After the ACD analysis, the necessary background to solve the identified tasks within ACD is discussed in Chapter 4–6. The theoretical part is divided into three main topics: system design and technology (Chapter 4), model-based system design (Chapter 5), and modelling strategies in Chapter 6. The knowledge, theories and processes discussed in these three theoretical chapters constitute – together with the ACD analysis from Chapter 3 – the foundation of the following application examples in Chapter 7.

Chapter 4 **Cascaded Systems Design** deals primarily with the analysis of different system design drivers. It presents how a systematic component analysis can be used to analyse, describe, enhance, and create systems. From the perspective of aircraft on-board power systems, the concept of **power** and **signal** components is presented (see Section 4.4). This concept allows for a systematic system build-up concerning both system architecture and component selection. Furthermore, it backs the modelling strategies presented in the following chapters well.

Chapter 5 **Model-Based System Design** focuses on the analytical models within any MBSE approach. Different model categories are presented (Section 5.2) and strategies for model management and process integration are detailed (in Section 5.3). Particular emphasis is given to addressing tool fidelity (which were earlier in Chapter 3 identified as an important topic in ACD), which hampers the desired step-less modelling and analysis fidelity increase with the development progress (see Figure 5.8). Highlighting future prospects, a semantic-based approach is shown (Section 5.3.1.a), and Extensible Markup Language (XML) is

identified as a useful implementation strategy on the road towards future unified modelling approaches (see Section 5.3.1.b). The chapter concludes by highlighting the importance of a graphical model and modelling representation (Section 5.3.3).

Placed between the theoretical and application parts, Chapter 6 **System Modelling** deals mainly with structural system modelling. Starting with the classic Dependency Structure Matrix (DSM) approach (Section 6.2), the similarity of different system modelling approaches is highlighted, resulting in a unified approach that describes a system as a graph set. One of the main topics of this chapter is the sorting and clustering of system components by their (multi-domain) relationships (Section 6.2.2).

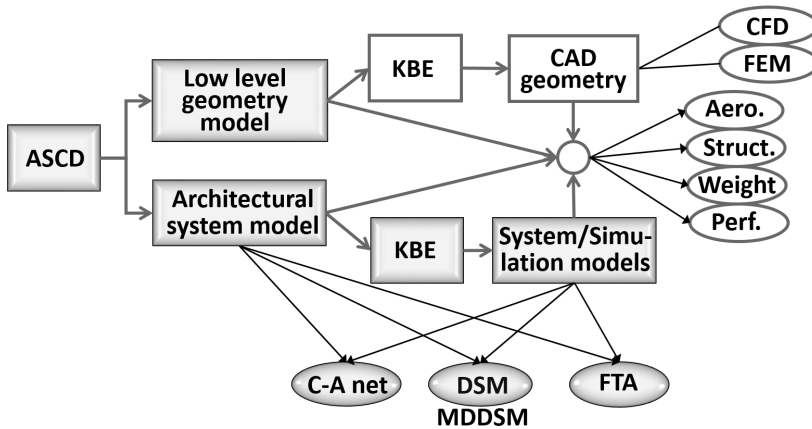
The **Example Model Implementations**, Chapter 7, shows the application of the topics discussed in the thesis. The following application use cases are presented in Sections 7.1–7.3:

**Usecase1:** Conceptual Aircraft Design Framework Development

**Usecase2:** Knowledge-Based Simulation Model Integration within Conceptual Aircraft Design

**Usecase3:** Graph-Based System Driven Design Modelling

After the use cases examples, a short conclusion is given in Chapter 8, followed by detailed conclusions in Section 8.2, including extensive use of cross-references. A schematic overview of the thesis domains can be found in Figure 1.2.



**Figure 1.2** Schematic overview of the ASCD process presented in this thesis. The thesis topics are shown highlighted in colour.

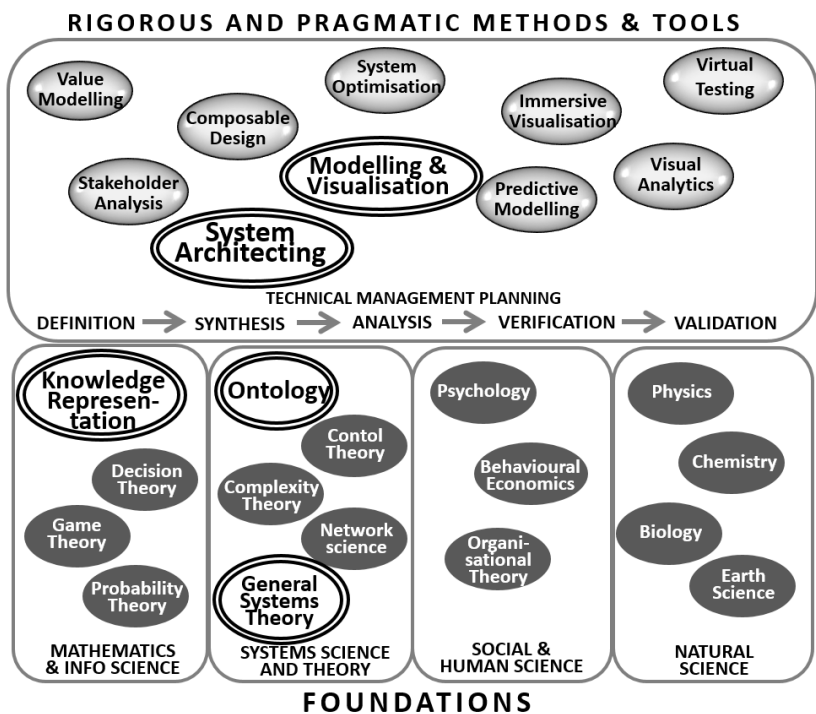
## 1.2 Thesis Domain

The scope of the thesis includes three main parts, conceptual aircraft design, modelling & simulation and systems engineering (design) (see Figure 1.1). The main focus on the former two, with the necessity to incorporate systems engineering (design) for both the modelling & simulation approach as well as the system architecture design.

The overall design process is aligned with the system engineering engine approach developed by [NASA, 2007], vaguely adopting the (product) development-V scheme [EIA632, 2003], starting with stakeholder and requirement identification, system decomposition and system integration with validation and verification.

Categorising the thesis domains from the application domain independent system engineering point of view of the International Council on Systems Engineering (INCOSSE), the thesis addresses the following areas (see Figure 1.3):

- systems architecture and modelling & simulation, on the foundation of
- knowledge representation and ontology.



**Figure 1.3** Thesis domains (marked with double edges) within the systems engineering view of the INCOSE society (adapted from [INCOSE, 2014])

### 1.2.1 Delimitations

ACD is an extreme representation of a Multidisciplinary Design Optimisation (MDO) problem. For the sake of simplicity, the topics manufacturing, cost and maintenance are excluded from the thesis; the reasons for these decisions can be found in Table 1.1.

Topic	Motivation
manufacturing	<ul style="list-style-type: none"> <li>• fidelity level in ACD too low: no/weak coupling between design and production</li> <li>• large scale effects</li> <li>• company-dependent infrastructure and expertise.</li> </ul>
cost	costs include many “soft factors”; highly dependent on the topic <i>manufacturing</i> (see above).
maintenance	<p>The maintenance strategy (e.g. preventive, predictive, run-to-failure or reliability centred maintenance) has indubitably an influence on system design. Maintenance has been excluded from the scope of this thesis in order to (a) simplify the design process (excluding this domain) and (b) the lack of maintenance-related design information such as:</p> <ul style="list-style-type: none"> <li>• cost: replacement/refurbish, etc. costs</li> <li>• mode of operation (preventive, corrective or reliability-centred maintenance)</li> </ul> <p>These topics are also related to the <i>cost</i> topic, excluded for the above-mentioned reasons.</p>
no advanced control theory/concepts	Control and behavioural modelling has been limited in this work to the minimum needed to model cyber-physical systems which inevitably include both structural and behavioural models. The reason is the different approach in behavioural modelling, which is outside the scope of this work.

**Table 1.1** *Excluded (design) topics and the reasons for their exclusion*

## 1.3 Research Method

### 1.3.1 Research Environment and Related Projects

The work evolved over the years from a Model-Based Development (MBD) approach for an aircraft conceptual design framework towards a Systems Engineering (SE) inspired Model-Based Systems Engineering (MBSE) approach to achieve the needs and challenges required for multidisciplinary ACD assessment. It is based on four research projects

with different topics ranging from physical prototype investigation and simulation-based system architecture development to conceptual aircraft design and systems engineering design. All the projects were within the National Aviation Engineering Research Programs (NFFP) funded by VINNOVA, the Swedish R&D government agency, working under the Ministry of Enterprise and Innovation [VINNOVA, 2016].

<b>Program</b>	<b>Project Name</b>	<b>Year</b>	<b>Topic</b>
GFF	GFF Demonstra- tor	2009	subscale physical prototypes
NFFP4 Clean Sky SFWA	FLUD / EDOC	2010- 2013	physical system simulation; thermal management system; electric ECS architecture; FMI/FMU co-simulation; simulation automation
NFFP5 2010-1251	Konceptmetodik	2011- 2014	conceptual aircraft design; parametric data structure
NFFP6 2014-0927	CADLAB	2015- 2016	CAD integrated ACD; para- metric design; (aircraft) sys- tem design

**Table 1.2** *Research project overview*

### 1.3.2 Research Questions

The research is based on the following research questions that were addressed within the above-listed research projects. The answers to the research questions can be found in Section 8.3 on page 132.

**RQI** Which tool-principles and type of information model suit the (multi-domain and multi-aspect) conceptual aircraft design process?

**RQII** Can Knowledge-Based Engineering (KBE) be applied in system architecture design to enhance and back the MBSE approach within conceptual design?

**RQIII** How can the different aspects of system design be supported by appropriate modelling?

### 1.3.3 Research Approach

The research questions originate from industrial needs that triggered the research projects listed above. Some of these questions arose during the study of the initial research questions. These follow-up questions are of two categories: They were either needed directly to answer the initial problem statement or they represent interesting aspects or alternative solutions that were found during the investigations. The latter were extremely fruitful to avoid any kind of inbreeding in the research within the conceptual aircraft design community only and enlarged the fields of study (to be in contact with) to additional branches such as fluid power, mechatronic systems and system engineering. Additionally, collaborations with researchers from outside Europe – namely Brazil – contributed to a wider problem decomposition due to their different social background and perception. These cooperations triggered an enlargement of the possible solution design space (involving theories, methods, processes and tools). One example is the Channel-Agency Net (C-A net) modelling approach (see Section 6.5.1).

**Validation** of the research (outcomes) was done in trial by implementations, a pragmatic but reasonable way.

**Verification** of the research outcomes – if applicable – was (partly) conducted by the industry partner in the research projects listed earlier in Table 1.2.

## 1.4 Contribution

The most significant contribution – from the author’s point of view – is the abstract analysis of the (conventional, state-of-the-art ()) multi-aspect conceptual aircraft design process with respect to software and systems engineering. The traditional mechanical engineering approach – mainly focusing on aerodynamics, structure and propulsion – is extended with system engineering perspectives to address the design challenges of future, highly-integrated, more electrical on-board power system designs. Using object-oriented techniques and multi-aspect modelling, a holistic design approach seems to be realisable. Other contributions of this work include:

- The transition from conventional ACD towards a multi-domain ASCD approach encompassing Systems Engineering (SE) and Software Engineering (global, whole thesis)

- The similarity findings between Multi-Domain Dependency Structure Matrix (MDDSM) and C-A net modelling (Sections 6.5–6.5.2)
- The definition of an XML-based information model with a careful universal aircraft geometry parametrisation (Section 3.2, Usecase1 in Section 7.1 and [VII])
- The KBE-based approach for system configurations (Chapter 6 and Usecase2 in Section 7.2)
- The automated integration of simulation models using KBE methodology into the design process (Section 6.1, Usecase2 in Section 7.2 and [IV])
- The extended use of XML, XSD, and XSLT for model support, enabling CAD and system integration using XML-based parametric modelling.
- The similarity detection by MDDSM and C-A net. This model analogy enables the unified handling of these modelling concepts as one graph model in combination with the OOP implementation principles (Usecase3 in Section 7.3)
- The unified graph-based modelling approach (Usecase3 in Section 7.3)

### **1.4.1 Publications**

This monograph is based on – but not limited to – the following publications by the author:

- I** Staack, I., D. Lundström, and P. Krus (2010). “*Subscale Flight Testing at Linköping University*”. In: *27th International Congress of the Aeronautical Sciences*, ICAS, Nice, France.
- II** Staack, I., H. Ellström, M. Bergman, P. Sarwe, and P. Krus (2012). “*More Electrical Environmental Control System Simulation*”. In: *Proceedings of the 28th International Congress of the Aeronautical Sciences*, ICAS, Brisbane, Australia.
- III** Staack, I., R. C. Munjuruly, P. Berry, T. Melin, K. Amadori, C. Jouannet, D. Lundström, and P. Krus (2012). “*Parametric Aircraft Conceptual Design Space*”. In: *Proceedings of the 28th Inter-*



*national Congress of the Aeronautical Sciences*. ICAS, Brisbane, Australia.

- IV Staack, I. and P. Krus (2013). “*Integration of On-Board Power Systems Simulation in Conceptual Aircraft Design*”. In: *Proceedings of the 4th CEAS European Air & Space Conference*. Council of the European Aerospace Societies (CEAS). Linköping University Electronic Press, 2013, p. 709.
- V Staack, I., R. C. Munjulury, T. Melin, P. Krus, and A. Abdalla (2014). “*Conceptual aircraft design model management demonstrated on a 4th generation fighter*”. In: *29th Congress of the International Council of the Aeronautical Sciences*. ICAS, St. Petersburg, Russia.
- VI Staack, I., P. Krus (2017). “*Integration of Aircraft On-Board Systems Simulation in Aircraft Systems Conceptual Design*”. In: *CEAS Aeronautical Journal*. Submitted

Additional publications with contribution by the author:

- VII Munjuruly, R. C., I. Staack, P. Krus, and P. Berry (2016). “*A knowledge-based integrated aircraft conceptual design framework*”. In: *CEAS Aeronautical Journal* 7.1, pp. 95–105. issn: 18695590.
- VIII Scholz, D., R. Seresinhe, I. Staack, and C. Lawson (2013). “*Fuel Consumption Due to Shaft Power Off-takes from the Engine*”. In: *Proceedings of the 4th International Workshop on aircraft System Technologies (AST)*. Hamburg, Germany, pp. 169–179.
- IX Jouannet, C., P. Berry, T. Melin, K. Amadori, D. Lundström, and I. Staack (2012). “*Subscale Flight Testing used in Conceptual Design*”. In: *Aircraft Engineering and Space Technology* 84.3, pp. 192–199.

## 1.5 Remarks

The thesis applies the standard nomenclature that is currently used within the European aviation community. To make the formulations more precise, some deviations from the standard terms are made. The following listing explains these alterations, which help to render the weak

semantic of written language into a more precisely defined use of that terms within this thesis:

- The symbol **&** is used for set expressions like **modelling and simulation** (written: **modelling & simulation**) for easier readings.
- Instead of differentiation between **aircraft** and **aeroplane**, the term **aircraft** is used throughout the thesis. This is done to highlight that the system & modelling theory and processes (Sections 6.5–6.5.2) are not limited to aeroplanes only. However, Chapter 2, Chapter 3, and all implementation work have been applied on fixed-wing aeroplanes only.
- The theoretical chapters, Chapter 4 - Chapter 6 are mainly application domain independent (but in this work related to aircraft design). Consequently, the word **aircraft** may be substituted by the word **product** in these chapters.
- Instead of the term **systems of systems (SoS)** for a composition of (sub-)systems within a project, product or network, the term **cascaded systems** is used. SoS is used in the sense according to the definition by [L. Hu et al., 2015] or [Maier, 1998] (for further explanation see Section 4.1.1).
- In Section 4.4.1 “*Power Components*” and following chapters, the term energy **transformation** is used for a transformation from one energy domain (e.g. electric) into another domain (e.g. hydraulic). A transformation thus involves two energy domains. In contrast, the term **adaption** is used for a single domain energy alteration such as adjustment of pressure or flow level.
- In imitation of the **use case diagram** in UML, the term **use case** is used in this work to denote the application examples **Usecase1** to **Usecase3** in Chapter 7.

# 2

## Design Influence on Aircraft Performance

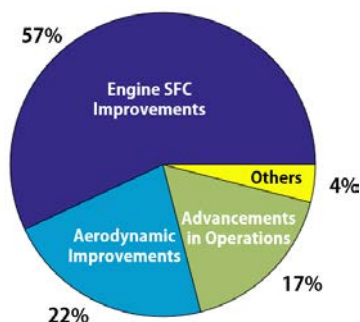
*This Chapter identifies the influence of design (parameter) on aircraft performance, focusing on total fuel consumption. The performance and efficiency of (civil) aircraft are investigated with particular emphasis on the influence on performance of the on-board power system's design. The importance and contribution to performance of the on-board systems is highlighted and the relationship between (system) weight, drag, and energy efficiency is analysed. It serves as an introduction to conceptual design process analysis in Chapter 3.*

### 2.1 Civil Aircraft Performance and Efficiency

To focus on the right topics within any product development, one has to have a clear picture of the design influences, thus the sensibilities between the design parameters (and eventually the uncertainty parameters) and the design benchmark. To analyse trends and the impact of new technologies (like experience gained, changed social and environmental conditions and politics), design history trend analyses are useful.

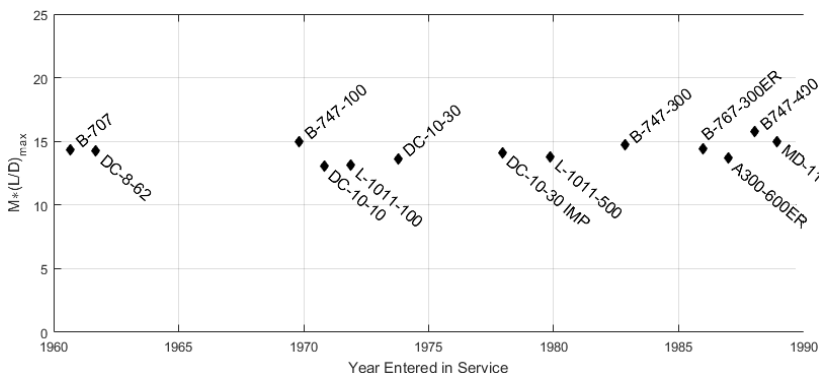
#### 2.1.1 Historic Trends and Future Estimates of Aircraft Fuel Efficiency

In recent decades, energy consumption reduction has mainly been based on engine efficiency, aerodynamic improvements and more efficient aircraft usage [Lee et al., 2001], see Figure 2.1.



**Figure 2.1** Advancements split-up of US commercial fleet improvements between 1969 and 1995 (data from [Lee et al., 2001])

Whereas the improvement trend is clearly visible in the Specific Fuel Consumption (SFC) enhancement, other operational characteristics seem to stagnate. Examples include the structural weight fraction (OEW/MTOW) – counteracted by increasing mission distances – or the  $M^*(L/D)$  measure (shown in Figure 2.2), counteracting the aerodynamic enhancements from a slight increase in the design Mach number. How-



**Figure 2.2** Long-haul airliner drag evolution between 1960 and 1990 (adapted from [La Rocca, 2011])

ever, the stagnation of certain parameters does not necessarily mean that no (technology) advancement occurred but may be penalised by the mode of operation or cross-coupling effects of other design changes. Examples of the latter type include:

- structure penalty: additional weight due to the negative influence of higher aspect ratio wings, lower chord thickness and more aero-

dynamic shaping

- weight penalty: due to additional on-board power systems such as gust alleviation and load distribution system<sup>1</sup>, enhanced comfort (cabin pressure, noise shielding), heavier high Bypass Ratio (BPR) engine installation, possibly additional weight due to larger landing gear (ground clearance)
- drag penalty: higher drag due to the larger (high BPR) engine and nacelle size.

Only the propulsion SFC value is an independent system characteristic<sup>2</sup>, almost independent of the other design parameters but with a huge impact on the weight and aerodynamic characteristics of an aircraft.

### 2.1.2 Fuel Economics – Aircraft Design, Operation and Equipment

A fleet fuel efficiency analysis of the top largest airlines operating transatlantic passenger flights (between Europe and North America) shows a tremendous difference in fuel efficiency of up to 51% excess fuel per passenger kilometre between the best (Norwegian, mostly Boeing 787) and the worst (British Airways, mostly Boeing 747-400) competitor [Kwan et al., 2015].

$$\begin{aligned} \frac{paxkm}{L} = & -24.745 - 33.737 * (RL \text{ margin}) \\ & + 21.691 * (seat \text{ density}) + 33.897 * (pax \text{ loadfactor}) \\ & + 37.730 * (freight \text{ share}) \end{aligned} \quad (2.1)$$

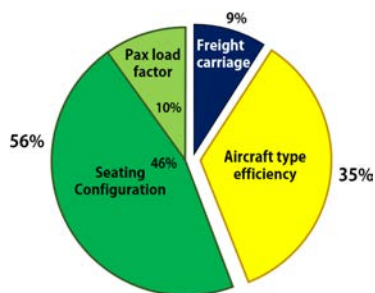
where

- $L$  the fuel burn in litres.
- $RL$  the aircraft-specific fuel burn, compared with the industrial reference line.

---

<sup>1</sup>One of the key factors enabling high aspect ratio wings as present on A350, B-787 and B-777X aircraft.

<sup>2</sup>Actually, the pure engine SFC is a component characteristic (of the engine) and not a system characteristic (of the aircraft). Taking into consideration the installation losses (inlet/outlet and interference), the SFC can be seen as an aircraft system characteristic.



**Figure 2.3** Fuel consumption key drivers, based on transatlantic fleet analysis (adapted from [Kwan et al., 2015])

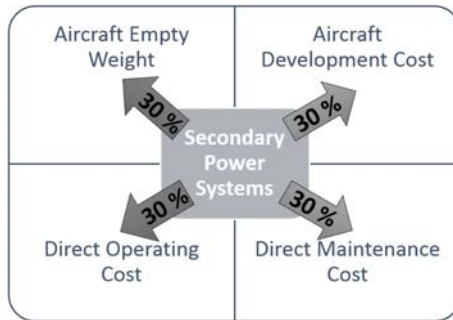
This fuel consumption analysis is based on scheduled flight data which were fed into a (0-dim, refer to Section 5.2.2.a, page 56) simulation tool. The findings of this study show that the major contribution to fuel efficiency is related to cabin efficiency (aka seat density) and only 35% is affected by the aircraft technology used. Outliers are Air Berlin, operating relatively old Airbus A330-200 aircraft but with a dense seat configuration, and Norwegian, with both a high-density cabin layout and the youngest fleet of Boeing 787-800. Icelandair is rated precisely on the industrial average, operating single-aisle Boeing 757 aircraft with an average fleet age of 18 years but counterbalancing this penalty with the highest seat density of all competitors and a good load factor.

## 2.2 Performance Contribution of On-Board Systems

On-board power systems are necessary for performance, safety, controllability, and comfort in any type of aircraft [Chakraborty and Mavris, 2016]. Focusing on the energetic system properties, the on-board systems can be categorised as

- Power Generation Systems (PGS)
- Power Distribution Systems (PDS), and
- Power Consumption Systems (PCS)

Usual types of secondary on-board power are electric, hydraulic, and pneumatic power. The primary source of energy is almost any time jet-fuel is converted into thrust or power through a thermodynamic process. A high-level analysis by [Liscouët - Hanke, 2008] comes to the conclusion that roughly 30% of the aircraft weight, the development costs, the Direct Operating Costs (DOC), and the direct maintenance costs are contributed by the (on-board) secondary power systems (see Figure 2.4). System design optimisation methods and tools should be capable of taking into consideration these multi-domain aspects such as engineering process effort, physical system properties, fuel consumption and operator costs.



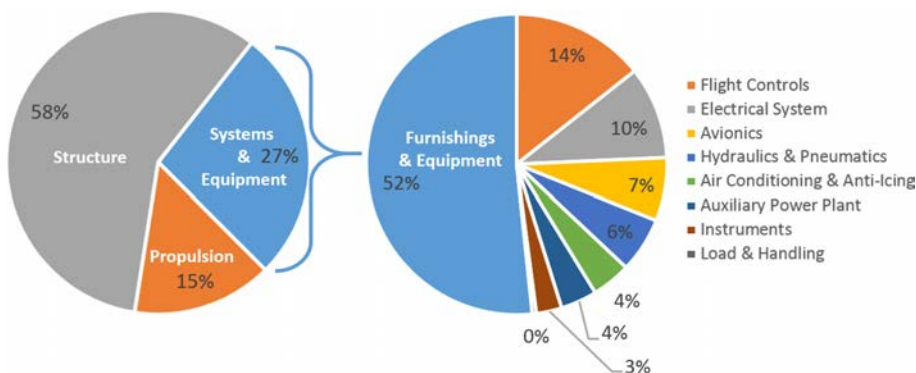
**Figure 2.4** On-board systems significance on aircraft weight and over-all costs (adapted from [Liscouët - Hanke, 2008])

Ignoring the cost side and focusing on energy consumption only, the right pay-off between (component/system) efficiency and weight – and possibly the impact on drag – has to be found. [Scholz, 1998] shows a system analysis process taking into consideration the (additional) fuel consumption due to:

- fixed mass variation
- variable mass variation (mainly fuel)
- engine power off-takes
  - bleed air off-takes
  - mechanical (shaft) power off-takes
- ram air off-takes

- additional drag

A weight breakdown of a Boeing 737-800 confirms the 30% rule stated by Liscouët - Hanke (see Figure 2.5). The figure also shows that approximately 50% of the systems & equipment weight is related to the on-board (power) systems, whereas the other 50% is attributed to cabin furnishing & equipment. These values are for civil transport aircraft, with even higher values of the on-board systems contribution on military aircraft (trainer, fighter, airlift, and tanker).



**Figure 2.5** Weight break-down of a Boeing 737-800 (data based on a weight estimation from [Bruner et al., 2010])

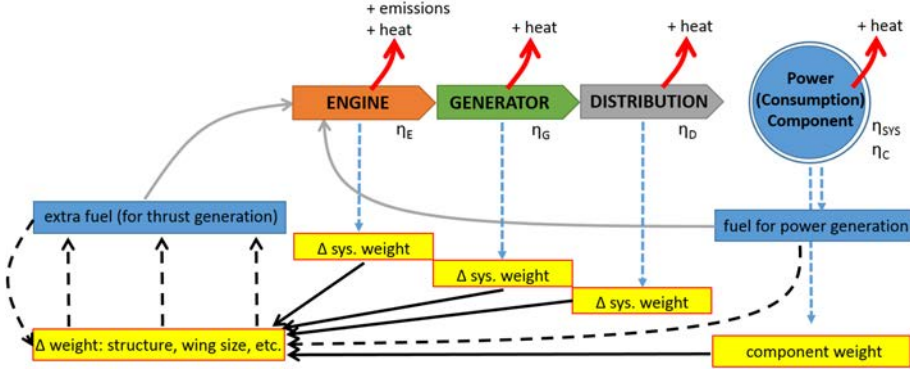
### 2.2.1 Secondary Power Off-Takes Efficiency

Figure 2.6 illustrates the scaling relationships of shaft power off-takes and (secondary) electric power consumption. Changes in the mechanical power off-takes directly impact on the fuel consumption but also have effects on the fuel to mechanical power efficiency. Indirectly it can also influence engine weight due to engine redesign (e.g. in engine core and accessory gearbox design). Different sources indicate an almost linear relationship between SFC and secondary power off-takes [Slingerland et al., 2007] [Dollmayer, 2007] [VIII]. However, a slightly exponential relationship is expected from the effects of the (optimal) operating point of the compressor and turbine through the power off-takes<sup>3</sup>. [Slingerland et al., 2007] show that concerning extracted exergy, bleed air off-takes

<sup>3</sup>For a more detailed power off-take investigation and explanation of the rescaling of the engine, see [Dollmayer, 2007].



have a less detrimental effect on SFC than shaft power off-takes have. For engine bleed air off-takes, an upper limit clearly exists with a de-



**Figure 2.6** Impact of electric power off-takes by power consumption systems/components on the vehicle growth factor: weight and efficiency scaling effects<sup>4</sup>.

clining limit for high BPR engines due to the low remaining core air mass flow [VIII]. The Degree of Subsystem Electrification (DSE) (see Section 4.3 on page 37) or, in other words, the kind of power extraction, may also influence the way of operating, as shown by [Seresinhe et al., 2013].

## 2.2.2 Growth Factor

Similar to the (component) efficiency scaling of a subsystem, the same principle is present on aircraft level, as shown in Figure 2.6. Additional subsystem weight(s) imposes a weight overhead from increased vehicle size, reinforced structure and additional fuel (weight), compensating for the extra fuel burn. The latter aspect is largely based on the mission profile and operation mode. For long-distance missions, fuel efficiency versus system weight becomes more important than for short-haul missions, which becomes visible in the design (e.g. braced-wing design for a short-haul aircraft, sacrificing the aerodynamic finesse in favour of structural weight). This effect has been already known since the advent of aviation, formulated by Luis C. Breguet in his famous range equation:

$$RANGE = \frac{Velocity}{TSFC} * \frac{L}{D} * \ln \left( 1 + \frac{W_{fuel}}{W_{payload} + W_{OEI}} \right) \quad (2.2)$$

For a certain aircraft configuration and mission profile, a growth factor due to weight or (propulsion or secondary energy) efficiency variation can be calculated [Ballhaus, 1955]. This (payload) growth factor represents the sensitivity of the overall design to a change in the payload requirement (with all the residual requirements fixed). Besides payload alterations growth factors for changes in range, installed thrust, or power are also common [Rugg, 1970]. These growth factors give useful feedback on the (cost) impact of the basic vehicle characteristics. Usual (payload weight) growth factors are around three for current airliners, with outliers for extreme aircraft designs such as long-endurance solar powered aircraft with a growth factor value of up to 4.5 [Ross, 2016]. A problem with growth factors – and in general with globally applied sensitivity analysis – is the irreconisable shadowing of the relationships between the design, the mode of operation, and the requirements.

---

<sup>4</sup>Compare this figure with Figure 4.9 on page 45.

# 3

## The Conceptual Aircraft Design Process

*This Chapter investigates the requirements, processes and tool capabilities needed for the conceptual aircraft design phase in detail. Based on the findings in this chapter, an example implementation of an ACD framework called CADLab is performed, see the Usecase1 example in Section 7.1.*

Initial sizing in the conceptual design stage is an iterative process, establishing a rough geometry based on significant requirements. The sizing process usually incorporates aerodynamics, weight & balance (structure), and propulsion system. In the classic approach, this constraint diagram – also denoted as the sizing diagram – is used to represent requirement fulfilment. With the help of this diagram the qualitative fulfilment/failure of the basic requirements can be displayed.

### 3.1 Process Characteristics

The following topics characterize the ACD phase:

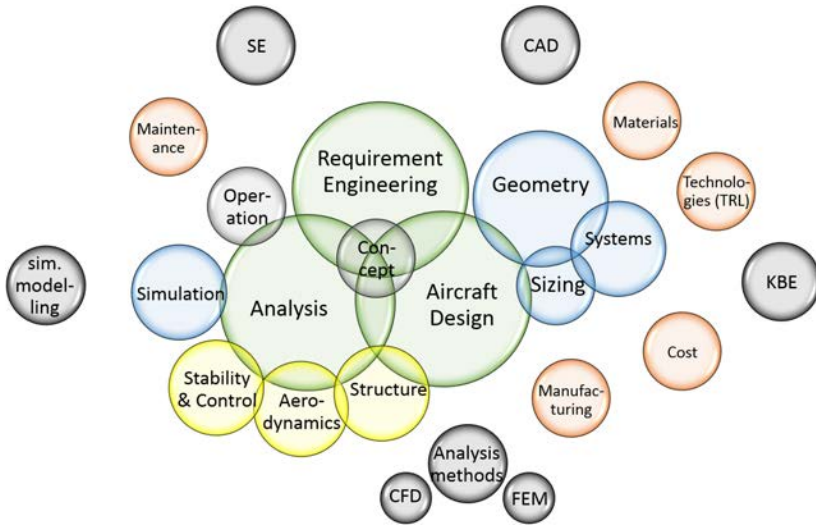
- **efficient:** short-term, low-efforts, "one man show"
- **flexible:** just-in-time adjustments for the project-/campaign-specific topics
- **transparent:** support the user in understanding the design features, design conflicts, and the limiting require-

ments

- **multi-modal:** enable automatic and manual mode for use within optimizations or manual design

Of all the topics mentioned above, ACD is mainly characterised by the efficiency of the process to find and estimate the performance of an aircraft configuration, based on fundamental requirements. These demands distinguish the ACD clearly from the subsequent preliminary design and detailed design phases. The main focus during ACD is usually on aerodynamics and structure to find the most suitable configuration (concept) for the given requirements. Besides the typical topics of structure, aerodynamic, and stability & control, many other subjects may also be of interest. Figure 3.1 gives a comprehensive overview of the different topics that may be addressed during initial design.

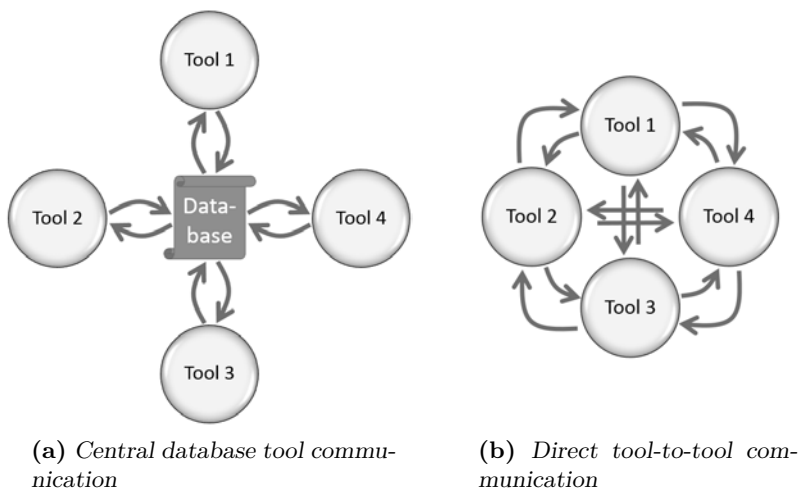
The principal characteristics of the conceptual design phase mentioned earlier – low-effort and short-term task, vague and incomplete requirement formulation, new emerging technologies and large uncertainties – distinguish it clearly from the preliminary or detailed design phases. As the effort should be low, it cannot be carried out by experts on each topic but may be addressed by one person or a small team. From this it follows that specific expert systems (such as e.g. DOORS [IBM, 2016] for requirement handling) may not be appropriate and are consequently not regularly applied.



**Figure 3.1** Selection of topics related to Aircraft Systems Conceptual Design (ASCD)

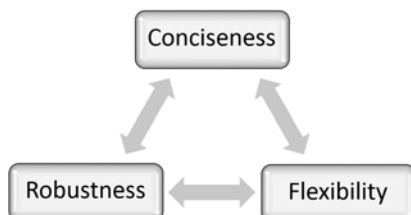
## 3.2 Information Model

A central database approach in an open (non-proprietary) format is favourable to make all the (project-) related data accessible for every application within a framework (see Figure 3.2). By this, only one data translation process per tool towards the database is needed. The drawback of this approach is that any tool-to-tool communication occurs via the database, incorporating four translations in a single tool-to-tool communication process. This requires a stable, “well-defined” translation process to avoid value runaways during iteration (e.g. within an optimisation process) by calculation (e.g. due to truncation) or translation inaccuracies. The latter may occur due to database incompatibilities with the tool format (leading to truncation errors due to different parameters in the tool and the database) or due to fidelity level mismatches. A common database – also referred to as a common namespace [Böhnke, 2015] – has to fulfil the needs and domain-specific aspects of the integrated tools. The development of such a common namespace and the related design and analysis routines (including automation) account for the main topics (in MDO) research [La Rocca and van Tooren, 2007].



