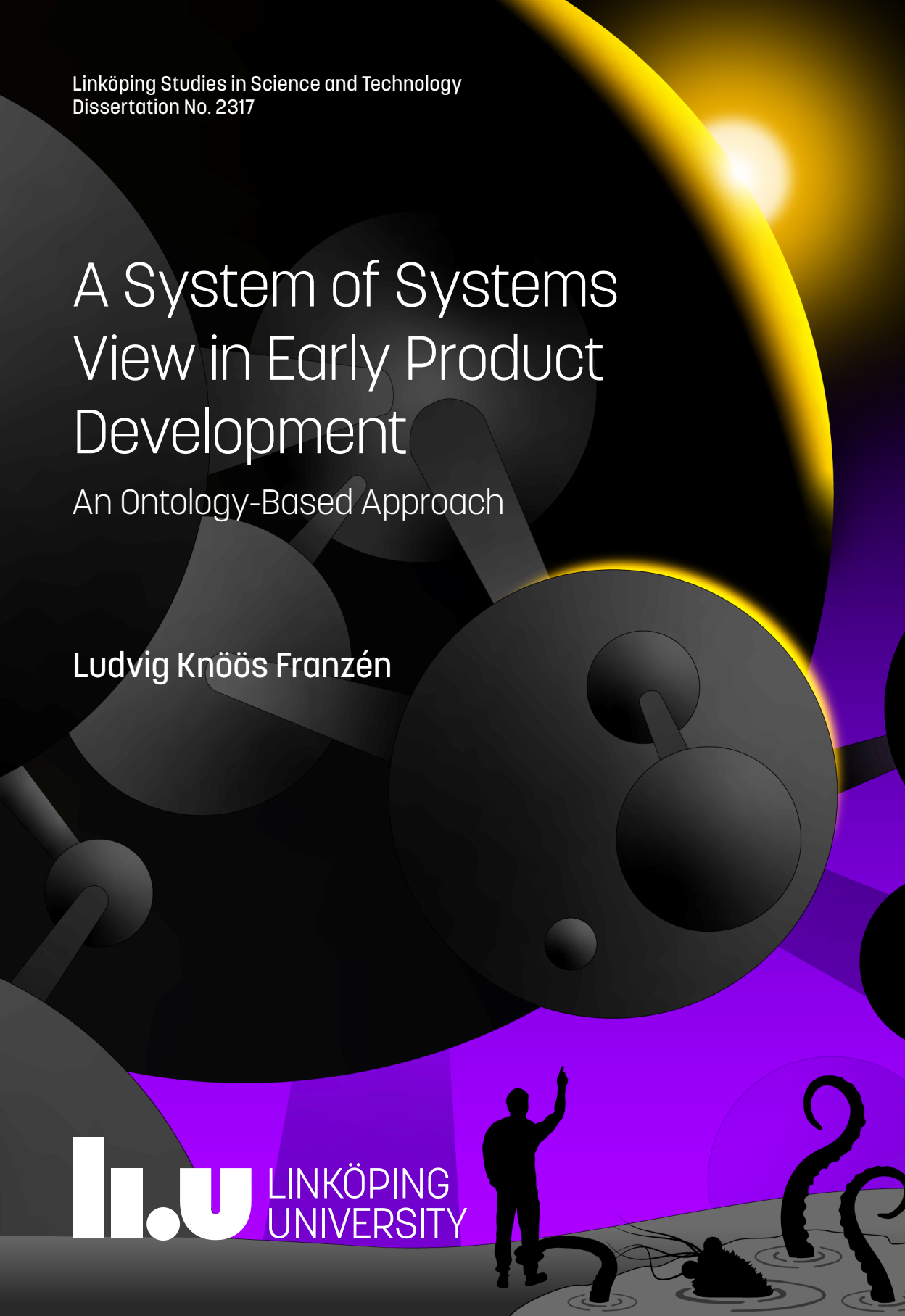
Linköping Studies in Science and Technology
Dissertation No. 2317

A System of Systems View in Early Product Development

An Ontology-Based Approach

Ludvig Knöös Franzén



LINKÖPING UNIVERSITY

Linköping Studies in Science and Technology
Dissertations No. 2317

A System of Systems View in Early Product Development

An Ontology-Based Approach

Ludvig Knöös Franzén

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

Linköping 2023

A System of Systems View in Early Product Development
An Ontology-Based Approach

ISBN 978-91-8075-165-0 (Print)
ISBN 978-91-8075-166-7 (PDF)
DOI https://doi.org/10.3384/9789180751667
ISSN 0345-7524

Cover: Ludvig Knöös Franzén, 2023

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 2023.

To my Friends and Family

> Allting har en ände utom korven som har två

Pappa

Abstract

The concept of system-of-systems is becoming increasingly common and relevant in many engineering applications. Today's highly interconnected world entails that more and more systems have dependencies on other systems. This increasing number of interdependencies results in new levels of complexity that must be managed in the early development of new products. Different viewpoints must also be handled to understand the many layers of a system-of-systems and its surrounding context. An ever-changing future results in uncertainty about the operational environment but also other aspects, such as available technologies, which complicates the matter even further. Traditional approaches for early product development can be used to some extent, but the complexity, scale and sheer number of interconnections in system-of-systems require a holistic perspective to obtain an early understanding of the problem, design spaces, and multiple aspects involved.

This dissertation aims to present a method that has been developed to address the demand for a more holistic system-of-systems view in early product development. Overall, the method consists of four parts that together show how design spaces for system-of-systems can be generated and later processed to find suitable solutions. Search and rescue operations have been used as examples of system-of-systems throughout this work and in the development of the presented method. The first two parts of the method are based on architecture frameworks and ontologies with description logic reasoning capabilities. An architecture framework is here used to break down system-of-system needs into functions to be fulfilled by constituent systems. Ontologies are thereafter used to represent the outcome and the resulting system-of-system design spaces with involved entities and their relationships. Description logic reasoning can subsequently be used to process the available design spaces and suggest suitable system-of-system solutions. The last two parts of the method build upon a concept exploration and estimation approach, together with visual analytics. The approach illustrates how individual system concepts can be estimated from an ontology-represented design space, and how visual analytics can be used to explore different system-of-system viewpoints at an early stage. Based on the outcomes of the presented method, this dissertation contributes a holistic take on early product development from a system-of-systems perspective.

Populärvetenskaplig sammanfattning

En system-av-system-vy i tidig produktutveckling

Ett ontologibaserat tillvägagångsätt

Konceptet system-av-system blir allt vanligare och relevant i många tekniska tillämpningsområden. Dagens högt sammanlänkade värld innebär att fler och fler system har beroenden med andra system. Detta ökande antal av ömsesidiga beroenden resulterar i nya nivåer av komplexitet som måste hanteras i den tidiga utvecklingen av nya produkter. Olika synsätt måste också hanteras för att förstå de många skikt som ett system-av-system består av och dess omgivande omständigheter. En ständigt föränderlig framtid resulterar i osäkerhet om den operativa miljön men även andra aspekter, såsom tillgängliga teknologier, vilket komplicerar saken ytterligare. Traditionella metoder för tidig produktutveckling kan användas i viss utsträckning, men komplexiteten, skalan och det stora antalet sammankopplingar i system-av-system kräver ett mer holistiskt synsätt för att skapa en tidig förståelse av problemet, designrymden och andra inblandade aspekter.

Denna avhandling syftar till att presentera en metod som har utvecklats för att möta efterfrågan på en mer holistisk system-av-system-vy i tidig produktutveckling. Sammantaget består metoden av fyra delar som tillsammans visar hur designrymder för system-av-system kan genereras och senare bearbetas för att hitta lämpliga lösningar. Sök- och räddningsinsatser har använts som exempel på system-av-system genom hela detta arbete och i utvecklingen av den presenterade metoden. De två första delarna av metoden är baserade på arkitekturramverk och ontologier med beskrivningslogiska resonemangsförmågor. Ett arkitekturramverk används här för att bryta ned system-av-system-behov i funktioner som ska uppfyllas av involverade system. Ontologier används därefter för att representera resultatet och de resulterande system-av-system-designrymderna med involverade entiteter och deras relationer. Beskrivningslogiska resonemang kan sedan användas för att bearbeta de

tillgängliga designrymderna och föreslå lämpliga system-av-system-lösningar. De två sista delarna av metoden bygger på en konceptutforsknings och uppskattningsmetod tillsammans med visuell analysteknik. Tillvägagångssättet illustrerar hur individuella systemkoncept kan estimeras från en ontologirepresenterad designrymd, och hur visuell analysteknik kan användas för att utforska olika system-av-system-perspektiv i ett tidigt skede. Baserat på resultaten av den presenterade metoden bidrar denna avhandling med en helhetssyn på tidig produktutveckling utifrån en system-av-system-vy.

Acknowledgements

This work has been carried out at the Division of Fluid and Mechatronic Systems (FLUMES) at Linköping University. The research has been performed as part of the System-of-Systems Trade Space Exploration (S2TEP) project which is part of the National Aeronautics Research Programme (NFFP7) funded by the Swedish Agency for Innovation (VINNOVA) and the Swedish Armed Forces. The project has also been done in collaboration with Saab Aeronautics.

I would like to start by expressing my gratitude to my main supervisors, Professor Petter Krus and Dr. Christopher Jouannet, who have been both sources of inspiration and excellent guides in my academic journey. I also offer my deepest gratitude to my co-supervisors Dr. Ingo Staack and Dr. Kristian Amadori. Thank you for your endless support, dedication and inspiration. I would additionally like to thank you all for the interesting and enjoyable discussions that we have had about most things in this universe. Sorry for all the jokes that have slipped out my mouth at the most appropriate of moments... I would like to particularly thank Rita Enquist and Jenny Dagberg for their great support in helping me navigate the ocean of administration at the university. Special thanks go to all of my colleagues at Flumes. Thank you all for the many interesting discussions, activities, feeding frenzies, and enjoyable coffee drinking sessions that we have had.

My gratitude also goes to my family and especially my parents and sister for unconditionally supporting me throughout all of my educational endeavours so far. You're the best! Finally, I would like to express my deepest gratitude to my wife Mathilda, who has supported me in every way since the day we first met. Thank you and our "little" Wille-dog for enduring my endless repetitions about my everyday concerns. This dissertation is dedicated to you.
Tack!