**Figure 3.2** *Inter-tool communication strategies*

### 3.2.1 Parametric Design



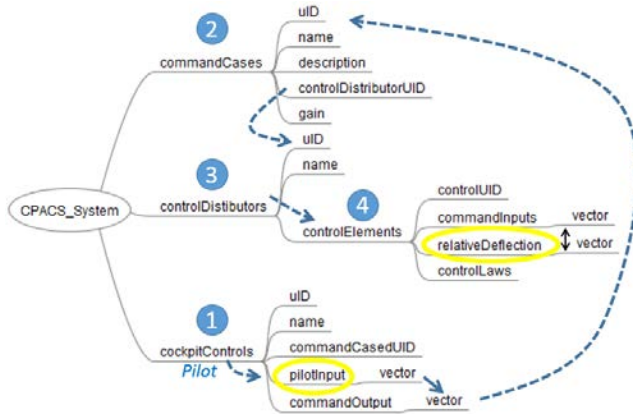
**Figure 3.3** *Characteristics of a good parametric design (according to [Sóbester et al., 2015])*

The foundation of a robust and interpretable database is the underlying parametrisation scheme. According to [Sóbester et al., 2015], a parametric design should focus on conciseness, robustness, and flexibility. Unfortunately, these are three contradictory targets, for which a good compromise has to be found (see Figure 3.3).

### 3.2.2 Namespace Hierarchy and Data Structure

Conceptual aircraft data usually do not include behavioural data, but some data formats include (static) functional or behavioural information within the data setup. One example is the pitch control path in

the Common Parametric Aircraft Configuration Schema (CPACS) data format [CPACS v.2.3, 2016], where the signal path from the pilot's stick deflection to the control surface deflection is implicitly included in the data setup (see Figure 3.4). This implicit relational information has somehow to be machine-interpretable documented, for both the user and the analysis tools (for more details see Section 5.3.1).



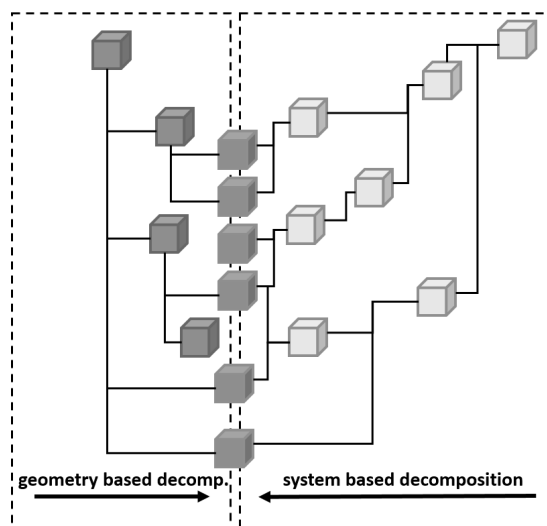
**Figure 3.4** Example of an implicit functional relation within a CPACS data format

An additional problem to be addressed is the hierarchy concept of the entire applications. A flat tool hierarchy, with each tool acting as a master, and no database update control, can result in the above-mentioned unintended value runaways as well as data inconsistency.

Within conceptual aircraft design, the major work tasks focus on sizing and placement of airframe entities and components, each representing a geometric task resulting in a definition of the Outer Mold Line (OML) shape and weight distribution. From this geometric model, basic (first order) analyses are conducted such as areas and wetted area calculation, (cross-section) area distribution and so forth. Deeper (2<sup>nd</sup> order) analysis such as drag estimation, structure analysis or more detailed on-board system analysis (e.g. fuel system) thus require a functional and behavioural approach (see also Section 5.2 "Model Types" on page 50).

**Trim Drag Example:** To address the trim drag, the functional information concerning how to maintain longitudinal trim has to be included in the model using a functional model. On a standard drake configuration, this might be performed by generating an up-/down-force on a

(more or less) horizontal lifting surface in the back of the aircraft that perhaps is user-denoted/tagged as a horizontal tail. In the case of an all-moving elevator, this can be achieved by changing the incidence angle of the elevator or by a deflection of the trailing edge control surface(s) on this lifting surface. Additionally, especially under prolonged cruise flight regimes, trim might even be arranged by active load shifting, usually implemented by intertank transfer to a rear located trim tank. In addition to the behavioural information, this also requires additionally a structural model (components) of the fuel system to enable the transfer of a certain amount of fuel to the trim tank, involving valves, piping, and pumps. In a functional view, the initially geometry-based product tree is therefore tweaked towards a more system-related product-oriented decomposition, as shown in Figure 3.5. A typical aircraft system hierarchy break-up, close to the Air Transport Association Specification 100 (ATA 100) chapter structure, can be found in [Jackson, 1997].



**Figure 3.5** Product trees of different target domains: geometric vs. system/functional decomposition (adapted from [Baslev, 2010])

### 3.3 Domain-Specific Tool Implementation

Just as important as the data structure is the internal build-up of the entire application(s), addressing topics such as clarity, maintainability,



adaptability and extensibility (cf. the ACD characteristics listing at the beginning of Section 3.1). Common implementation concepts are:

- function-based
- object-oriented, and
- graph-based

One main topic of Object-Oriented Programming (OOP) is the association of object related data and routines to an object entity, usually denoted as a `class`. Concepts of OOP are encapsulation, inheritance, and polymorphism [Cellier, 1996] – similar to the inseparable relationships of real physical objects in an aircraft. These concepts distinguish the OOP from the (purely) function-based approach with the drawback of greater complexity. Graph-based approaches are not yet state-of-the-art for ACD but are already used for product/system design, often as a holistic approach including requirement handling and manufacturing [Rudolph, 2014].

### **3.3.1 Geometry/CAD Domain Handling and Integration**

It is natural that a Computer Aided Design (CAD) (related) model is set up according to a geometry-based hierarchy in contrast to a functional model, such as the schematic (or system simulation) of the fuel system. The CAD domain usually serves the following three main purposes:

- geometry modelling
- analysis capability, direct or via special export formats (e.g. weight distribution, volume measurements, Computational Fluid Dynamics (CFD)/Finite Element Method (FEM) meshes)
- visualization

Even on very simple 3D geometries quiet complex mathematical algorithms and calculation procedures have to be performed. Due to a vast parameter overdetermination of the basic geometry elements of an aircraft such as fuselage or lifting surface (e.g. wings) bodies, an acausal implementation is to be preferred. An example of such an implementation is the **VAMPzero** by the DLR (see the more detailed explanation

in [La Rocca, Jansen, et al., 2013]) for initial aircraft sizing. The basic geometric kernels of “easy” CAD modeller such as SUMO or openVSP [openVSP, 2016] are already quite huge and complex to develop.

Since “CAD work is to 99% about interacting with the system GUI” [La Rocca, 2011], the use of a mature, established CAD<sup>1</sup> environment should be preferred. However, the direct application-unspecific application domain “complete” CAD tools such as CATIA® or proEngineer create several problems when applied to the ACD process. First of all, the user is overwhelmed by the various functionalities to build up a geometry. Also, starting from a blank sheet every time is a cumbersome approach, but most importantly, the interpretation (by other tools) of the individually modelled CAD geometry lacks functional as well as behavioural information.

The approach selected in CADLab is to apply an overlay in a complete CAD environment (here CATIA® V5) and by doing so limit the type of geometric instances. A detailed description of the CADLab framework and the included CAD implementation can be found in the Usecase1 example in Section 7.1 respectively Section 7.1.2.b on page 96. An already existing ASCD tool is ASTRID [Chiesa, Di Meo, et al., 2012] which is based on OOP principles but do not incorporate the CAD domain during the design process [Chiesa, Fioriti, et al., 2012].

### 3.3.2 Causal versus Acausal Implementation

It is also favourable to go for an acausal implementation in the other domains, similar to the CAD/geometry definition task. This implementation method supports the ability to conduct work steps in an arbitrary order, allow for a vast selection of overdetermined parameters, and enable a flexible usage of the embedded equations<sup>2</sup>.

BeX, a stand-alone conceptual aircraft design sizing tool implemented in Excel/VBA macros [Berry, 2015] enables an acausal use of a causal implementation using local optimisations, either through the default Excel (optimisation) function Solver or the Goal seek function for

---

<sup>1</sup>CAD program knowledge collected during education and work experience is often related to one of the established commercial CAD/CAM tools.

<sup>2</sup>In common programming techniques, mathematical equation formulations are strict causal and represent causal, one-directional assignments only. Certain programming/simulation languages, however, have the possibility to enable a mix of both, causal assignments and acausal equations like for example in the Modelica language [Fritzson, 2004].

single variable problems. Certain parameter values (cells) are divided into a calculated output and an estimated input part to avoid numerical loops. The user may possibly manually update the input value with the output value. Update control is thus up to the user with the problem of keeping the data consistent in favour of flexible use. Due to the nature of **Excel**, data, data representation, figures, and the GUI are in one and the same window, limiting clarity but providing access to any properties for the experienced user.



# 4

## Cascaded Systems Design

*This chapter deals with the analysis of system and subsystem relationships, system requirements and the main system architecture design drivers. The concept of energy conversion for physical power systems is also presented.*

### 4.1 System Nomenclature

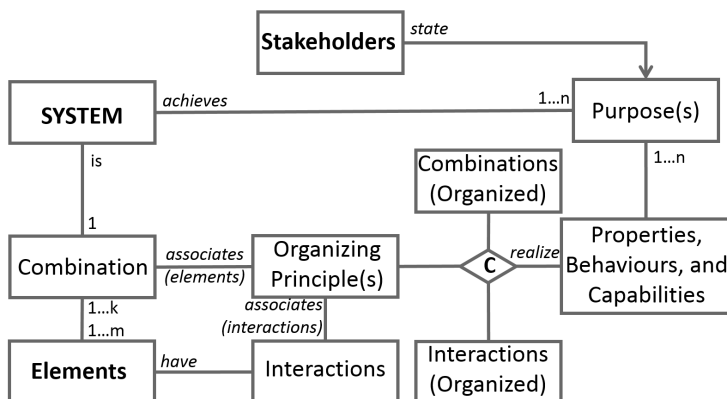
*“A system is a combination of interacting elements organised to achieve one or more stated purposes” [INCOSE, 2015].*

One application related approach to describing a system is the PICARD system theory by [Martin, 2012], decomposing a system as a holistic image of six attributes, see Figure 4.1.



**Figure 4.1** The PICARD theory of systems by [Martin, 2012]

This system definition more precisely stated and extended with the relationships by [Dickerson et al., 2010] results in the diagram shown in Figure 4.2. Other system definitions, such as [ISO 15288, 2008], have



**Figure 4.2** Extended diagram representation of the system definition (adapted from [Dickerson et al., 2010])

the fact that a system fulfils one or several purposes is in common. This function fulfilment thus defines the minimum possible division of a system into subsystems.

### 4.1.1 Inter-System Relationships

A system might be divided into subsystems according to the above-mentioned system definition in such a way that every subsystem again represents a system with a distinct purpose. The term *subsystem* highlights that a certain system is placed within the context of another system, usually denoted as the parent system. System denomination usually depends on the system of interest that the user has in mind. The system (of interest) operation may require some kind of interaction with enabling systems (see [ISO 15288, 2008]). If these different systems are composed together, enabling new functionality or higher performance (of already system-inherent functions), the term Systems of Systems (SoS) is used [L. Hu et al., 2015]. Typical SoS applications favour of the fusion of information from different domains; examples include defence systems (mainly information fusion), multi-vehicle missions scenarios [Roberts et al., 2016] or smart electric power networks (information and energy fusion) [Mavris and Griendling, 2016].

System (instances) operate in an environment, and the systems purpose typically requires some means of resource exchange or communication. For these reasons, systems usually provide system boundaries for information, matter or energy exchange. According to [DoD, 2010], system boundaries to(wards) other (compatible) systems are called system interfaces<sup>1</sup> which represent a refinement of a system boundary. A comprehensive overview of system nomenclature and definitions within the engineering system design context can be found in [Dickerson et al., 2010].

To unify the terminology within this thesis and to shift the focus from the SE-tailored definition towards a system modelling and simulation dominated context (which can be seen as a subsystem within systems engineering), a few nomenclature adjustments are introduced: System boundaries (including system interfaces) are denoted as ports with resources shared across these virtual or real element ports. These resources can be of any type of information, matter, or energy. With these changes, the nomenclature becomes congruent with the Channel-Agency net definition presented later in Section 6.5.

## **4.1.2 System Properties**

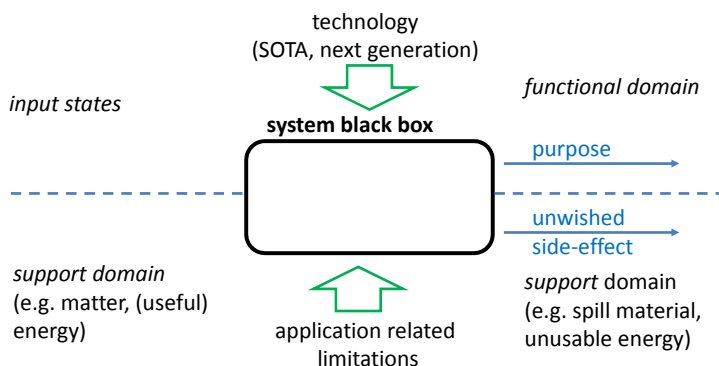
The easiest possible abstraction of a system is the black-box representation, nowadays often replaced by a SysML/UML state diagram, reducing a system into its primary interfaces, functions, and properties. In the context of aerial vehicle design, significant system features may be:

- system weight
- system power demand (and thus efficiency)
- system volume and possibly limitations in component shape, split-up, and placement

Usual semi-empirical methods take into consideration the two first points, whereas the third is seldom addressed but is especially important for Unmanned Aerial Vehicle (UAV) and military aircraft because of the limited space available.

---

<sup>1</sup>A system interfaces might be of any complexity level like a physical pipe for matter exchange, a high-level signal interface (like RS-232 or CAN) or any combination of matter, energy, and information exchange.



**Figure 4.3** A universal (sub-)system black box model representation

## 4.2 System Design Drivers

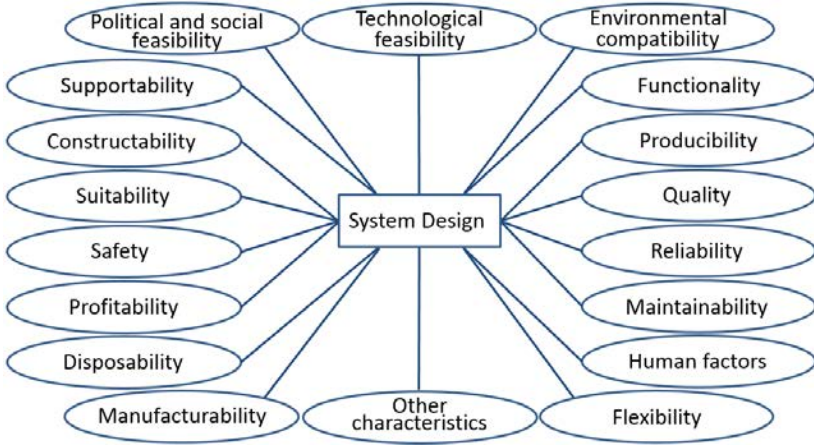
A particular system design is the result of all influence factors from all system stakeholders during its whole life-cycle. Figure 4.4 illustrates the extremely wide spread of influencing factors that have to be taken into account (and thus modelled) during the design process. Some of these factors may be active only during one phase of the products life-cycle, whereas others may be related to several or all phases. Obviously, during a particular project, the primary focus may only be on a few categories, e.g. reliability in the case of a Primary Flight Control System (PFCS) (see also limitations of this thesis in Section 1.2.1). Certain topics may be excluded, others extremely simplified.

From an MDO point of view, only a limited number of requirements may be valid and influence the design (in architectural and quantity measure) at a certain life cycle phase or in a certain operation mode. A large number of requirements might be inactive during certain life-cycle stages or particular operation states, or may not apply to the chosen system architecture at all.

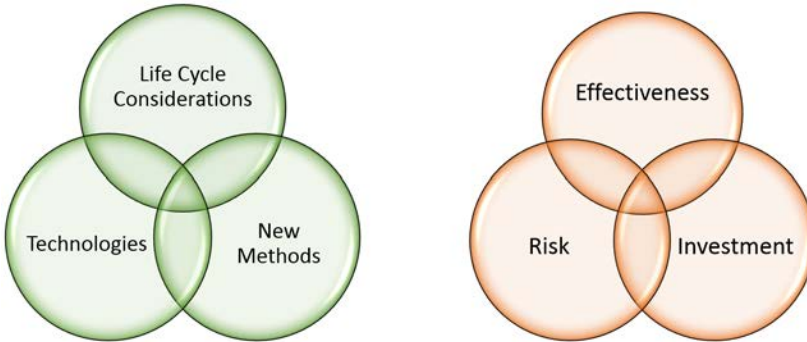
### 4.2.1 System Performance and Efficiency

Systems are compositions of components with specific relationships to fulfil the dedicated purpose(es), also denoted as functions. A system performance or efficiency benchmark has to consider the system function fulfilment among the (performance-related) properties of the system of interest. A system benchmark may thus differ between the different stakeholders' points of view. The most common holistic approach is the





**Figure 4.4** Common system design influencing factors (adapted from [Herzog, 2004]; primary source: [Martin, 1996])



**Figure 4.5** The three important elements of the new design process paradigm (adapted from [Kirby, 2001]) to the left and the economic technology assessment properties to the right

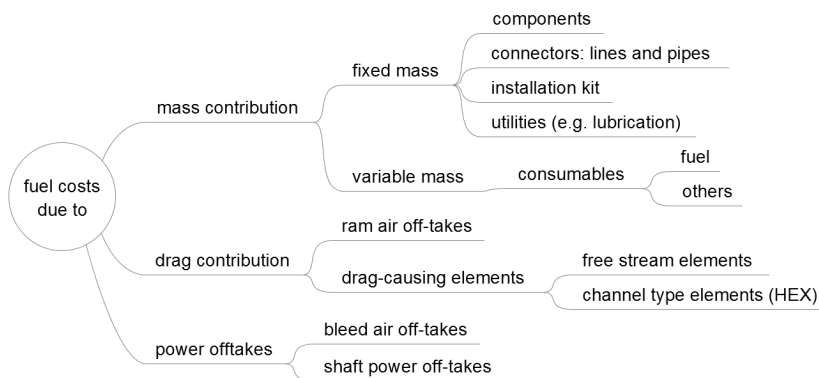
complete life-cycle assessment (e.g. based on [ISO 14040, 2006] regarding the environmental impact of the product). With regards to aircraft systems more specifically, a lumped systematic cost approach like the *DOCsys* method by Scholz is favourable [Scholz, 1998]. It represents a DOC-based estimation method similar to the one recommended by [ATA, 1967]. Scholz describes the system DOC as:

$$DOC_{SYS} = Depreciation_{SYS} + Fuel_{SYS} + DMC_{SYS} \quad (4.1)$$

where

- *DMC* are the (system caused) Direct Maintenance Costs.

Aircraft operators may extend this approach by adding delay and cancellation costs as well as capital costs due to Spare Holding Costs (SHC). The fuel cost due to the system can be split up according to the scheme shown in Figure 4.6. Important to note is that system costs depend on



**Figure 4.6** Mission-specific fuel cost contribution factors of an on-board (power consuming) system

both the operator-specific (utilisation and maintenance strategy) and a mission-specific fuel contribution itemised in Figure 4.6. Further on-board system properties of interest (with the aircraft as the system of interest) besides weight, drag contribution, and secondary power off-takes include:

- volume and component placement limitations (especially on supersonic vehicles)
- system component (and installation) contribution to the centre of gravity location and shift (in the case of variable component mass).

## 4.2.2 System Faults, Safety and Reliability

In addition to the performance or cost benchmarks, system reliability is another important system design driver, both for safety reasons and the DOC influence due to delays, cancellations/Aircraft on Ground (AOG) events and SHC effects.

The systems reliability can be calculated using a Fault Tree Analysis (FTA), taking into consideration the individual components Mean Time

Between Failure (MTBF) and the arrangement/relationships of the components [Vesely et al., 2002]. Using an FTA, both fault frequency and the fault set (combinations of failures) are addressed, but it comprises no failure effect assessment. For this purpose, a Failure Mode and Effects Analysis (FMEA) may be performed, targeting the consequences and workaround strategies of particular system faults combinations. The hazard level of the FMEA cases for JAR/FAR-25 defines five Design Assurance Level (DAL) categories – each addressed with a certain acceptable minimum reliability value – ranging from catastrophic ( $1e^{-9}/h$ ) (A) to no effect (E) [DO-254, 2000].

Addressing system reliability in an automated design process is, in addition to an FTA with constant component probability values, a challenging task. Deriving a fault tree structure from any graph/network structure requires a process to break up loops in the network to maintain a strict tree structure. The automatic integration of an FMEA seems very challenging due to the (often manually conducted) assessment of the fault consequences including the consideration of lowered requirements and the relocation of system functions to other, fault-unaffected, components. This type of load balancing in a derated system mode has to be defined in the behavioural part of the system model. An overview of system safety and reliability methods in early design phases can be found in [C. Johansson, 2013] and an example of the FTA modelling and integration is given in Usecase3 in Section 7.3.3.

One hurdle to integrated the certification requirements dressing the system's reliability is the general characteristic of requirements. Requirement formulation should be in a manner stating what a system should comply with and not how it should be achieved [Herzog, 2004], [Robertson et al., 2013]. This implementation-independent formulation makes it hard to link the requirements towards the implemented system automatically because it requires a high-level interpretation of the requirements. Quality measures for requirements can be found in [Pohl et al., 2011] and [IEEE Std. 830, 1998].

## **4.3 System and Component Technology Classification**

For the system design drivers' analysis, system decomposition and classification are applied. The Decomposition to split the system of interest

into smaller units, decomposition can be performed on different criteria such as functionality, technology domain or supplier. A common split up of aircraft systems is addressed in the superseded ATA 100 numbering system [ATA 100, 1999] or its successors S1000D (military)/ATA i2000 Spec (civil) [A4A, 2016]. A common coarse-level categorization of the on-board power systems – with the focus on the energy expenditure – is the division into power generation, power distribution, and power consumption system (see Section 2.2). Focusing on the overall function (of each system), [Pahl et al., 2007] group the functions into:

- conversion of energy
- conversion of material
- conversion of signals

The included components may be rated according to their primary purpose as:

- power/energy/matter
  - producer (generator)
  - consumer
  - converter
- signal
  - regulator (controller, e.g. PID)
  - calculator (logic, e.g. sensor fusion)

Other classifications may be related to the system power domain, e.g. the DSE [Chakraborty, 2015], which describes the extent of installed electric secondary power in comparison with the alternative (pneumatic and hydraulic) installed power capacities.

#### **4.3.1 Natural Order of System Technologies**

In a classic product development approach, different (technology) alternatives of a product are presented in a morphological matrix, from a functional point of view also known as the function-mean matrix [Pahl et al., 2007]. The vast number of combinations in the morphological matrix may require a reduction to adequate combinations only; Focusing

on (new) technologies, a Technology Capability Matrix (TCM) [Kirby, 2001] may be used as a filter on the morphological matrix in order to reduce the combinations of useful technology implementations. Basic TCM addressed incompatibilities originate from:

- competing for the same purpose
- combinational (extreme) degradation effects<sup>2</sup>
- introducing (too high) certification difficulties

At a more detailed level, focusing on the system-subsystem cascading, further technology (domain) limitations may be identified. These are:

- energy transformation incompatibilities
- impractical combinations; these may be determined by:
  - detailed investigation of the energy transformation
  - impractical component size (scale)
  - complexity: Unnecessarily high complexity of implementation
  - experts judgement
- Technology Readiness Level (TRL)<sup>3</sup>: high level of perceived technology risk versus system reliability importance [Chakraborty, 2015]
- control: Stability issues (e.g. time domain, feedback loop delay, real time system)

Since the energy transformation matrix (see [Culp, 1991]) is relatively dense, no or only a few completely incompatible system-subsystem configurations may exist. To significantly reduce the number of combinations, more detailed transformation analysis, focusing on the practicalness and feasibility of the solutions may be evaluated utilising the component/power scale, the TRL, or other appropriate measures.

---

<sup>2</sup>E.g. composite wing in combination with hybrid laminar flow technology as shown by [Kirby, 2001]

<sup>3</sup>Alternatively to the most common TRL index, other, for implementation more precise, measures as the the System Readiness Level (SRL), the Integration Readiness Level (IRL), or the Manufacturing Readiness Level (MRL) may be used [GAO, 2016].

Based on a risk reduction by the TRL and the system/technology application experience from already realised systems, [Chakraborty, 2015] derives following TCM rules for a More Electric Aircraft (MEA) architecture of cascaded systems from the expert judgement mentioned above:

1. *“Successive actuation packages feature progressively more electrification: An actuation function once electrified is never de-electrified in a higher package”*
2. *“No package features a combination where a more critical actuation function has been electrified while a less critical function remains conventional (hydraulic)”*
3. *“When competing actuation technologies differ in the perceived level of technological risk, the technology perceived to be riskier is introduced to less critical actuation functions first”*

[Dunker et al., 2015] shows a Flight Control System (FCS) optimisation based on a hybrid electro-hydraulic system setup considering the above-mentioned system implementation rules to ensure the systems' reliability.

#### **4.3.2 Technology Scalability**

Another important component and system design factor is the scale: most (energy conversion/adaption) technologies are limited to a certain application range, both, through

- infeasible regions (due to technology-inherent design space limitations, e.g. by physical effects)  
and
- impractical regions (due to improper, inefficient or unusual design)

Problematic for (KBE-based) automated design is the necessity for design space limitations of the component sizing rules, which – besides the naturally addressed primary properties like efficiency – are often related to secondary effects such as heat load or cavitation. In addition to operational aspects, impractical component and technology scales are off-design point states or operations in regions where certain technology dependent disadvantages outmatch the technology-dependent advantages. This means that the system-disadvantageous effects have to be addressed and modelled in the surrogate (component) model.

**Aerospace Example: Propulsion System** The aircraft's propulsion system is a typical system where the system design is largely influenced by the scale of application and the operation design point. While electric-propeller solutions dominate for small UAVs, a transition towards (two-stroke and four-stroke) engine driven propeller systems, turbo-prop configurations and finally kerosene-jet driven propulsion systems predominate for medium-size and large aircraft.

## **4.4 The Power and Signal Component Concept**

Based on the energetic classification as mentioned in the listing in Section 4.3, the system components can be split into power (handling) and signal (handling) components. Besides modelling and simulation aspects (see e.g. the *Hopsan* Transition Line Model (TLM) method in Section 7.2.5), this categorisation also allows for a systematic design process, enabling an analysis or technology, component, and system selection based on the findings presented in the following sections.

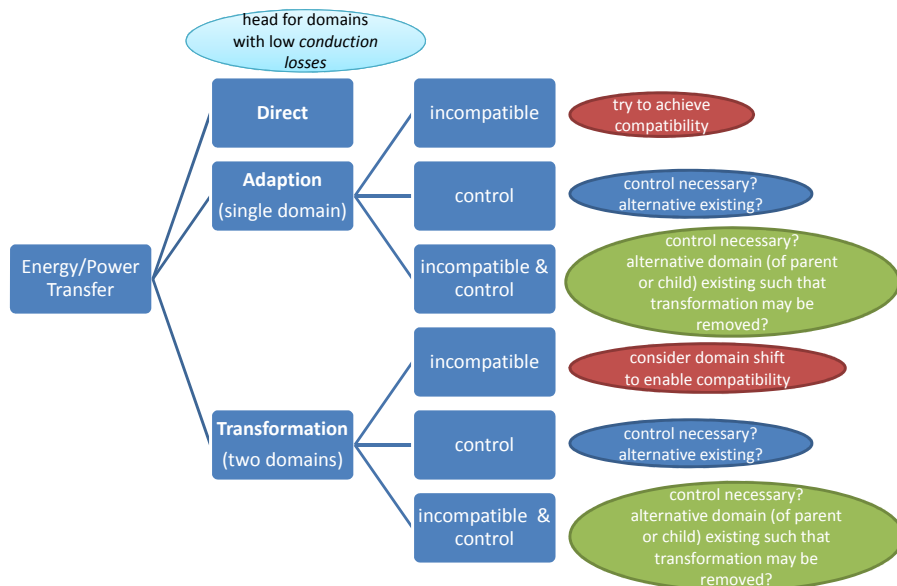
### **4.4.1 Power Components**

Usual energy forms within power components may be of a mechanical, electrical, (nuclear,) electromagnetic (light), chemical or thermal nature [Allen, 2014]. Primary energy sources are often based on chemical energy ("fuel") due to their relatively high energy density. An alternative energy classification is a categorisation into potential and kinetic types of energy. A grading of these powers and energy domains may be performed with respect to the target system's scope; addressed properties are energy/power transformation, transfer efficiency, inherent risks (heat, fire, toxicology), or power/energy densities of the matter or the power component itself. Depending on the energy domain, energy conversions may be irreversible and come along with energy losses. Based on the transformation ability, energy sources can be ranked, with electric energy as the most valuable one at the top.

#### **4.4.1.a Energy Transformations**

As defined above in Section 4.3, resources such as matter, energy and information may be exchanged between any systems. Physical power systems' performance is inevitably associated with the conversion or

adaption<sup>4</sup> of power and energy, and possibly – if existent – the conversion of matter. These processes are limited by physical laws as well as by the implemented technology (level). In contrast, information exchange can be simplified as a lossless process. This simplification is adequate for power systems but may not apply to stand-alone processing units, cyber systems (e.g. software) and information systems.



**Figure 4.7** Energy transfers and the usual motivations of energy adaptations and transformations (of power-consuming systems)

Important to note is the frequent absence of direct energy level conversions within one domain. This conversion incapability can be solved by transferring the energy into another domain and then back into the original domain. Examples include heat pumps, DC-DC converters and automatic transmissions.

**Technology Selection for Energy Transformation:** An extreme example of how technology selection can influence the system (or energy transformation) efficiency is the electrically driven photon emitter, usually denoted as a bulb. In the classic (a better term might be “histor-

<sup>4</sup>Observe the use of the terms *transformation* and *adaption* defined in Section 1.5 “Remarks” on page 11.



ical”) implementation as an incandescent filament bulb, electric energy transforms via a metallic (usually tungsten, earlier carbon) resistor into heat. This heat makes the filament incandescent and it then, according to the Kirchhoff’s law, emits a temperature-dependent spectrum of photons. The total efficiency is limited to the temperature-dependent spectrum shape and the filament temperature, which is restricted by the material and expected lifetime. Small improvements had been made by adding a halogen gas to reduce the filament degradation and allow for higher temperatures. LED-based bulbs, however, use another physical effect to transfer electric energy into photons of a particular wavelength: electroluminescence. In this case, the technical efficiency is not limited and can theoretically reach 100%. Nonetheless, after various technology changes the electrical interface remained unchanged; it is still the Edison Screw E27 (DIN 40400), invented by Thomas A. Edison, inspired at that time by the screwed connectors of the gas-driven bulbs at those days (system compatibility, see Figure 4.8) [Utterback, 1996].



**Figure 4.8** *Light bulbs of different technologies with the deliberate similarity of the (newest) filament LED bulb (right) and the classic filament bulb (left). The system interface is still based on the historical Edison "E27" socket.*

#### 4.4.1.b Power Control/Energy Adaption

Energy loss due to the control of matter and power is another typical kind of energy losses within (power) systems or power components.

Usual analogue control elements are based on throttling (e.g. resistors, nozzles, orifice and proportional valves), which are necessary lossy processes. Consequently, to avoid losses through energy conversion and control, the following recommendations can be given:

- prefer a *high level* of energy
- avoid energy conversions where possible <sup>5</sup>
- perform energy conversion only from “higher energy levels” to “lower energy levels” with appropriate physical effects
- try to adjust the energy/matter properties to needs (thus avoiding unnecessary “equal” power or matter translations)
- avoid control by throttling

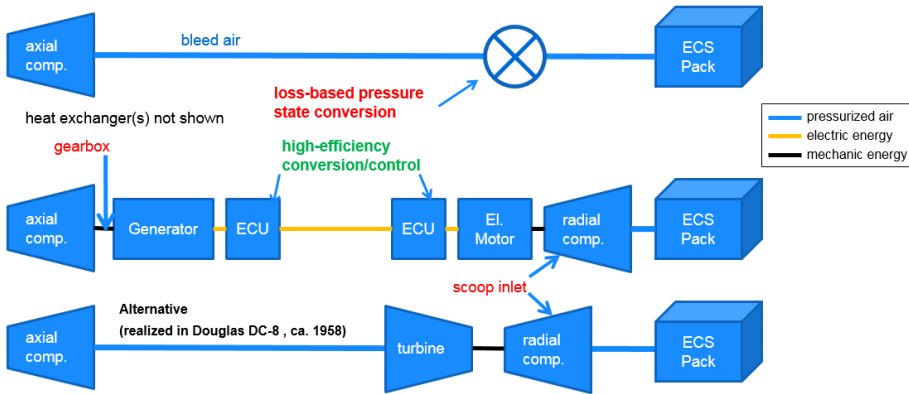
A schema for energy transformation and conversion related analysis of power components is given in Figure 4.7. With the help of this examination, unnecessary or inadequate energy/power paths in the system design can be discovered and mitigated.

**Control Losses on Different ECS Architectures:** Applied on aircraft subsystems, the conventional Environmental Control System (ECS) includes multiple thermal energy conversions with the above-mentioned negative system performance impacts. On a conventional ECS (see Figure 4.9), bleed air is tapped from the engine core and conditioned in a complicated process involving heat exchanger, bootstrap unit, water separator and ozone converter to the required temperature and pressure level for flight deck and cabin air conditioning. The system is self-powered by the high pressure potential of the bleed air and follows the above-formulated system rules well by holding the energy flow in one domain (pressurised air only). More detailed information regarding conventional ECS can be found in [Hunt et al., 1995] and [Moir et al., 2008].

In a new MEA architecture – with the Boeing 787 as the first of its kind – an electrically driven system with a scoop inlet for ambient air intake replaces the bleed air driven ECS. As shown in Figure 4.9 (middle), the conventional ECS layout involves several inter-disciplinary energy

---

<sup>5</sup>This rule is similar to the well-known rule to strive for short and direct load-paths within structural design.



**Figure 4.9** Different ECS schematics: Conventional bleed air system at the top, new MEA system design (middle) and the unique system design of the Douglas DC-8 with a pneumatic-pneumatic energy level converter (turbine-radial compressor) unit.

transformations, namely from thermodynamic (within the engine) via mechanical energy into electrical energy and then almost the whole way back from electrical energy to mechanical energy and finally into thermodynamic energy<sup>6</sup>. The important system design drivers (besides the bleed air versus shaft-power engine off-take efficiency topic, addressed in Section 2.2.1) are the control losses of the conventional design over the extremely wide operating range. Instead of throttling and final temperature fine-control with the help of hot trim air, the system control in the MEA architecture is relocated from the thermodynamic domain into the electrical domain with its superior control efficiency mechanism. An interesting approach to overcome the control problem – and at the same time abandon the engine bleed air from the fresh air system(!) – has been made by the Douglas company with an ECS system including a bleed air-driven turbine compressor unit to condition ambient air from a scoop inlet. In this way, the turbine-compressor unit acts as a level transducer and can be efficiently controlled by geometry changes in the compressor or turbine (e.g. via controllable guide blades or the rotational speed).