Linköping, March 2023

Ludvig Knöös Franzén

Abbreviations

ABS	Agent-Based Simulations
AFOV	Angular Field of View
AI	Artificial Intelligence
ASDL	Aerospace Design Laboratory
BFO	Basic Formal Ontology
BORO	Business Objects Reference Ontology
CAD	Computer Aided Design
CATIA	Computer Aided Three-Dimensional Interactive Application
CFD	Computational Fluid Dynamics
CONOPS	Concept of Operations
CPACS	Common Parametric Aircraft Configuration Schema
CS	Constituent Systems
DoD	Department of Defense
DoDAF	US Department of Defense Architecture Framework
DOE	Design Of Experiments
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DSM	Design Structure Matrix
F/M tree	Function/Means tree
FEAF	Federal Enterprise Architecture Framework
FLAR	Forward Looking Airborne Radar
FMI	Functional Mock-up Interface
GRG	Generalized Reduced Gradient
HFM	High-Fidelity Model
HT	Horizontal Tail
IAMSAR	International Aeronautical and Maritime Search and Rescue
INCOSE	International Council on Systems Engineering

IRMA	Interactive Reconfigurable Matrix of Alternatives
JRCC	Joint Rescue Co-ordination Centre
LFM	Low-Fidelity Model
LKP	Last Known Position
LLM	Large Language Models
MBSE	Model Based Systems Engineering
MDO	Multi-disciplinary Design Optimizations
MFM	Multi-Fidelity Model
MIT	Massachusetts Institute of Technology
MODAF	UK Ministry of Defence Architecture Framework
MOE	Measures of Effectiveness
MOO	Multi-Objective Optimizations
MOP	Measures of Performance
MTOW	Maximum Take-off Weight
MTTD	Mean Time To Detection
NAF	NATO Architecture Framework
OBSE	Ontology-Based Systems Engineering
OpenVSP	Open Vehicle Sketch Pad
OWL	Web Ontology Language
OWL-DL	Web Ontology Language - Description Logics
PCA	Principal Component Analysis
PDF	Probability Density Function
POD	Probability Of Detection
QFD	Quality Function Deployment
RAPID	Robust Aircraft Parametric Interactive Design
RDF	Resource Description Framework
SAR	Search and Rescue
SE	Systems Engineering
SERC	Systems Engineering Research Center
SFC	Specific Fuel Consumption
SMA	Swedish Maritime Administration
SoS	System-of-Systems
SoSE	System-of-Systems Engineering
SPARQL	SPARQL Protocol And RDF QUERY Language
SQWRL	Semantic Query-Enhanced Web Rule Language
SS	Sub-Systems
SSP	System Structure and Parameterization
SVD	Singular Value Decomposition
SWRL	Semantic Web Rule Language
SysML	Systems Modelling Language
TOGAF	The Open Group Architectural Framework
TSE	Trade Space Explorations

TTD	Time To Detection
UAF	Unified Architecture Framework
UAV	Unmanned Aerial Vehicle
UML	Unified Modelling Language
UPDM	Unified Profile for DoDAF and MODAF
VA	Visual Analytics
VT	Vertical Tail
XML	Extensible Markup Language

Papers

The compiled work presented in this dissertation is based on the following publications that can be regarded as its foundation. The publications are chronologically appended and are, apart from minor formatting changes, reproduced in their original forms. The publications are also referred to by their Roman numerals throughout this dissertation. Chapter 7 provides a short summary of the content in each appended paper.

[I] L. Knöös Franzén, I. Staack, C. Jouannet, and P. Krus, "An Ontological Approach to System of Systems Engineering in Product Development," in *Proceedings of the 10th Aerospace Technology Congress (FTF)*, Stockholm: Swedish Society of Aeronautics and Astronautics, 2019, pp. 35–44. DOI: 10.3384/ecp19162004.

[II] L. Knöös Franzén, S. Schön, A. Papageorgiou, *et al.*, "A System of Systems Approach for Search and Rescue Missions," in *Proceedings of the AIAA Scitech 2020 Forum*, Orlando, Florida: American Institute of Aeronautics and Astronautics, 2020, pp. 1–16. DOI: 10.2514/6.2020-0455.

[III] L. Knöös Franzén, I. Staack, P. Krus, *et al.*, "A Breakdown of System of Systems Needs Using Architecture Frameworks, Ontologies and Description Logic Reasoning," *Aerospace*, vol. 8, no. 4, 2021, ISSN: 2226-4310. DOI: 10.3390/aerospace8040118.

[IV] L. Knöös Franzén, I. Staack, P. Krus, *et al.*, "Ontology-Represented Design Space Processing," in *Proceedings of the AIAA AVIATION 2021 Forum*, Virtual Event: American Institute of Aeronautics and Astronautics, 2021, pp. 1–15. DOI: 10.2514/6.2021-2426.

[V] L. Knöös Franzén, S. Schön, I. Staack, *et al.*, "Exploring the Impact of Model Fidelity Through Interactive Visualizations for System of Systems," in *Proceedings of the AIAA Scitech 2022 Forum*, Virtual Event: American Institute of Aeronautics and Astronautics, 2022, pp. 1–14. DOI: 10.2514/6.2022-1467.

[VI] L. Knöös Franzén, R. C. Munjulury, and P. Krus, "Ontology-Assisted Aircraft Concept Generation," in *Proceedings of the 33rd Congress of the International Council of the Aeronautical Sciences*, Stockholm, Sweden, 2022, pp. 1–17.

[VII] L. Knöös Franzén, I. Staack, P. Krus, and K. Amadori, "Optimization Framework for Early Conceptual Design of Helicopters," *Aerospace*, vol. 9, no. 10, 2022, ISSN: 2226-4310. DOI: 10.3390/aerospace9100598.

The author of this dissertation is the main author of all the appended publications and has been responsible for the formulation of ideas, conceptualization, modelling, writing and more. The co-authors have mainly had supervisory roles, although publications [II], [V] and [VI] have partly been produced by some of the co-authors.

Additional Publications

The publications listed below are not included in this dissertation. They are nevertheless relevant for the topic and constitute important parts of the work and its background.

[VIII] L. Franzén and E. Magnusson, "Weight Penalty Methods for Conceptual Aircraft Design," Master's Thesis Report LIU-IEI-TEK-A-18/03188-SE, 2018.

[IX] K. Amadori, C. Jouannet, L. Knöös Franzén, and E. Magnusson, "Weight Estimation for Conceptual Design: Refurbishing and Tweaking of Older Methods," in *Proceedings of the AIAA Scitech 2019 Forum*, San Diego, California: American Institute of Aeronautics and Astronautics, 2019, pp. 1–11. DOI: 10.2514/6.2019-0257.

[X] L. Knöös Franzén, "A System of Systems View in Aerospace Product Development," in *Book of abstracts for the 3rd ECATS Conference, Making Aviation Environmentally Sustainable*, vol. 1, Gothenburg, Sweden (Online): ECATS and the Swedish Aerospace Research Center (SARC), 2020, pp. 245–249, ISBN: 978-1-910029-58-9.

[XII] L. Knöös Franzén, "System of Systems: Trade Space Exploration: with Ontology and Description Logic Reasoning," in *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Online Streaming: IC3K, 2020, pp. 1–7.

[XIII] L. Knöös Franzén, *An Ontological and Reasoning Approach to System of Systems*. Linköping, Sweden: Linköping University Electronic Press, Licentiate Thesis No. 1907, 2021, ISBN: 978-91-7929-635-3.

[XIV] A. Oprea, R. Hällqvist, L. Knöös Franzén, *et al.*, "Connecting System Simulation to Aircraft Concept Development," in *Proceedings of the 32nd Congress of the International Council of the Aeronautical Sciences*, Virtual Event, 2021, pp. 1–17.

[XV] F. Larsson, L. Knöös Franzén, and R. Reichenwallner, "An Estimator for Aircraft Actuator Characteristics Using Singular Value Decomposition," in *Proceedings of the 6th Workshop on Innovative Engineering for Fluid Power*, São Paulo, Brazil, 2022, pp. 1–11.

[XVI] J. Lovaco, L. Knöös Franzén, and P. Krus, "Agent-Based Simulation and Ontology Integration for System-of-System Exploration," in *Proceedings of the Interdisciplinary Conference on Innovation, Design, Entrepreneurship, and Sustainable System*, São Paulo, Brazil, 2022, pp. 1–10.

This dissertation is based on the author's licentiate thesis, [XIII], and can be regarded as a continuation of it. Consequently, it reuses a significant part of the previously published work.

Contents

Appended Papers

1
Introduction

Systems are all around us and have long been a focal point in engineering and product development. However, the concept of System-of-Systems (SoS) is becoming more and more prominent in many engineering disciplines from both an industrial and an academic perspective [1]. Many systems are today increasingly connected and interacting with other systems and their operational environments in SoSs. This results in unique capabilities but also new levels of complexity that must be understood and managed during the development of new system solutions. An SoS context with interoperating systems also entails long-term evolving requirements. This, together with a constantly changing operational environment, further complicates the matter and introduces uncertainty that leads to higher levels of risk during development. This is especially true for products such as aerospace systems, which often have inherently long development times, spanning several years or even decades. Solutions with long development time must therefore be resilient to change or risk becoming obsolete before they are even produced due to changes in the outside world. The often long expected lifespans of aerospace systems also make them especially susceptible to changes throughout their operational life and in initially specified requirements. A holistic view of the development process is consequently needed to account for factors such as long-term evolving requirements and the many interdependencies that exist. This implies a shift of focus, where the delivery of capabilities should be ensured over time and throughout changing circumstances, instead of a more traditional fulfilment of fixed sets of requirements only. A holistic view thereby implies that more aspects than just single system solutions must be considered in the early phases of product development, which ultimately adds to the overall complexity and problem scale. Traditional approaches for early product development still apply, but the many interdependencies in an SoS create a need for new methods to handle the multiple aspects involved. Consequently, this dissertation aims to illustrate how an SoS perspective can be used to approach and facilitate early product development from a holistic point of view.

1.1 Background

The term "system" can be found in almost any area of research. A general definition of a system is that it is an arrangement of parts or elements that together produce results, behaviour or meaning not obtainable by the individual elements alone [2]. An aircraft is in this sense a system that consists of different elements or parts that together make the system airworthy. The elements of a system can themselves be systems and are typically called sub-systems. It might be tempting to say that a complex system, such as an aircraft, also is an SoS since it is composed of systems and sub-systems. However, an aircraft is not by definition an SoS. What is then the difference between a complex system and an SoS, and why distinguish them in the first place? The International Council on Systems Engineering (INCOSE) defines an SoS as an interoperating collection of Constituent Systems (CS) that usually produce results that the individual systems cannot achieve alone [3]. A CS can be part of more than one SoS and can also be composed of several sub-systems that in turn can include their own sub-systems and system elements, or parts. An illustration of a hierarchy for SoSs, CSs, sub-systems and system elements can be seen in Figure 1.1.

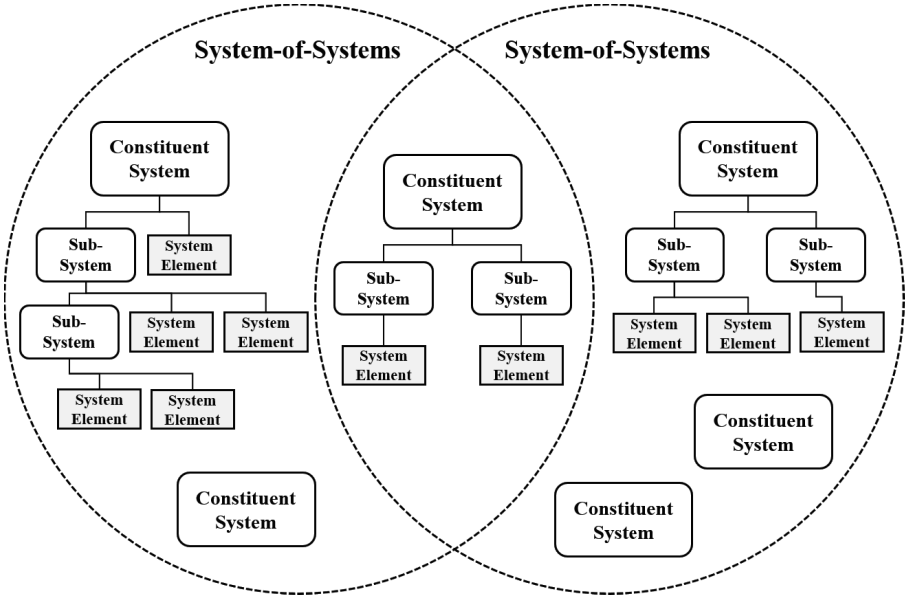


Figure 1.1 *An example of a hierarchy comprising System-of-Systems (SoS), Constituent Systems (CS), sub-systems and their corresponding system elements.*

The difference between a system and an SoS can be hard to discern from their quite similar definitions. However, an SoS can be distinguished from a

system based on five different characteristic properties introduced by Mark W. Maier [4]. These describe how a system can be better understood as an SoS if it has the following characteristic properties:

- **Operational independence**
- **Managerial independence**
- Geographical distribution
- Emergent behaviour
- Evolutionary development

The two independence properties marked in bold are the main distinguishing features of an SoS compared with a complex system under this definition. Operational independence specifies that each of the CSs in an SoS should be able to operate on their own and still be able to fulfil their individual purposes if disconnected from the other CSs. Similarly, managerial independence requires that the CSs can be developed by different manufacturers and that they can be managed and maintained by different organizations. Consequently, CSs may be evolving towards meeting their own needs and goals instead of the SoS's.

A geographical distribution of the CSs means that the distance between them is typically large and that the main form of interactions between them are in terms of information exchange rather than physical connections. Emergent behaviours can appear once CSs are interacting. Consequently, the emergent behaviour property describes the collective ability of the CSs to produce unique behaviours through collaboration. These can, for example, be capabilities that the individual constituents cannot achieve on their own, but can also correspond to unwanted behaviours that may arise through system interactions. Finally, evolutionary development implies that an SoS is under constant evolution and that its composition can vary. Consequently, new CSs can be added and old ones removed over time. The geographical distribution, emergent behaviour and evolutionary development properties are typically found in SoSs but may also be experienced by complex systems.

The definition just listed is used throughout this dissertation to distinguish a system from an SoS. Based on the definitions above, SoSs in the aerospace sector are, for example, entire air defence systems, the air transport system, aircraft carriers with assigned aircraft squadrons, drone swarms, and Search and Rescue (SAR) systems, to name a few. But, are not the SoSs just mentioned also systems? It might not come as a surprise, but the answer is yes! An SoS is a system and the name even says so. However, as Maier also explains in his article, different classifications can lead to different approaches in, for example, system development [4]. An SoS perspective might therefore highlight certain properties of interest that a singular system's view would not.

The development of systems and SoSs can be referred to as Systems Engineering (SE) and System-of-Systems Engineering (SoSE) respectively. The

field of SE can be defined as "*a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods*" [2]. SE consequently involves the engineering design of systems, but also the identification of stakeholder needs and the influence of the operational environments and other external factors that must be accounted for during development [5]. SoSE focuses on the integration of CSs to obtain capabilities that the singular systems cannot achieve on their own [6]. SoSE thereby facilitates the development of interoperable system solutions that can deliver capabilities over a long-term perspective. Having an SoS perspective and a capability-based focus is becoming a necessity in today's complex product development, especially in the aerospace sector.

1.1.1 Previous Research at Linköping University

Previous research related to this dissertation has proposed a holistic approach for product development in the context of SoSE and aerospace [1]. This approach assumes that a holistic design model for SoSs can be divided into five main levels of interest. The goal of the model is to create an early understanding of an SoS under development and how its design can be holistically explored. The five levels are illustrated in Figure 1.2.

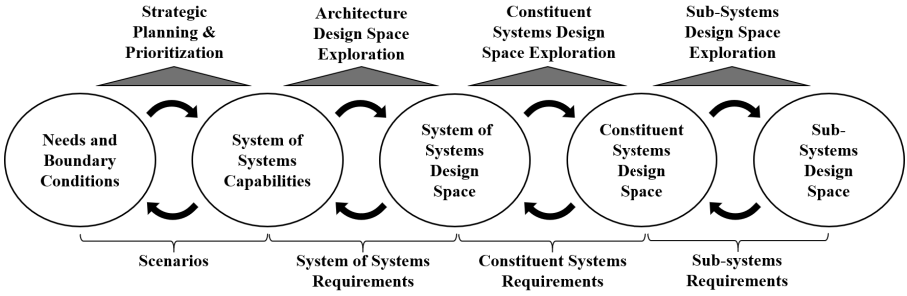


Figure 1.2 *An overview of the holistic design model for SoSs. Adapted from [1].*

Starting from the left, the *Needs and Boundary Conditions* level in Figure 1.2 focuses on analysing the overall needs of the SoS, which are typically those of the customers or other stakeholders involved. The boundary conditions of an SoS have an influence on the needs and can be aspects such as politics, economy, technology and more. Time frames are also an important aspect that may affect the needs, as boundary conditions are constantly changing. The next level describes the SoS capabilities required to meet the needs. These capabilities are determined by analysing different scenarios to understand the influence of changing boundary conditions. The capabilities most resilient to

changes can, for example, be identified through such analyses and by exploring the design space. The *System of Systems Design Space* level is used to investigate how the desired capabilities can be achieved. This is done by performing architecture design space explorations to find all valid SoS solutions. This consequently generates an SoS design space where each valid SoS solution is represented. Each SoS is composed of CSs that together achieve the capabilities. The *Constituent System Design Space* level focuses on the design of the individual systems of the SoS. Here, traditional product development processes can be used to design the systems based on derived requirements from the previous levels. Finally, the *Sub-System Design Space* level is used to explore how the sub-systems can be designed. The model just described is recurrent and has no specific starting point. This means that an analysis could start at the sub-system level and involve investigations into whether additional capabilities can be achieved by incorporating a new technology, for example.

The holistic SoS design model is a central part of this dissertation and the related method for approaching the problem outlined in the beginning of this chapter.

1.2 Aim and Research Questions

The overall purpose of this dissertation is to illustrate how early product development for systems can be approached from an SoS perspective. Accordingly, the dissertation aims to provide realizations of the holistic SoS design model in Figure 1.2. The work thereby illustrates how a produced method and related parts contribute at the different levels and how together they enable holistic SoS analyses at an early development stage. On a more specific level, this dissertation intends to answer the following research questions derived from the problem outlined in the previous sections:

- **RQ1:** How can needs from a system-of-systems be broken down into required capabilities and subsequently functions to be performed by constituent systems in a standardized and consistent way?

- **RQ2:** How can a design space for system-of-systems be represented in a flexible manner that also allows for traceable processing?

- **RQ3:** In what way can an overarching system-of-systems perspective generate requirements and be used in the exploration of new constituent system designs?

- **RQ4:** How can decision support be facilitated in the context of system-of-systems in early product development?

1.3 Delimitations

The highly combinatorial nature of SoSs makes the development of methods for approaching it difficult without introducing some delimitations. This dissertation mainly focuses on design processes from an SoS perspective and how an SoS design space representation can be created and processed. Consequently, other aspects of SoSs, such as different simulation techniques, are therefore not covered in detail. Additional delimitations considered in this dissertation are listed below:

- **No detailed evaluation of particular solutions:** The interaction between CSs can create emergent SoS capabilities. Different SoS solutions can therefore result in similar available capabilities and measures of effectiveness. This dissertation focuses on the methods and approaches that can be used to create an SoS solution. The performance of particular SoS solutions is therefore not measured or evaluated in detail.

- **No investigation of behaviour models or different simulation techniques:** Similar to the previous delimitation, no behaviour models that describe the interaction between systems are developed or further investigated in this dissertation. The possible Concept of Operations (CONOPS) for different SoSs is therefore not covered in detail either. Additionally, only Agent-Based Simulations (ABS) are illustrated as simulation techniques for SoSs in this dissertation.

- **Only Search and Rescue as application area:** There are many examples of SoS application areas that can be used as case studies to test produced methods. However, to keep the generated design spaces at a manageable level, only Search and Rescue (SAR) will be used as an application area.

- **Mainly aerospace systems:** As SoSs can typically involve CSs from different disciplines, this dissertation will mainly focus on the analysis and design of aerospace SAR system solutions.

- **Low fidelity levels:** Performed case studies are kept at a basic level with generally low fidelity to illustrate the utilization of produced methods and approaches.

- **No new definitions for SoSs:** The dissertation builds upon existing definitions of SoSs and will consequently not be used to provide new definitions for characterizing SoSs or differentiating them from complex systems.

1.4 Contribution

This work contributes a method for realizing parts of the holistic SoS design model illustrated in Figure 1.2. The dissertation thereby also provides examples of how an SoS perspective can be used in aerospace product development. The performed work is, however, kept at a general level so that the representation of any SoS is facilitated. Consequently, the presented method and related parts are applicable in other product development areas as well, and not just from an aerospace SoS perspective.

This dissertation also suggests how the presented method and corresponding parts can be used to process different design spaces in the early stages of product development. Additionally, the approaches described in this dissertation can be used in more traditional product development approaches for conceptual design to cover even more aspects. All the above-mentioned, together with a theoretical review of related areas and methods, contributes to the overall understanding and knowledge of SoS perspectives in engineering.