<sup>6</sup>Also interesting to note is the substitution of an axial compressor (in the conventional ECS) with a radial compressor (in case of the MEA ECS architecture). Compared with e.g. the development of jet engines, the common technology trend had been from radial to axial compressors due to the higher maximum efficiency.

#### **4.4.1.c Exergy versus Energy Efficiency**

Energy conversions are usually described with respect to their absolute efficiency. For thermodynamic transformation based power systems, it is useful to have an alternative or additional measure based on the system/component exergy [Grönstedt et al., 2014].

While the former represents an absolute boundary for the reachable limit, the latter shows the difference between the theoretical optimum with the current technology state and the achieved efficiency. Exergy-loss focused analysis thereby enlightens the real potential of the entire components. Depending on the analysis objective, overall system design or component design, energy efficiency or exergy may be used for evaluation.

#### **4.4.2 Signal Components (and Power Electronics)**

A signal component is either an electric signal processing unit (e.g. a printed board without any processing units on it) or the combination of software and the software executing hardware (the processing units). A signal component of the latter type cannot be described and handled by an architectural or structural model only but requires a behavioural model also (see also Section 5.2).

The signal components' computational performance has made tremendous advances over the last 50 years, following Moore's Law, stated in 1965 [Waldrop, 2016] (see also Figure 5.9). Alongside transistors' miniaturisation and count rise, energy efficiency – ranked as Floating Point Operations Per Second (FLOPS) per watt – has evolved considerably. Nowadays, one-chip microcontrollers have the calculation performance of older personal computer without the need for active cooling anymore.

Unlike the above-discussed power components, signal components have to be handled in a different way because the signal component's purpose within a system is not sufficiently defined by the element's function itself. A processing unit, for example, may be used as a controller for any system control purpose or to fulfil any other data processing purpose. In contrast to signal components, power components have a tighter coupling between the element function and the system function, as demonstrated in the following example: An electric motor represents an electrical to rotational mechanical power transformer on its element level, which is consistent with the item's function from the system point of view (to perform a particular function, e.g. generate torque on a

wheel/shaft). This fact makes a (universal) bottom-up approach of signal components very hard, because of (the structural model unit) may fulfil several functional purposes.

Another problem is the signal component relationship on secondary effects. From an aircraft system performance perspective, size and energy consumption are the dominating design properties. However, weight, volume, and size properties of signal components are mainly based on secondary aspects such as power supply, cooling, (sealed) housings<sup>7</sup>, connectors, (line) installation and other requirements like Electro-Magnetic Compatibility (EMC), vibration, and separation. Hence, these secondary effects have to be considered. Because of these secondary effects there is a weight and energy saving potential related to the component miniaturisation and power reduction<sup>8</sup>.

Reducing the heat load and lowering the cooling demand will be one part of future signal component improvements. New Integrated Circuit (IC) technologies with higher upper temperature limits (as e.g. Gallium-Nitride) have the capability to trigger huge weight savings by lowering the cooling load of the Thermal Management System (TMS) [Ganev et al., 2013].

Not categorised as signal components but also associated with the aspects explained above (like housing, isolation, and cooling) are the power electric components such as converters or inverters; Current research states a power density of 1...2[kg/kW]<sup>9</sup> for (line replaceable) Motor Control(ler) Electronics (MCE) units for JAR/FAR-25 aircraft in a power range of 5...30[kW] [Todeschi et al., 2016]. The same source also identified Electro-Magnetic Interference (EMI)/EMC as a potential show-stopper. Directly related to the magnetic field density and for this reason the volumetric power density of the component, this fact may hamper further development of components with higher power density.

---

<sup>7</sup>Housing weight due to gas-tight housing requirement to prevent condensation events due to pressure variation over each flight cycle. This type of housings are required for unpressurised and humid locations (ECS, galley, and battery compartment).

<sup>8</sup>See also the growth factor in Section 2.2.2.

<sup>9</sup>A very low value compared to the density pressure of electric motors or other electric components in non-aerospace applications.

## 4.5 Cyber-physical Systems

Besides the physical function instantiation process, the split-up between software and hardware<sup>10</sup> in cyber-physical system implementations also have to be solved. The emergence of more and more software-based solutions can be explained by the topics discussed in Section 4.4.2. Because of the different nature of software and hardware (power) components, the applied processes and tools should be capable of handling these subjects.

Due to the higher increment of calculation power (see Figure 5.9) compared with the increase in system (power) capability measures (e.g. thrust-to-weight ratio), the software part becomes increasingly important, shifting the domain boundary from the physical to the imaginary parts of the system. This trend has already gone so far that as an initial guess, the cost of a fighter aircraft may not be stated/calculated based on the total system weight, but by the millions of lines of code (MLoC) that are incorporated in the product [Hammarström, 2016].

**Software Remark 1** Even though software developed may be hardware-independent, it requires hardware to be executed on; these components (e.g. a microcontroller) may be stand-alone (categorised as signal components) or part of a power (electric) component (e.g. an MCE). In fact, almost any power electronic components contain some (calculation) signal elements. These items may be insignificant for the weight, power and size estimation, but important from component focused functional or behavioural view.

**Software Remark 2** If software is part of the system of interest, both, the software itself and the software hardware components have to be addressed within the reliability analyses (see Section 4.2.2). Recommendations for considering software are given in [DO-178C, 2011], for electronic hardware in [RTCA, 2000].

---

<sup>10</sup>In order to differentiate between software and “*non-software*” (thus often power components) related hardware, the former is denominated as “*software hardware*”.

# 5

## Model-Based System Design

*In “Cascaded Systems Design” (Chapter 4), models have already been applied to describe the principles of system design. In this chapter, a detailed analysis of models and modelling techniques for physical system design including physical system simulation models is given. Additionally, the knowledge of systems is specified leading towards a systematic nomenclature enabling (semi-)automated system engineering processes such as Knowledge-Based Engineering (KBE) for system architecture design.*

*The purpose of this chapter is to analyse and classify the different model types and their integration. These investigations provide the necessary background to form a sound information model and types of analysis selection of an ASCD framework.*

### 5.1 The Use of Models

*“A model is a relational structure for which the interpretation of a (logical) sentence in the predicate calculus becomes valid” [Dickerson et al., 2010]*

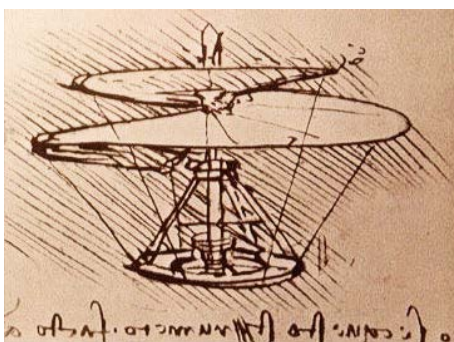
#### 5.1.1 Model Abstraction

A model is the fundamental way of human thinking, providing a complexity reduction and a syntax highlighting on the context in focus. Models can be of any type: drawings, language specifications, physical

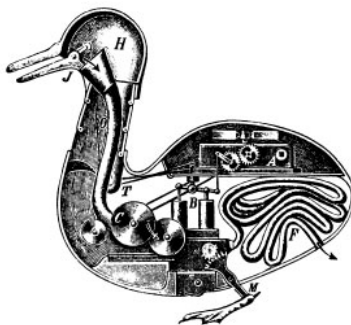
models, (UML) diagrams, code snippets, huge software programs or simply human imagination. In the context of systems engineering, modelling is the abstraction activity of a system (see also Figure 5.3). Modelling languages (e.g. UML) are denoted as technologies that bring precision into the abstraction [Dickerson et al., 2010].

### 5.1.2 Early Adopters

Models have been known to exist since long time, the famous drawings of technical inventions by Leonard da Vinci in the 15<sup>th</sup> century being one example (see Figure 5.1). Particularly interesting in the system modelling context is the digesting duck by René Descartes (Figure 5.1), a drawing (model) of the model of “an animal body as a complex machine with the bones, muscles and organs replaced with cogs, pistons and cams” [Wikipedia, 2016] from 1662. As a model of a model, it represents a metamodel of digestion.



(a) A helicopter study by Leonardo Da Vinci, 1493



(b) Digesting Duck model by René Descartes, 1662

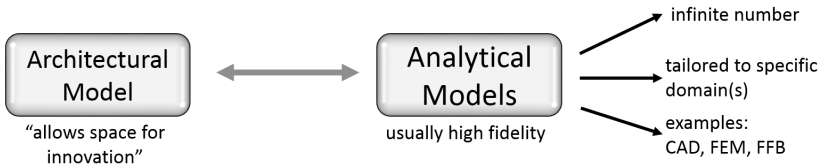
**Figure 5.1** Examples of early model adopters

## 5.2 Model Types

A model is a relational structure and a collection of mathematical relationships on this set [Dickerson et al., 2010]. (System) of models may be of a physical type but are more often of a theoretical nature. (Applied) models are usually not all-embracing but tailored to a certain purpose. The developing action of an analytical model, tailored to a specific pur-

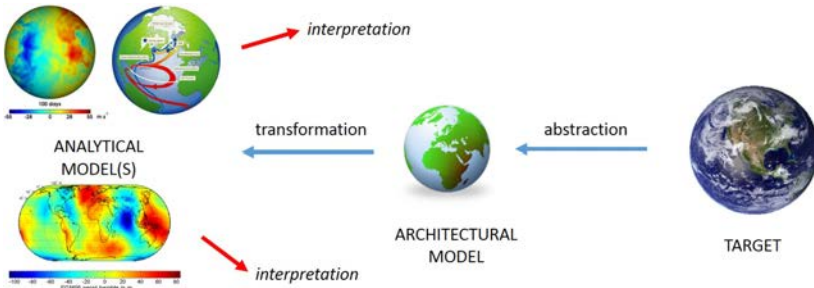


pose by limiting it to one domain, requires an architectural model of the matter/system of interest. This all-encompassing architectural model is usually not explicitly formulated and in one location but exists individually in the imagination (aka understanding) of every person involved in the development process of the system. The derivation process from the



**Figure 5.2** The architectural versus analytical (domain-dependent) model relationship

architectural model into an analytical model represents a model transformation, whereas the architectural model is an abstraction of the target (see Figure 5.3).

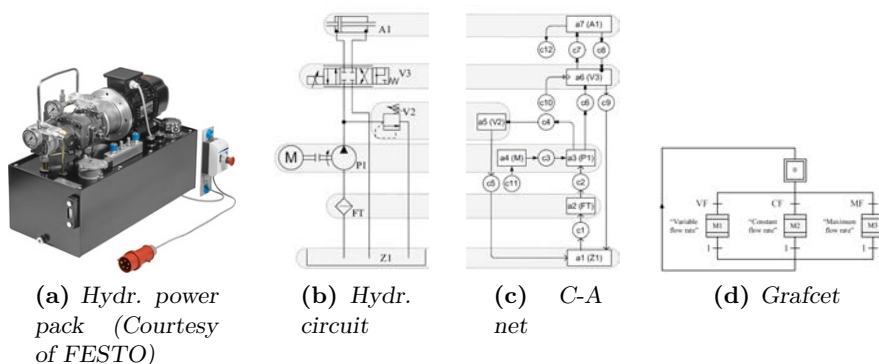


**Figure 5.3** Schematic of architectural model abstraction from the target (system) and the transformation of different analytical models from the architectural (reference) model

The crux of the matter is the general absence of an all-embracing architectural model, which (as mentioned above) may be spread information at different locations as well as the own imagination and interpretation of every stakeholder<sup>1</sup>. Because analytical models represent only a single or limited domain view, transformed from the architectural model which in turn is the abstraction of the target, the interpretation of every one

<sup>1</sup>Although not explicitly addressed in this thesis, a holistic sustainable product development approach addresses all possible stakeholders of a system may differ dramatically in their expertise background, target/goals and motivation.

of these models is dependent on the human individual. Model knowledge is only partly included in the (analytical) models directly. Another part of model knowledge is defined by the notation/specification of the model, including the underlying theory. Figure 5.4 shows the tree analytical models of a hydraulic power pack unit on a high abstraction level: structural, functional and behavioural (see also Table 5.1) [Porciúncula et al., 2016]. Whereas all of these analytical models consist of a graphical notation (hydraulic system symbols according to [Standard, 2007-03-15]), C-A net [Reisig, 1992], Grafcet [IEC 60848, 1999]), models without a suitable graphical or imaginable<sup>2</sup> representation may also exist.



**Figure 5.4** Different (analytical) models of a hydraulic power supply pack. From left to right: product photo, structural hydraulic circuit drawing (ISO 1219), functional C-A net and behavioural Grafcet representation (b-d from [Porciúncula et al., 2016]).

## 5.2.1 Physical Models and Simulation

Besides theoretical models in physical product/system development, two kinds of models are regularly used:

- simulation models, and
- physical models

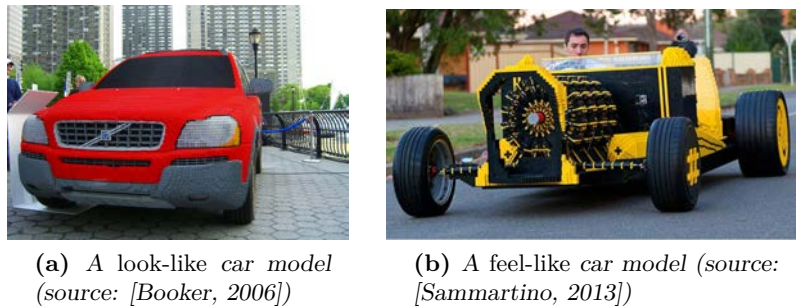
<sup>2</sup>Imaginable model: related to our macro/micro physic world view or related to the size of the system of interest. An example of a missing graphical/imaginable (depending on the state of education) is the wave-particle dualism model for electrons.

functional model(ling)	structural model(ling)	behavioural model(ling)	
what the system does or should do ( <i>why</i> )	describes <i>where</i> the functions are implemented	<i>how</i> and <i>when</i> functions are executed	
the system's ability to fulfil a certain purpose	set of system elements and their relationships	relationship of inputs, internal states and outputs	
functional structure channel-agency net (C-A net) production flow schema	function-mean tree or table	continuous state model	discrete state model
UML engineering circuit diagrams/drawings (electric, hydraulic, pneumatic, etc.)		transfer function; state space description	Petri net (event-driven); mark flow graph; automation grafcet

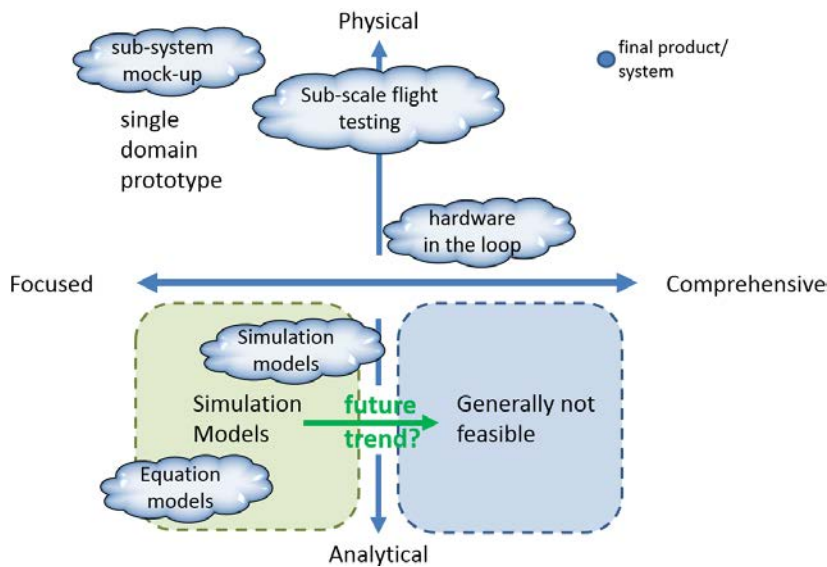
**Table 5.1** *Modelling concepts for system representations (adapted from [Porciúncula et al., 2016])*

In a product development context, both kinds are denoted as prototypes [Ulrich et al., 2012]. Figure 5.5 shows two physical – and due to the use of Lego bricks also parametric – car prototypes of different model purpose. Contrary to accepted opinion, interpreting prototypes as tangible artefacts, prototypes may consist of analytical (thus non-physical) models only. [Ulrich et al., 2012] categorises prototypes by means of their physical/analytical degree and comprehensiveness (see Figure 5.6). [Hallberg, 2012] adapts this classification for the use of low-cost demonstrators, mainly applied in the early design phases (conceptual design). One example of such a low-cost demonstrator used for subscale flight-testing is shown in Figure 5.7 (see [I]).

Worth noting is the absence of analytical, comprehensive models and the almost complete y-axis range through combinations of physical and analytical models like e.g. hardware in the loop or power in the loop models in Figure 5.6. These facts depict the relation mentioned above that an analytical model (or prototype) is a transformation from the



**Figure 5.5** Physical LEGO models of a car: A geometric design model (a) and a functional model (b). Nomenclature in the sub-figures according to [Ulrich et al., 2012].



**Figure 5.6** Types of prototypes (figure from [Ulrich et al., 2012], extended by [Hallberg, 2012] and adapted by the author)

architectural model and only represents a limited degree of the target domains of the final product (see Figures 5.2 and 5.3).

Besides all above mentioned topics, models – especially physical, non-functional models as shown in Figure 5.5a – provide a foundation for communication, imagination and supports imagination and supports the mental model abstraction process. Also in this area, a transition from



**Figure 5.7** *Example of a functional subscale flight testing model within the aircraft conceptual design process ([I] and [IX])*

physical (and also non-physical) prototypes towards augmented reality implementation can be observed.

## 5.2.2 Computational Simulation Models

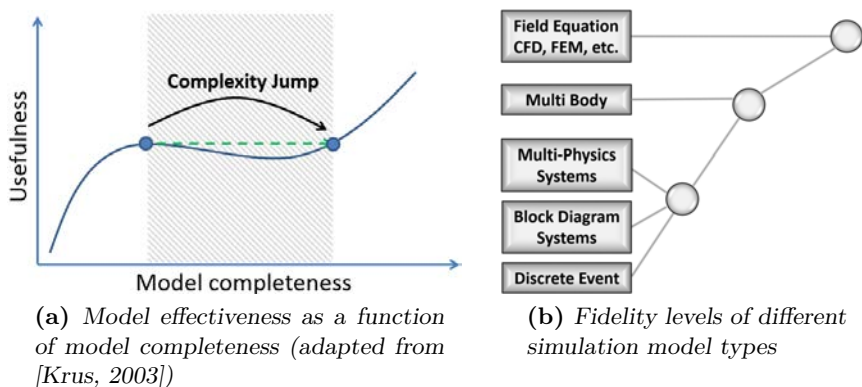
As shown in Figure 5.6, simulation models may be used for focused, analytical design exploration. For physical systems, many aspect-dependent tools exist, ranging from discrete event simulations, system simulation (e.g. *Simulink*, *Amesim*, *Modelica* and *Hopsan*), up to high-fidelity CAD and FEM tools (see Figure 5.8b). These tools are nowadays the foundation for product developing, enabling new, model-based product development processes such as Model-Based Component Acquisition (MBCA) or MBSE.

With advanced model integration concepts (e.g. Functional Mockup Interface (FMI), standards [Modelica Association, 2016]), and the use of all-embracing CAD tools, which are more and more turning into multi-domain Computer Aided Engineering (CAE) tool suites (like the *CATIA*® V6 environment [Dassault Systèmes, 2016]), simulation models are moving towards more comprehensive, more integrated models than ever seen before. This fact makes it necessary to deal with system and model(ing) complexity issues, as examined in the following chapters.

### 5.2.2.a Model Fidelity and Computational Efforts

To avoid unnecessary overhead, the degree of model abstraction and transformations should align to the required analysis level. Based on the information and independence axioms<sup>3</sup> of [Suh, 1990], a simpler model is superior to a more complex one if it fulfils the same purpose (with equal accuracy). Historically, the transition of the Geocentric model to the Heliocentric model followed the second axiom of Suh. The Geocentric model was not a wrong model, only more complex in calculation than the replacement model; from a historical perspective it was the right model – it matched the Christian philosophy.

Additionally, modelling effort is nonlinearly associated with model accuracy with (possibly) a transition zone between low-fidelity and high-fidelity models in which modelling effort increases but model improvement stagnates (see Figure 5.8) [Krus, 2003].



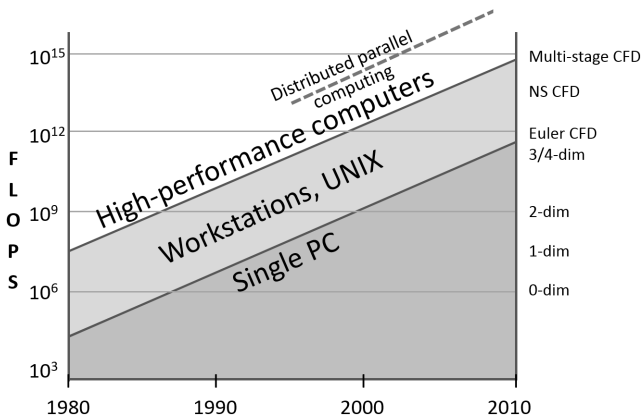
**Figure 5.8** Model complexity and modelling techniques' fidelity

Similar to the model usefulness versus model completeness graph shown in Figure 5.8a, [NASA, 2007] sketches a common trend of cost versus effectiveness of development processes. Interpreting costs as model complexity (mainly the effort to build and tune the model), it does not identify a saddle point as [Krus, 2015] does, but states that the amount of efforts spent is limited at the very beginning in conjunction with a pleasing steep increase in effectiveness.

<sup>3</sup>The independence axiom is not explained in this work. However, its principles of splitting up functions within a system (e.g. by functional decomposition) are the basis of the following system description and design automation.

Semi-empirical models – typical representatives of 0<sup>dim</sup> methods – are very efficient but due to their nature a robust, conservative estimate as long as extrapolation is avoided. The main drawbacks of these methods are related to the available data that they are based on: Limited data, design space constraints and technologies restrictions (limited to older up to state-of-the-art implementations) are some of these. In addition, the unknown precision of the data in terms of performance and design maturity, e.g. by design influence by unknown requirements such as fleet design, production and enterprise experience, can result in misleading, conservative state-of-the-art estimations [Böhnke, 2015].

Simulation model fidelity can be described as the number of dimensions (time and space) ranging from 0<sup>dim</sup> up to 4<sup>dim</sup> models. Currently, 2<sup>dim</sup> and 3<sup>dim</sup> models often focus on Research and Development (R&D) application on component level [RTO/NATO, 2007], while conventional cyber-physical system simulations are usually based on 0<sup>dim</sup> models (see Figure 5.9).

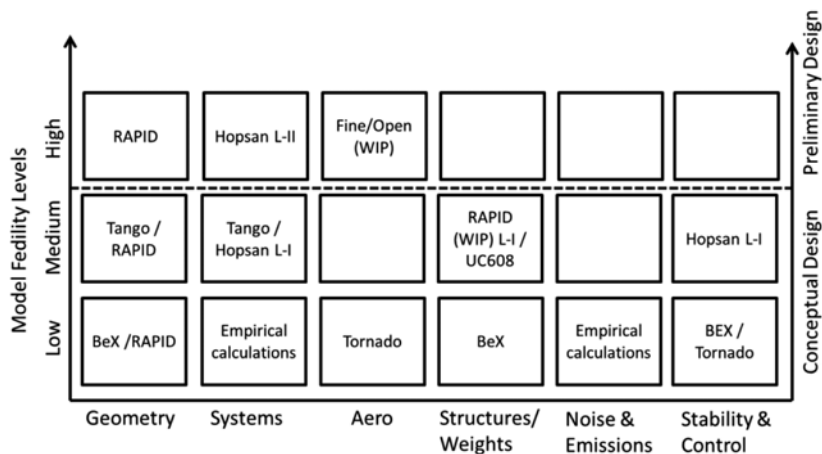


**Figure 5.9** Comparison between different simulation fidelity levels and their performance requirements (adapted from [Visser, 2015])

### 5.2.2.b ACD Tool Fidelity

In the context of ACD, simulation models of different comprehensiveness are used: from complex multi-domain SoS simulations (e.g. air combat simulations), simple single-flight mission simulations (addressing e.g. fuel consumption) down to single-domain system simulations like propulsion or on-board power systems simulations. Because model fidelity level is

related to model purpose, a tool–fidelity classification can only represent a qualitative indication: for system design such as for the hydraulic PFCS layout, a 0<sup>dim</sup> or 1<sup>dim</sup> simulation tool already represents high fidelity (on its abstraction level). However, the same simulation tool might only serve for low or medium fidelity for single component development (e.g. hydraulic pump design), and one might head for 3<sup>dim</sup> or 4<sup>dim</sup> tools to study secondary effects of interest (e.g. flow simulation for pressure spikes, pulsation, flow separation or cavitation). Alternatively to the fidelity classification by the tool dimensionality, in aircraft design a four-stage index (ranging from Level-0 to Level-3) is frequently used, describing tool fidelity by analysis type [Moerland et al., 2015]. ACD usually applies to level-0 and level-1 methods, and preliminarily level-2 and detailed design level-3 methods [Nickol, 2004].



**Figure 5.10** Model fidelity within the CADLab environment [Munjulury, 2014]

[Munjulury, 2014] states the tool fidelity classification of the CADLab framework (refer to Usecase1 on page 93) for application within the conceptual aircraft design context, as shown in Figure 5.10, adapted from a tool-independent grading by [Nickol, 2004] (see Figure 5.11). The noticeable difference is that the fidelity gap within the geometry domain for medium-level tools has been filled by the CAD environment integrated into the CADLab framework. Here, RAPID serves as a layer over the CAD environment and allows for an efficient geometry representation on a lower level than a standard CAD environment only (see the more detailed description in the Usecase1 example in Section 7.1.2.b on page



96). [Zhang, 2015] states that tool frameworks in “wing design practice evolution” – including virtual aircraft, aerodynamic, and structure – are heading towards almost reaching fidelity levels that have hitherto been restricted to physical (prototype) flight testing only. These frameworks, however, make extensive usage of 3<sup>dim</sup> or 4<sup>dim</sup> tools and are thereby restricted to the preliminary or detailed design phase.

High	CFD Methods (e.g. FUN2D, MUSEC, TETRUS)	FEM Methods (e.g. NASTRAN)	Level 2 Noise Prediction (e.g. AVATAR)			“Big Iron” CAD (e.g. CATIA, ProE)
Medium	Vortex Lattice Methods (e.g. WINGDES, VORVIEW)	Level 1 Structural Analysis (e.g. ELAPS, PDCyl/PDARb)	Level 1 Noise Prediction (e.g. ANOPP, PBOOM)	<u>GAP</u>	Vortex Lattice Methods (e.g. VORSTAB)	<u>GAP</u>
Low	Empirical Methods (e.g. EDET, O’Brinski)	Empirical Methods (e.g. FLOPS, ACSINT)	Empirical Methods (e.g. FLOPS)	Empirical Methods (e.g. FLOPS)	<u>GAP</u> Datcom? (Not well integrated)	Vehicle Sketch Pad (VSP)
	Aero	Structures/ Weights	Noise	Emissions	S&C	Geometry

**Figure 5.11** Tool fidelity level of different aerospace disciplines within a variable fidelity framework (by courtesy of [Nickol, 2004])

### 5.2.2.c Uncertainty Handling

Tightly coupled with the fidelity level is the uncertainty level of the results. Especially when dealing with – and merging – the results of analysis methods of different fidelity, some measure addressing the uncertainty is needed. In the classic approach, without such measures, uncertainty management is solely based on the experience and knowledge of the user<sup>4</sup>. As illustrated in Figure 5.8a, an increase in model fidelity may come at the cost of deteriorated model accuracy in certain cases, typically on low fidelity levels in the transition from semi-empirical to low-level analytical methods. Addressing uncertainty is a difficult task for several reasons:

- Quantitative uncertainty handling within the model (as shown by [Eek, 2016]) needs significant modelling overhead that might coun-

<sup>4</sup>Respectively the model provider.

terbalance the efficiency target between spent effort and resulting quality.

- (b) Different kinds of uncertainties – aleatoric and epistemic – that have to be addressed [Chakraborty and Mavris, 2016]. During ACD, especially the former predominate in particular (due to lack of knowledge of the product) and cannot be modelled.
- (c) A global tool-based fidelity classification (as presented in Figure 5.10 and Figure 5.11) might be misleading because of the result accuracy variations of one tool applied to different topics (such as lift versus drag in the aerodynamics domain). Furthermore, it leads to losing sight of the distinct project features that may require higher or lower fidelity in a certain task than on average. This requires a project- and topic-specific scale of what should be assessed as low-, medium- or high-fidelity.

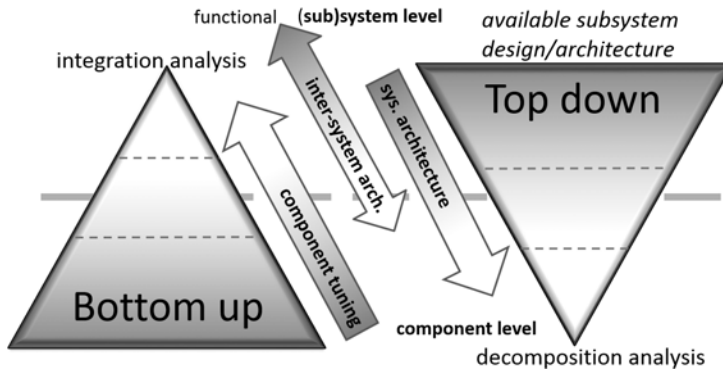
### **5.3 Engineering System Design: Process Integration**

Product development SE concepts are normally based on modelling and simulation (MBSE). An overview of frameworks that enable the communication/process between the design task (induced by the stakeholders), modelling and simulation can be found in [Haveman et al., 2015]. Further SE references are [INCOSE, 2015], [NASA, 2007], [DoD, 2008] or [Martin, 1996]. Four main SE approaches are identified by [Herzog, 2004]:

- I. top-down analysis approach
- II. life cycle orientation
- III. requirement analysis and understanding
- IV. emphasis on the interdisciplinary approach

All have in common (during the conceptual SE stage) the fact that information gap problem between the low formality, multidisciplinary, sparse data on the one hand and the need for a strict semantic and complete knowledge of system architecture and implementation on the other:

- unification of model and other parallel R&D activities
- integration into system design process
- cyber-physical models



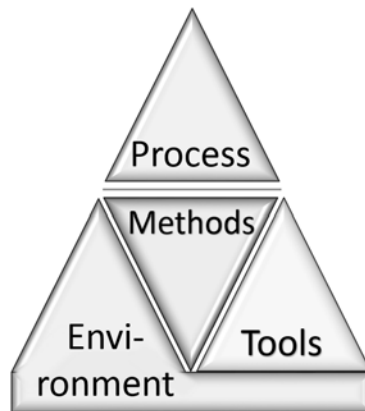
**Figure 5.12** *The simulation model implementation problem: system architecture (macroscopic) versus dependent microscopic component sizing*

Whatever approach is used during the design process is used, the top-down/bottom-up relationship problem is inevitable and has to be solved (see Figure 5.12).

During method and process development, it is important to match the surrounding and prerequisites, not only regarding technical issues but also social aspects and organisation culture. [Martin, 1994] developed the PMTE paradigm pyramid, which sets processes at the top of the methods, tools, and environment, but highlights the fact that all four parts relate to each other with bidirectional influences. Figure 5.13 shows an adapted PMTE pyramid that emphasises the importance of the environment and tools on the methods, at the same time as the process should effectuate the method.

### 5.3.1 Model Semantics and Ontology for System Engineering

The system knowledge (as analysed in Section 5.3.2 on page 64) has to be structured and saved in a concise, unambiguous manner to make it available for automated processes. The implicit knowledge in particular has to be provided by defining the context of the knowledge and the vocabulary.



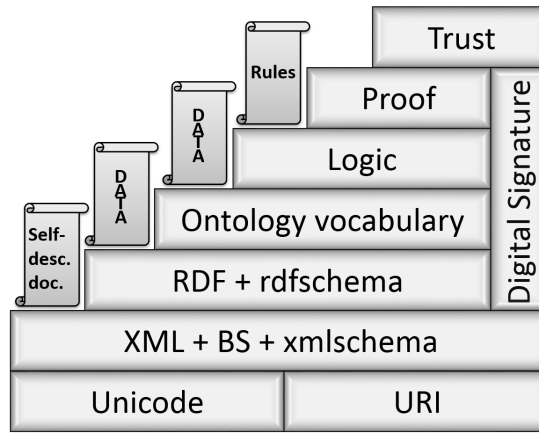
**Figure 5.13** The adapted PMTE Paradigm Pyramid with the process (definition) at the top and tools and environment(s) at the bottom (adapted from [Martin, 1994])

#### 5.3.1.a The Semantic Web Approach

One approach for adding semantics to any (sort of) data (e.g. a model) and thereby making it machine-processable is the Semantic Web approach, invented by [Berners-Lee, 2000]. As shown in Figure 5.14, it is based on the XML language. The descriptive base is the Resource Description Framework (RDF) describing the resources by *Subject-Object-Literal* triples [Daconta et al., 2003]. These triples can be represented as a directed graph (see also Usecase3 on page 115) or interpreted as a normal subject-predicate-object sentence in the English language. Resource Description Frameworks (RDF) can be grouped as containers or as reification, which is one of the differences between an RDF (a *graph*) and an XML (a *tree*) model [Berners-Lee, 1998]. On top of the RDF/RDF schema, the ontology of the (RDF) vocabulary is defined (e.g. by Ontology Web Language (OWL)).

#### 5.3.1.b XML as a Universal Exchange Format: On the Way Towards a Universal Model?

Already in 2003 B. Johansson et al. showed the use of web service standards. Based on an XML-based data repository, that allows for integration of distributed models for system simulation and optimization via standardised interfaces through a sequencer [B. Johansson et al., 2003]. Later, [Larsson, 2006] presented a strategy to derive application-



**Figure 5.14** *The Semantic Web layers, based on XML (adapted from [Berners-Lee, 2000])*

dependent equation-based models (in any language) from that on an XML basis, **Hopsan** as well as **Modelica**. Other approaches, like [Braun, 2015], make use of simulation language inherent pros and cons and unify the interface between the languages by means of standardised interface definition that includes executable simulation files, called FMI [Modelica Association, 2016]<sup>5</sup>. Both cases rely on XML as the semantic foundation for knowledge/data exchange. [Pop et al., 2003] show a **ModelicaXML** notation, based on the ideas and concept of the semantic web, introduced in Section 5.3.1.a on the facing page that would allow for an application-independent export of the model and offer additional debugging possibilities through (OWL) validation processes. Various standardised XML-based nomenclatures make use of the fixed semantic of tags in the XML format and the sound integration of validation schemas. Extending the view to other areas, these are for example **eCl@ss** (for product classification), ISO/IEC 81346 standard series (Building construction, Reference Designation System [Baslev, 2015]), and **TEI** (Text Encoding Initiative; for literature research). Other well-established XML formats include XML conformal HTML (**XHTML**) for web pages and Scalable Vector Graphics (**SVG**) for two-dimensional vector graphics. The widely used **FreeMind** program is also based on a graph-like XML data structure (see Usecase1 on page 93).