From a more specific point of view, this dissertation shows how ontology and description logic reasoning can be used in an SoSE perspective and to represent as well as process a design space. The work also shows how an architecture framework can support early product development and be used to obtain the functions to be performed by an SoS to meet overarching needs. Different approaches for concept exploration and estimations are thereafter introduced and utilized to provide solutions to the identified functions and requirements. Finally, Visual Analytics (VA) techniques are introduced as an overarching scope on all aspects of a holistic SoS design model.

1.5 Methodology

The scientific methodology of this dissertation is largely built upon hypothetico-deductive practices [7]. The formulation of hypotheses is done based on theories formed from literature studies. The hypotheses are then tested using experiments designed to challenge the assumptions. If the obtained results support the theories and prove the hypotheses right, they consequently form new methods and models to mimic reality. If proven to be wrong, new hypotheses can be formed based on the acquired knowledge from the performed experiments. Needless to say, the principles of the hypothetico-deductive process are highly iterative. This is illustrated in Figure 1.3.

On a more specific level, the formulation of the method and related parts within this work has been based on the outcome of a literature review. The theoretical background has here been investigated to find similar initiatives, existing methods, procedures and tools from areas such as SE, SoSE and more. As mentioned in section 1.1.1, the holistic SoS design model from [1] and its five levels of interest have been a central element of the approaches described in this dissertation. In that article, Staack et al. also suggest different key enablers

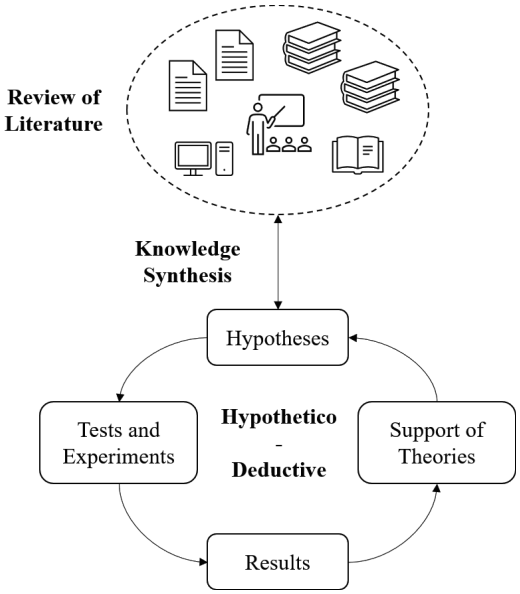


Figure 1.3 *An illustration of the iterative process of acquiring knowledge to form and test hypotheses.*

for holistic design engineering. These key enablers have partly been used to narrow down the initial scope of the performed literature study as well. The results from reviewing the theoretical background have subsequently been used to formulate hypotheses and theories that have been believed to answer the posed research questions and the aim of the dissertation. The hypotheses have thereafter been used to form a method based on four related parts, referred to as approaches, that have each contributed to the seven research papers on which this dissertation is built, namely papers [I]–[VII]. Each approach has been obtained by synthesising the knowledge from the literature review and utilizing the iterative process shown in Figure 1.3. The corresponding hypotheses have then been tested through the case studies of the appended papers in order to show their utility and to either update or confirm their hypotheses. Finally, the knowledge gained from each paper has been used to provide answers to the research questions of this dissertation. Figure 1.4 shows an illustration of the overall methodology and workflow that has been used to perform this work.

As Figure 1.4 shows, four approaches have been investigated and established from the work presented in the appended papers. Consequently, these are the parts from which the proposed method has been formulated. The method and appended papers have together resulted in this dissertation and have thereby been used to approach the posed aim and research questions.

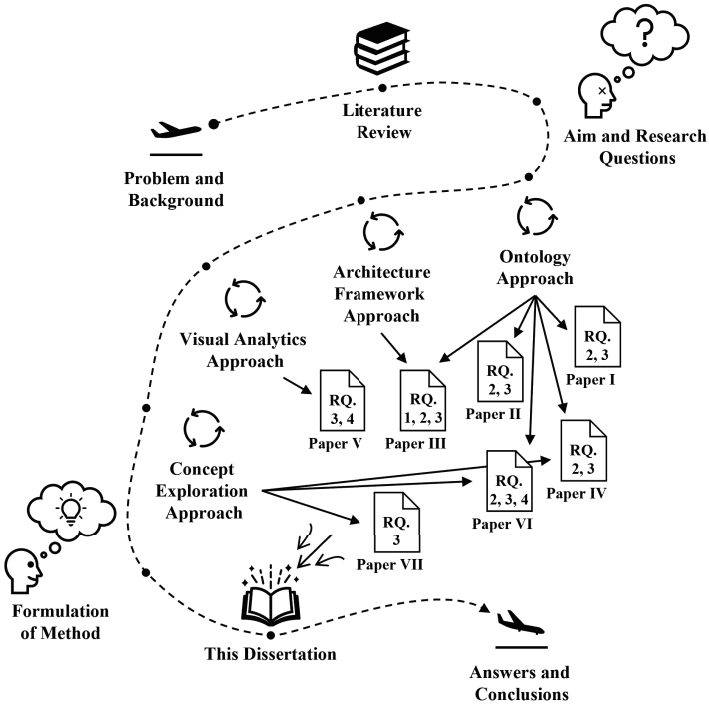


Figure 1.4 *An illustration of the methodology that has been used to traverse the different steps leading to this dissertation. The figure also shows where the different research questions (RQ) have been addressed.*

1.6 Dissertation Outline

This dissertation is structured in the following way: Chapter 2 provides a theoretical background and introduction to the different fields related to the presented research. Chapter 3 introduces the method and corresponding parts that have been used to provide answers to the research questions and the aim of the dissertation described in the introduction. The method, approaches and corresponding results are then discussed in chapter 4. The overall content of the dissertation is also discussed here. Chapter 5 concludes the dissertation by answering the research questions and highlighting the most important conclusions. A brief outlook on future possibilities is presented in chapter 6. Finally, chapter 7 provides a short summary of the appended papers.

2
Theoretical Background

The focus of this dissertation is on early product development from a holistic System-of-Systems (SoS) perspective, and there are many ways of approaching the problem outlined in the introduction. This chapter therefore provides both theory and background on the most closely related fields connected to this dissertation and their corresponding methods and approaches. Similar research initiatives are also highlighted in this chapter.

2.1 System of Systems

The introductory chapter briefly touched upon the definitions of systems and SoSs. SoSs are similar to singular systems in many ways, and there are even those who believe that it can be misleading to treat them differently [8]. Still, others suggest that it can be valuable to recognize them as different in order to facilitate understanding [4]. There are various ways of distinguishing a system from an SoS, such as the "ABCDE model" proposed by Boardman et al. [9]. A widely used definition is based on Maier's five characteristic properties which were mentioned previously in section 1.1. Maier's definition is used to distinguish a system from an SoS throughout this dissertation. Also, an SoS tends to have additional differences compared to a system, as explained in [10]. These are listed in Table 2.1.

Table 2.1 *The differences that systems tend to have compared with systems of systems (SoS). Adapted from [10].*

System	System of systems
A clear set of stakeholders	Multiple levels of stakeholders that may have mixed and competing interests
Objectives and purposes are clearly defined	Multiple objectives and purposes that can contradict each other
Explicit management structure and accountabilities	No clear accountability, and the management structures can be different
The operational priorities are clear, and priorities can also be resolved	Multiple operational priorities that can sometimes be different. No clear routes for resolving priorities
The systems' ownership is clear, and resources can be moved between elements	Can include multiple owners that make their own resourcing decisions
Single life cycle	Multiple asynchronous life cycles

SoSs can also be categorized into different types, depending on their degree of centralized control [11]. The different categorizations are:

- *Directed system of systems*
- *Acknowledged system of systems*
- *Collaborative system of systems*
- *Virtual system of systems*

The *directed* category involves SoSs that have a high degree of centralized control. A *directed* SoS is typically both built and managed with the goal of fulfilling specific purposes. The Constituent Systems (CS) still have an operational independence, but are subordinated to the overall purpose of the SoS. An integrated air defence network is as an example of a *directed* SoS that is centrally managed to defend a certain region against enemy systems [11].

An *acknowledged* SoS has a lesser degree of centralized control than a *directed* one. In this sense, the CSs have more freedom and retain their independent properties, such as ownership, objectives and development. However, they do have commonly recognized objectives for the SoS. The INCOSE Systems of Systems Primer gives an air traffic control system as an example of an acknowledged SoS where the individual CSs have common goals and follow overall regulations and protocols [10].

Collaborative SoSs have CSs that voluntarily choose to participate to meet the overall purposes of the SoS. Unlike *directed* and *acknowledged* SoSs, a *collaborative* SoS has no centralized control or overall directing authority. *Collaborative* SoSs do however have commonly agreed central purposes and follow standards, regulations and working practices. An example of a *collaborative* SoS, also given by [10], is electrical grids where CSs together produce electricity and distribute it to customers.

Finally, a *virtual* SoS type is typically an SoS with no centralized control, authority or even a commonly recognized purpose. They often experience emergent behaviours on a large scale due to the lack of management and the consequential self-organization of the CSs. The internet is an example of an SoS that would be classified as a *virtual* type [10].

Furthermore, an SoS does not necessarily have to be of a specific type throughout its entire life cycle [10]. This depends on its current operating modes and the way it is viewed. However, issues from misclassification between the different categories just listed can also arise. For example, a collaborative SoS misclassified as a directed one could give a false sense of control over the SoS and its operating CSs. Another example is that misclassification of a virtual SoS can lead, for example, to unexpected emergent behaviours [4].

While research related to SoS can be found all over the world, the field is dominated by initiatives from the United States [12]. Actors such as Massachusetts Institute of Technology (MIT) [13], MITRE corporation [14], [15], Georgia Institute of Technology's Aerospace Design Laboratory (ASDL) [16]–[18], Purdue University [19], [20], the Systems Engineering Research Center (SERC) [21]–[23] and the U.S. Department of Defense (DoD) [11], [24] have contributed to research within the field.

2.2 System of Systems Engineering

An SoS is rarely developed from scratch [25]. As previously mentioned, the life cycles of the CSs are often asynchronous, and SoSs are consequently rather formed over time as systems interoperate and collaborate to achieve new capabilities. System-of-Systems Engineering (SoSE) can be seen as Systems Engineering (SE) at an SoS level. SoSE involves the planning, analysis, organization and integration of CSs to obtain capabilities not attainable by the individual systems alone [24], [26]. Compared with SE, SoSE involves a set of similar but different processes [24]. SE can also be distinguished from SoSE based on various differences displayed in Table 2.2.

Table 2.2 *A comparison between different attributes of SE and SoSE. Adapted from [27].*

Property	Systems Engineering	System of Systems Engineering
Scope	A single complex system	Multiple complex systems that are integrated
Objective	Optimization against fixed sets of requirements	Satisfying and sustaining capabilities to meet stakeholder needs over time
Boundaries	Static and persistent throughout the system's life cycle	Dynamic and changing over time
Problem	Defined	Emergent
Structure	Hierarchy	Network
Goals	Unitary	Pluralistic and aggregated
Approach	Process	Methodology
Timeframe	Single system life cycle	Continuous with multiple constituent system life cycles

SoSE is a relatively young and immature area compared with SE [12]. Consequently, fewer approaches and standard practices are available. Of course, SE approaches apply to some degree, but the higher level of abstraction needed for SoSs calls for more holistic courses of action. Capability engineering is, for instance, a field which is similar to SoSE in many ways. It focuses on the identification, evaluation and integration of capabilities to ensure that they are properly designed and sustained from an interoperability perspective [28]. Capability engineering thereby extends SE principles to an SoS perspective. It is explained in [29] that capability engineering starts with the assessment of desired capabilities, followed by the identification of resources and viable options for achieving them. The options for achieving desired capabilities can then be assessed and selected. Another related field and part of SoSE is mission engineering which focuses on the operational mission context of a system or SoS [30]. Mission engineering is a top-down approach which facilitates the planning and identification of capabilities to achieve a desired mission effect. It can, among other things, thereby be used to identify and address mission capability gaps. Such identified gaps may, for example, be used to generate requirements for new SoS architectures and CSs to be acquired or designed.

An obstacle in SoSE is that it is difficult, and sometimes near impossible, to ensure that capabilities can be met through prototype testing [10]. The reason for this is that the complexity and overall large scale, with multiple CSs, introduce both economic and physical barriers [15]. Consequently, modelling and simulation for complex systems and SoSs becomes a valuable alternative in early design.

2.2.1 Model Based Systems Engineering

Model Based Systems Engineering (MBSE) is, as the name implies, systems engineering with a special focus on modelling aspects [3]. MBSE is used to increase the understanding of systems and their architectures, both during the development of new systems and in the deployment of existing ones [5]. MBSE has been used to model SoSs and to support the translations between capabilities and requirements, as shown in [31] and [32] for example. Different modelling languages are found within MBSE that describe systems, their functionality, architecture and other properties. Two of these languages are the Unified Modelling Language (UML) and the Systems Modelling Language (SysML), both of which have been used to model and describe important aspects of SoSs [29], [31], [33]. These aspects include factors such as the architecture and information exchange between CSs, as well as their resulting emergent behaviours in the overall SoS.

Understanding and predicting emergent behaviours is an important aspect in the development of complex systems and SoSs. The emergent behaviours of an SoS can be both wanted and unwanted [34]. The authors of [34] give bee colony behaviour and their collective harvesting capability as an example of wanted emergent behaviour. Similarly, market crashes are given as an example of unwanted behaviours in [34]. Early development includes the identification of possible emergent behaviours so that they can be either facilitated or eliminated. Unique capabilities achieved through interoperation and collaboration between CSs are examples of wanted emergent behaviours in SoSs where the sum is greater than the parts. MBSE is a means by which to investigate these different behaviours early in the development process, and to facilitate the emergence of wanted behaviours and capabilities. In a similar way, unwanted behaviours can be avoided by analysing SoS architectures and the interoperation between CSs with MBSE.

2.2.2 Architecture Frameworks

The architecture of systems and SoSs is an important consideration in MBSE, as previously mentioned. The *Systems Engineering Guide for Systems of Systems* describes how the architecture of an SoS defines the way the CSs work together to meet stakeholder needs [24]. It also explains that an SoS architecture inevitably includes the Concept of Operations (CONOPS), which describes how CSs can be employed. SoS architectures also contain descriptions of the CSs and their functions, as well as their external and internal dependencies and relationships. Moreover, the communication and data flow, as well as the end-to-end functionality of the SoS, can also be described through an architecture. The book *"System of Systems Modeling and Analysis"* ([35]) explains that the different architectural aspects of an SoS can be divided into three classes of design variables at an early SoS modelling phase, namely:

- *Composition*
- *Configuration*
- *Control*

Here, *Composition* describes aspects such as which CS and resources are to be included and what functions should be fulfilled. *Configuration* aims to describe the interconnections between constituents and how these evolve over time. Finally, *Control* describes the degree of control or influence that each CS has in the SoS. Both *configuration* and *control* include aspects of a dynamic nature compared to the more static aspects of *composition* variables.

SoS architectures can also be represented through something known as architecture frameworks. These have been developed for the purpose of describing the architectures of complex systems and SoSs. Architecture frameworks are also designed to capture the viewpoints of different stakeholders connected to an SoS. These different viewpoints may, for example, be operational, service, strategic, security and overall system aspects [14]. Architecture frameworks that have been used to model SoSs in an MBSE-focused approach include the US Department of Defense Architecture Framework (DoDAF), the UK Ministry of Defence Architecture Framework (MODAF) and the NATO Architecture Framework (NAF) [32]. There are also other examples of architecture frameworks in SE and SoSE, such as the Federal Enterprise Architecture Framework (FEAF), The Open Group Architectural Framework (TOGAF) and the Zachman Framework [14]. The Unified Profile for DoDAF and MODAF (UPDM) is another framework that supports the development of system architectures. As the name implies, it unifies DoDAF and MODAF to provide a common model for them. UPDM has been further developed into what is known as the Unified Architecture Framework (UAF) [36].

The Unified Architecture Framework

The UAF is designed to consistently model SoS architectures using an MBSE approach [37]. It is closely connected with UML and SysML, and thereby allows for improved interoperability with other related tools and standards. The UAF supports analyses of system specifications, designs and verifications from different stakeholder views. Consequently, it allows stakeholders to analyse specific areas of interest in a holistic way that facilitates the fulfilment of desired capabilities in the SoS [36]. Like DoDAF and MODAF, the UAF introduces a number of different viewpoints that are designed to capture different stakeholders' areas of interest. These are presented in an overarching domain meta-model that describes the definition of the viewpoints and their corresponding concepts and relationships [38]. The UAF grid, or matrix, shows how the viewpoints correspond to different domains and model kinds within the UAF. An illustration of the UAF matrix can be seen in Figure 2.1.

Domains / Models	Taxonomy Tx	Structure Sr	Connectivity Cn	Processes Pr	States St	Interaction Scenarios Is	Information If	Parameters Pm	Constraints Ct	Roadmap Rm	Traceability Tr
Metadata Md	Metadata Taxonomy Md-Tx	Metadata Structure Md-Sr	Metadata Connectivity Md-Cn	Metadata Processes Md-Pr	Metadata States Md-St	-			Metadata Constraints Md-Ct	Metadata Roadmap Md-Rm	Metadata Traceability Md-Tr
Strategic St	Strategic Taxonomy St-Tx	Strategic Structure St-Sr	Strategic Connectivity St-Cn	-	Strategic States St-St	-			Strategic Constraints St-Ct	Strategic Deployment St-Rm Strategic Phasng St-Rm	Strategic Traceability St-Tr
Operational Op	Operational Taxonomy Op-Tx	Operational Structure Op-Sr	Operational Connectivity Op-Cn	Operational Processes Op-Pr	Operational States Op-St	Operational Interaction Scenarios Op-Is			Operational Constraints Op-Ct	-	Operational Traceability Op-Tr
Services Sv	Service Taxonomy Sv-Tx	Service Structure Sv-Sr	Service Connectivity Sv-Cn	Service Processes Sv-Pr	Service States Sv-St	Service Interaction Scenarios Sv-Is	Conceptual Data Model,	Environment Pm-En	Service Constraints Sv-Ct	Service Roadmap Sv-Rm	Service Traceability Sv-Tr
Personnel Pr	Personnel Taxonomy Pr-Tx	Personnel Structure Pr-Sr	Personnel Connectivity Pr-Cn	Personnel Processes Pr-Pr	Personnel States Pr-St	Personnel Interaction Scenarios Pr-Is	Logic Data Model,		Competence, Drivers, Performance Pr-Ct	Personnel Availability, Personnel Evolution, Personnel Forecast Pr-Rm	Personnel Traceability Pr-Tr
Resources Rs	Resource Taxonomy Rs-Tx	Resource Structure Rs-Sr	Resource Connectivity Rs-Cn	Resource Processes Rs-Pr	Resource States Rs-St	Resource Interaction Scenarios Rs-Is	Physical schema, real world results	Measurements Pm-Me	Resource Constraints Rs-Ct	Resource evolution, Resource forecast Rs-Rm	Resource Traceability Rs-Tr
Security Sc	Security Taxonomy Sc-Tx	Security Structure Sc-Sr	Security Connectivity Sc-Cn	Security Processes Sc-Pr	-	-			Security Constraints Sc-Ct	-	Security Traceability Sc-Tr
Projects Pj	Project Taxonomy Pj-Tx	Project Structure Pj-Sr	Project Connectivity Pj-Cn	Project Processes Pj-Pr	-	-			-	Project Roadmap Pj-Rm	Project Traceability Pj-Tr
Standards Sd	Standards Taxonomy Sd-Tx	Standards Structure Sd-Sr	-	-	-	-			-	Standards Roadmap Sd-Rm	Standards Traceability Sd-Tr
Actual Resources Ar		Actual Resource Structure Ar-Sr	Actual Resources Connectivity Ar-Cn	Simulation					Parametric Execution/ Evaluation	-	-
Dictionary Dc											
Summary & Overview Sm-Ov											
Requirements Req											

Figure 2.1 *The Unified Architecture Framework (UAF) matrix describing how the different viewpoints correspond to domains and model kinds. Adapted from [38].*

The different viewpoints are each associated with models designed to describe different areas of interest in a UML or SysML manner. The *Strategic Taxonomy (St-Tx)* in Figure 2.1 can, for example, be used to list all the capabilities of the SoS in question [39]. Besides being used for modelling SoSs from different stakeholder viewpoints, the UAF can also be used to perform trade studies for SoS architecture development, as explained in [40]. One difference with the UAF compared with DoDAF and MODAF is that the UAF is designed to capture system architecture descriptions not just in the defence sector, but in commercial sectors as well.

2.2.3 System of Systems Modelling and Simulation

MBSE and architecture frameworks are one of many ways to model SoSs. As mentioned earlier, a desired outcome when modelling SoSs is to understand possible emergent behaviours. According to [9], emergent behaviours are foreseen

and appropriately tested during development of singular systems. However, these are harder to foresee in SoSs, as they must be rich in emergent behaviour to achieve a broad range of capabilities. In SoSs, all emergent behaviour cannot deliberately be designed in and there is always a risk of unintended consequences that are not discovered during testing [9].