<sup>5</sup>The container/setup information of an FMI is also in XML format.

### 5.3.2 System Knowledge in Knowledge-Based Engineering

According to [Van der Laan, 2008], KBE knowledge can be split into two parts: factual and heuristic. The first is common sense knowledge, publicly available and state-of-the-art; the latter is specific knowledge (of persons, societies or companies) and usually based on experience. Another approach is to split knowledge into product and process knowledge [Van der Laan, 2008]; both are needed in a KBE process. In a classic approach, according to [Stein, 1995], knowledge-based systems may be build up by the following domains:

knowledge-based system = domain-independent inference engine  
+ domain-specific knowledge base  
+ problem-specific database

KBE is not limited to classical implementations; it might be coupled with databases (e.g. MySQL) or realised by graph-based design languages as shown by [Groß et al., 2012]. There, the types of design rules are categorised in the following manner:

- axiomatic rules      without input/preconditions
- insertion rules      with some existing instances
- architectural rules    establishing links and relationships between existing instances
- modification rules    that may add but also modify existent instances
- non-visual rules      coded rules, lacking a graphical representation.

KBE is particular advantageous within aerospace industry/application for the following reasons:

- strict and comprehensive rules of certification
- extremely long product life cycle time: problem of saving expert knowledge and maintaining training status
- huge enterprises, lot of
  - heuristic knowledge (not taught at school)
  - statistic/feedback from the predecessor product and tight/long-term customer relationships

The strict certification rules (for civil transportation aircraft under the terms of JAR/FAR-25) seem to make KBE especially beneficial because these regulations impose a lot of the design rules (or limitations). Furthermore, they are quite static and project-independent such that the work can be reused. In contrast, certification rules are usually vague, technology-independently formulated such that a direct (technology/instance-specific) rule derivation may not be possible.

Certification rules are often related to reliability issues. Whereas system reliability assessments (e.g. realised by an FTA) may be directly included and automated, the more sophisticated, “indirect” reliability assessment of derated or alternative system modes is harder or impossible to integrate. In a conventional design process, system failure modes and their consequences are investigated by a systematic analysis such as FMEA. Even though this is a systematic approach, it is often conducted manually, e.g. using a spreadsheet. Problematic for automation are the intuitive “what if” case processes during the investigation as well as the creativity needed to come up with clever (function) relocation and load balancing of the flawed system function(s). In modern, tightly integrated systems (such as the more or all-electric aircraft), it is both a blessing and a curse action due to

- the sheer infinite modes of failures<sup>6</sup>
- the sheer infinitely enlarged control and alternative paths in the system design

One benefit of integrated system design is the ability to effectuate system reliability through a global approach. Taking into account several systems – both on the system level of interest as well as upwards/downwards in the system hierarchy – enables the use of alternative components or systems for a certain (failed) function. Instead of the concept of (two parallel) redundant systems each capable of taking over the whole load of the failed system (possibly on a derated level)<sup>7</sup>, this approach

---

<sup>6</sup>As an example of number of (unimaginable) failure combinations on a modern aircraft design see A-380 Qantas flight 32 [ATSB, 2013] where a single failure (uncontained engine failure) resulted in 53 primary error messages that took approx. 50 minutes for an experienced crew to work through.

<sup>7</sup>However, mainly due to enhanced component reliability, the number of redundant systems seems to shrink in modern design, as seen from the number of engines on civil aircraft (down to two even for Extended Operation (ETOPS)) or the number of ECS packs (with only the Boeing B-747 and Douglas DC-10 with three packs).

avoids over-dimensioning of a system due to reliability requirements. The main problem with the derated, relocated, or load balanced system performance investigation is its behavioural dependency, which is not described by the functional and structural models (in the KBE approach). Thus software and control regimes/logic have to be part of the KBE approach if reliability and system safety issues should be included. Systems with a high saving potential are the Electric Power Distribution System (EPDS) and the ECS/heat management system; both are characterised by a high DAL category, large power spikes in the operation cycle and a high degree of system integration (tight integration with many couplings and inter-system dependencies)

### **5.3.3 Visual Model Representations**

As shown earlier in Section 5.2, models may lack a correct and comprehensive graphical representation. Not until a graphical representation is present does a model become powerful and useful; tools (and frameworks) get intelligible, and thereby understandable to the user, enabling an efficient way of cognitive (system) understanding (transfer of knowledge) and situation awareness (overview) [Jändel et al., 2016]. This overview can in return lead to new inputs or adjustments to the underlying (meta)model by the user. Closing the loop between computational data processing and human decision-taking, graphical representations are an important factor within modelling and simulation (as illustrated in Figure 5.15), or, as stated by [Mavris, Pinon, et al., 2010], model visualisation<sup>8</sup> “...*reduces the user’s cognitive burden by combining and leveraging both human and electronic data processing strengths and capabilities*”. One example of a close human-machine domain link is the use of Digital Mock-Ups (DMU) in an integrated product development (IDP) process [Holmberg, 2000].

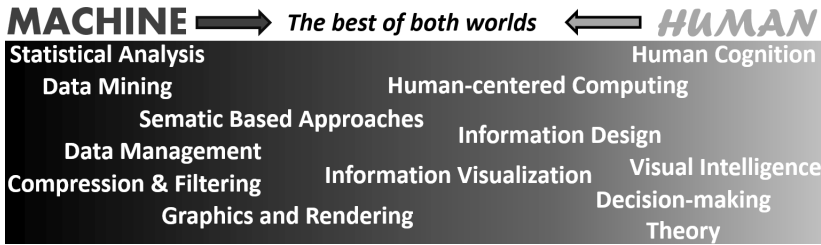
#### **5.3.3.a Human Cognitive Aesthetic**

One strength of human cognitive capability is the decision-making with incomplete data; however, cognitive capabilities are more limited than normally anticipated (known as Miller’s law) [Miller, 1956]. Based on

---

<sup>8</sup>[Mavris, Pinon, et al., 2010] use the more general word *information* (visualisation) instead of the here more precisely used term *model*. However, with the scope of representing architectural (meta)models, the word *information* may be replaced with *model*.



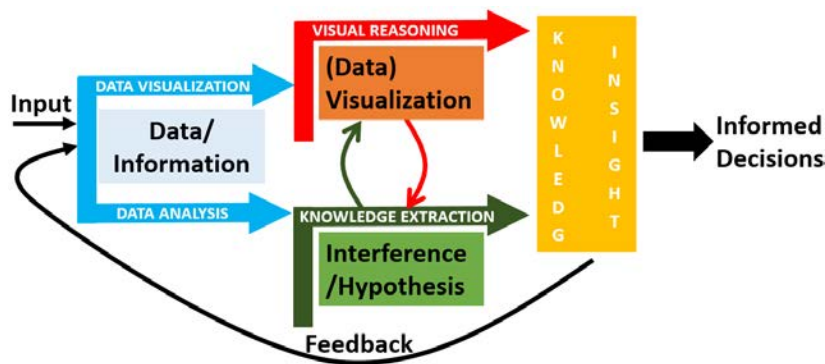


**Figure 5.15** Human and electronic data processing comparison with information modelling as a link in-between (adapted from [Mavris, Pinon, et al., 2010])

short term decision making, Miller showed that humans tend to lose their decision accuracy when the problem (number of solutions) becomes bigger than 2-3 bytes (seven solutions on average). If applicable, the hierarchical model structure (flat vs. deep) should therefore be adapted to serve both human cognitive and aesthetical needs. The latter is tool dependent and depends on the graphic rendering.

### 5.3.3.b Graphic Tool Implementation

Graphical system (simulation) representation problems cannot be reduced to the model hierarchy only but also relate to human cognitive capability and drawing aesthetics. While there are no strict criteria for aesthetics, certain rules should be followed to attain a well-arranged representation; avoid edge-crossing, use symmetries, highlight parallelism, orient edges on vertical, horizontal or diagonal lines and arrange them evenly in space are some of these [Y. Hu, 2006]. User interaction from data may be done through data visualisation, data analysis (hypothesis generation) or the data analysis visualisation (see Figure 5.16) [Mavris, Pinon, et al., 2010]. For an (ACD) framework, this means that both model (data) visualisation for model editing as well as result visualisation (through data analysis of the results from different sources) should be integrated. In the case of a **Hopsan** simulation model, the graphical limitation is primary due to the number of connections (thus edges in the graph notation) and secondary due to the number of components. For good readability, crossing and stacked connections should be avoided. Clustering connections into buses should be considered to enhance graphical representation of complex systems to prevent confusing overloaded layouts. It might also be helpful to add user-defined



**Figure 5.16** The visual analytic and enabled reasoning process (adapted from [Mavris, Pinon, et al., 2010])

directional indications to highlight the causal relationship.

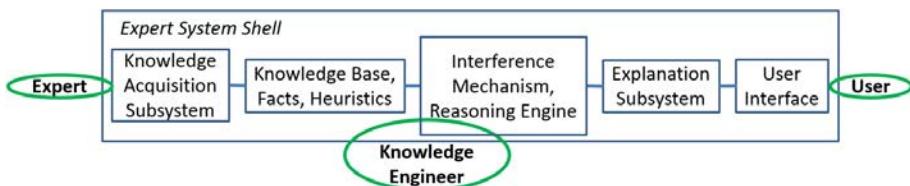
# 6

## System Modelling Techniques

*In this chapter, approaches for modelling the (system) models are presented, based on the system and model theories presented in Chapter 4 “Cascaded Systems Design” and Chapter 5 “Model-Based System Design”. Alongside the model theories, processes, and notations, implementation examples -- where necessary and applicable – are given.*

### 6.1 KBE Model Translation Methodology

Based on the theory presented in Section 5.3.2, the implementation of the KBE process is the computer application that makes use of the stored knowledge for solving problems in a specific domain ([La Rocca, 2011] [Edward et al., 1993]). The main topics of KBE are the knowledge base and the reasoning engine [Edward et al., 1993], see Figure 6.1. The



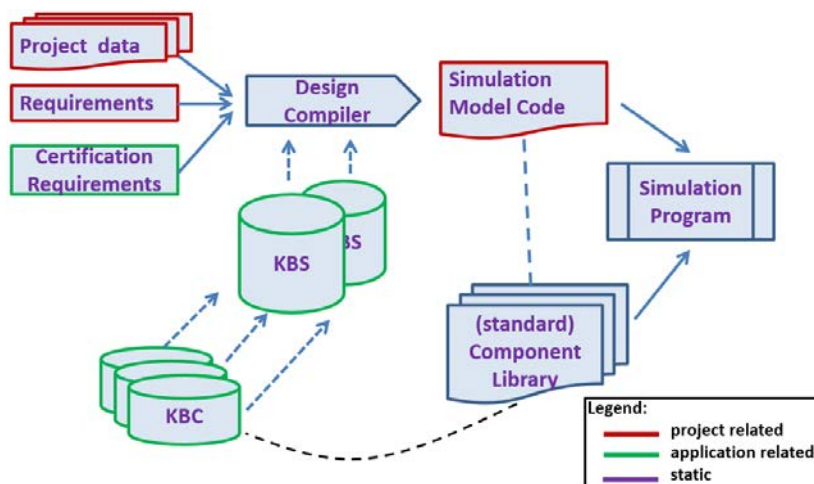
**Figure 6.1** KBE/expert system tools and the interaction of the involved active stakeholders (adapted from [Edward et al., 1993])

reasoning engine, also called a design compiler (see Figure 6.2), can

be of forward/backward chaining inference techniques (*IF – THEN* rules) up to case/frame-based reasoning implementation [Edward et al., 1993]. The latter builds on the idea of storing relevant knowledge of an object in a single data structure denoted as a frame. Synonymous in programming is the OOP approach, where objects are addressed by classes that contain both data (properties) and routines (functions) for data calculation, handling and editing. A frame-based system can thus be seen as the application of OOP paradigms in the field of expert systems [La Rocca, 2011].

Within the transition (instantiation) step from pure project data to a simulation model, the combination of two processes is essential:

- transforming the project data by means of an interpreter into the simulation system with its components and parameters
- adding the additional information of the (pre-known) Knowledge Base (KB) such as the general architecture of a subsystem and the required components (instances)



**Figure 6.2** Schematic of the KBE-based translation process, including a component (KBC) and one or several system (KBS) related knowledge bases

Figure 6.2 shows a comprehensive overview of a system and system component-based KBE approach, adapted for the conceptual aircraft design phase process. Project domain related data (shown in green)

extends the specific project-related data (shown in red); the domain-related data might, for example, be a KBE library for JAR/FAR-25 aircraft system knowledge (see Usecase2 on page 100).

## 6.2 Dependency Structure Matrix (DSM): Meta-data Modelling

The Dependency Structure Matrix (DSM), also called design structure matrices (DSM)<sup>1</sup> is a widely used matrix to describe, visualise and edit relationships between and within organisations, processes, and product modelling. It is based on a square matrix and may be applied to any abstraction level, leading to four main categories of DSMs ([Eppinger et al., 2012]):

- component-based
- people/team/organisation-based
- activity-based, and
- parameter-based.

DSM was first mentioned by D. Steward in 1981 with a focus on process management [Steward, 1981]. Characteristic of a DSM are the identical elements and element sorting line-wise and column-wise. This distinguishes a pure DSM from the related Domain Mapping Matrix (DMM) (different column- and line-wise elements) and the Multiple Domain Matrix (MDM), a combination of DMM and DSMs [DSMWEB, 2015]. An overview of the growing number of DSM-related publications can be found in [Browning, 2016]; tools and basic concepts are explained in [DSMWEB, 2015]. Alternative names for DSM are interaction matrix, N-squared or N<sup>2</sup> (N<sup>2</sup>) matrix/diagram [NASA, 2007].

In the following, the focus is on static component-based DSMs only with the purpose of system architecture modelling. Physical system DSMs thus represent component-based DSMs and are mainly used for structural design and pattern recognition. A product development approach utilising all four DSM domains listed above for system decomposition and integration can be found in [Browning, 2001].

---

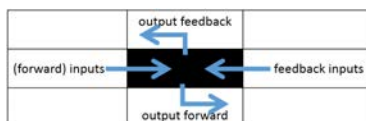
<sup>1</sup>Although Design Structure Matrix (DSM) is the most commonly used DSM denomination, the more specific and non-ambiguous denomination Dependency Structure Matrix (DSM) is – if appropriate – to be preferred and is used in this work.

DSM Data Types	Representation	Domain	Application	Analysis Method
Team-based	Multi-team interface characteristics	static	Organisational design, interface management, team integration	Clustering
Component-based	Multi-component relationships	static	System architecture: engineering and design	Clustering
Activity-based	Activity input/output relationships	time-based	Project scheduling	Sequencing, Partitioning
Parameter-based	Input/output relations of computational tasks	time-based	Low level activity sequencing. Design of computational process	Sequencing, Partitioning

**Table 6.1** The four different DSM data types, their application and the usually applied analysis methods (figure adapted from [Yassine, 2004])

## 6.2.1 DSM Nomenclature and Notation

Two different DSM notations exist: The IR/FAD convention with the forward DSM component relations in the lower triangle and the feedbacks on the upper triangle and the IC/FBD convention which is the other way around. This thesis uses the IR/FAD notation (see Figure 6.3). Mathematically, the representations can easily be exchanged for a linear algebraic transpose of the matrix. A (non-symmetric) DSM



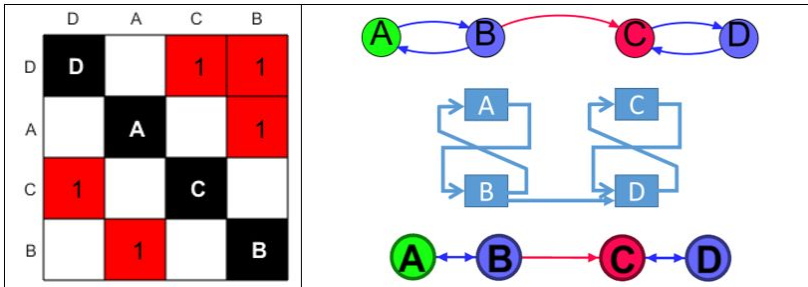
**Figure 6.3** Reading pattern of the IR/FAD input in rows DSM notation used in this work

is a matrix representation of a (directed) graph [Yu, Goldberg, et al., 2009], representing the causal dependencies between components. This fact enables the application of graph theory to the design problem. Depending on the analysis topic, DSM or graph representation may be preferable. Unlike the graph model, the DSM matrix is the graphical representation of itself in a compact, flexible, scalable and concise format [Eppinger et al., 2012]. Usual DSM operations are (according to

[Yassine, 2004]):

- **partitioning:** The process of reordering (manipulation of DSM rows and columns)  
**Target:** DSM transformation into a lower triangular form<sup>2</sup>
- **clustering:** The definition of DSM element subsets that are mutually exclusive or minimally interacting  
**Target:** Minimisation of inter-cluster connections
- **tearing:** Selection of feedback marks to be removed to enable a lower triangular matrix  
**Target:** Elimination of feedback elements
- **banding:** Alternating colour marking bands to show independent (parallel or concurrent) elements. Feedback marks are usually not considered in this action.  
**Target:** Visual highlighting of parallel/concurrent elements (no change in DSM shape)

While the latter is a graphical modification only and tearing is usually not appropriate for a (physical system) element-based DSM, only partitioning and clustering are topics discussed in this work.

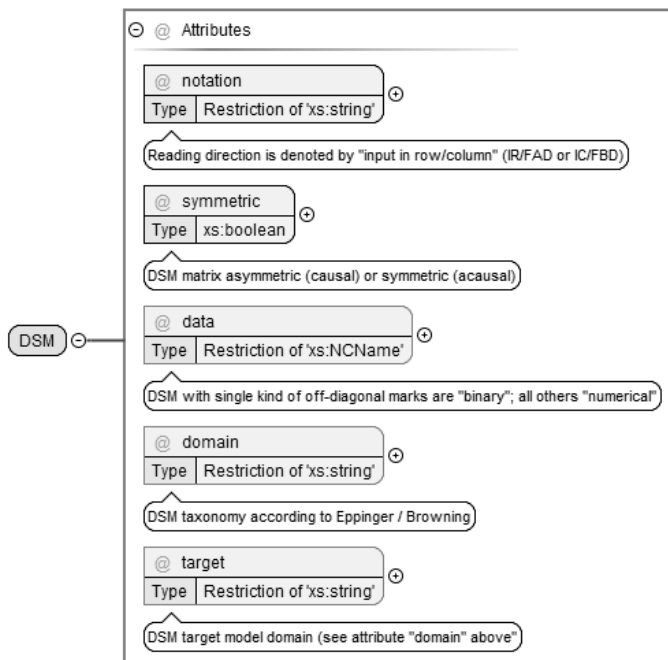


**Figure 6.4** Example DSM matrix representation (left) and different graph representations of a *Hello World* physical component DSM (see Figure 6.11 on page 85). DSM notation is IR/FAD.

### 6.2.2 Sorting and Hierarchical Clustering of DSM

Although DSM clustering and sequencing (partitioning) pursue different objectives, they are associated in that clustering usually involves

<sup>2</sup>In a physical system DSM, it is unlikely that a pure lower triangle exist.



**Figure 6.5** *DSM attributes required for a distinct DSM setup (authors own notation)*

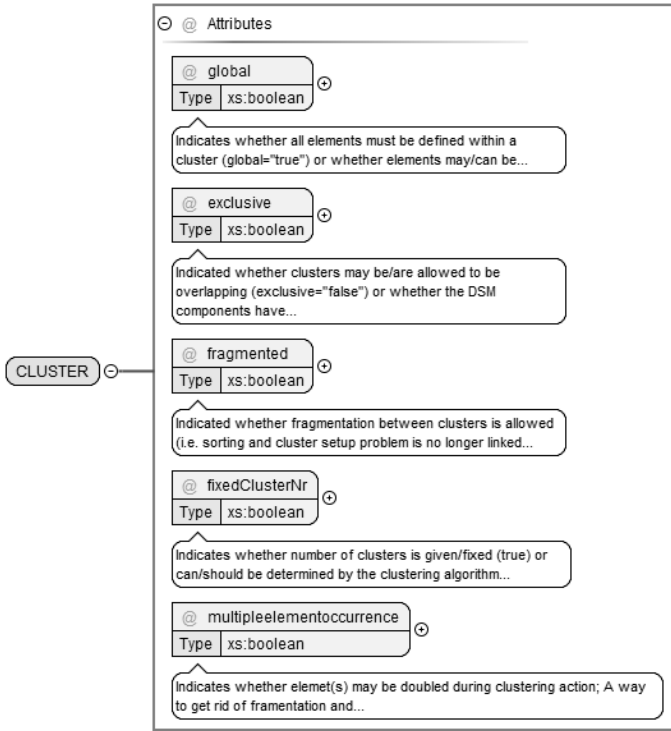
sequencing of the DSM. In (physical system) DSM, clustering enables a smart way of partitioning system components into different system groups (aka clusters) depending on their relationships and properties. A benchmark to the DSM cluster solution may be based on the cluster properties, e.g.:

- cluster size(s)
- cluster modularity
- cluster density
- (element) multiplex (only in non-exclusive DSM/clusters)

and/or the remaining structure of the DSM such as:

- number of relationships (in the case of non-boolean DSM; this represents an abstract relationship density value and not the absolute number of connections between the elements)





**Figure 6.6** DSM cluster attributes required for a distinct cluster setup (author's own notation)

- weighted number of connections, taking into consideration the length of a relation (the distance between the two linked elements)

Common problem within MDO is the precise objective function formulation with a proper weighting of the underlying single objectives. Applied on a  $2^{\text{dim}}$  matrix with its quadratic growth of the element relationships with the matrix size, the weighting factor problem has to be taken into account when using the for example cluster density as a design criterion. With the easiest setup, represented by a Boolean DSM with exclusive clusters of unlimited size (thus the  $1...n$  numbers of clusters), the number of possible combinations is defined by the Stirling number of the 2<sup>nd</sup> order:

$$S_{n,k} = \frac{1}{k!} \sum_{i=1}^k (-1)^{k-j} \binom{k}{j} j^n \quad (6.1)$$

where

- $k$  is the number of clusters.
- $j$  is the number of elements of the DSM.

The number of combinations literally explodes with DSM size and a small matrix with 15 elements results already in a design space of  $1.383e^{+09}$  combinations. Even worse is the case for non-exclusive (thus overlapping) clusters, where the number of combinations becomes sheer infinite.

However, for the presented use for simulation system sorting, only exclusive, cluster fragmentation-free and possibly global sorting is applied to address the/all components to different subsystems with a minimum number of inter-system connections. This method might not be appropriate if certain systems have a strong relationship with subsystems, as is typical for PDSs. [Helmer et al., 2010] address this problem in their research and call these systems bus systems. These bus systems can be solved by the application of an overlapping (thus non-exclusive) cluster setup or duplication of the bus components within the matrix.

### **6.2.2.a Sequencing/Partitioning**

Several approaches for lower-triangle sorting of  $2^{\text{dim}}$  matrices are known, with Tarjan's graph algorithm one of the absolute favourites. This algorithm is for example also used in *Modelica* for sorting the model equation system into a Block Lower Triangular (BLT) [Fritzson, 2004].

However, with the unlikelihood of a clear BLT solution for physical system (element) DSMs<sup>3</sup>, the partitioning has to be a trade-off between the feedback relation lengths and the number of feedback loops. This weighting action is problem-specific and requires proper algorithm-tuning to the analysis objectives.

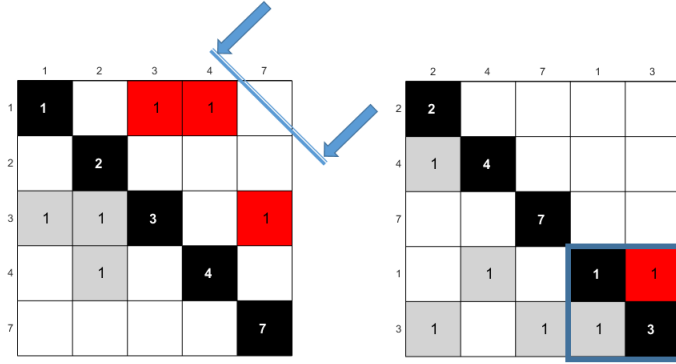
### **6.2.2.b Clustering**

The setup of a clustering algorithm very much depends on the application topic. In general, the quality of cluster setup may rely on the following classic, always existing properties:

- number of clusters
- cluster size(s)

---

<sup>3</sup>Due to the bidirectional nature of any physical connection, a BLT solution for a physical system/component DSM is very unlikely.



**Figure 6.7** Visualisation of the feedback loop lengths (cells marked in red) and the desired relocation direction within the sequencing procedure. The right matrix shows one sorting solution with the inevitable feedback loop between elements number 1 and 3.

- cluster density
- inter-cluster relationships

A basic measure can be:

$$Obj = \alpha \sum_{i=1}^M C_i^2 + \beta I_0 \quad (6.2)$$

where

- $C$  is the cluster size.
- $I_o$  is the number of outer cluster relationships.
- $\alpha, \beta$  are the cluster size and relation tuning parameters.

Furthermore, other problem-dependent cluster benchmark properties may be taken into account. Element/cluster importance measures like lead time of a process (time domain) or any other quantifier such as size property (e.g. team or code size), domain, multi-domain, price or effort may be used.

The usual approach for cluster optimisation is to use Generic Algorithms (GA) ([Yu, Goldberg, et al., 2009], [Jung et al., 2013]). By applying this method, the combined problem of clustering and positioning can be addressed individually by the dividing of the chromosome into two parts: a cluster setup part and a positioning part. Now standard GA transformation mechanisms (such as crossover and mutation) can

be applied individually to both sections of the chromosome. [Jung et al., 2013] show the application of such an algorithm to an asymmetric, numeric and global cluster configuration. During clustering, different problems such as path dependency and dimension topology issues have to be addressed [Jung et al., 2013]. These problems are related to the dimensional arrangement of the components and the clusters within the  $2^{\text{dim}}$  matrix compared with the  $N^{\text{dim}}$  target system, resulting in unresolvable inter-cluster relationships, a negative metric of the cluster quality assessment.

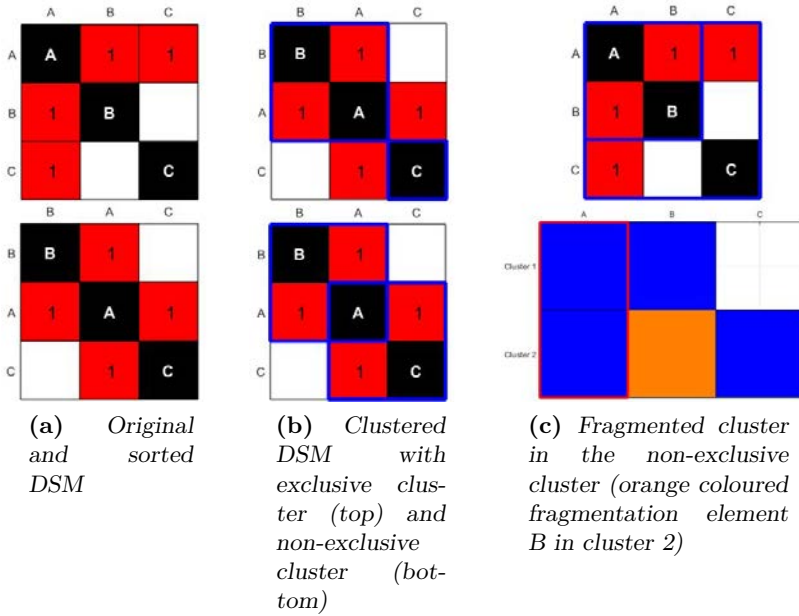
**Desirable Cluster Properties** Of great importance is a clear formulation of the intended cluster setup. A proposal for cluster configuration attributes is given in Figure 6.6. A clear formulation is essential because of the cluster algorithm notation being heavily dependent on this setup. The (low-level) implementation of the cluster sorting algorithm notation also relies on the cluster setup (see Table 6.2) and Figure 6.6).

**Cluster Setup Notations** As indicated in Table 6.2, some notations are not indistinct or even incompatible with certain cluster setup properties. It has to be pointed out that exclusive (cluster elements) and (cluster) fragmentation are two distinctly coupled attributes in the sense that fragmentation is never a desired feature, caused by an odd combination of positioned DSM and cluster setup. Fragmentation may be solvable (read: cleared) by DSM sorting only. In cases where fragmentation cannot be resolved by repartitioning of the DSM, the concept of DSM element duplication may be used, denoted by the DSM attribute `multipleelementoccurence`; an additional element instance may resolve the fragmentation, but comes at the cost of a larger, varying DSM size and challenging DSM interpretation. The DSM dimension becomes dynamic with changes in size for different cluster setups. An algorithm using this dynamic DSM size concept can be found in [Thebeau, 2001].

Cluster exclusivity of the DSM elements, meaning the non-overlapping of the entire clusters is the default feature of common clustering algorithms. However, when modelling system architectures, the identification and highlighting of linkage components that are part of two or more

---

<sup>3</sup>The `clusterMatrix` representation is ambiguous if the number of clusters becomes larger than the number of DSM elements. The main advantage of the `clusterMatrix` notation is only present if a constant matrix size of  $N \times N$  elements is used in a way that direct matrix operations can be performed on the DSM.

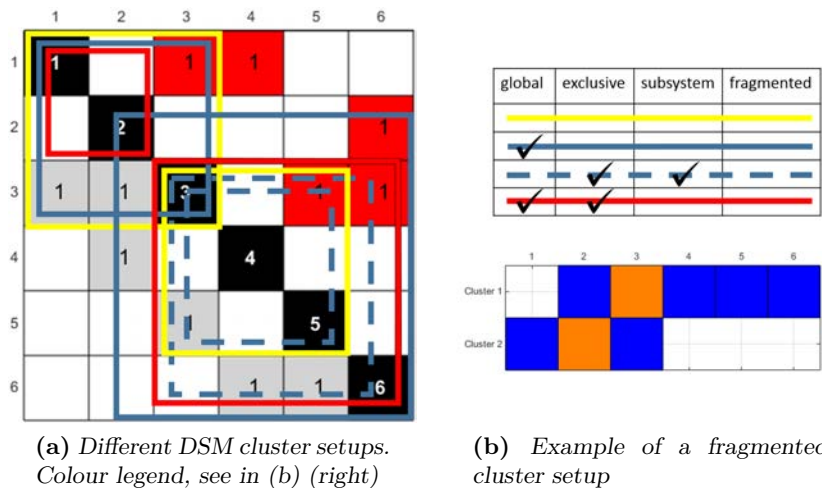


**Figure 6.8** Different DSM cluster setup examples

clusters might be of interest, especially in adaption of the so-called bus components (see [Yu, Yassine, et al., 2007] or [Helmer et al., 2010]). Another common clustering attribute is the number of clusters that can either be used as input for constant cluster number algorithms or as a result output, obtaining the best match for a given design problem.

### 6.2.2.c Algorithm Limitations

Due to the multi-objective penalty function for the sorting and clustering optimisation, the weighting factors of the single objectives have to be adjusted for every new problem, depending on the DSM properties (structure and size), the DSM and cluster attribute setup and the purpose of the clustering/sorting. Furthermore, multi-level clustering suffers from the same problems as any multi-stage processes in general (such as system or product development). Treating it in a similar way to the traditional V-model process by a decomposition (here: the clustering of the subsequent product/cluster layers) with a final benchmark being available only after the composition of the whole (cluster)system, makes it a global iterative process affecting the design process. On larger



**Figure 6.9** DSM cluster setups. Fragmented clusters cannot be displayed in the DSM matrix with the used cluster markings with coloured squares (see left side)

matrices, this results in huge time penalties due to the vast number of possible combinations.

Name	Example	Advantages	Disadvantages
<b>clusterMatrix</b> (default nomenclature)	clusterMat= [1,0,1,0,1,0; 0,1,0,1,0,1; 0,0,0,0,0,0; 0,0,0,0,0,0; 0,0,0,0,0,0]	<ul style="list-style-type: none"> <li>• Most flexible (empty or multiple elements)</li> <li>• (usually) Constant size (NxN, even if less than N clusters)</li> <li>• Enables direct matrix operations with the DSM</li> </ul>	<ul style="list-style-type: none"> <li>• Manual editing and readability</li> <li>• (memory) Size</li> <li>• Remark: Necessary to state whether linked with the original or a repositioned DSM</li> </ul>
<b>clusterCell</b>	clusterCell= {[1,3,5],[2,4,6]}	<ul style="list-style-type: none"> <li>• User friendly setup</li> <li>• Component sorting and cluster sorting already included</li> <li>• Good for indexing (e.g. in combination with labels)</li> </ul>	<ul style="list-style-type: none"> <li>• Computational speed</li> </ul>
<b>clusterVector</b>	clusterVector= [1,2,1,2,1,2]	Often used/needed within GA clustering optimisation	<ul style="list-style-type: none"> <li>• Limited to exclusive clusterSetup only</li> <li>• Erroneous if non-global clusterSetup (global = false)</li> </ul>
<b>clusterDividing-points</b>	N/A ClusterDiv= [3,6]	Often used within GA-based clustering	<ul style="list-style-type: none"> <li>• Coupled with DSM sorting</li> <li>• Limited to exclusive clusterSetup only</li> <li>• ClusterSetup fragmented not possible</li> <li>• Order of clusters coupled with DSM positioning</li> </ul>

**Table 6.2** Different DSM cluster setup notations used within algorithm implementation. All setups apart the **clusterCell** are related to a certain sorted DSM.

## 6.3 Multi-Domain Dependency Structure Matrices (MDDSM)

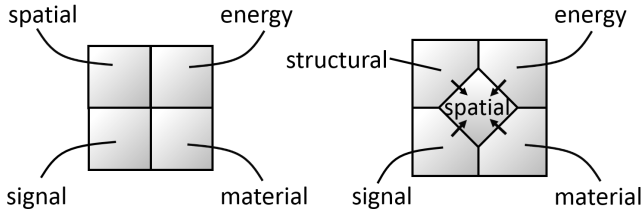
Modelling a physical (component-based) system limits the inter-element relationships to one kind of relationship only which is a drastic simplification and fails to include a great deal of system information such as:

- technology and relationship domains
- energy and material flow
- physical component placement
- the importance of the individual elements regarding product performance
- component properties (such as mass and volume)
- resource consumption (energy, material, cost, and time)
- heat radiation and other emissions

This single-domain DSM drawback can be mitigated by adding more domains to the DSM, extending the usual square  $2^{\text{dim}}$  ( $N \times N$ ) matrix into a composite  $3^{\text{dim}}$  ( $N \times N \times D$ ) matrix with  $D$  different domain relationships matrices. Another issue is the lack of a natural diagrammatic ( $2^{\text{dim}}$ ) representation of a multi-domain  $3^{\text{dim}}$  structure so that a graphical solution of the Multi-Domain Dependency Structure Matrix (MDDSM) has to be found. A possible decomposition of a  $3^{\text{dim}}$  space into a  $2^{\text{dim}}$  space can be achieved by cascading the data and presenting the third dimension within the others so that the  $D$  dimensions are represented within the  $N \times N$  cells of the first and second dimension. For product architecture (composite DSM) modelling [Eppinger et al., 2012] and [Helmer et al., 2010] propose a setup of the  $2^{\text{dim}}$  MDDSM representation where the energy, structure, signal, and material relationships modelled. This results in a four-domain MDDSM ( $N \times N \times 4$ ). A qualitative relationship assessment may be performed by either an integer [Eppinger et al., 2012] or a floating point rating in the range  $[-2, +2]$  [Helmer et al., 2010]. The spatial value calculation in the Helmer et al. notation (see Figure 6.10, right) is a reduction of the four-domain values down to the most significant value only. Manual adjustments are



required if none of these values is significant or in situations where the user prefers a solution which deviates from the merging algorithm<sup>4</sup>. A detailed explanation of the rating scheme can be found in [Helmer et al., 2010]. With the help of this scheme, the MDDSM is reduced to a normal DSM and the standard analysis methods (explained in Section 6.2.2) can therefore be applied.