The INCOSE Systems Engineering Handbook elaborates on the understanding of emergence by referring to complicated versus complex systems [3]. A complicated system can be understood by breaking it down into parts and then reassembling the parts to get an overall understanding of the whole. Consequently, the fixed relationships in a complicated system allow for reasonably reliable predictions of the system's characteristics and emergence. However, this is not as straightforward in complex systems. Here, the interconnections between the parts give rise to emergent properties that may disappear if the system is broken down and investigated as individually isolated parts. Simulation approaches can, however, be used to facilitate the understanding of emergent behaviours in complex systems and SoSs. System dynamics is one example that can be used to investigate and increase the understanding of systems through simulations [41]. Another technique, that also has been the main simulation approach used in this dissertation, is Agent-Based Simulations (ABS).

Agent-Based Simulations

Agent-Based Simulations (ABS) can be used to simulate and analyse SoSs, and thereby enhance the understanding of SoS dynamics and performances in the early stages of system design [42]. ABS builds upon the interaction between agents that each have their own set of individual behaviours and rules of interaction with other agents. An agent may, for example, correspond to a known CS, such as an aircraft, in an SoS. An agent might also represent a new CS to be developed. ABS can consequently be used to evaluate SoSs, their capabilities and involved CS in different operational scenarios and circumstances. This can thereby give initial performance requirements for new CSs to be developed from an SoS perspective. Metrics for evaluating the performances and efficacy of different SoSs must, however, be established prior to that. The *Mission Engineering Guide* ([30]) describes how the many kinds of metrics or measures of a mission can be divided into two broad categories; namely Measures of Effectiveness (MOE) and Measures of Performance (MOP). MOE can be introduced as a means for assessing how well an SoS meets, for example, desired capabilities through simulations. MOE can be defined as *"The metrics by which an acquirer will measure satisfaction with products produced by the technical effort"* [43], or as a measurable attribute of success within an overall mission [30]. The emphasized focus on capabilities in SoS analyses makes MOE an important measure for determining SoS performances, for example. MOP, on the other hand, is used to indicate the performance of individual CSs used in the SoS and overall operation or mission. An MOP may in this sense be the range or manoeuvrability of a specific CS [30].

ABS is typically used as a "bottom-up" modelling approach, meaning that modelling on a system's scale gives rise to collective and emergent phenomena through interactions between the modelled individuals, entities or systems. Common application areas for ABS include studies of biological systems and studies for understanding emerging behaviours in social sciences [44], [45]. There are several studies that have used ABS to investigate SoSs. One such study is presented in [18], which shows how ABS can be used to investigate the interoperability levels between systems and how it impacts the overall performance of an SoS. The results of this are thereafter used to perform trade studies to examine network and vehicle performances. A study presented in [46] utilizes ABS to evaluate how changes in system design parameters affect an SoS. In this case, the SoS is a group of coordinated unmanned aerial vehicles that together perform a maritime SAR mission. A similar ABS study, presented in [47], illustrates how different factors, such as the operational environment, affect the performance of unmanned aerial vehicles in an SoS context. The study in [47] also utilizes an optimization framework to efficiently explore the design space for the development of new unmanned aerial vehicles. Additionally, the book *"Modeling and Simulation Support for System of Systems Engineering Applications"* provides considerable details on how ABS can be used to model and simulate SoSs, as well as illustrating several application examples from multiple domains [42].

Model Fidelity

Model fidelity becomes an important topic to consider when modelling and simulating SoSs. Since, SoSs typically involve collaborating CSs that are complex in their own right, some modelling aspects must be simplified to allow for inexpensive simulations from a computational point of view [48], [49]. The purpose of the analysis also dictates the resolution of the models to be used. The modelling and simulation fidelity should therefore be of a degree that sufficiently captures relevant aspects of the problem at hand [50]. Model fidelity can be defined as the degree to which a model is true to what it represents [51]. Consequently, model fidelity does not necessarily dictate the level of detail, but rather the value that a model brings in its context, or simply the model's accuracy. Models can be categorized into different types depending on the fidelity level. For example, [49] uses the terms Low-Fidelity Model (LFM) and High-Fidelity Model (HFM). The paper explains that HFMs are typically accurate but at the expense of computational time, while LFMs are computationally cheaper but with less accuracy. A so-called Multi-Fidelity Model (MFM) is a combination between HFMs and LFMs. Naturally, this kind of model leverages the computational efficiency of LFMs while relying on HFMs to establish a desired level of accuracy [49].

From an SoS perspective, an SoS model can, for example, be composed of several different domain-specific models that each represent entities such as involved vehicles, the environment or behaviours at different fidelity levels.

MOEs can, in such a case, be used to obtain a perception of how well and accurate the overall SoS model fulfils its purpose. Sensitivity analyses may thereafter be performed to establish yet another perception of how influential the different domain-specific models are on the collective results of the SoS model. High sensitivity to the results from a specific model would consequently call for a higher fidelity model. Similarly, computational effort could be saved by relying on LFMs for models with little or no sensitivity to the outcome of the overall SoS model. However, low sensitivity may in some cases also indicate a need for higher fidelity to capture relevant relationships.

Model Integration and Co-Simulation

As established earlier, SoS models can be a composition of many models. It is therefore relevant to address how such constituent models can be combined and simulated together – especially when models can be developed in different tools or, for example, by different organizations with dissimilar modelling techniques. One way to approach model integration and interoperability is to rely on established standards, such as the Functional Mock-up Interface (FMI) [52] and System Structure and Parameterization (SSP) standards [53]. The goal of integrating different domain-specific models under a common standard is typically to enable the concept of co-simulation. Essentially, co-simulation implies that different models are capable of communicating and exchanging information with each other during run time. The degree of transparency regarding the models and their inner workings may vary. This might, for example, depend on whether or not the models include sensitive information. The models in a co-simulation can consequently be run in a "black box" mode, which means that a model simply converts one or more inputs into a number of outputs. The inner workings of the models are therefore not necessarily exposed in an explicit way which could, for example, be beneficial when working with proprietary models from different organizations and actors [54]. In an SoS context, model integration and co-simulation may be used with models representing, for example, different systems to ultimately simulate the workings of an SoS. The fidelity of the various models can also be explored this way, given the modularity of models under a co-simulation standard [55].

This section of the dissertation has so far dealt with MBSE and different modelling and simulation techniques for SoSs. Another approach that has been used to model systems and SoSs, in a similar way to MBSE, involves using ontologies. This is further explained in section 2.3.2.

2.3 Ontology

Ontology is a formal and explicit representation of a given domain that involves knowledge of the included entities and the relationships between them [56]. A more formal definition is that an ontology is an "explicit specification of a conceptualization" [57]. Ontologies are, in this sense, a way of representing domain knowledge and managing it through a common understanding of the content [56]. Simply put, ontologies are used to represent entities and their relationships. This is typically done using something known as semantic triples [58]. A triple consists of three parts: a subject, a predicate, and an object. This is illustrated in Figure 2.2, which also presents an example of a triple relating two entities together.

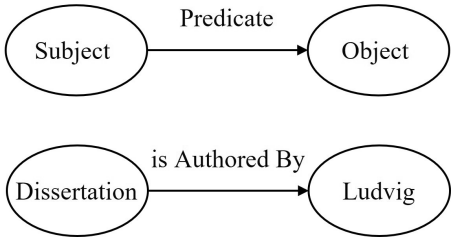


Figure 2.2 *The elements of a semantic triple that forms the foundation of an ontology.*

As shown in Figure 2.2, the *subject* entity in a triple points to the *object* entity using a *predicate* relationship. The example triple describing that this *dissertation is authored by Ludvig* could just as well have been defined in the inverse direction. Such a triple could in that case manifest as *Ludvig is author of Dissertation*, where *Ludvig* would be the subject pointing to the *Dissertation* object through an *is author of* predicate. This way of describing entities and their relationships can be interconnected and used to create chains of triples pointing to each other. This can thereby lead to large networks of entities and relationships in an ontology or knowledge base. Figure 2.3 depicts an ontology that describes different entities and the relationships that exist between them.

The creation of ontology models can be referred to as ontological engineering, and this includes descriptions of the languages, methods and principles available for this purpose. An ontology can be implemented in different ontology languages, the most common of which are the Resource Description Framework (RDF) and the Web Ontology Language (OWL) [58]. OWL is based on RDF, but has the advantage of being better equipped for description logics and constraints checking [59]. An ontology produced in OWL consists of individuals, classes, and their properties and relationships, which together are used to describe concepts of the domain in question with triples. Classes can be seen as collections of individuals. For example, the small ontology in Figure 2.3 has a

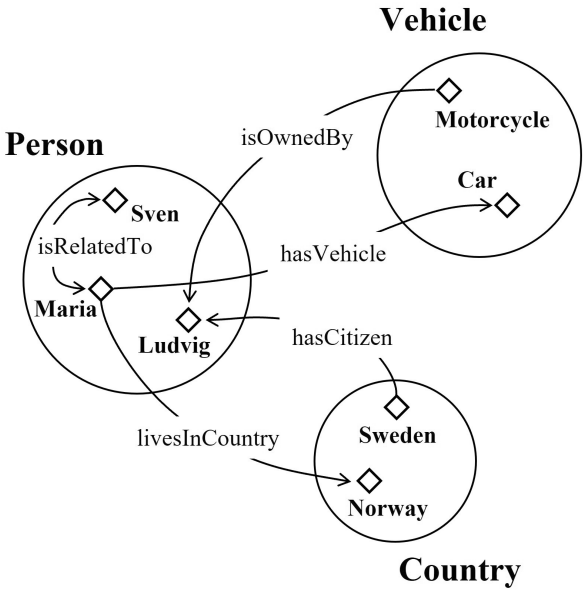


Figure 2.3 *An illustration of an ontology that includes entities and their corresponding relationships.*

class called *Vehicle* that includes two individuals, *Motorcycle* and *Car*, which in turn are related to other instances of different classes within the ontology.

OWL-implemented ontologies also feature the *open world assumption* and the *non-unique naming assumption*. The *open world assumption* implies that new information about the domain can appear at any time, and no excluding conclusions can therefore be drawn; more information that is simply not yet known could always come to light [58]. A *closed world assumption* regards non-existing data as false, while an *open world assumption* regards it as simply unknown. Consequently, no assumptions are made about incomplete data in an OWL ontology. The *non-unique naming assumption* simply implies that different entities within the ontology may be referred to using different names by people and organizations for example. The *open world assumption* and the *non-unique naming assumption* give OWL ontologies an advantage in terms of interoperability and scalability compared to relational databases, for example.