**Figure 6.10** The composite DSM/MDDSM notation with energy, spatial, signal and material relationships in the four cell sectors (left) [Eppinger et al., 2012] and the enhanced concept with the spatial value derivation of the four properties used for perspective reduction.

## 6.4 Graph Theory and Representation

The element/cluster relationship perspective reduction by Helmer et al. (as explained above) is obviously a process where information is lost. This domain reduction down to a single, abstract domain is especially adverse for human system analysis, interpretation and graphical representation. For these purposes, it is preferable to change to graph representation which is – as already stated at the beginning of Section 6.2.1 – just another interpretation of the matrix data, substituting<sup>5</sup> the

- elements (or merged clusters) into vertices (V), and
- relationships into edges (E).

The DSM size  $N$  is equal to the number of vertices  $n$ , denoted as the order of the graph, and the number of edges  $m$  defines the size of the

<sup>4</sup>Due to more profound information/knowledge and individual expertise rather than the explicit MDDSM data available.

<sup>5</sup>Actually, the substitution in this case is done merely for notation convention purposes.

graph. Furthermore, the density of a graph  $G$  can also be explained by:

$$\delta(G) = \frac{m}{\binom{n}{2}} \quad (6.3)$$

where

- $m$  is the number of edges.
- $n$  is the number of vertices.

DSM	Graph	Graph: Number of
elements	vertices/nodes $V$	order of the graph
relations (inner/outer if clusters defined)	edges $E$	size of the graph

**Table 6.3** *DSM vs. graph notation*

In cluster theory, a common problem is the analysis of huge (communication) data with the targets like

- clustering graphs (into communities) or the splitting-up of networks: identifying the cut positions (weakest relation, network modularity)
- seeking the shortest distance between vertices (component relationships)
- analysing graph/cluster statistics (degree, modularity, density and centrality)

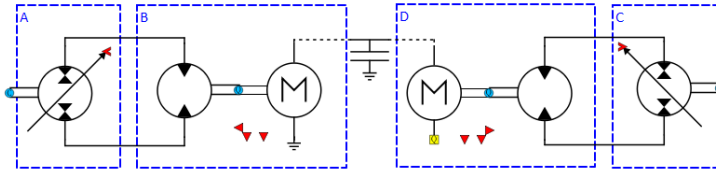
A comprehensive overview of graph clustering theory and application can be found in [Schaeffer, 2007], and an application example is given in Usecase3, Section 7.3.2. In common network analysis tools, additional attributes may be defined besides the pure vertex-edge definitions.

Related to MDDSM in Section 6.3, physical system properties may be determined as edge attributes (e.g. material, energy, structural and signal) or as vertex attributes (such as the propulsive energy domain for power elements, the component domain, the functional information and so forth). In the graphical representation, these characteristics may be depicted by

- size, shape, colour, and labels of the vertices

- colour, width, line style, and labels of the edges

in a user-selectable fashion. An example of such a graph representation is given in Figure 6.12, based on the **Hello World** system pictured in Figure 6.11 as an example of a physical multi-domain system consisting of four blocks.



**Figure 6.11** Physical circuit diagram interpretation of the system “Hello World” example, based on ISO 1219 with the four A-D blocks indicated. Solid connection lines represent hydraulic pipes while dashed lines are used for electric connections.<sup>6</sup>



**Figure 6.12** Graph representation of the system *Hello World* example with colour notation of the vertices by the drive (power input) domain and edge colour notation by the type of connection.











Through the application of network analysis tools, advanced sorting algorithms (e.g. [Y. Hu, 2006] or [Fruchterman et al., 1991]) can be used directly and adapted to the requirements of the design task. These tools may also include a `graph_ical(!)` user interface for direct user interaction (e.g. interaction of component placement during sorting), as well as extended statistical graph and cluster analysis capabilities.

## 6.5 Channel Agency Networks for System Modelling

Channel-Agency Net (C-A net) was invented in the 90s and for the first time introduced by [Reisig, 1992]. It is a modelling technique loosely

<sup>6</sup>Although this is the physical representation of a **Hopsan** simulation model, it is not an executable simulation due to the inclusion of some modelling errors but merely used for visualisation purposes.

based on the Petri net concept. Unlike Petri nets, it does not represent transition actions but works in the manner of (functional) components. Like Petri nets, a C-A net uses two distinct graphical elements, namely agencies and channels with edges in-between. Channel to channel and agency to agency connections are prohibited<sup>7</sup>. The agencies may model any functional activity which in the case of physical system modelling equals the elements or components of the system. The channels depict abstract resources of any kind. The causal edges between the channel/agency components indicate the technical<sup>8</sup> direction of the resource interaction defined within the related channel. The C-A net edges may indicate the matter category by means of a graphical notation of an arrowhead shape, induced as a more intuitive tool in graphical representation (see Figure 6.13).

Basic elements				Directed arcs		Hidden channels
Symbol	Generic name	Functional view	Structural view	Symbol	Resource type	Symbol
	Active unit	Activity (function)	Agency		Information	
	Passive unit	Resource	Channel		Energy	
					Matter	
					Energy and matter	

**Figure 6.13** C-A net element representation (by courtesy of [Porciúncula et al., 2016])

### 6.5.1 Modelling with C-A Nets

Both, [Belan et al., 2010] and earlier [De Negri, 1996] point out that a C-A net approach was mainly invented for the conceptual (product development) phase, starting out with a coarse and abstract definition. The(ir) long-term vision, however, was for this method to have the flexibility to gradually be refined alongside the development progress. For this reason, a distinct but interchangeable nomenclature is a necessity to back up this modelling approach. Possible model refinements include:

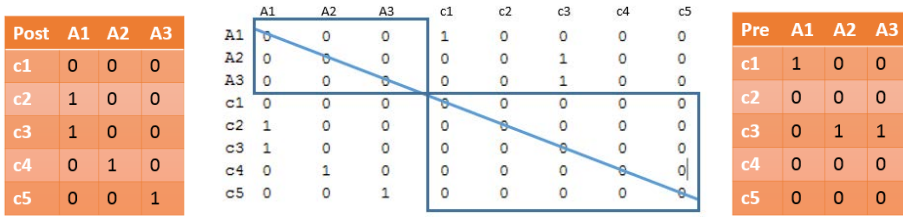
- substitution of the channel (*C*) respectively the agency (*A*) components with *C – A – C* or *A – C – A* subsystems

<sup>7</sup>Observe the similarity to the Hopsan TLM setup!

<sup>8</sup>The physical nomenclature of electric current flow direction, technical and physical, is used in the edge notation of channel resources.

- refinement of the channel
- refinement of the agency

Up to now, no distinct C-A net and tool exist. De Negri makes use of standard drawing tools to create C-A net representations which are then transformed manually into a matrix representation. A representation of this agency-matrix sorted matrix is given in Figure 6.14. The conver-



**Figure 6.14** The C-A net input (right) and output (left) matrix relationships in the pre/post notation by [Porciúncula et al., 2016] and the complete DSM, sorted by agencies and channels.

sion of the (agency-channel sorted) C-A net DSM to the *Pre* and *Post* matrices can be done by:

```
postM = sortedDSM(NA+1:end, 1:NA)
preM  = sortedDSM(1:NA, NA+1:end)'
```

where,  $N_A$  (or  $NA$ )

is the number of agencies

### 6.5.1.a C-A Net Channel Definition

According to [De Negri, 1998], three channel resource types may be modelled:

- information
- energy
- matter
- matter & energy combined.

Energy is used as an abstract instance with the meaning of its useful form for the system application ("useful energy") and not energy in a

physical sense. It is left open for discussion whether (d) represents a category of its own or is a combination of (a) and (b).

Applied on an example of a hydro-electric power plant (given by [De Negri and Belan, 2013]), the drive medium of the turbine represents the useful input energy for the system scope. The input resource "pressurised water", consisting of a matter flow (a) and useful energy (b) can thus be modelled by the matter & energy category (d) (see Figure 6.15). The output of the turbine is the depressurised water, so only matter (c) without useful energy even though the water still contains potential, kinetic, and thermal energy. From the water turbines' point of view, however, the term "useful energy" is only applicable to the pressurised input water while the outlet water no longer has any energetic value for the system it has passed. A refined power plant model with a focus on the water inlet and outlet design will however identify and require useful energy (in the form of kinetic or potential energy) for discharging the water. This creates a conflict between the channel elements connecting the turbine outlet (agency) and the discharge channel (agency). The turbine agency therefore has to be updated in the refined model as well. Care has to be taken regarding:

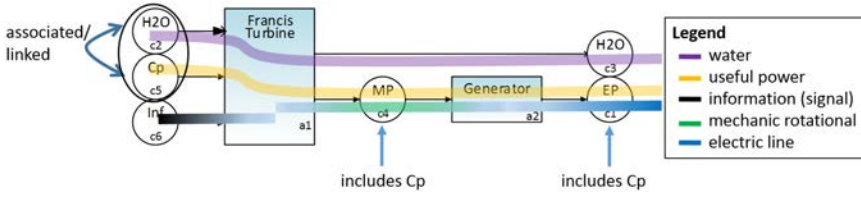
- the combination of matter and useful energy, and
- the definition of matter-less useful energy. In the currently used notation, these are electric energy<sup>9</sup>, mechanical (shaft) energy, and radiation heat energy.

#### **6.5.1.b C-A Net Agency Definition**

The agencies serve a functional purpose. An agency definition may include type checks and functional relationship information of the input and output channels. In physical power systems most components serve power conversions or adaptations (see Section 4.3). These relationships are described by the (logic) transition of the input to the output resources (see Figure 6.14). Consequently, the model architecture can be analysed regarding consistency with the help of the relationships. The incoming and outgoing channels resources on the system level can also be identified; these resources are the systems' interactions with the environment (see Figure 6.16).

---

<sup>9</sup>With a causal, user-defined functional flow direction, even in case of an alternating current system.



**Figure 6.15** Graphical visualisation of the channel properties in a hydro-electric power plant model with the transitions in the agencies described by behavioural descriptions (adapted from [De Negri and Belan, 2013])

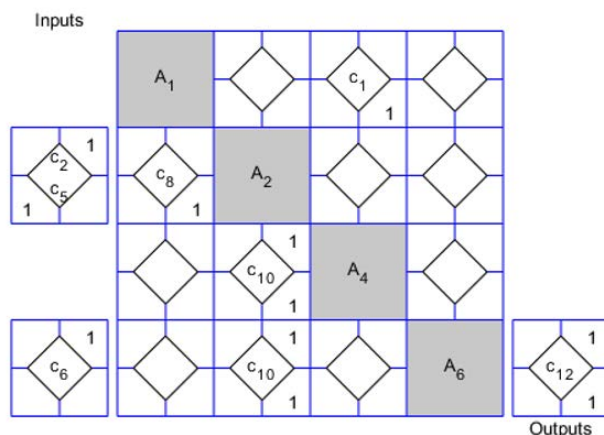
### 6.5.2 Extending the C-A Net model

[Porciúncula et al., 2016] combines the structural C-A net model analysis with a behavioural Grafcet description of the model, resulting in reduced C-A net models for every operating mode of the system. By applying FTAs on these reduced C-A net models, the reliability of the overall system can be assessed [Porciúncula, 2009]. With this approach, the structural (and partially the functional) system information is used together with the behavioural information to conduct a reliability analysis. The observant reader may already have noticed the analogousness of the C-A net, MDDSM, and the graph model; indeed, the main difference is the absence of explicit channel components in the MDDSM and the lack of spatial information in the C-A net definition. The former is a trivial problem. Any channel – apart from the input/output resources – can therefore be modelled as a single or multiple relation between two agencies. For input/output resource representation, the author suggests rendering the system boundaries in the appropriate agency line to the left (inputs) or right (outputs) of the C-A net MDDSM (see Figure 6.16).

The bipartite properties of both the C-A net and the **Hopsan** model offer an ideal symbiosis. With only minor adjustments, C-A net can be modelled with the help of **Hopsan**. The needed modifications are:

- two components, one for channel and one for agency definition with a large number of possible string input parameters, are defined
- the type of resources have to be defined by global string parameters with identical name and value content (e.g. "e\_usefullEnergy")

The **Hopsan** C-A net model can now easily be translated from the



**Figure 6.16** An extended MDDSM to represent a C-A net. Instead of the spatial value in the cell's centre, the channel's name is displayed. Note that all "location" elements ( $2^{nd}$  cell sectors) are empty due to the lack of geometric structure information in a conventional C-A net.

C-A net notation into an MDDSM or a graph set with the help of an Extensible Stylesheet Language Transformation (XSLT) (see Usecase2 on page 115).

**Extended C-A net description** Based on the C-A net criteria and criticism mentioned earlier, an Extensible Markup Language (XML) Schema Definition (XSD) definition has been initiated that fulfils all the needs and allows for a multi-disciplinary tool implementation (and thus automation). Ensuring compatibility with the MDDSM, locational information has been added to the C-A net. This information was lacking in the C-A net notation before but is valuable for the geometrical component arrangement calculation<sup>10</sup>.

<sup>10</sup>Component Placement: Any distance between two related components will negatively affect the system's performance (regarding weight, volume, efficiency, and complexity). The penalties diverge vastly and range from low impacts for signal lines over medium impacts for energy conducting lines, with or without material flow, towards the highest impacts in the form of mechanical force and energy links like linkages, pushrods or rotating axles. Additionally, effects such as resonance frequency, stiffness, and allowance for clearance may furthermore limit the placement of components containing mechanical force or energy links.



# 7

## Example Model Implementations

*Based on the theoretical parts presented in Chapters 4–6, application examples are given below.*

The presented theoretical parts had been applied to different test cases. In order to verify the theoretic approach presented in the Chapters 2–6 its practical value was tested by facing it with a variants of possible scenarios. Important test requirements were the following:

1. **Validation:** to check whether the proposed theories and processes can be adapted and implemented
2. **Verification:** to ensure that the proposed theory/process fulfils the desired objective and also improve the current state-of-the-art
3. **Deficiency Detection:** to highlight and eliminate new problems that arise due to the application and implementation or the suggested theory/process

The accompanying (use cases) implementation work was a key element in pointing out any theory shortcomings as well as impracticable concepts and implementations such as unnecessary complexity and lack of transparency for the user. Furthermore, new ideas arose and it helped to identify additional problems in areas which had not been sufficiently observed before, for example the importance and complexity of DSM sorting and clustering. The use case implementation work can therefore

be seen as an application born out of necessity for the optimal/ideal process development and turned out to be a very important part of the chosen research approach (see Section 1.3.3).

**Application Overview** The following application use cases are presented:

**Usecase1:** Conceptual Aircraft Design Framework Development

**Usecase2:** Knowledge Based Simulation Model Integration within Conceptual Aircraft Design

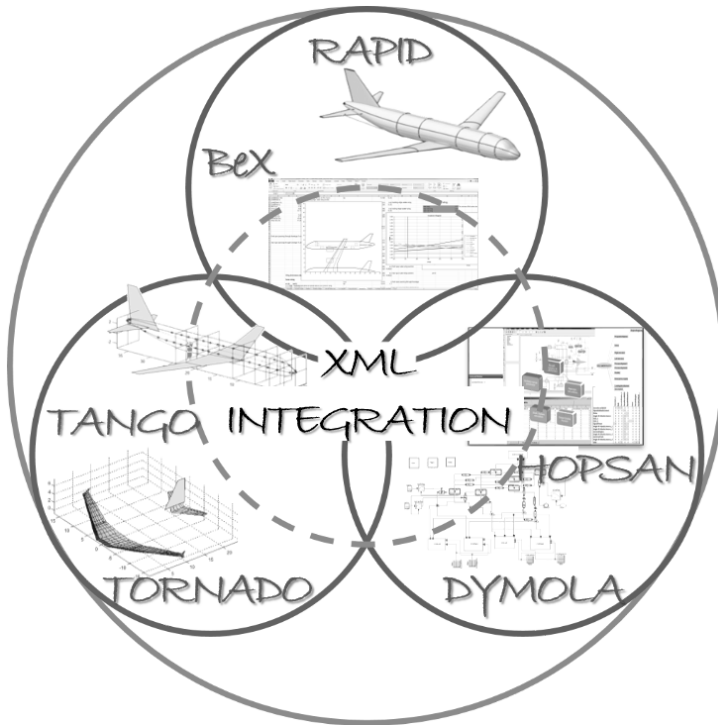
**Usecase3:** Graph-Based System Driven Design Modelling

## 7.1 Usecase1: Conceptual Aircraft Design Framework

*Based on the ideas presented in Chapter 2 “Design Influence on Aircraft Performance”, a conceptual aircraft design tool framework called CADLab [III] around a central XML database were developed around a XML database.*

### 7.1.1 CADlab Framework Overview

Based on the theory investigations in Chapter 2, the framework was built up around a parametric database as illustrated in Figure 7.1.



**Figure 7.1** Overview of the CADLab framework consisting of three main parts: CAD module, estimation, analysis and assessment module, and a simulation & system architecture module

All the modules communicate and interact with the central XML database. One of the goals of developing this framework was to enable

parallel functionality that allows the user to select the most suitable tool for any topic (a tool-dominated working process, see the PMTE pyramid, Figure 5.13). The CATIA® V5 CAD environment is integrated into the framework using a geometry-oriented design overlay named RAPID (see [VII], [III]).

This KBE-based CAD tool includes an aircraft sizing module that serves as a fast configurator for the initial layout, usually based on a conceptual sizing. In later stages of design, more detailed tasks like structure, cockpit design and system placement can be performed within the same model. An application example of the presented framework can be found in [V].

#### **7.1.1.a Framework Design**

A thorough environment and tool analysis in order to define suitable work processes and methods for the framework implementation has been performed with respect to the PMTE pyramid paradigm (see Section 5.3). Table 7.1 shows the basic outcomes of this initial framework analysis.

Based on the analysis shown in Section 3.1 “*Process Characteristics*”, the framework should support two core principles: flexibility – allowance for usage based on the “right tool for right action” principle – and support an understandable and transparent implementation.

#### **7.1.1.b Tool Communication and Database Update**

Communication between the CATIA® environment and the central database is established by VBA scripts, transferring the CAD model data into a geometry-based XML product tree. XSLT definitions are used to translate the data between the applications, see Figure 7.2. Referring to Figure 3.5 on page 26, the RAPID XML schema represents the geometry-based view whereas the **Tango** data format accounts for a mix of both, a geometric- and system-based object decomposition. Both style sheet definitions (denoted by “*Matlab*” and “*CATIA*” in Figure 7.2) present the fundamental design approach differences in the CAD and the technical computing/programming language domains.

Prerequisite	Motivation
integration of a full CAD environment	<ul style="list-style-type: none"> <li>• following a (preliminary) design process highly related on CAD</li> <li>• creation of a useful tool for geometry domain ("the right tool for the right action")</li> <li>• utilising the geometric modelling experience with CATIA<sup>®</sup>/CAD environments</li> </ul>
Matlab <sup>®</sup> programming language	<ul style="list-style-type: none"> <li>• expertise, practice and education of the staff</li> <li>• flexibility of an interpreter language (debugging capability); enables fast code adaption to project-individual needs ("efficiency")</li> <li>• integration of an existing Matlab<sup>®</sup> in-house tool</li> <li>• existence of licenses</li> </ul>
XML database	<ul style="list-style-type: none"> <li>• suitable SE standard (see research/theory parts of this thesis)</li> <li>• trend and results from other research institutes: especially the XML-based CPACS definition by the DLR (see [Nagel et al., 2012])</li> <li>• Matlab<sup>®</sup> support for Java Document Object Models (DOMs)</li> <li>• data exchange capability to 3<sup>rd</sup> party analysis tools</li> </ul>

**Table 7.1** Framework requisites (on the general framework outline)

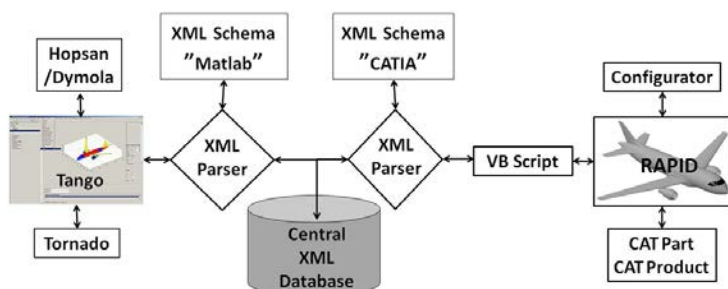
### 7.1.2 Tool Implementation

Due to the domain dependency on the product decomposition and the different implementation approaches between a conventional programming language and CAD (graphical environment editor) based implementation, the implementation approach between **Tango** and **RAPID** differs substantially from each other.

#### 7.1.2.a Tango Implementation

Tango is implemented using Matlab<sup>®</sup> OOP principles, mimicking the class concept known from C++/Java but with significant differences compared to these languages<sup>1</sup> [The Mathworks, Inc., 2015b]. The XML-

<sup>1</sup>OOP was introduced in Matlab<sup>®</sup> Version 2008a based on a new object-oriented framework, though only in a rudimentary fashion. Since the object-oriented capabil-



**Figure 7.2** The data translation (parsing) action within CADLab between the two main applications *Tango* and *RAPID*

based import/export routines make use of the underlying Java DOM application classes in *Matlab*<sup>®</sup>. The data is handled object-orientedly within geometrical or functional classes in such a way that every class includes its class-related XML parsing functionality. This method allows for greater flexibility and quick replacement or appending of new classes. As a consequence, the “*Matlab*” schema (in Figure 7.2) represents a one-to-one copy of the class arrangements within *Tango*. Since *Tango* is class-based implemented according to the OOP principles presented in Section 3.3 and data arrangement is in a user-friendly setup, close to the central CADLab database layout. Every object class contains its own data parser. Besides the geometry and system modelling capability, design analysis is a topic that can be performed in *Tango*. Analysis capability is divided into two main levels. The low-level analysis (refer to Figure 5.10) is mainly based on textbook formulas like [Raymer, 2006], [Roskam, 1985], [Torenbeek et al., 2009] and [Gudmundsson, 2014]. For a higher fidelity level, more advanced, physics-based analytical models like *Tornado* or system simulations are incorporated. The latter are described explicitly in the following Usecase2 example on page 100.

### 7.1.2.b RAPID Implementation

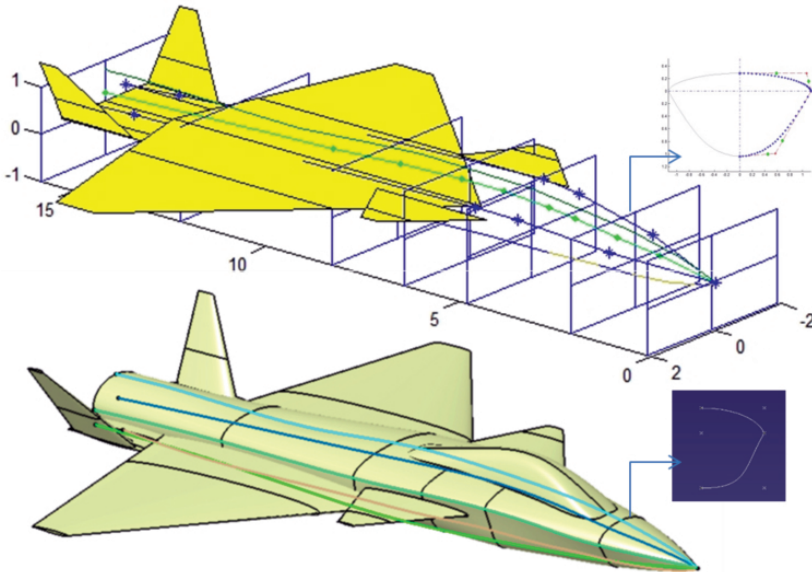
The *RAPID* tool is a CAD overlay for the *CATIA*<sup>®</sup> V5 software (see the explanation of this concept in Section 3.3.1 on page 27). The implementation is based on **Power Copy (PC)** (comparable with a 1:1 copy of a geometric instance) and **User Defined Feature (UDF)**. Both are methods for geometry-related KBE [Munjulury, 2014]. In *RAPID*, PCs and

ities are being continuously improved and extended with newer version releases.

UDFs are used to define geometric instances which match the CADLab database description.

A VBA script acts as a RAPID internal interpreter for a geometry XML dataset derived from the central XML dataset (see Figure 7.2). The templates are saved either as PCs or UDFs; PC uses VB script and UDF uses Engineering Knowledge Language (EKL) to instantiate. A VB script, controlling the PCs, acts as a RAPID-internal interpreter of the geometry-adjusted XML dataset and is derived from the central XML dataset (see Figure 7.2). The UDFs are expressed by scripts in Engineering Knowledge Language (EKL). Unlike PCs, the UDFs allow for a parameter update even after instantiation, whereas the setup of a PC is unmodifiably defined during the instantiation (the reproduction process from a design template).

The VBA script serves as both data interface and top-level KBE automation routine within RAPID. The intermediate step of adding a "CATIA"-like XML dataset between the CADLab information model and the RAPID VBA interface is done to maintain a reasonable complexity of both, the XSLT file and the VB script. This split-up allows for the language-dependent advantages and disadvantages of XSLT and VB.



**Figure 7.3** Example of different geometrical illustrations of the same dataset in *Tango* (top) and *RAPID* (bottom)

### **7.1.3 The CADLab Database Namespace**

The developed CADLab data structure focuses on the good parametric design principles, these are, according to [Sóbestor et al., 2015]: conciseness, robustness, and flexibility. Referring to the design decomposition principles mentioned above, the basic dataset setup is a geometric-based decomposition close to the one found in the CPACS standard [CPACS v.2.3, 2016].

The trade-off of the CADLab parametrisation sacrifices flexibility in favour of robustness – to allow for optimization and iterative tool changes – and conciseness – to maintain transparency and causal parameter behaviour (see Figure 3.3 on page 24). The result is a compromise, optimized for the use of standard fuselage-wing aircraft configuration. Lifting surface shapes are rather strictly defined, based on the airfoil notation by Melin ([Melin et al., 2011], [Melin, 2013]). To enable modelling of other configurations than conventional fuselage-wing designs, the rigour has been lowered by a more flexible but less concise definition of the fuselage. This object is primarily intended to model the fuselage, but can also be used for any non-lifting device modelling.

### **7.1.4 Usecase Review and Conclusions**

Implementation work on the framework revealed the problems and challenges examined in Chapter 2. The way of conducting the ACD work is highly influenced by the decision to rely on acausal programming. One challenge of CAD implementation is the absence of a universal physical model that includes the proprietary CATIA<sup>®</sup> environment; an open-source CAD such as openCASCADE may be preferable for future development. Interrelated with the acausal implementation topic (addressed in Section 3.3.1, on page 27) are the geometry handling and GUI representation (the latter has been addressed in Section 5.3.3) capabilities: even the most basic geometric bodies require a great deal of effort and should be avoided. For future framework development, the use of so-called "poor man" CAD environments should be considered.

Another tool-inherent limitation are the GUI incapacities of Matlab<sup>®</sup>. A more complex and modular environment would be favourable. Especially promising and worth looking into seems to be the Python language; while retaining the advantages of an interpreter language, it additionally offers a wide range of open-source GUIs to choose from. The OOP implementation enables flexible object adjustments.



However, compared with conventional (function-based) programming, the price that has to be paid in terms of complexity overhead. If not familiar with this concept, it requires special education and training the engineers involved.

Huge effort have been invested in the parametrisation of the geometry domain. The use of spline curves provides great flexibility to ensure design freedom (size of the design space) of complex shapes such as those needed for fuselage shape adjustments for wave drag reductions. However, distinct parameter influence may be sacrificed. As a compromise for the fuselage, nacelle and other non-lifting surface component models, cubic Beziér curves have been used for the cross-sectional shape definition while the longitudinal shape is based on spline curves [Munjur, 2014]. Another approach to reduce the number of parameters and enhancing the distinct model behaviour is to use the Singular Value Decomposition (SVD) method [Krus, 2016]. This bottom-up modelling approach identifies useful artificial influence parameters by means of statistical data or physical simulation model results. It may replace or extend a sensitivity analysis, supporting the designer to generate sound surrogate models with a design parameter reduction.

The requirement handling and the addressing of (qualitative) reliability measures remains an open topic. Here, more future work is necessary, first of all to include fidelity-based processes in the data exchange and subsequently present the fidelity measure in an appropriate manner

## **7.2 Usecase2: KBE-Based Simulation Model Integration within Conceptual Aircraft Design**

*This application example is based on but not limited to [IV].*

A total aircraft system simulation with the primary flight control (hydraulic power) system-based on conceptual aircraft data has been chosen to demonstrate the implementation (process) of the KBE methodology shown in Section 6.1. A flexible aircraft system top-level architecture can be defined as illustrated in Figure 7.15. This architecture is a logical composition of the physical instances needed to describe a whole aircraft in a flat hierarchy. The behavioural modules namely the attitude control autopilot, have been taken directly from [Krus, 2009]. The PFCS has been chosen since it represents the “perfect” on-board power system for design automation in many ways. The outstanding characteristics are:

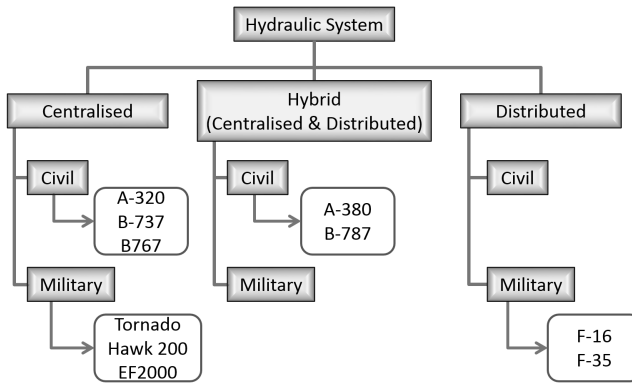
- clear and project-independent (static) functionality of the roll, pitch and yaw control
- high reliability-focused designs: a (total) system loss/failure will result in a catastrophic event, thus leading to a highly redundant system build-up (see DAL category in Section 4.2.2)
- clearly defined system interfaces: On the output side, the actuator linkage to the control surfaces can be seen as the system interface between the (P)FCS actuation system interacting with the structural/geometrical/aerodynamic model of the aircraft
- huge design space: Many different configurations can be found on historical and current aircraft
- part of the aircraft control system and for this reason
  - present in any modern aircraft
  - required for closed-loop aircraft control (simulation, calculation, concept)

### **7.2.1 Aircraft Hydraulic System Topology**

Aircraft hydraulic systems differ from stationary or (non-aviation) mobile applications. A comprehensive technical overview of commercial airliners’ hydraulic systems can be found in [AIR5005A, 2010].

The usual PFCS hydraulic actuator power system topologies can be categorized into three different system layouts (see Figure 7.4):

- centralised system
- distributed system
- hybrid system



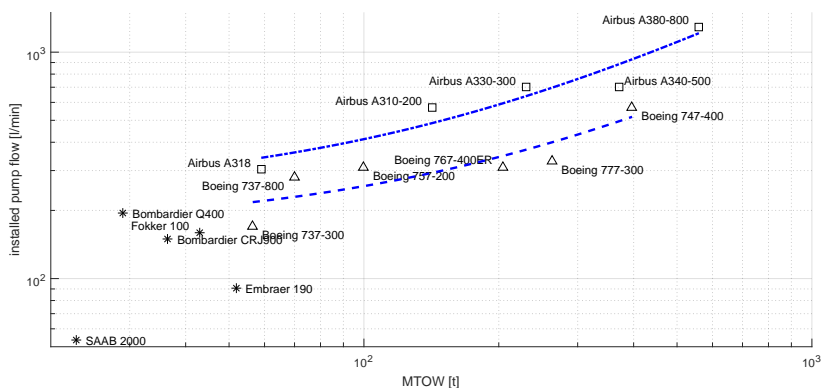
**Figure 7.4** Hydraulic system top level architecture layout concepts

The naming refers to the supply power generation concept. In civil aircraft applications, the centralised layout is usually realized with a trend towards hybrid systems in the latest aircraft generations (Airbus A380, Boeing 787) whereas in military applications both centralised and distributed systems can be found. In the following civil aircraft application example, only the centralised approach is considered.

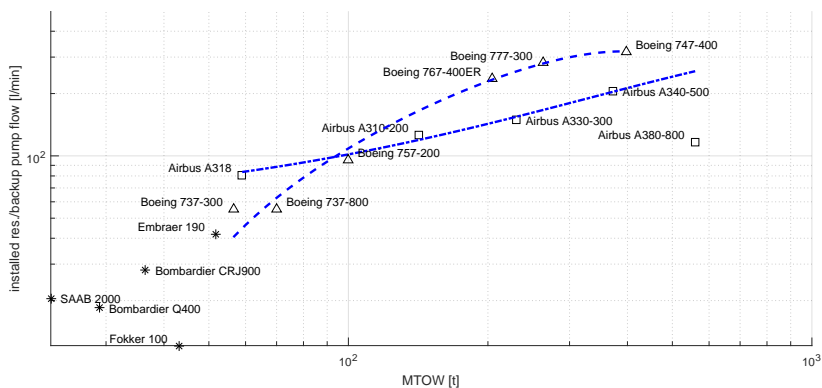
## 7.2.2 Statistical Analysis of Hydraulic Flight Control Systems Properties

Statistical PFCS analysis of current FAR/JAR-25 aircraft (e.g. based on [AIR5005A, 2010]) reveals design parameter relationships that can be used for the formulation of the KBE rules; primarily for component sizing and secondarily for system architecture setup. Even on a basic level, the system strategies differ considerably among the enterprises (see Figure 7.5).

A typical system architecture of a JAR/FAR-25 certified aircraft is shown in Figure 7.7 using the Airbus A320 family as an example. These



(a) *main pump flow capacity*



(b) *installed back-up flow capacity*

**Figure 7.5** Civil aircraft hydraulic system statistics: Installed pump capacity in relation to aircraft weight. Observe the different values and trends between Airbus and Boeing aircraft. Data on the Airbus is A-380 excluded in the trendline in (b) due to system layout difference (electrohydraulic standby system) (data based on [AIR5005A, 2010]).

types of systems usually consist of three independent hydraulic systems, here called *yellow*, *green* and *blue*. As a result of the analysed system designs (data from [AIR5005A, 2010]), only the Fokker 100 and the Boeing 747 deviate significantly from the "three hydraulic subsystems" solution with a two-subsystem design in the Fokker and a four(!)-hydraulic subsystem design in the Boeing 747<sup>2</sup>. This analysis shows that there

<sup>2</sup>The Boeing B-747 hydraulic design reliability analysis becomes especially interesting in light of the accident investigations of the last 45 years of operation: At least



**Figure 7.6** *Unconventional example of the hydraulic actuation system on the aileron of an F-104 Starfighter with ten(!) actuators for one control surface (5 each for system A and B). The actuator casing in this example is also part of the (ultra-thin) wing structure. (photo of the German 22+58 F-104 G by courtesy of Robert Dietz, [www.f104g.de](http://www.f104g.de)).*

are certain common architectural design strategies in the build-up of a PFCS system of JAR/FAR-25 aircraft. These features make KBE rule-based design practical and trustworthy.