The Web Ontology Language - Description Logics (OWL-DL) is a subset of OWL and features description logic reasoning capabilities. Description logic reasoning, hereinafter referred to simply as reasoning, can check an implemented ontology model for inconsistencies. However, reasoning can also be used to infer complex relationships from simpler ones [60]. Consequently, reasoning can be used to expand the captured knowledge by inferring new triples,

thereby contributing to the scalability of ontology models. Reasoning in OWL is based on the *open world assumption*, and there are different types of reasoner algorithms that each support different features [60], [61]. Automated reasoning over large ontologies does, however, require large computational resources and involves a cost in terms of computational time, as shown in [62] and [63]. Computational resources are consequently a limiting factor when it comes to scalability through reasoning. However, there are various optimization techniques that can contribute to the efficiency of reasoning, as explained in [63]. Utilizing heuristics in the ontology can also contribute to the efficiency of reasoning in domains with a large number of axioms [64].

2.3.1 Top-Level and Meta Ontologies

The interoperability and scalability of an ontology can be further increased by utilizing a domain-neutral top-level ontology structure [65]. A top-level ontology acts as a framework for organizing domain-specific ontologies. It also supports the creation of new ontologies, as well as the re-use of existing ones. Top-level ontologies can also be referred to as upper-level ontologies, and there are many examples of proposed structures for this purpose. Commonly used top-level ontologies include the Basic Formal Ontology (BFO), the Business Objects Reference Ontology (BORO) and the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [66], [67]. A comparison between different top-level ontologies is presented in [67]. Consequently, ontologies can be used to describe other ontologies. This can be seen as a meta ontology structure, where more abstract and higher-level ontologies are used to describe lower-level and more specific domain ontologies. A mid-level ontology fits in between top-level and domain ontologies. Such a mid-level ontology might describe a set of domain ontologies and their subsequent sub-ontologies. An example of this is a mid-level ontology describing vehicles. The corresponding more specific domain ontologies can then describe airborne vehicle systems, which in turn have sub-ontologies describing aircraft control surface actuation systems, for instance. Figure 2.4 shows an illustration of what a hierarchy within a meta ontology structure could look like for the example just described.

A domain-specific ontology can, in this sense, describe certain areas of interest, such as individual systems, requirements or parts. Domain ontologies are found in many areas of research. An example of an ontology for aircraft design is presented in [68]. Another domain example is presented in [69], which describes an ontology for information systems interoperability.

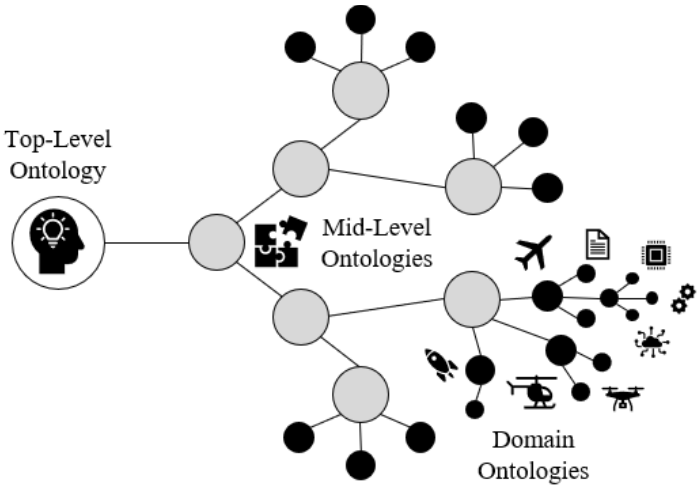


Figure 2.4 *An illustration of the hierarchy between top-level, mid-level and domain ontologies in a meta ontology structure.*

2.3.2 Ontology Based Systems Engineering

Ontologies can be seen as alternatives to modelling languages such as UML and SysML within MBSE. Ontology is used in a similar way to model and describe entities and their relationships in a domain. Ontologies do, however, feature increased interoperability and scalability compared to the languages mentioned previously. This comes from the ontologies' ability to describe a domain from different terminologies and viewpoints [70], [71]. Reasoning is also an advantage with ontologies, since it allows for automatic consistency checking and classification of relationships in the model. Ontology models have been used to represent relevant domain knowledge in SE, SoSE and capability engineering [72]–[74]. Ontologies have also been used to enable reasoning over SysML-represented content, to improve flexibility and to organize different domains in order to enhance the exchange of information [75], [76]. SysML requirement diagrams can also be transformed into an OWL file to enhance the understanding of the semantic context, as proven by a case study in [77]. Consequently, the study in [77] also shows that UML and SysML models can be used to automatically generate an ontology. Using ontology in SE may be referred to as Ontology-Based Systems Engineering (OBSE). There are many contributions that an ontology can bring to SE in addition to the benefits mentioned above. Yang et al. present a summary of these together with a detailed state-of-the-art review of OBSE [78]. However, ontology also has its limitations. One of the drawbacks previously discussed is computational time when using automatic reasoning. The number of implicit relationships within an ontology model can affect and increase the computational effort, as a reasoner must rebuild the

original ontology structure. More details about how implicit information can increase the computational time for reasoning in OWL are explained and discussed in [79], where reasoning and inferred relationships are used to a high degree to build up an ontology. Another disadvantage of ontologies and reasoning concerns numerical calculation and optimization capabilities. While OWL ontologies can represent and reason over data and numerical values, mathematical operations are limited. One way of enabling some basic and limited calculation capabilities is through different rule languages [80].

The Semantic Web Rule and Query Languages

A common rule language for semantic web technologies is the Semantic Web Rule Language (SWRL) [81]. SWRL was developed to enable certain inference capabilities that cannot be made through regular description logics. SWRL semantics are based on OWL-DL and thereby complement the language with more expressiveness [82]. SWRL is applied on top of an ontology model. Certain SWRL-compatible reasoner algorithms can thereafter be used to classify the ontology while incorporating any included rule in the inference process. Added rules with SWRL thereby enable additional functionality, for example reasoning over mathematical expressions [82]. SWRL cannot be used to query an ontology based on rules, as it is solely a language for rules and not queries. Queries can, for example, be used to ask an ontology questions on its represented knowledge about the domain in question. Consequently, queries enable filtering and retrieval of information from the ontology model. This is typically done using a query language called SPARQL Protocol And RDF QUERY Language (SPARQL) [58]. SPARQL is mainly oriented towards RDF and thereby has its limits when it comes to queries on OWL ontologies. The combined rule and query language Semantic Query-Enhanced Web Rule Language (SQWRL) was consequently introduced as a query language for OWL ontologies [83]. SQWRL enables, for example, debugging of SWRL rules through the added querying capabilities [81].

2.4 Product and Engineering System Development

As mentioned in the introduction, Systems Engineering (SE) can be used to create systems and to develop new products. Many SE approaches are applicable in early product development from an SoS perspective. Staack et al. suggest that holistic analyses of an SoS can give the functional requirements that must be fulfilled by the CSs in order to meet the overarching capabilities [1]. These requirements can then be used as input for a continued product development process, where early system solutions are proposed through conceptual design studies. The many alternatives, which are able to fulfil overarching SoS capabilities and CS requirements, for example, can be represented as elements in different design spaces.

2.4.1 Design and Trade Space Explorations

A design space contains possible solutions to a given design problem. The size of an available design space can be reduced by excluding areas where desired or specified requirements cannot be fulfilled. The feasible solutions can consequently be found by exploring the available design space of alternatives [16]. Design space explorations are also used to increase the information and understanding about a new system in early development, thus reducing the overall uncertainty [84]. This can, for example, be done by investigating a system's sensitivity to changes in initial requirements or relevant design parameters/-variables. The overall goal of a design space exploration is to find the best valid solution that fulfils all requirements and design objectives in the best way possible [85].

The sheer size of a design space can be a challenge in explorations [86]. Design space reductions are therefore an important aspect. This is especially true for SoSs that typically have large available design spaces due to their highly combinatorial nature. Areas where non-valid solutions are found can quickly be ruled out by screening and excluding parts of the design space. Figure 2.5 illustrates a reduced design space representation.

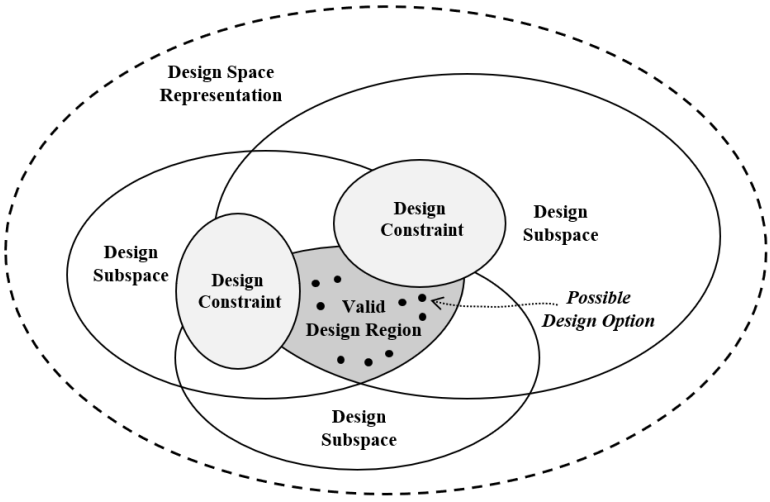


Figure 2.5 *A design space representation that has been reduced by different valid subspaces and design constraints. Each point in the valid design region represents a valid design option.*

As seen in Figure 2.5, a design space can be reduced by excluding areas that are unfeasible from different perspectives. This creates design subspaces of valid solutions that together intersect into a smaller area of overall suitable designs. This space can subsequently be further reduced by inserting design constraints, such as requirements or desired performances. The valid region

then contains all the possible design options that are feasible and fulfil the given requirements. In contrast to reductions, design space expansions can be performed to add more possible options to the design and thereby open up new useful areas of the design space [87]. Parallels to the *open world assumption* in OWL can be drawn here, as a design space may contain more relevant information that simply is not known yet. The size of a design space can consequently be infinite, and the available design space of alternatives is better referred to as a design space representation, as new alternatives may appear through technological advances, for example. The non-stationary boundaries of the design space representation make product development complex from an SoS perspective, due to the evolutionary development and emergent behaviour characteristics.