### 7.2.3 KBE Hydraulic System Implementation

Each hydraulic system can be characterised as a closed loop containing the functional parts in form of supply system, accumulator, actuators and power transfer system (see Figure 7.8). To enable a KBE implementation according to the process explained in Section 6.1 (especially in Figure 6.2 on page 70) four categories of the Knowledge Base System (KBS) blocks have been defined:

#### I Fixed ports, static system

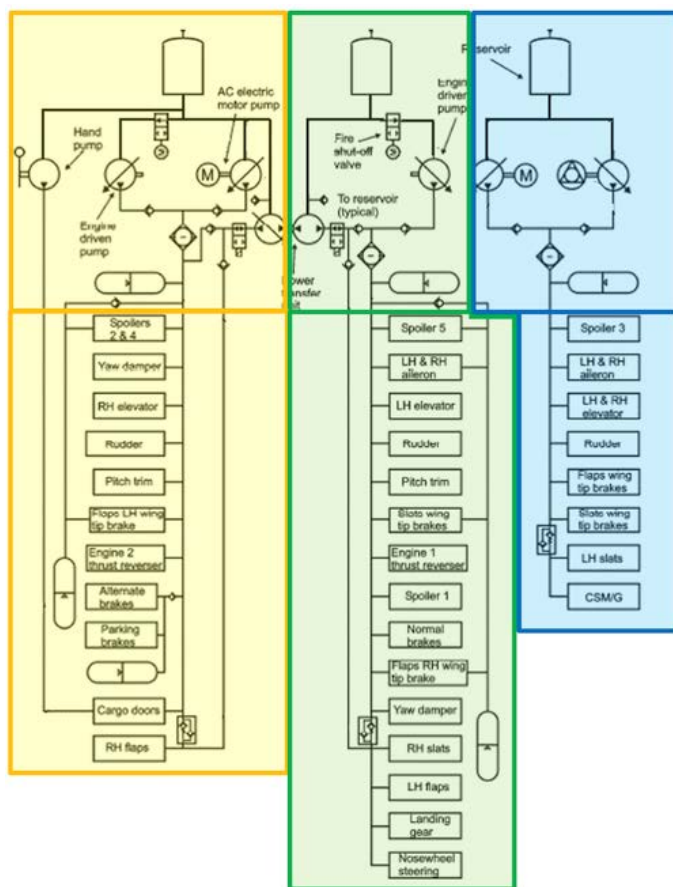
This represents a static system layout with fixed defined system ports. This type can be handled as a (complex) single component (Knowledge Base Component (KBC)). These system types require only parameter adaption during instantiation, e.g. a simple propulsion system model where only the engine deck data is updated.

#### II Fixed ports, repetitive system

A fixed system component composition with an adaptable number of occurrences. An example is the mission system defined by a

---

one accident (AWA flight 845, [Roskam, 2007], [NTSB, 1972]) can be found, where three out of its four hydraulic systems became inoperable and the fourth system enabled a safe landing, saving the live of 199 passengers.

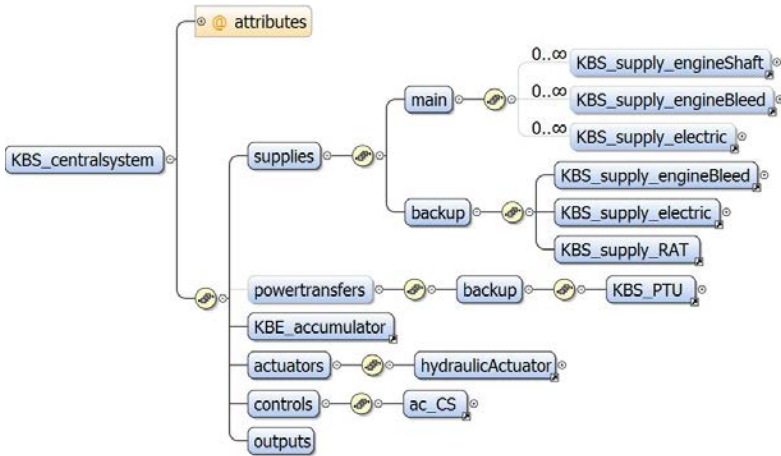


**Figure 7.7** Airbus A-320 Hydraulic system layout (adapted from [AIR5005A, 2010])

state machine (with static number and functionality of input/output ports) which works with a flexible number of repetitive elements of the same shape but with individual parameter settings.

### III Fixed port, flexible system

System with flexible internal component composition but with a fixed defined number (and functionality) of the input/output ports. An example is the PFCS controller with its roll, pitch and yaw commands (fixed ports!) but with significant changes in the controller layout between different projects (e.g. stable versus unstable configuration).



**Figure 7.8** System architecture description of one centralised hydraulic system branch (based on XSD)

#### IV Flexible ports, flexible system

A system with flexible system component composition as well as a varying number and functionality of the input/output ports. An example is the hydraulic actuator system.

These Knowledge Base System (KBS) category definitions limit the possible actions within the system-building process and thereby enable an easier implementation of the design compiler.

Although automated simulation model generation is the primary objective, it may not be applicable for every complex system (especially KBS category III or IV). In this case, the KBS translation may require user interaction, integrating the engineer (with specific expertise) into the system architecture generating process [Haskins et al., 2007]. This interaction can be defined as a configurator process; within the configurator, different design rules may be implemented to support the user during the designing process to prevent impossible combinations [Andersson, 2012].

### 7.2.4 Cascaded Systems Simulation Metamodel Implementation

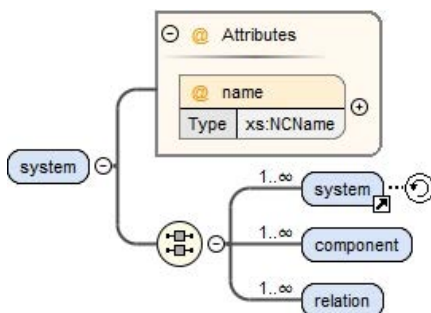
Usual 0<sup>-dim</sup> or 1<sup>-dim</sup> physics simulation tools (such as Modelica, Amesim, Hopsan; see Section 5.2.2.a) depend on two data sources:

- project-related sources in the form of
  - system architecture/relationships
  - component parameter value setup (often referred to as *tuning* or *sizing* parameter)
- domain-related (tool-dependent) component libraries.

Advanced users may also develop their own library components and subsystems which in turn can also be seen and handled as components. The easiest OOP definition of a cascaded system which may contain unlimited numbers of its class may look like the following XSD example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="system">
    <xs:complexType>
      <xs:all>
        <xs:element ref="system" maxOccurs="unbounded"/>
        <xs:element name="component" maxOccurs="unbounded"/>
        <xs:element name="relation" maxOccurs="unbounded"/>
      </xs:all>
        <xs:attribute name="name" type="xs:NCName"/>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Figure 7.9 on the current page shows the graphical representation of this XML schema (XSD).



**Figure 7.9** The cascaded system model: graphical representation of the easiest possible system schema definition



## 7.2.5 Implementation of the Hopsan Metamodel

In a **Hopsan** simulation model, project information is distributed to the following files:

- The **Hopsan** model file (ending .hmf): The project-related architecture, component tuning and graphical representation
- The library files, including the component files for every component:
  - Component header file (.hpp)
  - Component code file (.cpp)
  - Component port setup and graphical properties file (.xml)
  - Component graphic (usually a .svg vector graphic file<sup>3</sup>)

Depending on the solver strategy, the components may additionally contain parts of the solver routine. Unlike a central solver approach (like e.g. in **Modelica**), where no (direct) solver part is located in the component code, this is the case in some distributed solvers like **Hopsan** with its TLM concept. Here, the TLM solver has to be built-in every  $C$  or  $Q$  component but not in signal ( $S$ ) components (for details see [Axin et al., 2010]).

The simulation metamodel includes all the information in the simulation model file but lacks the behavioural information about the components. In short, it is a reduction of the dynamic simulation model into a static black-box model. In theory, the behavioural component data may also be included in the metamodel but the complexity of this task does not allow it to be part of the thesis<sup>4</sup>. Because of the usual simulation tool-dependent component modelling, this work has to be done for every simulation tool/language. Depending on whether the traceability between the metamodel and the model source data is ensured (and implemented!), referring to Section 5.2, the simulation metamodel serves either

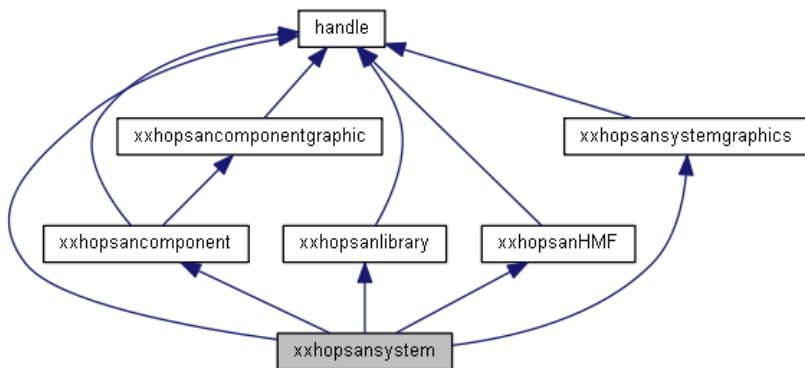
- as an *architectural* model for the different analysis models
- or

---

<sup>3</sup>A scalable vector graphic (SVG) is also based on XML.

<sup>4</sup>This concept would lead to a universal meta-simulation notation. Related work for model translations, e.g. between **Modelica** and **Hopsan**, can be found in [Braun, 2015] or [Sjölund, 2015].

- as an *analytical* model from which different (domain) representations can be derived (see Figure 7.10)



**Figure 7.10** Overview of the class arrangement of the *Hopsan* meta-model in *Matlab*<sup>®</sup>, called *xxhopsansystem*. The figure is taken from the automatic code documentation by *Doxygen*

The simulation metamodel has been implemented in *Matlab*<sup>®</sup> based on the principles of OOP. A setup of the different classes can be found in Figure 7.10. This package can be seen as an assembly of parts of the *Hopsan* core, the GUI and the component libraries. Unlike in *Hopsan*, no interactive modelling GUI is included. Instead, high-level commands are available that enable a command line like build-up, modification, and analysis of a *Hopsan*-like network of components. The same commands are used by the design compiler in an automated design process.

Even though a parser for the logical operations within every (library) component is available, the *Hopsan* core solver functionality does not<sup>5</sup> – represent a real metamodel of the simulation model.

## 7.2.6 KBE Process on the Metamodel

The KBE process is implemented on top of the *Hopsan* metamodel explained above. In this implementation example, the design rules are

<sup>5</sup>A rudimentary parser of the C++ to *Matlab*<sup>®</sup> code has been developed but never used for calculation. The *Hopsan* core could be implemented to solve the equations but the author did not consider this to be reasonable. Performance would have been much worse due to the huge execution speed differences between C++ and a *Matlab*<sup>®</sup> OOP implementation. A call of the *Hopsan* core via the *Hopsan* client interface (CLI) is the faster and straightforward option.

spread over the design compiler, the KBSs, and the KBEs. XSD files partly model the architecture of the KBSs. This offers the potential to use the file during the whole process: for the system architecture (rules) development, the documentation and the instruction source for the design compiler. All other information, the component sizing inside the KBC/KBS, and the design compiler, are implemented in Matlab<sup>®</sup>. A problem with the design compiler and KBS implementation is the handling of the *III* and *IV* order category (for a definition see Section 7.2.3 on page 103). Another issue is the interdependency or gap – depending on viewpoint – between the top-down (here: KBS definitions) and the bottom-up (KBEs including parameter tuning) relationship (see Figure 5.12 on page 61).

### 7.2.6.a Hydraulic Actuator Model Definition

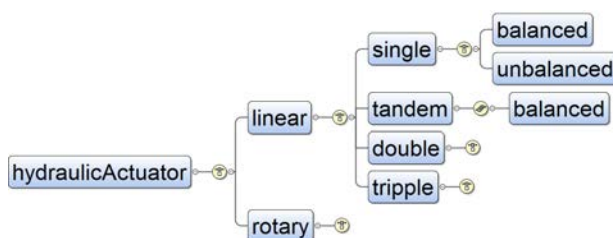
The definition (of the PFCS-related parts) of one of the centralised hydraulic systems (illustrated for the Airbus A-320 family in Section 7.2) can be described by a tree architecture as shown in Figure 7.8. Each element beginning with the prefix KBS refers to a system layout defined by the designer. The KBC prefix denotes elements that are directly linked to simulation components and thus represent the smallest possible unit that cannot contain any subsequent KBS/KBC definitions.

Defining the global system layout as **centralised** only allows the actuator to be of the hydraulic input type. Thereafter, the actuator type selection can be refined by the kind of application. A table of typical actuator type applications within the FCS is presented in [Wang et al., 2015]. Figure 7.11 shows a hydraulic actuator configuration tree which can be simplified for the present PFCS case to:

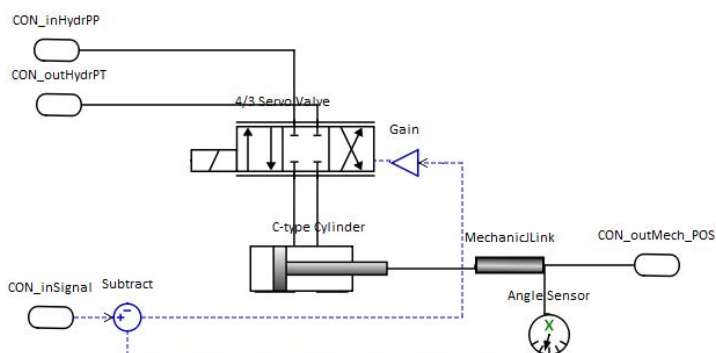
- linear type only
- civil application: usually single or double type; military application: usually tandem type

This system can be instantiated by the compiler process. In the described centralised, civil PFCS case, the actuator selection comes down to:

```
hydraulicActuator(type="linear", housing="single",
                  subtype="unbalanced");
```



**Figure 7.11** Hydraulic Actuator design tree. Double and triple linear actuators can be seen as combinations of single actuators; rotary type branch not shown.

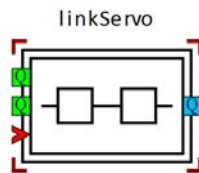


**Figure 7.12** Example of the KBS instantiation of a linear single actuator (KBS Type I) in Hopsan

This declaration is translated into the (Hopsan) actuator subsystem illustrated in Figure 7.12. For the overlying KBS, this object appears as a black box with only the system input/output ports visible, as shown in Figure 7.13. The configured actuator encapsulates no further KBS definitions (KBCs only). The created simulation object code consists only of library components with their parameters, the connections and the system ports (see Figure 7.14).

### 7.2.6.b Simulation Model Integration and Instantiation

Based on the work performed by [Krus, 2009], a (Hopsan) simulation conformal topology for an aircraft has been defined and visualised in Figure 7.15. With the help of the aircraft geometry and aerodynamics data

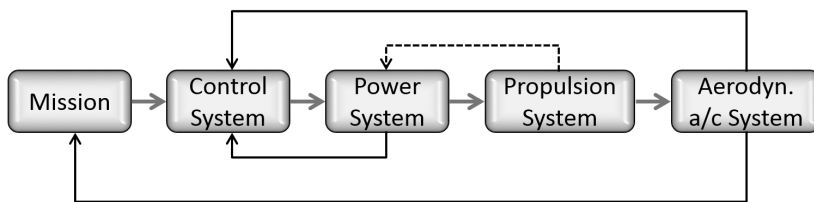


**Figure 7.13** Top level view of the linear actuator system (without defined system symbol) with the highlighted system ports hydraulic (green), mechanical (blue) and control signal input (red)

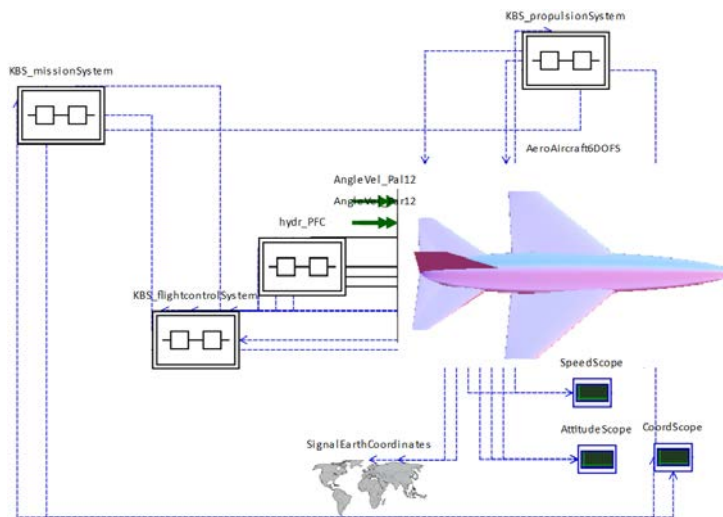
▼ system	@name	KBS_LinkServo			
	@typename	Subsystem			
▼ objects	▼ component	(6 rows)	@name	@typename	parameters
		1	4/3 Servo Valve	Hydraulic43Valve	> parameters
		2	C-type Cylinder	HydraulicCylinderC	> parameters
		3	Gain	SignalGain	> parameters
		4	Subtract	SignalSubtract	> parameters
		5	MechanicJLink	MechanicsTranslationalLosslessConnector	
		6	Angle Sensor	MechanicsPositionSensor	> parameters
	▼ systemport	(4 rows)	@name	hopsangui	
		1	CON_inSignal	> hopsangui	
		2	CON_inHydrPP	> hopsangui	
		3	CON_outHydrPT	> hopsangui	
		4	CON_outMech_POS	> hopsangui	
▼	> connections				

**Figure 7.14** The created *Hopsan* (sub)system definition of the linear actuator in an XML editor view (insignificant details such as GUI information are hidden)

(see Usecase1 on page 93) provided by the CADLab framework, a whole aircraft simulation system is built up with a Six Degrees of Freedom (6DOF) aircraft model as the central part. Executing the Matlab<sup>®</sup> design compiler script initiates the KBS-defined system with the top layer architecture according to Figure 7.15. During the compilation process the user is incorporated in the configuration of the hydraulic system layout using configurator GUIs.



**Figure 7.15** Top level aircraft system description

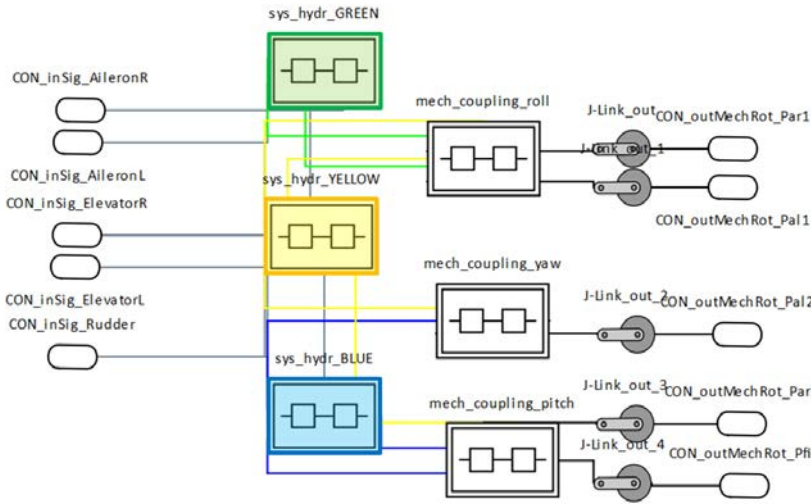


**Figure 7.16** The generated Hopsan simulation model (top level view; the components are manually rearranged for distinct appearance)

## 7.2.7 Usecase Review and Conclusions

The statistical analysis of PFCS shows clear structural trends and sizing information for the hydraulic system that can be used to build up a knowledge base for such systems. In this example, the creation of executable simulation models from such a knowledge-based description has been shown. The application example implementation was performed with a rather crude ad hoc implementation of the KBE compiler in Matlab® with the focus on methodology testing and development of the KBS/KBE nomenclature. The following shortcomings were detected during this work:

- (a) need for a graphical representation of the generated system model: This topic requires complex component placing algorithms (refer



**Figure 7.17** The generated centralised hydraulic system simulation model with the three hydraulic subsystems; comparable with Figure 7.7 (components manually rearranged)

to Usecase3 on page 115).

- (b) insufficient KBE definition assistance for the developer: Other program languages and development environments should be tested for the KBS/KBE definition.
- (c) aircraft system layout is mainly driven by reliability requirements. This makes it necessary to involve FTA and FMEA during the KBS definition process (see Usecase3).
- (d) time-consuming KBS definition: Only justifiable if the KBS definitions are reusable in other/future projects or are integrated into an iterative (e.g. optimization) process.

The first two topics (a) and (b) deal with common modelling problems such as the importance of a graphical model visualisation (see Section 5.3.3 on page 66) and the tool utility (which is addressed in Usecase1 and Chapter 3). The first is an implementation issue only while the latter is an unsolved topic that requires further investigation.

The third topic (c) is an application-specific problem, in particular in the case of the PFCS, where the reliability requirements are the driving design rules and consequently have to be either directly included into

the KBE process, or, if neglected, need to be addressed outside the KBE process in a similar way to an optimization problem. In a real generic system composition, it cannot be neglected as in this limited implementation example. Ideas and first steps of implementing FTA into the design process are presented in Usecase3. Topic (d) is a general KBE-specific theme, requiring time-consuming expert inputs for both the creation and maintenance of the KBE. This is an inevitable subject in the absence of automatic machine learning or other intelligent processes that automatically add and administer the information in the knowledge base.

One benefit of integrating a whole vehicle into the simulation model is the ability to perform a simulation-based component sizing on behalf of the simulation results [Jouannet and Krus, 2006]. By making use of this method the (cumbersome) initial component sizing approach (as seen in [Ingram et al., 2015] or [Scholz, 1996] for FCS actuator sizing) can either be skipped or conducted with reduced accuracy as it will be refined afterwards based on the simulation (results) itself. To consider all dimensioning load cases (which represent the extreme points in the design space) several mission simulations may be needed. Theoretically, it should be possible to set up self-learning algorithms based on the outcome of the simulation model. The other benefit is that – unlike conventional actuator size estimation approaches – the energy consumption can be addressed taking the stability and control effects of the aircraft into account<sup>6</sup>.

---

<sup>6</sup>Configuration and stability have a substantial impact on how and with which load the control surfaces are used during flight. In general, unstable configurations require quasi-continuous position updates at rather low force loads.



## 7.3 Usecase3: Graph-Based Driven System Modelling

*Based on the modelling theory in Chapter 5 “Model-Based System Design” and Chapter 6 “System Modelling”, these application example snippets highlight the idea and possibilities of (XML-based) well-defined data formats.*

With the availability of a metamodel (or in general, any structural model; see Section 5.2.1), the simulation model might be analysed prior to the execution. Potential motivations may be performance issues (simulation time and effort) or the analysis of simulations that cannot be executed due to any modelling error. It might well also serve as a simulation model debugger if the simulation itself fails, e.g. due to value drift (to zero, infinity, or *NaN*) or stability issues (value oscillations) as a result of a too stiff or unstable model.

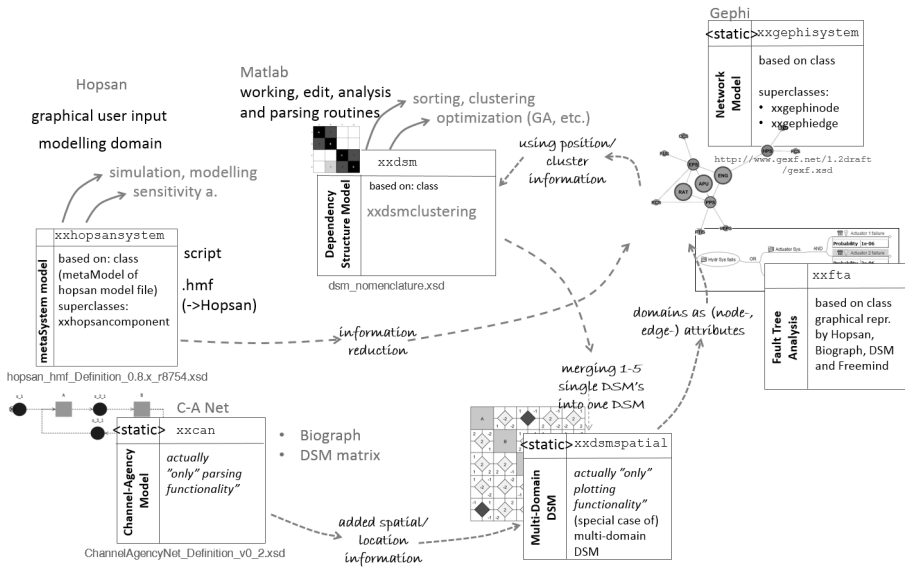
Figure 7.18 shows a conglomeration of (mainly) architectural models which all represent some kind of node/lattice – edge structure. Most of these concepts are based on XML DOMs with almost identical information content, making a seamless transition between these analyses possible (e.g. by XSLT). A common feature of system architecture analysis is a DSM analysis of a system in order to perform the following tasks:

- analysis and investigation of the model structure
- the sorting, relocating and split up of the project applying various rules such as:
  - different component domains
  - graph/network analysis

Different analysis aspects of the system meta model discussed above are described in the following sections. Table 7.2 on the next page highlights the differences between the modelling methods and tools where the FTA tree notably shows a variation with its strict tree structure. This particular nature implies one of the challenges of deriving an FTA model from an architectural (e.g. simulation) model: the detection and breakup strategy of loop structures, or, in other words, the conversion of a network into a (strict) tree structure.

Method/ Purpose	Tool	Node Vertex	Edge Syntax	Edge Type	Type	Format/ Standard	Remark/ Visualiza- tion
C-A Net: Meta- model modelling		2 types	directional	user-defined (li- brary; project spec.)	Net	no, own (XML)	visualization in Hop- san (special libr.) or XML editor (schema- based)
Physic modelling & simula- tion	Hopsan	library	directional (power comp.) bidirectional (signal comp.)	library (fixed) and 1type (sig- nal)	Net	yes, .hmf (XML)	own GUI (based on QT libr.)
FTA	xxFTA	2(3) types	directional	1type	Tree	yes, XTOM (XML)	
DSM		<string>	directional	1type, strength property (if not boolean)	Net	yes/no	
(S)MDM		4/5 <string>	directional	1type, strength property (if not boolean)	Net	no	
	Gephi	1 type	directional or bidirectional	1type, strength property (if not boolean)	Net	yes	visualization; use of advanced (interactive) sorting algorithms
	CATIA				Tree	yes (pro- prietary)	

**Table 7.2** Comparison of the methodologies and implementations used



**Figure 7.18** Overview of the developed model analysis and processing suitable for the model-based system development approach

### 7.3.1 The DSM Modelling Process

#### 7.3.1.a Problem of Implicit and Explicit Information Formulation

Beside the topics how to achieve consistency of the functional, structural and behavioural models (see section 5.2.1) and the sorting and separation problem, a third issue is the presence of implicit expressed properties for certain domains. Depending on the model type, certain domain features may be expressed explicit while others remain implicit. In ACD this problem is especially present in CAD domain. On the example of a hydraulic valve, [Sethson, 1999] shows a methodology of identifying functional matter flow<sup>7</sup> paths by the evaluation of CAD data. This process enables the creation of a (simulation) model based on the CAD data without the use of meta-information in the CAD domain, e.g. in the form of tags on the respective surfaces.

A comparable difficulty arises when attempting to coordinate a physical multi-domain simulation in form of a Hopsan model with a C-A net model. While the causality of the input/output relationship may be

<sup>7</sup>In the case of a hydraulic valve the possible paths of the hydraulic fluid through the valve which changes for different valve positions.

explicitly modelled in a C-A net, this does not apply to **Hopsan** models. In the **Hopsan** model, these causalities may be obtained by investigating the mathematical operations within the components. Alternatively – as a straightforward simulation-based method which takes advantage of the operability of the simulation model – these relationships can be obtained by means of a sensitivity analysis (on the component level).

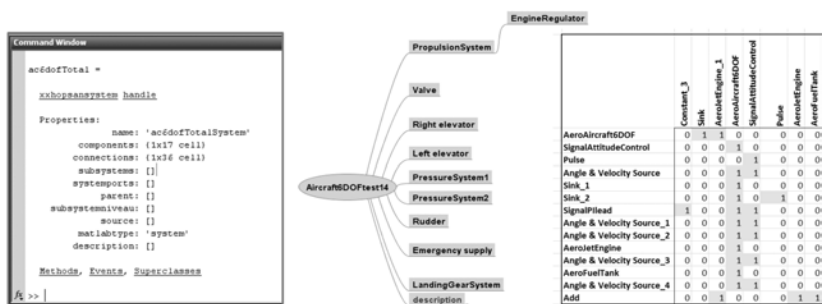
Further problems with multi-domain models are modelling shortcomings of non-matter-based effects such as heat conduction and radiation. This are common issues in component-based physical system models where the environment and spatial position usually are not modelled<sup>8</sup>. Fields of importance for those kinds of applications are in general components of high power density, systems with temperature sensitive materials (such as kerosene fuel, semiconductor material, aluminium) and temperature-dependent material properties (such as hydraulic liquids). Aviation-related examples are the fuel system [Gavel, 2007], the hydraulic actuation system [Behr et al., 2013], the ECS [II] or the Thermal Management System (TMS) [Seki et al., 2015] of an aircraft.

### **7.3.1.b Model Representations**

In addition to the usual simulation result analysis can the simulation model itself being investigated prior model execution. This enables additional analysis capabilities within the KBE-based simulation model integration process like that presented in the Usecase2 example on page 100. Figure 7.19 shows different model analysis and access methods that are available to the developer during the simulation model generation process (thus prior model execution). These are (from left to right) the metamodel itself (instances of the classes), a component connection analysis (connection matrix) and a tree view of the subsystems and component hierarchy. To fit industrial documentation standards it is possible to export any analysis data in the most common formats, e.g. **Excel** or **HyperText Markup Language (HTML)**. Figure 2 Since the simulation metamodel represents a graph of different kinds of vertices (thus subsystems or simulation library components) and edges (single- or bi-directional connections), exporting to any graph notation may be

---

<sup>8</sup>In particular heat dissipation effects by radiation and convection into the ambient air can only be rudimentary dealt with in a 0<sup>-dim</sup> or 1<sup>-dim</sup> component-based simulation environment. For detailed analysis of these effects, 3<sup>-dim</sup> or 4<sup>-dim</sup> modelling methods, e.g. CFD may have to be considered.



**Figure 7.19** Overview of different simulation representation and analysis in Matlab®, Excel or XML-based mind maps (aircraft model from Figure 7.16)

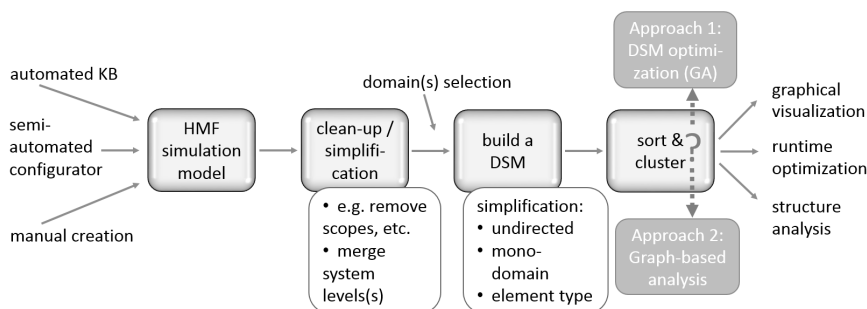
served. Data transfer to both GraphML [GraphML, 2015] and Graph Exchange XML Format (GeXF) [GEXF, 2015] notations have been applied to this use case to show this capability.

### 7.3.2 Interactive System Partitioning and Component Placement by the Application of Network Analysis Sorting Algorithms

The result of an automatic system/simulation model generating and instantiation process can be used directly for evaluation either of the model itself or by simulation execution. Manual interaction, however requires a comprehensive, well-arranged model representation like that shown in Figure 7.16. To obtain such a user-convenient graphical representation (see also Section 5.3.3) the automatically generated graphical component selection and its arrangement process can be based on domain, aspect or application-specific clustering (see Section 6.2).

A sensible inter-system break up should not be defined in advance but should arise as a consequence of the created topology by the KBE process. To perform this task, the DSM techniques presented in Section 6.2.2 and Section 6.4 can be applied. These techniques form a process to convert a simulation model data into a component-based DSM (see Table 6.1 on page 72) and support the subsequent analysis by either:

- (a) DSM sorting and cluster optimizations, or the application of
- (b) Graph-based sorting and cluster routines with a GUI



**Figure 7.20** The DSM-based model topology analysis process

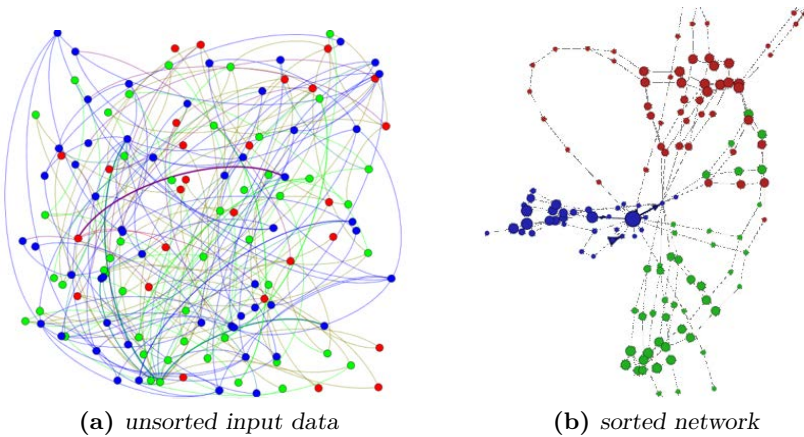
The former is an approach that requires the definition of one's own sorting criteria based on DSM properties like cluster density, inner/outer connectivity and possibly the presence of bus properties. The latter includes both available complex network sorting algorithm (like e.g. [Y. Hu, 2006]) and the use of interactive GUI (WYSIWYG). This allows for the application of existing tools and the direct interaction of the user during the algorithm tuning. This user participation is also required in the former case to adapt the optimization setup to the problem. Figure 7.20 shows an example of such a DSM-based simulation model analysis process.

### 7.3.2.a Graph-Based Sorting & Clustering Routines Including a GUI

In the graphical graph-based approach, the open source graph visualization platform **Gephi** has been used. Following the procedure shown in Figure 7.21, a clean-up of the simulation file<sup>9</sup> is performed and the simulation project tree is flattened into a single basic layer. Depending on the task, certain (component or edge) domains might be eliminated or merged (see e.g. Section 6.3 that addresses the domain reduction). The resulting network can then be presented as one or several DSMs or – as a graph setup – be further analysed and processed by the network visualization tool.

Within the network analysis tool (here: **Gephi**), the user can interact and control the split-up level and thus the granularity of the system until a satisfactory solution is found. The original simulation model data is then updated with the respective subsystem split-up and the

<sup>9</sup>A simulation tool dependent process that for example removes user interface components.



**Figure 7.21** The graphical clustering approach based on a network. The node colours in (a) represent the *Hopsan* simulation component type (of the aircraft model shown in Figure 7.16), whereas the node colours in (b) identify the system relationship (sorting algorithm: Yifan-Hu, network split-up by modularity property).

node positioning by means of the network properties (in this case the graphical node information and the node positions). The clustering process can be recursively applied on the created subsystems until a sufficient data structure is achieved.

### 7.3.3 Fault Tree Analysis Modelling

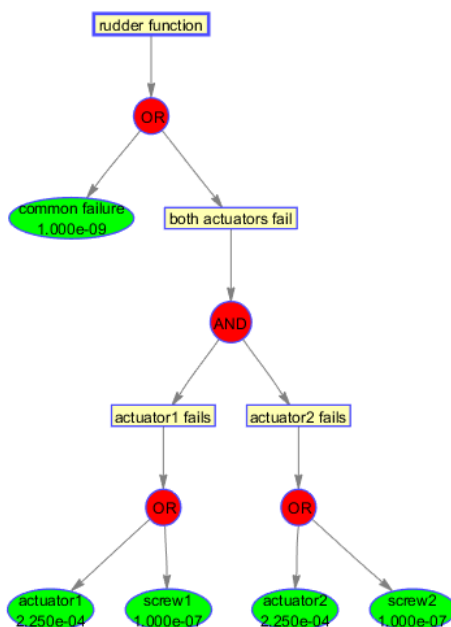
As already motivated in Section 4.2.2 and Section 7.2, investigations of system reliability must not be neglected in safety-critical systems.

Taking advantage of the alternating component type similarities between FTA and *Hopsan*, FTA tree modelling is naturally supported by the *Hopsan* GUI. With the help of a small external library consisting of eight components, FTA modelling can be performed stand-alone or mixed up within any *Hopsan* simulation model.

The actual FTA calculations can either be executed directly in *Hopsan* or performed in a dedicated FTA environment, e.g. the `xxFTA.m` class in *Matlab*<sup>®</sup>, see Figure 7.18). The advantage of the latter method is the extended FTA analysis capabilities such as the cut set order representation. Furthermore, depending on the size of the FTA problem, the component probability value spread and the amount of available ran-

dom access memory (RAM) size, direct Monte Carlo FTA simulations can be applied, utilising the key feature of *Matlab*<sup>®</sup> – matrix and vector operations.

The FTA tree can be presented in a well-conditioned manner by using the *Matlab*<sup>®</sup> *Biograph* module, which is part of the *Bioinformatics* toolbox: Graphical results of FTA using the *hierarchical* layout option<sup>10</sup> are satisfactory (see Figure 7.22). Unfortunately, no further information on the layout algorithm within this function has been published yet.



**Figure 7.22** FTA tree representation in *Matlab*<sup>®</sup> with the help of the *Biograph* toolbox

### 7.3.4 Graphical C-A Net Modelling Approach

Based on the experience gained from applying *Hopsan* as the GUI for FTA modelling (on the preceding page), an ad hoc trial to adapt the *Hopsan* GUI for C-A net modelling was performed.

<sup>10</sup>Hierarchical layout: Primary sorting by the topological order of the graph and secondary on the nodes from top to bottom, while minimizing crossing edges, see [The Mathworks, Inc., 2015a].



#### **7.3.4.a GUI/Component-Based C-A Net Implementation**

A similar approach to that in **xxFTA** (in Section 7.3.3 on page 121) was used. The OOP structure made it possible to adapt the **xxFTA** class to serve C-A net modelling instead of FTA modelling with only minor adjustments.

Similar to FTA modelling, the model is based on a specific C-A net component library containing two component types only – channels and agencies. After several trials a convenient solution of the model and channel definition was found: each channel property (signal, matter, energy; for definitions see Section 6.5.1.a) has to be defined by the user as a system parameter of string type with the same name and value. During the build-up of the C-A net topology these system properties are easily accessible through a pop-up window in the **Hopsan** GUI for every channel component. A good C-A net modelling strategy with this implementation approach is primarily to define all channel properties and subsequently perform the modelling of the system relationships.

#### **7.3.4.b XML Editor-Based C-A Net Implementation**

As an alternative to the presented GUI/component-based C-A net model, a script-based input approach within a XML editor environment can be performed. With the presence of an XML Schema (XSD), modern XML editors support the user with active tool-tips and suggestions as well as just-in-time validation with error highlighting and optional correction advices<sup>11</sup>. The main task of C-A net modelling is to provide a smooth transition of the model fidelity, starting from a coarse model (e.g. electric signal specified as one universal signal channel only) towards the final detailed level model (e.g. electric signal protocol specified). The challenge of the schema definition is to allow for a certain flexibility but ensure robustness and an efficient modelling approach without too much overhead at low fidelity levels<sup>12</sup>.

---

<sup>11</sup>In this project, the XML editor **oXygen** was used; other XML editors may offer similar functionality.

<sup>12</sup>A very similar task/design compromise to the geometry parametrisation presented in Section 3.2.1.

### **7.3.5 Usecase Review and Conclusions**

Handling product properties and dependencies can be tackled in various ways. In most cases, the data can be described as a graph-set. Using XML-based datasets and class-based OOP, a fast implementation of the different modelling approaches can be maintained. To obviate the GUI programming effort, existing tools providing a GUI can easily be integrated using Extensible Stylesheet Language Transformation (XSLT). Extensible Markup Language (XML) Schema Definition (XSD) enable dataset validation checks or may be applied for modelling support, as shown by the editor-based C-A net modelling approach (Section 7.3.4.a on the preceding page). XSD is an appropriate format for dataset validation. However, shortcomings in terms of capability (e.g. the lack of formal mathematical expressions) and high implementation effort for complex content checks were revealed in the Usecase1 example. Various criticism on the XSD standard (e.g. in [Møller et al., 2006]) confirm these shortcomings. In particular the limitation in content-related element declarations based on attribute or element, as well as the inappropriate way of redefining elements limits its usefulness for complex and flexible product tree validation.

Based on the findings in the System Modelling sections (Section 6.2, Section 6.3 and Section 6.5), the modelling approach may be unified and DSM matrix-based sorting/clustering routines may be used at any time. As a low-effort alternative, the export to network analysis tools for clustering was examined and was rated positively. Finally, the parallel implementation of FTA was demonstrated. The topic of reducing the network's structure of the product properties and dependencies into a strict tree dependency (see Table 7.2), however, remains unresolved. This topic could be resolved by modelling the system as a C-A net. The additional causal dependency information in this type of model allows for a logical analysis of the loop structures within the model-network to detect their parallel or sequential kind, resulting in *AND* respectively *OR* couplings in the FTA tree. The extension of this C-A net-based approach with a KBC for the C-A net/**Hopsan** component relationships information (see Figure 6.2) enables direct simulation model generation (see Figure 7.18). Alternatively to this KB-based approach, the C-A net components could be built up based on the simulation tools' components, containing only the additional causal port relationships information.

# 8

## Discussion and Conclusion

### 8.1 Discussion

Handling all the aspects and domains within Aircraft Conceptual Design (ACD) while maintaining the specific characteristics of this design phase is a difficult task. The current trend towards highly integrated, multi-domain frameworks aiming for higher fidelity jeopardizes the provision of a concise, flexible work process.

Furthermore, with shrinking design enhancement margins due to the levels technology has reached today and enhanced performance requirements in ACD, hitherto disregarded topics the (on-board power) subsystem design become an inevitable part of the conceptual design. The current MEA trend with more tighter integrated systems will continue, probably resulting in even more complex architectures, incorporating hybrid propulsion technologies. These trends clearly mark a breakpoint at which the behavioural model has to become part of the concept study.

To deal with subsystem architecture in ASCD, efficient (modelling) processes to design and evaluate cyber-physical systems have to be found. These methods need to be streamlined to try to avoid unnecessary workload and expert knowledge to fit into the ACD context. One possibility shown in this thesis is the use of KBE methods within an object-oriented framework. Notwithstanding that KBE originates from the CAD domains, integrating high-level 3D geometries into a multi-dimensional framework is still a challenging task.

Enabling automated reuse of knowledge by means of KBE seems prof-

itable for repetitive, project-independent work topics like the certification requirements. In the presented Usecase2, the application of a simulation model is made possible by using KBE, eliminating the manual work of composing and tuning the simulation model. Chapter 6 proved that several modelling approaches designated for early product development stages are consistent and allow for a unified model based on graph theory. Interpreting a system as a graph network enables other analysis topics to be integrated for example a reliability analysis. To enable an efficient, tight model integration process and easy model transformations, the use of XML is a promising, low-hanging fruit. Unlike more complex, semantic-based scenarios like the Semantic Web approach, the application of XML can be easily accomplished. Especially in combination with the associated schema (XSD) and translation (XSLT) processes it provides a sound foundation for shared database operations. Most of the standard programming languages<sup>1</sup> or tools provide Document Object Model (DOM) for efficient XML handling, XSD validation processes and XSLT-based transformations. As has been shown in the use case examples, the combination of OOP, XML, KBE, and CAD enables an easy, streamlined and flexible model-based ASCD framework implementation without the need for substantial software development efforts.

## 8.2 Detailed Discussion

**Alternative usage** The main task of ACD is not to come up with a final design but rather to find the best configuration for the given – possibly vague and incomplete – requirements with the right degree of credibility to convince management and stakeholders [McMasters et al., 2004]. Nowadays, only very few of the evaluated concepts are taken further to the preliminary design phase (and far fewer reach the detail design stage and are realised). ACD should therefore be seen more as a recursive process that enables the designer to investigate and analyse the certain topic of a given problem. The system understanding and insight gained can be used for requirement analysis, enabling a result-based negotiation of the customer’s needs by relating the product costs or the design impact to the requirements. Another topic is the analysis

---

<sup>1</sup>See also the **Tools, Software, and Programming Languages** acronyms list on Page xiii, which also states the programming language used for the tools. Almost all programming languages in this listing support XML DOM capability or provide special packages or libraries for this purpose.

of the impact of new technologies on product performance as well as on the product design itself. Furthermore, universities and academia make frequent use of ACD for student education and training.

**Resource limitations** Multi-domain resource limitations in social and technical aspects are the specific characteristics of the ACD process with both negative and positive consequences. The assessment of the included work topics, processes, methods and tools used (see Figure 5.13) has to be performed mainly on an efficiency measure to accommodate the trade-off between effort and result (enhancement). This fact – shown in Section 5.2.2.b – limits the use of simulation tools to mainly 0<sup>-dim</sup> or 1<sup>-dim</sup> models.

**Clear and visible Process** Associated with the uncertainty topic is the need for understandable, transparent and plausible routines with an appropriate GUI to avoid any discrepancy between the user's expectations of the tool's capabilities and the results. Applying detailed CAD geometry on low-fidelity aerodynamic analysis tools (such as Vortex Lattice Method (VLM)) might be misleading if the user is not adequately informed through for example a graphic which illustrates the simplified geometry used by the calculation routine. Alternatively, the result accuracy should be stated for the user.

**Database** A key requisite to derive an analytical model from an architectural model is the accessibility of the data as well as the existence of a comprehensive transformation process (see Figure 5.3). The central XML-based database approach supports either demand, enabling a distinct transformation process between the models based on the parametric design information.

**Parametric design** Parametric design has to provide a concise, robust and flexible design description (see chap:conAD:ParametricDesign). Unfortunately, these three features constitute a design conflict that requires careful balancing supported by a sound parametrisation of the data with respect to the modelling and analysis needs. In the CADLab framework (see Section 7.1), the flexibility of the parametric design has largely been sacrificed for a slim, robust and concise geometry description. Basically oriented on a product tree of a standard aircraft configuration, it consists of a central body (the fuselage) and several attached lifting surfaces (like

wings, empennage surfaces, and canards). It has been shown that dealing with the various product domains like outer shape geometry (OML) or the architecture and functionality of on-board systems requires adapting data to these domains. These data rearrangements might require a global change of the model/product (tree) structure during the transformation process (see Section 3.2 Information Model and Figure 3.5).

**Simulation tools** Many analysis methods are based on simulations (see Section 5.2.2 and Figure 5.10 and Figure 5.11). This means that many of the transformation processes from the architectural model to the analytical models (see Figure 5.3) are comparable with the manual modelling process in the various tools, including all the tool-dependent rules and characteristics (see Figure 5.8b on page 56). Usecase 3 on page 115 is one example which shows that such modelling characteristics can be used to advantage, as in the *Hopsan* case the  $C-Q$  component sequencing. The modelling technique expressed by the dimensionality also sets the accuracy and the modelling effort which has to match both the available data and the required analysis uncertainty (see above). It has been shown (in Section 5.2.2.a/Figure 5.8a) that the modelling outcome as a matter of expedience relates in a nonlinear relationship to the model's completeness. This leads to the conclusion that a complexity jump (in the meaning of modelling technique and level of detail) has to be made at a certain point to overcome an inopportune modelling region where the model's usefulness stagnates or even deteriorates. Not only in ACD but with complex product development in general the focus lies on product enhancement<sup>2</sup>. A complexity jump like the one mentioned above exists in-between semi-empirical, low-fidelity approaches and higher level, product property-based methods. Special methods and processes for modelling may be applied to overcome this area of inappropriate model fidelity.

The physical simulations additionally require (at least parts of the) behavioural model in addition to the functional system information. This part is often processed within the functional model without the explicit information of whether it depends on the functional model itself or the behavioural (control) part of the system. The simulation result thereby depends on all three factors the control system, the physical system ar-

---

<sup>2</sup>Product enhancement seen as a technology evolution and not a product revolution. Advantageous during product development in the first case is the availability of expertise and statistical data.

chitecture and the component characteristics. Given that fact, it is hard to attribute the system's shortcomings to the control/software part, the physical system part or an inadequate interaction of both parts.

**Knowledge-based engineering** One approach to solve the complexity jump issue discussed above is the use of KBE methodology, aimed at the automatic reuse and reprocessing of domain- and problem- specific knowledge of an enterprise for a specific project (see Section 5.3.2).

During conceptual design with its incomplete dataset of the yet unknown product properties, parts of the model have to be established. Based on various sources of knowledge, KBE can help to find appropriate product/model properties like (sub-)system architectures as shown in the Usecase2 example on page 100.

The drawbacks of any KBE approach are the added complexity, implementation overhead, and additional maintenance effort which may only be applicable if it is regularly used. ACD – with the investigation of multiple solutions based on the initial vague requirements (thus leaving a huge design space to explore) – is such a case where KBE can be beneficially applied. The Usecase2 example shows the advantages of a KBE approach which incorporates the link towards a system modelling (0<sup>dim</sup>) tool from a KBE implementation based on the process shown in Section 6.1, Figure 6.2. In fact, the coupling information relates to both the modelling principles – in the *Hopsan* case the methodology of power ( $C$  and  $Q$  components) and signal ( $S$ ) components – and the specific library information (see Section 7.2.5). The last enables the linkage between functional instances during the compiler process and the component libraries.

**Unified modelling** From the point of view of a unified modelling approach (discussed in Chapter 5), the downside of a KBE setup like the one presented is that the tool- or model-theory specific rules and dependencies on component libraries are incorporated by the KBE process. In the Usecase2 example shown on page 100, the functional part of the components has not been included in the KBE solver but was described by meta-information of the components.

One example would be an electric hydraulic pump (e.g. in the Usecase2 example) which is treated in the KBE process as a black box with one electric and two hydraulic interfaces that are capable of enhancing the pressure difference on the hydraulic ports. However, the equations

and relation between electric power consumption, pressure rise and fluid mass flow are not part of the solver, if not explicitly specified by the user. Such detailed rules are for example needed for the component parameter sizing, addressing the problem of top-down versus bottom-up approaches as shown in Figure 5.12 on page 61.

For a unified modelling approach, however, the model equations should be part of that very same model and then be derived towards the characteristic of the respective simulation tools as for example shown by [Larsson, 2006]. The solver theory in the different modelling approaches may therefore differ dramatically (refer to the differences between the centralised *Modelica* approach and the distributed TLM approach used in *Hopsan*; see Section 5.3.1.b on page 62), which may make translations very complicated or even impossible.

**Semantic approaches** Based on the universal model idea discussed above, several approaches have been applied to enable an efficient way of describing a model. *UML* is one of these tools mainly used within software development with the concretisation into *SysML* as a tool for product development modelling. Both are XML-based, as is the semantic web approach (see Section 5.3.1.a) on the lowest level. This approach is far more abstract and with the different layer also much more complex, but may present a future path towards a unified, all-embracing data description. Whatever implementation language is chosen for modelling, its usefulness depends on both the expedience of the language itself and its graphical representation models. In case of a high (huge or inadequate) abstraction level between the modelling language and the modelling topic, a graphical illustration may be adequate to support the domain-specific modelling needs (see Figure 5.3.3). One example of an Integrated Development Environment (IDE) enabling a graphical model environment is the *Eclipse* platform which is individually adjustable for different programming languages.

**XML** The first step towards unified modelling may be the use of Extensible Markup Language (XML) for data exchange and translation (interpretation). Various advantages (as shown in Section 7.3, Section 7.1.1.b) come along with the strict validation, the ASCII text-based format and the availability of style definitions (XSD format) for format validation and translation schemes (based on the XSLT format). Unfortunately, the formats' inherent drawback is the limited and extremely inappro-



priate inclusion of numeric data – a shortcoming that applies for any non-binary data format. A way to deal with this drawback is to exclude large numeric datasets into binary files with only the link and explanation left in the XML document.

**Graph-based model implementation** Besides the extended use of XML, the Usecase3 example shows the use and nature of graph-based models for the conceptual product development stage. Unlike the traditional strict data tree product structure (like the XML format or as often applied in single-domain models such as CAD geometries; see Figure 3.5), single-domain and especially multi-domain models are more of a network or graph-like structure, as has been shown in Sections 6.2–6.5. Consequently, the use of graph notation and graph-based implementations may be beneficial to describe model properties. One way to address graph-like data in a two-dimensional and efficient computational matter is to use matrix notations<sup>3</sup>.

**DSM and C-A Net modelling and analysis** DSM, as the representative of a matrix notation for different tasks within product development, is the classic example commonly used to visualise graph-like dependencies (see Section 6.2). Addressing multi-domain problems, the DSM may be extended in the third dimension by several single-domain DSMs, with the absence of a comprehensive two-dimensional visual representation. One solution to this problem can be found with the help of the Multi-Domain Dependency Structure Matrix (MDDSM) notation, presented in Section 6.3. This notation complies precisely with the channel definition of the C-A net terminology provided in Section 6.5. The only difference is the absence of the spatial arrangement information in the C-A net notation. Given that, DSM, MDDSM and C-A net models can be treated as one single model, enabling powerful graph mathematics to be used (see Section 6.4 and Section 7.3.2). It has been shown (in Section 6.2.2) that sorting algorithms rely heavily on the DSM properties, shape, and the cluster configuration (see Figure 6.5 and Figure 6.6). Given the large number of possible setup combinations (see Equation 6.1 on page 75) and the sorting tasks of different intentions, the tuning of the applied

---

<sup>3</sup>Event though the use of matrices for product properties and relationship modelling might be seen as an inefficient matter due to the generally sparse matrices, in the case of using **Matlab**<sup>®</sup>, mathematical operations speed up enormously through the use of matrix operations.

sorting algorithms (usually based on generic algorithms) is a non-trivial issue with great influence on the result. Alternatively, interactive GUI-based network analysis tools may be preferred instead of a cumbersome optimization algorithm setup tuning (see Section 7.3.2.a).

**Graph-based or acausal implementation** Tackling the problem of addressing the – physical system inherent – bidirectional dependencies can be achieved in various ways: by the split-up into directional relationships as for example performed in DSM matrices, in the form of real acausal relationships as supported in certain simulation tools like *Modelica* or by special implementations like the TLM concept used in *Hopsan*. Each solution has its task-dependent characteristic advantages and disadvantages, which makes a global tool/methodology ranking impossible. Despite the fact that a complete CAD environment has been included in the ACD framework, unforeseen efforts had to be made to implement a rudimentary geometry kernel in the *Matlab*<sup>®</sup> environment. This task is an ideal candidate for acausal programming due to the vast over-definition of the aircraft geometry through different parameters that the user is interested in for various reasons. For future development, this part should – if geometry handling outside the complete CAD environment is needed – be based on available geometry kernels (such as *openCASCADE*) or complete geometry modelling tools like *openVSP*.

## 8.3 Answers to the Research Questions

*This Chapter serves as a response to the Research Questions stated in Section 1.3.2 on page 8. To avoid repetitions, the questions will only be answered in short form, with references to the corresponding chapters in this work.*

**RQ1** An enabler for an efficient multi-domain and multi-aspect ACD is the ability to change instantly between different applications and to keep the user informed and updated through an understandable and transparent process. As shown in theory (Section 3.2) and praxis (Section 7.1), the foundation to enable this task is a sound model parametrisation (see Section 3.3.1) that makes the model accessible and interpretable for the respective parsers between the holistic model information and the entire single-domain applications (see also Figures 5.2 and 5.3). For efficient geometry domain representation, acausal programming should be

considered (Section 3.3.2). To head towards a flexible and robust implementation, OOP techniques are appropriate due to their data content, functionality and rule applications being within one entity possibly along with graphical representation and user interface functions. Such an implementation allows for domain-specific product (tree) setups as shown in Figure 3.4. An implementation of such a framework is presented in the Usecase1 example on page 93.

**RQ2** The answer is yes, KBE can support the use of simulation models during the ACD. As theoretically motivated in Chapter 5 the models used within MBSE can be of a very diverse nature, addressing different degrees of completeness (Figure 5.6), implementation methods (physical or analytical), modelling techniques (theory and implementation language) and model fidelity (see Section 5.2.2.a) depending on the scope of the model. KBE is one possibility to integrate the collective information (see Section 5.3.2 on page 64) and make automated use of it during the design process (see Section 6.1). The implementation of such a process which integrates an automatically generated system simulation model is shown in the Usecase2 example on page 69ff.

**RQ3** Managing the various aspects of system design requires different, domain-dependent modelling and analysis methods (see therefore also RQ1 above). The appropriate modelling choice may be (further) bisected into the modelling task on one hand and the analysis task (in case of a simulation model the model execution) on the other hand. Convenient modelling is usually performed by means of aspect-specific programming languages and notations (see Figure 5.4, Table 5.1 and Section 5.2.1). Merging the different domains – structural, functional, and behavioural – is both blessing and curse for a designer trying to achieve a balance between integrity, consistency, and ambiguity on one side and complexity and modelling appropriateness on the other. The complexity of a multi-aspect model turns into a hurdle already at a relatively low fidelity level, as illustrated by the problem of MDDSM sorting (in Section 6.3). However, Usecase3 shows the benefits of a central model – in this case a graph model – that enables automated transformation towards aspect-specific analytical models (see Figure 7.18 on page 117). In the future, more enhanced and holistic approaches may arise; possible trends are semantic approaches like the semantic web (see Figure 5.14 on page 63).

## **8.4 Conclusion**

By careful analysis of the work tasks and the special needs of the conceptual aircraft design phase, KBE-based design approach has been presented that includes CAD and on-board power systems. On-board power systems have been added to the ACD (becoming ASCD) phase to enhance the result accuracy and take into account into the overall system design, comprising propulsion, power generation and power consumption systems. However, adding subsystem design requires more attention than extending the ACD process by system simulation models only. For this reason, a thorough analysis of modelling & simulation techniques has been performed with respect to their capabilities to back a smooth model-based design approach. Based on the theoretical findings, a conceptual design framework based on a central XML-based dataspace has been developed incorporating a complete 3D CAD environment. By extensive use of KBE methods, automated generation of simulation models becomes possible. Based on the incomplete data, these processes incorporate additional data from the knowledge base. KBE seems to fit well in the application of ACD because of the high influence of safety and redundancy-focused certification requirements concerning the design. All the aircraft's (sub-)system designs are tailored to the redundancy requirements. For this reason, it is necessary to include reliability measures within the design process. Including simulation models in the (ACD) design process requires an interpretable, preferably parameterised information model and appropriate modelling approaches. Both topics have been shown in this thesis. By means of methodical modelling approaches, system designs of (cyber-physical) power systems can be systematically derived from rules that for example address the technology level, power transformation & conversions, and scale.

The use of OOP principles and XML-based data with the extensive use of XSD and XSLT enables a relatively simple approach to maintain multi-domain models compared to other, more complex approaches, focussing on unified modelling. This method enables a type of pseudo-unified modelling approach that enables the integration of different models backed by the flexibility of a graph network.

## 8.5 Future Work

Due to the large scope of MBSE in general and specifically in ACD, not all topics of these engineering tasks could be addressed and solved within this thesis. It has been shown that for a full KBE integration, all available domains as well as project-specific information has to be used. Especially difficult hereby is the integration of requirement engineering in order to make tracking of the design decisions and the design impact possible. Related to this topic, additional work towards the inclusion of reliability assessments (e.g. by means of FTA and FMEA) has to be done. As shown in the Usecase3 example on page 115, the inclusion of an FTA is an easy task if the fault tree structure is known. However, deriving a strict FTA tree structure from a multi-domain, non-strict system architecture is still an existing problem that needs to be solved.

Backing the KBE approach by enabling the interpretation of the existing information is another unresolved topic for future research. With today's knowledge it is not foreseeable which approach will offer the best alternative. In this thesis, the unified model approach and the semantic web approach have been discussed; time will tell whether the semantic/ontology approaches turn out to be beneficial in long-term.

For product development, commercial products that offer a multi-domain and simulation tool integrated environments are already available, for example the CATIA® V6 work suite [Dassault Systèmes, 2016]. However, based on a proprietary data format the integration process seems not to be transparent and a flexible adaption to other tools might be hampered.

Last but not least, the engineering process depends on the education, skills and social aspects of the people involved, most of them with a university education in engineering. As a side effect of the introduction of more complex processes, it is undoubted that the education of technical engineers has to be adapted to future needs [McMasters et al., 2004]. With today's cyber-physical systems, the clear boundaries between mechanical and software engineering are washed away whereas they still exist even in modern educational curricula [Hayhurst et al., 2012]. The understanding of this topic as a combined problem among all these disciplines will help to enhance the acceptance and enable the use of such processes by forming a solid foundation of the environment in which the processes and methods have to be applied (see Figure 5.13, the PMTE pyramide).



# Bibliography

- DO-178C (2011). *Software Considerations in Airborne Systems and Equipment Certification*. Tech. rep. Radio Technical Commission for Aeronautics (RTCA), SC-205, Inc., Washington, DC, U.S.
- DO-254 (2000). *DO-254/EUROCAE ED-80: Design assurance guidance for airborne electronic hardware*. Tech. rep. RTCA.
- A4A (2016). *Airlines for America (A4A) homepage: AS1000D and ATA i2200 Specification*. [Online; accessed 2016-03-07]. URL: <https://publications.airlines.org/>.
- AIR5005A (2010). *Aerospace - Commercial Aircraft Hydraulic Systems*. Tech. rep. A-6A1 Commercial Aircraft Committee, SAE International.
- Allen, J. S. (2014). *Principles of Energy Conversion*. Tech. rep. Department of Mechanical Engineering - Engineering Mechanics, Michigan Technological University, U.S.
- Andersson, H. (2012). “Variability and Customization of Simulator Products : A Product Line Approach in Model Based Systems Engineering.” Linköping Studies in Science and Technology. Dissertation No. 1427. Department of Management and Engineering (IEI), Linköping University, Sweden. ISBN: 978-91-7519-963-4.
- ATA (1967). *Standard Method of Estimating Comparative Direct Operating Costs (DOC) of Turbine Powered Transport Airplanes*. Tech. rep. Washington D.C.: Air Transport Association of America (ATA).
- ATA 100 (1999). *ATA Specification 100 - Specification for Manufacturers’ Technical Data*. Tech. rep. Revision No. 37. Washington D.C.: Air Transport Association of America (ATA).
- ATSB (2013). *Accident Report VH-OQA: In-flight uncontained engine failure Airbus A380-842, VH-OQA*. Tech. rep. Transport Safety Report, Australian Transport Safety Bureau (ATSB).

- Axin, M., R. Braun, A. Dell'Amico, B. Eriksson, P. Nordin, K. Pettersson, I. Staack, and P. Krus (2010). "Next Generation Simulation Software using Transmission Line Elements". In: *Fluid Power and Motion Control (FMPC)*, Bath, UK, pp. 265–276.
- Ballhaus, W. F. (1955). "Clear Design Thinking Using the Aircraft Growth Factor". In: *14th National Conference*. 0113. Society of Allied Weight Engineers (SAWE), Inc. Fort Worth, Texas.
- Baslev, H. (2010). *Structring Design Data ISO/IEC 81356*. Tech. rep. Baslev & Jacobsen ApS, Copenhagen, Denmark.
- Baslev, H. (2015). "Implementing Model Semantics and a (MB)SE Ontology in the Civil Engineering & Construction Sector". In: *Nordic Systems Engineering Tour 2015 (NoSE)*. INCOSE, Stockholm, Sweden.
- Behr, R. and V. Baumbach (2013). "Aircraft Hydraulic Thermal Control". In: *Proceedings of the 4th International Workshop on aircraft System Technologies (AST)*. Hamburg, Germany, pp. 51–59.
- Belan, H. C., V. J. De Negri, and R. Szpak (2010). "Channel/Instance Petri Nets for Structural and Functional Modeling of Industrial Equipment". In: *ABCM Symposium Series in Mechatronics*. Vol. 4. Associação Brasileira de Engenharia e Ciências Mecânicas (ABCM), Rio de Janeiro, Brazil, pp. 403–407.
- Berners-Lee, T. (1998). *Why RDF model is different from the XML model*. World Wide Web consortium (W3C). Online; accessed 2016-07-27. URL: <https://www.w3.org/DesignIssues/RDF-XML.html>.
- Berners-Lee, T. (2000). *Semantic Web on XML*. Keynote presentation for XML 2000 Conference and Exposition in Washington, World Wide Web consortium (W3C), DC, U.S. [Online; accessed 2016-07-20]. URL: <https://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html>.
- Berry, P. (2015). *Description of the BeX Sizing Program Usage*. Lecture Material 0.1. Course TMAL51, Division for Fluid and Mechatronic Systems, Linköping University (LiU), Sweden.
- Böhnke, D. (2015). "A Multi-Fidelity Workflow to Derive Physics-Based Conceptual Design Methods". PhD thesis. Technischen Universität Hamburg-Harburg (TUHH), Germany.
- Booker, A. (2006). *Foto: A full scale model of the Volvo XC90, completely made of legos*. [Online; accessed: 2016-09-13]. URL: <https://www.flickr.com/photos/armisteadbooker/145646807>.



- Braun, R. (2015). “Distributed System Simulation Methods: For Model-Based Product Development”. Linköping Studies in Science and Technology. Dissertation No. 1732. Department of Management and Engineering (IEI), Linköping University, Sweden. ISBN: 9789176858752.
- Browning, T. R. (2001). “Applying the design structure matrix to system decomposition and integration problems: a review and new directions”. In: *IEEE Transactions on Engineering Management* 48.3, pp. 292–306. ISSN: 0018-9391. DOI: 10.1109/17.946528.
- Browning, T. R. (2016). “Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities”. In: *IEEE Transactions on Engineering Management*, vol. 63. 1. Institution of Electrical Engineers (IEEE), New Jersey, U.S., pp. 27–52.
- Bruner, S., S. Baber, C. Harris, N. Caldwell, P. Keding, K. Rahrig, L. Pho, and R. Wlezian (2010). *NASA N+3 Subsonic Fixed Wing Silent Efficient Low-Emissions Commercial Transport (SELECT) Vehicle Study*. Revision A NASA/CR—2010-216798. National Aeronautics and Space Administration (NASA), Glenn Research Center, Ohio, U.S.
- Cellier, F. E. (1996). “Object-oriented modeling: means for dealing with system complexity”. In: *15th Benelux Meeting on Systems and Control, Mierlo, Netherlands*, pp. 53–64.
- Chakraborty, I. (2015). “Subsystem architecture sizing and analysis for aircraft conceptual design.” PhD thesis. Georgia Institute of Technology.
- Chakraborty, I. and D. N. Mavris (2016). “Assessing impact of epistemic and technological uncertainty on aircraft subsystem architectures”. In: *16th AIAA Aviation Technology, Integration, and Operations Conference*. American Institute of Aeronautics and Astronautics Inc, AIAA, Washington D.C, U.S., 24p.
- Chiesa, S., G. A. Di Meo, M. Fioriti, G. Medici, and N. Viola (2012). “ASTRID - Aircraft On Board Systems Sizing and Trade-Off Analysis in Initial Design”. In:
- Chiesa, S., M. Fioriti, and N. Viola (2012). “Systems Engineering - Practice and Theory”. In: InTech. Chap. Methodology for an Integrated Definition of a System and its Subsystems: The case-study of an airplane and its subsystems, pp. 13–38.

- CPACS v.2.3 (2016). *CPACS - Common Parametric Aircraft Configuration Schema*. [Online; accessed 2016-07-22]. URL: <https://github.com/DLR-LY/CPACS>.
- Culp, A. W. (1991). *Principles of Energy Conversion*. McGraw-Hill Series in mechanical engineering. McGraw-Hill, New-York.
- Daconta, M. C., L. J. Obrst, and K. T. Smith (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis, IN : Wiley Pub. ISBN: 0471432571.
- Dassault Systèmes (2016). *Catia®V6, 3D CAD/CAM Design Software*. [Online; accessed: 2016-09-07]. URL: <http://www.3ds.com/products-services/catia/products/v6/portfolio/>.
- De Negri, V. J. (1996). “Estruturação da modelagem de sistemas automáticos e sua aplicação a um banco de testes para sistemas hidráulicos”. PhD thesis. Universidade Federal de Santa Catarina, Florianópolis, Brazil.
- De Negri, V. J. (1998). “Conception of automatic test benches for hydraulic components”. In: *Proceedings of the power transmission and motion control (PTMC'98), Bath, UK*, pp. 29–41.
- De Negri, V. J. and H. C. Belan (2013). “Automatic System Design: Canhannel/Agency Nets”.
- Dickerson, C. and D. N. Mavris (2010). *Architecture and principles of systems engineering*. Complex and enterprise systems engineering series. Boca Raton, Fla.: CRC Press, p. 460. ISBN: 9781420072532.
- DoD (2008). *Systems Engineering Guide for Systems of Systems*. Tech. rep. Secretary of Defense for Acquisition and Technology, Department of Defence (DoD), Virginia, U.S.
- DoD (2010). *DoD Dictionary of Military and Associated Terms*. Tech. rep. Departement of Defence (DoD), Virginia, U.S.
- Dollmayer, J. (2007). “Methode zur Prognose des Einflusses von Flugzeugsystemen auf die Missionskraftstoffmasse”. PhD thesis. Technische Universität Hamburg-Harburg (TUHH), Germany.
- DSMWEB (2015). *DSMWEB - The DSM Community: Knowledge portal on structural complexity*. [Online; accessed 2015-08-20]. URL: <http://www.dsmweb.org/>.
- Dunker, C., R. Bornholdt, F. Thieleck, and R. Behr (2015). “Architecture and Parameter Optimization for Aircraft Electro-Hydraulic Power Generation and Distribution Systems”. In: *SAE Technical Paper*. SAE International. DOI: 10.4271/2015-01-2414.

- Edward, F., P. E. Friedland, B. B. Johnson, H. P. Nii, H. Schorr, and H. Shrobe (1993). *Knowledge-Based Systems in Japan*. Ed. by R. S. Englemore. Loyola College, Maryland, World Technology Evaluation Center (WTEC), Inc.; Baltimore, US.
- Eek, M. (2016). “On Credibility Assessment in Aircraft System Simulation”. Linköping Studies in Science and Technology. Dissertation No. 1758. Department of Management and Engineering (IEI), Linköping University, Sweden.
- EIA632 (2003). *Processes for Engineering a System*. Tech. rep. SAE International.
- Eppinger, S. D. and T. R. Browning (2012). *Design structure matrix methods and applications*. Engineering systems. Cambridge, Mass. : MIT Press. ISBN: 9780262017527.
- Fritzson, P. (2004). *Principles of object-oriented modeling and simulation with Modelica 2.1*. New York ; Chichester: Wiley, cop. ISBN: 0471471631.
- Fruchterman, T. M. and E. M. Reingold (1991). “Graph drawing by force-directed placement”. In: *Software - Practice and Experience* 21.11, pp. 1129–1164. ISSN: 00380644.
- Ganev, E. and M. Koerner (2013). “Power and Thermal Management for Future Aircraft”. In: *SAE Technical Paper*. SAE International. DOI: 10.4271/2013-01-2273.
- GAO (2016). *GAO Reports: Technology Readiness Assessment Guide*. Tech. rep. GAO-16-410G. U.S. Government Accountability Office (GAO), pp. 1–146.
- Gavel, H. (2007). “On Aircraft Fuel Systems: Conceptual Design and Modeling”. Linköping Studies in Science and Technology. Dissertation No. 1067. Department of Management and Engineering (IEI), Linköping University, Sweden. ISBN: 9789185643042.
- GEXF (2015). *GEXF — The Graph Exchange File Format: Language Specification*. [Online; accessed 2015-03-29]. URL: <https://gephi.org/gexf/format/>.
- GraphML (2015). *GraphML — The GraphML Specification*. [Online; accessed 2015-04-12]. URL: <http://graphml.graphdrawing.org/specification.html>.
- Grönstedt, T., O. Thulin, A. Lundbladh, M. Irannezhad, and L. Xu (2014). “First and Second Law Analysis of Future Aircraft Engines”. In: *Journal of Engineering for Gas Turbines and Power, American*

- Society of Mechanical Engineers (ASME)* 136.3.031202. ISSN: 978-079185513-3. DOI: 10.1115/1.4025727.
- Groß, J. and S. Rudolph (2012). “Generating simulation models from UML - A FireSat example”. In: *Proceedings of the 2012 Spring Simulation Multiconference, SpringSim 2012 - Theory of Modeling and Simulation: DEVS Integrative M and S Symposium*. Vol. 44. 4, pp. 182–189.
- Gudmundsson, S. (2014). *General aviation aircraft design: Applied Methods and Procedures*. Kidlington, Oxford, UK; Waltham, MA: Butterworth-Heinemann. ISBN: 9780123973085.
- Hallberg, P. (2012). *Low-Cost Demonstrators: Enhancing Product Development with the Use of Physical Representations*. Linköping Studies in Science and Technology. Thesis: 1563. Linköping: Linköping University, Department of Management and Engineering. ISBN: 9789175197272.
- Hammarström, P. (2016). “Gripen E Avionics Architecture - the new frontline against cost and complexity”. In: *Aerospace Technology Congress*. Fyg- och Rymdtekniska Förening (FTF), Stockholm, Sweden.
- Haskins, C., K. Forsber, and M. Krueger (2007). *Systems Engineering Handbook*. 3rd ed. International Council on Systems Engineering (INCOSE), Carlifornia, U.S.
- Haveman, S. P. and G. M. Bonnema (2015). “Communication of Simulation and Modelling Activities in Early Systems Engineering”. In: *2015 Conference on Systems Engineering Research, Procedia Computer Science*. Elsevier B.V., pp. 305–314.
- Hayhurst, D. R., K. T. Kedward, H. T. Soh, and K. L. Turner (2012). “Innovation-led Multi-disciplinary Undergraduate Design Teaching”. In: *Journal of Engineering Design* 23.3, pp. 159–184. ISSN: 09544828. DOI: 10.1080/09544828.2010.544248.
- Helmer, R., A. A. Yassine, and C. Meier (2010). “Systematic module and interface definition using component design structure matrix.” In: *Journal of Engineering Design* 21.6, pp. 647–675. ISSN: 09544828.
- Herzog, E. (2004). “An approach to systems engineering tool data representation and exchange”. Linköping Studies in Science and Technology. Dissertation No. 867. Department of Computer and Information Science (IDA), Linköping University, Sweden.
- Holmberg, G. (2000). “Integrated Product Development-a Key to Affordability”. In: *Proceedings of the 22th Congress of the International*

- Council of the Aeronautical Sciences*. ICAS, , Harrogate, UK. ISBN: ISBN 0 9533991 2 5.
- Hu, L., T. Yongliang, G. Yuan, B. Jinpeng, and Z. Jiangnan (2015). “System of systems oriented flight vehicle conceptual design: Perspectives and progresses”. In: *Chinese Journal of Aeronautics* 28.3, pp. 617–635. ISSN: 1000-9361. DOI: <http://dx.doi.org/10.1016/j.cja.2015.04.017>.
- Hu, Y. (2006). “Efficient and high Quality Force-Directed Graph Drawing”. In: *The Mathematica Journal*, vol. 10, Issue 1. Wolfram Media, Inc., pp. 37–71.
- Hunt, E. H., D. H. Reid, D. R. Space, and F. E. Tilton (1995). *Commercial Airliner Environmental Control System: Engineering Aspects of Cabin Air Quality*. Tech. rep. Aerospace Medical Association annual meeting, Anaheim, California, U.S.
- IBM (2016). *Rational DOORS: Dynamic Object Oriented Requirement Software*. [Online; accessed: 2016-09-07]. URL: [http://www.ibm.com/support/knowledgecenter/SSYQBZ\\_9.6.1/com.ibm.doors.requirements.doc/topics/c\\_welcome.html](http://www.ibm.com/support/knowledgecenter/SSYQBZ_9.6.1/com.ibm.doors.requirements.doc/topics/c_welcome.html).
- IEC 60848 (1999). *IEC 60848 Ed. 2: Specification language GRAFCET for sequential function charts*. Tech. rep. 3B/WG14, International Electrotechnical Commission.
- IEEE Std. 830 (1998). *Institute of Electrical and Electronics Engineers: IEEE Recommended Practice for Software Requirements Specifications (IEEE Std. 830-1998)*. Tech. rep. IEEE Compute Society, New York, U.S.
- INCOSE (2014). *A World in Motion: Systems Engineering Vision 2025*. International Council on Systems Engineering (INCOSE), Carlifornia, U.S.
- INCOSE (2015). *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 4th ed. John Wiley & Sons, p. 304. ISBN: 1118999401.
- Ingram, C., T. Dendinger, E. Inclan, Y. Charront, K. Handschuh, I. Chakraborty, E. Garcia, and D. N. Mavris (2015). “Integrating Subsystem Sizing into the More Electric Aircraft Conceptual Design Phase”. In: *Proceedings of the 53rd AIAA Aerospace Science Meeting*. Vol. 1676. AIAA 2015-1682. American Institute of Aeronautics and Astronautics (AIAA), FL, U.S., pp. 12542–12560.

- ISO 14040 (2006). *ISO 14040:2006 - Environmental management - Life cycle assessment - Principles and framework*. Tech. rep. International Organization for Standardization.
- ISO 15288 (2008). *ISO/IEC/IEEE 15288: Systems and Software Engineering - System Life Cycle Processes*. Tech. rep. IEEESTD, pp. c1–84. DOI: 10.1109/IEEESTD.2008.4475828.
- Jackson, S. (1997). *Systems engineering for commercial aircraft*. Aldershot : Ashgate, cop. 1997. ISBN: 0291398464.
- Jändel, M., P. Bivall, P. Hammar, R. Johansson, F. Kamrani, and M. J. Quas (2016). *Visual Analytics: Perspectives on the Field of Interactive Visualization*. FOI-R-4200-SE ISSN 1650-1942. Totalförsvarets forskningsinstitut (FOI), Stockholm, Sweden.
- Johansson, B., C. Jouannet, and P. Krus (2003). “Distributed Aircraft Analysis Using Web Service Technology”. In: *SAE Technical Paper*. SAE International. DOI: 10.4271/2003-01-3007.
- Johansson, C. (2013). “On System Safety and Reliability in Early Design Phases: Cost Focused Optimization Applied on Aircraft Systems.” Linköping Studies in Science and Technology. Dissertation No. 1600. Department of Management and Engineering (IEI), Linköping University, Sweden.
- Jouannet, C., P. Berry, T. Melin, K. Amadori, D. Lundström, and I. Staack (2012). “Subscale Flight Testing used in Conceptual Design”. In: *Aircraft Engineering and Space Technology* 84.3, pp. 192–199. ISSN: 1748-8842. DOI: 10.1108/00022661211222058.
- Jouannet, C. and P. Krus (2006). “Direct Simulation Based Optimization for Aircraft Design Including Systems”. In: *ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics (AIAA), Washington D.C, U.S.
- Jung, S., G.-B. Park, and D.-H. Choi (2013). “A Decomposition Method for Exploiting Parallel Computing Including the Determination of an Optimal Number of Subsystems”. In: *Journal of Mechanical Design*. Vol. 135. 041005. American Society of Mechanical Engineers (ASME), New York City, U.S.
- Kirby, M. R. (2001). “A Methodology for Technology Identification, Evaluation, and Selection in Conceptual and Preliminary Aircraft Design”. PhD thesis. Georgia Institute of Technology, U.S.
- Krus, P. (2003). “Simulation Based Optimisation for System Design”. In: *Proceedings of ICED 03, the 14th International Conference on Engineering Design*. ICED, Stockholm.

- Krus, P. (2009). “Whole Mission Simulation for Aircraft System Design and Optimisation”. In: *Proceedings of the 3th CEAS European Air & Space Conference*. Council of the European Aerospace Societies (CEAS). Royal Aeronautical Society, London.
- Krus, P. (2015). *Engineering System Design*. Vol. 1.0. LiU-IeI-R-07/0010-SE. Linköping University, Sweden.
- Krus, P. (2016). “Models Based on Singular Value Decomposition for Aircraft Design”. In: *Proceedings of the Aerospace Technology Congress 2016*. Flyg- och Rymdtekniska Förening (FTF), Stockholm, Sweden.
- Kwan, I. and D. Rutherford (2015). *Transatlantic Airline Fuel Efficiency Ranking 2014*. Tech. rep. Washington DC, U.S.: The International Council on Clean Transportation (icct).
- La Rocca, G. (2011). “Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization”. PhD thesis. Technische Universiteit Delft, Netherlands.
- La Rocca, G., J. Jansen, and T. Zill (2013). “Investigation of Multi-Fidelity and Variable-Fidelity Optimization Approaches for Collaborative Aircraft Design”. In: *Proceedings of the 4th CEAS European Air & Space Conference*. Council of the European Aerospace Societies (CEAS). Linköping University Electronic Press, 2013, p. 709.
- La Rocca, G. and M. J. van Tooren (2007). “A Knowledge Based Engineering Approach to Support Automatic Generation of FE Models in Aircraft Design”. In: *45th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics (AIAA), Washington D.C, U.S.
- Larsson, J. (2006). “A framework for implementation-independent simulation models”. In: *SIMULATION: Transactions of the Society for Modeling and Simulation International*, SAGE 82.9, pp. 563–579. ISSN: 00375497. DOI: 10.1177/0037549706071902.
- Lee, J. J., S. P. Lukachko, I. A. Waitz, and A. Schafer (2001). “Historical and Future Trends in Aircraft Performance, Cost, and Emissions.” In: *Annual Review of Energy and the Environment*, Annual Reviews 26, pp. 167–200. ISSN: 10563466.
- Liscouët - Hanke, S. (2008). “A Model-Based Methodology for Integrated Preliminary Sizing and Analysis of Aircraft Power System Architectures”. PhD thesis. Institut National des Sciences Appliquées (INSA) de Toulouse, Université De Toulouse, France.

- Maier, M. W. (1998). "Architecting principles for systems-of-systems". In: *Systems Engineering* 1.4, pp. 267–284. ISSN: 10981241. DOI: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.
- Martin, J. N. (1994). "The PMTE Paradigm: Exploring the Relationship Between Systems Engineering Process and Tools". In: *INCOSE International Symposium* 4.1, pp. 176–183. ISSN: 2334-5837. DOI: 10.1002/j.2334-5837.1994.tb01700.x.
- Martin, J. N. (1996). *Systems Engineering Guidebook: A Process for Developing Systems and Products*. 1st ed. CRC Press Inc.
- Martin, J. N. (2012). "Four Thought Patterns in Support of the Systems Approach". In: *International Federation for Systems Research (IFSR) Conversation: Team 4: Towards a Common Language for Systems Praxis*. Linz, Austria, pp. 25–27.
- Mavris, D. N. and K. Griendling (2016). "System and System of Systems Model driven Design and Engineering". In: *10th MODPROD Workshop on Model-Based Product Development*. Vol. 10. Linköping University, Sweden.
- Mavris, D. N., O. J. Pinon, and D. J. Fullmer (2010). "Systems Design and Modeling: A Visual Analytics Approach". In: *7th International Congress of the Aeronautical Sciences, ICAS, Nice, France*.
- McMasters, J. H. and R. M. Cummings (2004). "Rethinking the Airplane Design Process - An Early 21st Century Perspective". In: *42nd Aerospace Science Meeting*. AIAA 2004-0693. American Institute of Aeronautics and Astronautics (AIAA), Washington D.C, U.S., pp. 1–26.
- Melin, T. (2013). *Parametric Airfoil Catalog Part I, Archer A18 to Götingen 655: An Aerodynamic and Geometric Comparison Between Parametrized and Point Cloud Airfoils*. 1. Linköping University Electronic Press, Sweden, p. 561.
- Melin, T., K. Amadori, and P. Krus (2011). "Parametric wing profile description for conceptual design". In: *The International Conference of the European Aerospace societies, CEAS*. Venice, Italy.
- Miller, G. A. (1956). "The Magical Number Seven, Plus or Minus Two Some Limits on Our Capacity for Processing Information". In: *The Psychological Review, American Psychological Association*. Vol. 101. 2, pp. 343–352.
- Modelica Association (2016). *FMI-Standard: Functional Mock Up Interface*. Online; accessed 2016-04-14. URL: <https://www.fmi-standard.org>.



- Moerland, E., R. G. Becker, and B. Nagel (2015). "Collaborative understanding of disciplinary correlations using a low-fidelity physics-based aerospace toolkit". In: *CEAS Aeronautical Journal* 6.3, pp. 441–454. ISSN: 18695582.
- Moir, I. and A. G. Seabridge (2008). *Aircraft systems: Mechanical, electrical, and avionics subsystems Integration*. Aerospace series. Chichester, West Sussex, England; Hoboken, N.J. : J. Wiley & Sons Ltd., c2008. ISBN: 9780470059968.
- Møller, A. and M. Schwartzbach (2006). *An Introduction to XML and Web Technologies*. Addison-Wesley, Essex. UK. ISBN: 0-321-26966-7.
- Munjulury, R. C. (2014). *Knowledge Based Integrated Multidisciplinary Aircraft Conceptual Design*. Linköping Studies in Science and Technology. Thesis: 1661. Linköping : Department of Management and Engineering, Linköping University. ISBN: 9789175193281.
- Munjulury, R. C., I. Staack, P. Krus, and P. Berry (2016). "A knowledge-based integrated aircraft conceptual design framework". In: *CEAS Aeronautical Journal* 7.1, pp. 95–105. ISSN: 18695590.
- Nagel, B., D. Böhnke, V. Gollnick, P. Schmollgruber, A. Rizzi, G. La Rocca, and J. J. Alonso (2012). "Communication in Aircraft Design: Can we establish a Common Language?" In: *28th International Congress of the Aeronautical Sciences*. ICAS, Brisbane, Australia.
- NASA (2007). *Systems Engineering Handbook: NASA/SP-2007-6105 Rev1*. Tech. rep. National Aeronautics and Space Administration (NASA), Washington D.C., U.S.
- Nickol, C. L. (2004). "Conceptual Design Shop". In: *Presentation to Conceptual Aircraft Design Working Group (CADWG21)*.
- NTSB (1972). *Aircraft Accident Report, Pan American World Airways Inc., Boeing 747, N747PA, Flight 845, San Francisco, California, July 30, 1971*. AAR 72-17. Washington D.C.: National Transportation Safety Board (NTSB).
- openVSP (2016). *The NASA Open Source Parametric Geometry (open-VSP) Software*. [Online; accessed: 2016-10-18]. URL: <http://www.openvsp.org/>.
- Pahl, G., W. Beitz, J. Feldhusen, and K.-H. Grote (2007). *Engineering design : a systematic approach*. Berlin ; London : Springer. ISBN: 1846283183.
- Pohl, K. and C. Rupp (2011). *Requirements engineering fundamentals: A study guide for the certified professional (IREB)*. 1st. Rocky Nook, Inc., Santa Barbara, U.S.

- Pop, A. and P. Fritzson (2003). “ModelicaXML: A Modelica XML Representation with Applications”. In: *Proceedings of the 3rd International Modelica Conference (Modelica 2003)*, pp. 419–430.
- Porciúncula, G. S. (2009). “Methodology for the Reliability Analysis on the Automatic System Design”. PhD thesis. Universidade Federal de Santa Catarina, Florianópolis, Brazil.
- Porciúncula, G. S., H. C. Belan, V. J. De Negri, and A. Dias (2016). “Identification of the operational configurations of automatic systems for the design for reliability”. In: *The Brazilian Society of Mechanical Sciences and Engineering 2015, Springer Ltd.* 38.2, pp. 493–506. ISSN: 1678-5878. DOI: 10.1007/s40430-015-0334-4.
- Raymer, D. P. (2006). *Aircraft Design: A Conceptual Approach*. 4th ed. AIAA education series. Washington, D.C.: American Institute of Aeronautics and Astronautics. ISBN: 1563478293.
- Reisig, W. (1992). *A primer in Petri net design*. Springer Compass International. Springer, pp. I–XII, 1–120. ISBN: 978-3-540-52044-3.
- Roberts, W., K. Griendling, A. Gray, and D. N. Mavris (2016). “Unmanned Vehicle Collaboration Research Environment for Maritime Search and Rescue”. In: *30th Congress of the International Council of the Aeronautical Sciences*. ICAS, Daejeon, South Korea.
- Robertson, S. and J. Robertson (2013). *Mastering the requirements process: Getting requirements right*. 3rd. Addison-Wesley.
- Roskam, J. (1985). *Airplane design: Preliminary Sizing of Airplanes*. Ottawa, Kan. : Roskam Aviation and Engineering.
- Roskam, J. (2007). *Lessons learned in aircraft design : the devil is in the details*. Lawrence, Kansas, U.S. : Darcorporation. ISBN: 1884885586.
- Ross, H. (2016). “12th Research & Education (READ) & SCAD Conference”. In: *Around the World with Solar Power*. Polish Society of Aeronautics and Astronautics, Warsaw, Poland.
- RTCA (2000). *DO-254: Design Assurance Guidance for Airborne Electronic Hardware*. Tech. rep. Radio Technical Commission for Aeronautics (RTCA), SC-180, Inc., Washington, DC, U.S.
- RTO/NATO (2007). *Performance Prediction and Simulation of Gas Turbine Engine Operation for Aircraft, Marine, Vehicular, and Power Generation*. Tech. rep. RTO-TR-AVT-036. Research and Technology Organisation (RTA), North Atlantic Treaty Organisation (NATO).
- Rudolph, S. (2014). “Design languages for multi-disciplinary architectural synthesis and analysis of complex systems in the context of an

- aircraft cabin". In: *4th SCAD - Symposium on Collaboration in Aircraft Design*. CEAS Technical Committee Aircraft Design (TCAD).
- Rugg, H. R. (1970). "A Definition of the Aircraft Stretch Efficiency Factor". In: *29th Annual Conference*. Society of Allied Weight Engineers (SAWE), Inc. Washington, DC: Society of Allied Weight Engineers, Inc., p. 38.
- Sammartino, S. (2013). *The Super Awesome Micro Project: A full sized Lego car (crowd funded)*. [Online; accessed: 2016-09-13]. URL: [https://www.youtube.com/watch?v=\\_0bE4\\_nMCjE](https://www.youtube.com/watch?v=_0bE4_nMCjE).
- Schaeffer, S. E. (2007). "Graph clustering". In: *Computer Science Review, Elsevier Ltd.* 1.1, pp. 27–64. ISSN: 15740137. DOI: 10.1016/j.cosrev.2007.05.001.
- Scholz, D. (1996). "Development of a CAE tool for the design of flight control and hydraulic systems". In: *Aerotech 95*. Avionic systems, design, and software 11. Royal Aeronautical Society (RAeS), London, UK. IMECHE seminar publication, pp. 1–22.
- Scholz, D. (1998). "DOCsys - A Method to Evaluate Aircraft Systems". In: *Bewertung von Flugzeugen*. Workshop: DGLR Fachausschuß S2 - Luftfahrtsysteme, München. Deutsche Gesellschaft für Luft- und Raumfahrt, Bonn.
- Scholz, D., R. Seresinhe, I. Staack, and C. Lawson (2013). "Fuel Consumption Due to Shaft Power Off-takes from the Engine". In: *Proceedings of the 4th International Workshop on aircraft System Technologies (AST)*. Hamburg, Germany, pp. 169–179.
- Seki, N., N. Morioka, H. Saito, and H. Oyori (2015). "A Study of Air/-Fuel Integrated Thermal Management System". In: *SAE Technical Paper*. SAE International. DOI: 10.4271/2015-01-2419.
- Seresinhe, R., C. Lawson, and R. Sabatini (2013). "Environmental Impact Assessment, on the Operation of Conventional and More Electric Large Commercial Aircraft". In: *SAE Int. J. Aerosp.* 6, pp. 56–64. DOI: 10.4271/2013-01-2086.
- Sethson, M. (1999). "Complex cavity analysis: analytical fluid-power models using CAD information". Linköping Studies in Science and Technology. Dissertation No. 575. Dept. of Mechanical Engineering, Linköping University. ISBN: 9172194642.
- Sjölund, M. (2015). "Tools and Methods for Analysis, Debugging, and Performance Improvement of Equation-Based Models". Linköping Studies in Science and Technology. Dissertation No. 1664. Depart-

- ment of Computer and Information Science (IDA), Linköping University, Sweden.
- Slingerland, R. and S. Zandstra (2007). “Bleed Air Versus Electric Power off-Takes from a Turbofan Gas Turbine Over the Flight Cycle”. In: *Collection of Technical Papers - 7th AIAA Aviation Technology, Integration, and Operations Conference*. Vol. 2. Belfast, Northern Ireland, pp. 1516–1527.
- Sóbestor, A. and A. I. J. Forrester (2015). *Aircraft aerodynamic design: geometry and optimization*. Aerospace series. Chichester, West Sussex, United Kingdom : John Wiley & Sons, Ltd. ISBN: 9781118534731.
- Staack, I., H. Ellström, M. Bergman, P. Sarwe, and P. Krus (2012). “More Electrical Environmental Control System Simulation”. In: *Proceedings of the 28th International Congress of the Aeronautical Sciences*. ICAS, Brisbane, Australia.
- Staack, I. and P. Krus (2013). “Integration of On-Board Power Systems Simulation in Conceptual Aircraft Design”. In: *Proceedings of the 4th CEAS European Air & Space Conference*. Council of the European Aerospace Societies (CEAS). Linköping University Electronic Press, 2013, p. 709.
- Staack, I., D. Lundström, and P. Krus (2010). “Subscale Flight Testing at Linköping University”. In: *27th International Congress of the Aeronautical Sciences*. ICAS, Nice, France.
- Staack, I., R. C. Munjulury, T. Melin, P. Krus, and A. Abdalla (2014). “Conceptual aircraft design model management demonstrated on a 4th generation fighter”. In: *29th Congress of the International Council of the Aeronautical Sciences*. ICAS, St. Petersburg, Russia.
- Staack, I., R. C. Munjulury, P. Berry, T. Melin, K. Amadori, C. Jouanet, D. Lundström, and P. Krus (2012). “Parametric Aircraft Conceptual Design Space”. In: *Proceedings of the 28th International Congress of the Aeronautical Sciences*. ICAS, Brisbane, Australia.
- Standard, I. (2007-03-15). *ISO 1219-1 Fluid power systems and components - Graphical symbols and circuit diagrams*. Tech. rep. International Organization for Standardization.
- Stein, B. M. (1995). “Functional models in configuration systems”. PhD thesis. Department of Mathematics and Computer Science of the University of Paderborn, Germany.
- Steward, D. V. (1981). “The Design Structure System: A Method for Managing the Design of Complex Systems”. In: *IEEE Transactions*

- on *Engineering Management*. 28(3). Vol. EM-28. No. 3. Institution of Electrical Engineers (IEEE), New Jersey, U.S., pp. 71–74.
- Suh, N. P. (1990). *The Principles of Design*. Oxford University Press. ISBN: 0195043456.
- The Mathworks, Inc. (2015a). *The Mathworks: Bioinformatics Toolbox*. [Online; accessed: 2016-07-04]. URL: <http://se.mathworks.com/help/bioinfo/index.html>.
- The Mathworks, Inc. (2015b). *The Mathworks: Comparison of MATLAB and Other OO Languages*. [Online; accessed: 2016-06-20]. URL: [http://se.mathworks.com/help/matlab/matlab\\_oop/matlab-vs-other-oo-languages.html](http://se.mathworks.com/help/matlab/matlab_oop/matlab-vs-other-oo-languages.html).
- Thebeau, R. E. (2001). “Knowledge Management of System Interfaces and Interactions for Product Development Processes”. MA thesis. Massachusetts Institute of Technology (MIT), U.S.
- Todeschi, M. and F. Salas (2016). “Power Electronics for the Flight Control Actuators”. In: *Recent Advances in Aerospace Actuation Systems and Components*. Vol. 6. R3ASC. Institute National des Sciences Appliquées, Toulouse, France, pp. 1–9.
- Torenbeek, E., H. Wittenberg, and S. Calvert (2009). *Flight Physics: Essentials of Aeronautical Disciplines and Technology, with Historical Notes*. Dordrecht: Springer. ISBN: 9781402086632.
- Ullman, D. G. (2002). *The Mechanical Design Process*. 3rd ed. McGraw-Hill series in mechanical engineering. McGraw-Hill, New York. ISBN: 0072373385.
- Ulrich, K. T. and S. D. Eppinger (2012). *Product design and development*. Boston, Mass. : McGraw-Hill/Irwin. ISBN: 9780071086950.
- Utterback, J. M. (1996). *Mastering the Dynamics of Innovation*. Harvard Business School Press, Boston, Massachusetts, U.S. ISBN: 9780875847405.
- Van der Laan, A. H. (2008). “Knowledge based engineering support for aircraft component design”. PhD thesis. Faculty of Aerospace Engineering, Delft University of Technology, Netherland.
- Vesely, W., J. Dugan, J. Fragola, Minarick, and J. Railsback (2002). *Fault Tree Handbook with Aerospace Applications*. Handbook. Washington, DC: National Aeronautics and Space Administration.
- VINNOVA (2016). *VINNOVA: Nationellt flygtekniska forskningsprogrammet NFFP(6) The national aviation reserach program NFFP 6 2013-2016*. [Online; accessed 2015-03-20]. URL: <http://www.vinnova.se/sv/Var-verksamhet/Gransoverskridande>

- samverkan / Samverkansprogram / Nationella - flygtekniska - forskningsprogrammet/.
- Visser, W. P. (2015). *Generic Analysis Methods for Gas Turbine Engine Performance: The development of the gas turbine simulation program GSP*. Dissertation, Technische Universiteit Delft, Netherland. ISBN: 978-94-6259-492-0.
- Waldrop, M. M. (2016). "The chips are down for Moore's law". In: *Nature*. Vol. Volume 530. Issue 7589. Macmillan Publishers, pp. 144–147.
- Wang, S., M. Tomovic, and H. Liu (2015). *Commercial aircraft hydraulic systems*. Shanghai Jiao Tong University Press aerospace series. Waltham, MA : Academic Press, Elsevier. ISBN: 9780124199729.
- Weilkiens, T., J. G. Lamm, S. Roth, and M. Walker (2015). "Model-Based System Architecture: The V-Model". In: *Wiley Series in Systems Engineering and Management*, John Wiley & Sons, Inc. Chap. B, pp. 343–352. ISBN: 9781118893647.
- Wikipedia (2016). *Image: Digesting Duck — Wikipedia, The Free Encyclopedia*. [Online; accessed 2016-02-16]. URL: [https://en.wikipedia.org/wiki/Digesting\\_Duck](https://en.wikipedia.org/wiki/Digesting_Duck).
- Yassine, A. A. (2004). "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method". In: *Quaderni di Management (Italian Management Review)*. 9. Quaderni di Management (Italian Management Review).
- Yu, T., D. E. Goldberg, K. Sastry, C. F. Lima, and M. Pelikan (2009). "Dependency Structure Matrix, Genetic Algorithms, and Effective Recombination". In: *Evolutionary Computation*. Vol. 17. 4. Massachusetts Institute of Technology, pp. 595–626.
- Yu, T., A. A. Yassine, and D. E. Goldberg (2007). "An information theoretic method for developing modular architectures using genetic algorithms". In: *Research in Engineering Design*. Vol. 18 Issue 2. Springer, pp. 91–108.
- Zhang, M. (2015). "Contributions to Variable Fidelity MDO Framework for Collaborative and Integrated Aircraft Design." PhD thesis. KTH Royal Institute of Technology, Stockholm, Sweden. ISBN: 978-91-7595-606-0.

# Index

## A

- acausal ..... *see* implementation
- aesthetic ..... 67
- aircraft
  - Airbus A320 ..... 16, 101, 104
  - Airbus A380 ..... 101
  - Boeing 747 ..... 15, 102
  - Boeing 757 ..... 16
  - Boeing 787 ..... 15, 44, 101
  - Douglas DC-8 ..... 45
  - F-104 Starfighter ..... 103
  - Fokker 100 ..... 102
- analytical model ..... 50
- architectural model ..... 51
- ATA 100 ..... 38
- centralised layout, hydraulic .. 101, 109, 113
- certification ..... 65
- channel-agency net 52, **85**, 90, 117
  - agency ..... 88
  - channel ..... 87
  - extended ..... 89
  - matter ..... 87
  - useful energy ..... 87
- class ..... 108
- cluster
  - clustering ..... 73, 76
  - notation ..... 78, 81
  - properties ..... 74, 76, 78
- clustering ..... 120
- communication, tool to tool 23, 94
- constrain/sizing diagram ..... 21
- cyber-physical systems ..... 48

## B

- banding ..... 73
- black box model ..... 34
- bleed air ..... 18, 45
- block lower triangular ..... 76
- bus component ..... 79

## C

- C-A net ... *see* channel-agency net
- CAD, implementation ..... 96
- CADLab framework .... 28, 58, 93
- car model ..... 54

## D

- data format, CPACS ..... 25
- database ..... 23 f, 93
- degree of sys. electrification 19, 38
- dependency structure matrix .. *see* DSM
- design
  - assurance level ..... **37**, 66
  - compiler ..... 69
  - influence factors ..... 14, 34 f
  - process paradigm ..... 35

dimensions, tool ..... 57  
 direct operating costs ..... 35  
 domain, CAD ..... 27  
 DSM ..... 71, 120, 131  
     attributes ..... 74  
     categories ..... 71  
     cluster attributes ..... 75  
     partitioning ..... 73, 76, 119

## **E**

edge ..... 83  
 efficiency  
     exergy ..... 46  
     energy ..... 46  
     fleet ..... 15  
     fuel ..... 15, 36  
     power off-takes ..... 18  
     system ..... 44  
 energy  
     transformation ..... 39  
 environmental control system .. 44  
 exergy ..... 46  
 expert system ..... 69

## **F**

failure modes and effects analysis  
     37  
 fault tree analysis ... **36**, 113, 115,  
     121 f  
 flight control system ..... 100, 114  
 fragmentation ..... 78  
 fuel consumption, specific ..... 14  
 functional mockup interface .... 55

## **G**

generic algorithm ..... 77  
 Grafacet ..... 52  
 graph ..... 83, 115, 118, 120

density ..... 84  
 graphical setup ..... 67  
 graphical user interface ..... 85  
 growth factor ..... 19 f

## **H**

hierachy, tool ..... 25  
 hydraulic  
     actuator ..... 109 f  
     system ..... 100 f  
     system architecture ..... 104 f  
 hydro-electric power plant .... 88 f

## **I**

implementation  
     acausal ..... 28, 98  
     causal ..... 28  
 independence axiom ..... 56  
 integrated circuit ..... 47

## **K**

knowledge base system .. 105, 110,  
     113  
 knowledge-based engineering .. 64,  
     69, 103  
     implementation ..... 103  
     rules ..... 40, 44, 64

## **M**

MDDSM ..... 82, 89 f  
 metamodel ..... 107  
 Miller's law ..... 66  
 model  
     abstraction ..... 49  
     concepts ..... 53  
     decomposition ..... 26  
     fidelity ..... 56 f  
     geocentric/heliocentric ..... 56



- ul style="list-style-type: none;">
- hierarchy ..... 67
- integration ..... 55
- refinement ..... 86
- six degree of freedom ..... 111
- transformation ..... 51
- translation ..... 70
- ModelicaXML ..... 63
- Moore's law ..... 46
- more electric aircraft architecture
  - 40, 44
- multi-domain modelling ..... 82
- N**
- namespace ..... 23, 98
  - network ..... 37, 85, 119 f
- O**
- object oriented programming .. 27,
    - 95, 106, 108
  - on-board systems ..... 17
  - ontology ..... 61
  - otimization, local ..... 28
  - outer mold line ..... 25
- P**
- paradigm pyramid ..... 61
  - parametric ..... 98, 127
  - parametric design ..... 24
  - parametric model ..... 53
  - partitioning ..... *see* DSM
  - payload ..... 20
  - performance
    - computational ..... 46, 57
    - index ..... 13, 17
  - perti net ..... 86
  - PICARD theory ..... 31
  - PMTE Pyramide ..... 62
  - power component ..... 41
  - power electronics ..... 47
  - power off-take *see* secondary power
  - product tree ..... 26, 116
  - prototype ..... 53 f
- R**
- range equation ..... 19
  - reliability, system ..... 100
  - requirements ..... 21, 34, 37
- S**
- scaleability ..... 40
  - secondary effects ..... 47, 58
  - secondary power
    - consumption ..... 18
    - off-takes ..... 18 f, 36
    - system ..... 17
  - semantic ..... 61, 130
  - semantic web ..... 62
  - sensitivity analysis ..... 20, 99
  - signal component ..... 46
  - simulation
    - model ..... 55, 110
    - tools ..... 105, 128
  - singular value decomposition ... 99
  - sizing ..... 21
  - software
    - CATIA ..... 28, 94 f
    - Hopsan ..... 67, 89, 107, 110
    - Matlab ..... 108
    - Modelica .. 55, 63, 76, 105, 107
    - RAPID ..... 96
    - Tango ..... 95 f
  - solver, central ..... 107
  - solver, distributed ..... 107
  - spare holding costs ..... 36
  - stakeholder ..... 32, 51
  - Sterling number ..... 75

subsystem ..... 32  
system  
    decomposition ..... 26, 38, 71  
    knowledge ..... 61, 64  
    systems of systems ..... 32

## **T**

tearing ..... 73  
technology ..... 37  
technology readiness level ..... 40  
tool fidelity ..... 57, 59  
tools ..... *see* software

## **U**

UML ..... 50  
uncertainty ..... 59  
unified modelling ..... 62, 117

## **V**

validation ..... 9, 63, 91, 123  
verification ..... 9, 91  
vertex ..... 83  
visualisation ..... 66, 118 f

## **W**

weight ..... 14, 18  
weight breakdown ..... 18

## **X**

XML ..... 62, 115, 130  
    database ..... 95  
    schema (XSD) 74 f, 90, 96, 106  
    style sheet (XSLT) 94, 96, 130

“Aeronautics was neither an industry  
nor a science - It was a miracle

Igor Sikorsky, 1889-1972