Trade studies can be used to investigate how well the different elements in a design space meet the intended system objectives and requirements [88]. Such trade studies may be done by investigating how changes in design variables and system requirements affect the overall design space and available solutions. The trade studies that can be performed span an available trade space. A trade space is a set of possible design options with enumerated design variables [89]. A trade space is therefore a result view originating from, for example, modelling and simulation of different design options that may be explored. These trade space explorations give designers a perception of the available design space, and allow for facilitated and more elaborate decision-making. They also allow stakeholders to see the trade-off between system characteristics in terms of factors such as cost and performance. Trade Space Explorations (TSE) and design space explorations may seem similar, but they are not the same. While design space explorations focus on how different available design variables affect the characteristics of a system, TSE illustrates the trade-offs between the different design solutions in terms of their characteristics. TSE can thereby improve the understanding of how different design choices affect the overall solution, and how its expected delivered value changes accordingly. Consequently, a trade space is a way of representing critical decision parameters which can later be explored with the purpose of identifying the most robust solutions.

A "Tradespace Exploration design paradigm" is mentioned and described in [89]. This paradigm essentially implies that many design alternatives are investigated simultaneously and that no singular point solution is of particular interest. The focus is instead shifted towards observing trends in the resulting trade space, which ultimately results in facilitated insight and knowledge about the problem at hand. In short, the TSE paradigm aims to enable confidence in decision-making [90]. What [89] and [90] highlight about the TSE paradigm is very similar to the concept of set-based design. Set-based design is a practice where single point solutions are not chosen until sufficient knowledge about a design space has been gained [91]. The design options and requirements are here kept flexible throughout the development process. Consequently, several

possible design options are investigated simultaneously without committing to a specific solution. This enhances the flexibility of the design options and allows them to adapt over time.

2.4.2 Concept Generation Approaches

A design space of alternatives for concept generation can be established in different ways. SE has many approaches for system development that are applicable in an SoSE context. For example, a Quality Function Deployment (QFD) is an approach used to prioritize requirements and provide the relationships between technical requirements and stakeholder needs [16]. These derived requirements can then be broken down into a set of functional requirements describing what the system under development must be able to do. Consequently, the required functionality of the system can be used to choose suitable system elements that together meet the stated requirements.

Function/Means Tree

A Function/Means tree (F/M tree) is a method that can be used to perform functional breakdowns and to illustrate the design alternatives in concept generations [92], [93]. An F/M tree describes a hierarchy of functions and the corresponding means that implement them. A means can also be referred to as an alternative, which might be a system solution that implements one or more specific functions. Figure 2.6 shows an example of an F/M tree structure.

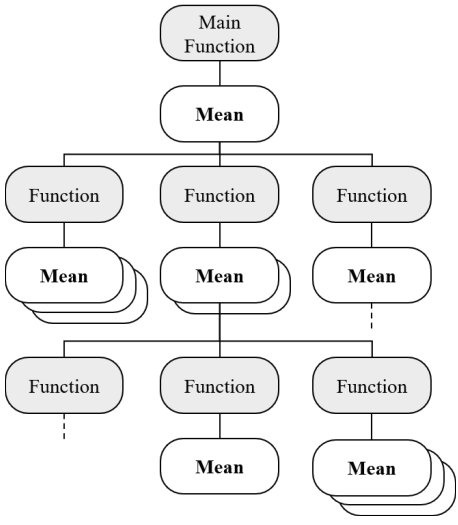


Figure 2.6 *An example of a Function/Means tree (F/M tree), showing the hierarchy between alternating functions and means.*

An F/M tree creates a hierarchy of functions and means, as illustrated in Figure 2.6. The top of the tree structure describes the main functions that the system should perform. The level beneath describes the different means or alternatives that can implement the main functions. Each means can subsequently be composed of lower-level functions that in turn are performed by lower-level means [93], [94]. This alternate decomposition can then be continued. The bottom level of means indicates sub-systems or system elements that may be used to fulfil the functions above. In this sense, a system concept corresponds to a set of particular means in the design space that the F/M tree spans.

Matrix-Based Approaches

Matrix-based approaches are commonly used to provide designers with an overview of an available design space and to generate concepts in product development. There are several examples of existing matrix-based approaches that can be used in an SoS approach to a design process. QFD was previously mentioned as an alternative for translating customer needs into prioritized requirements of a system to be developed. Another approach is a Design Structure Matrix (DSM) which can be used to display the connectivities and interdependencies between elements within a system's design [95].

An F/M tree can also be used in conjunction with matrix-based methods. One such approach is a morphological matrix, or a matrix of alternatives, which can be used to select alternatives for each function in the tree and thereby build up a concept for a new system. A morphological matrix provides a good overview for selecting suitable means for different functions [96]. The elements of the matrix represent the available design space of alternatives. Typically, the rows are used to represent the different functions, while the columns are used to list the corresponding alternatives of relevant system elements. The compatibility between design alternatives is not represented in a conventional matrix of alternatives. This can instead be done in a symmetrical compatibility matrix, where compatibilities between alternatives are typically indicated as either true or false.

An Interactive Reconfigurable Matrix of Alternatives (IRMA) builds upon the combined information of a matrix of alternatives and a compatibility matrix, and implements them in an interactive way that can be dynamically reconfigured depending on a designer's selections [16]. Consequently, an IRMA shows the potential incompatibilities within the design space and can at the same time be used to perform interactive "what-if" analyses. Design alternatives incompatible with chosen design elements can therefore be ruled out while simultaneously reducing the size of the available design space. Studies from Georgia Tech's ASDL have shown that matrix-based approaches, such as an IRMA, can be used to illustrate and create an available design space from knowledge captured in an ontology model [97]. In a similar way, an ontology and an IRMA can be used to model and prune a design space for cyber-physical

systems in a conceptual design context [98]. The pruning of an available design space is an important aspect in system design from an SoS perspective. The available solutions can grow at an exponential rate due to the combinatorial nature of a complex system or SoS. An IRMA can thus help by dynamically keeping relevant parts of the design space active. An additional benefit of an IRMA is that it provides an interactive visualization of the design space. This contributes to the early understanding of the interconnections and potential conflicts that exist in the design of a complex system [16].

2.4.3 Statistical Models and Optimization

Computationally inexpensive models of constituent systems are desired in explorations of the large design spaces that SoSs span. Many models used in the early stages of product development, SE and especially aircraft design build upon statistical estimations of existing solutions. Such estimations are typically combined with different optimization techniques to ultimately span a design space that can be used to explore different design alternatives. The most prominent solutions for continued evaluation and development can thereafter be identified. Statistical models can be created in a number of ways. However, this section only presents the techniques that have been utilized in the presented work and appended papers. Neural networks, deep learning and other Artificial Intelligence (AI) branches such as machine learning are therefore not covered.

Regressions

Regressions based on statistical analyses of existing solutions may be performed, for example, to identify trends that can be used to make predictive calculations about the performance of new systems under development. The reasoning behind this is that existing solutions build upon a collective knowledge that to some degree can be applicable to estimations for new design as well. The characteristic traits of a new design can thereby be predicted to some degree of accuracy, which is usually sufficient at an early conceptual design stage and in ruling out unfavourable regions of a design space [99]. There are several types of regressions that can be performed on statistical data. In general, a statistical regression can be described as a method for determining dependencies between variables of interest in a dataset. One of the simplest forms of regression is graph-based trend line fitting between two variables [100]. The relationship between the two variables is, for example, given as a linear equation. Other types of trend line equations, such as exponential functions, may provide a better fit against the underlying data, and it can be hard to determine the most appropriate expression. As with any statistics, more data is always preferable, since it reduces uncertainty and makes for more accurate estimations. However, some degree of intuition is almost always necessary in trend line fitting, which can make it prone to biases [100].

Regression analyses can also be made between more than just two variables. Such regression analyses are normally referred to as multiple regression analyses [101]. Typically, two or more independent variables with known values are used here to estimate the value of a single dependent parameter. Multiple regression analyses also include weightings so that the contribution of each predictor variable can be determined, as compared to a "regular" regression with only one variable. One way to enhance the fit of a linear regression model is to perform the regression on the logarithmic values of the underlying data set. The potential benefits of doing this include enabling the data to be better "centred", thereby allowing for a better fit and analysis of datasets that are not normally distributed [102].

The regression methods described so far are fairly simple ways of creating predictive models and of establishing the relationships between variables in datasets. More advanced methods for creating predictive models include techniques such as symbolic regressions with genetic algorithms and Singular Value Decomposition (SVD).

Symbolic Regression

Symbolic regression is a regression analysis that utilizes optimization to find a mathematical expression that best fits an underlying dataset. Lengthy mathematical expressions are here penalized so that both accuracy and simplicity are taken into account [103]. There are several different kinds of optimization techniques. However, optimizations with genetic algorithms are particularly suitable for symbolic regressions. The reason for this is that mathematical operators – such as addition, subtraction, and multiplication – can be seen as the different available chromosomes. These mathematical operators are consequently the design parameters that can vary throughout the optimization. The goal of the optimization is typically to minimize aspects such as the mean square error, while simultaneously penalizing the complexity of the resulting expression or individual. In this sense, it reduces some of the bias from the regression techniques discussed before and results in "simple" but accurate prediction models that have been derived from statistical data and optimization. Ultimately, a symbolic regression results in an expression, or formula, that has been optimized to fit the underlying data and to be simple in terms of length and complexity. An illustration of a symbolic regression process is presented in Figure 2.7.

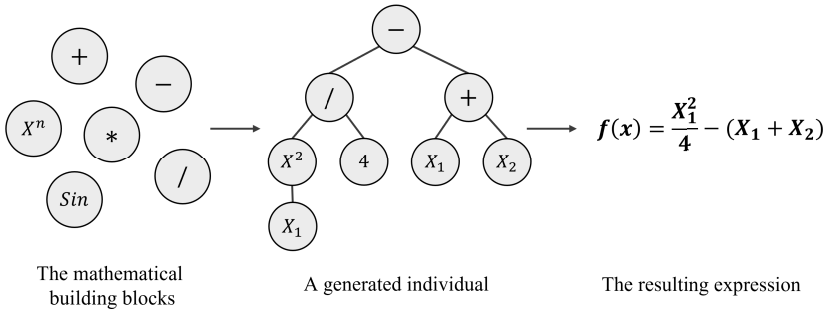


Figure 2.7 *An illustration of a symbolic regression process. Reproduced from paper [VII].*

Singular Value Decomposition

A Singular Value Decomposition (SVD) is a form of statistical analysis that can be used to create estimation models from statistical data [99]. A benefit, and powerful feature, of SVD compared to other estimation methods is that only a small number of input parameters are required to make an estimate of the remaining ones in a dataset [104]. An SVD analysis is also related to something known as a Principal Component Analysis (PCA), which can be used to determine the driving parameters in a data set and thereby reduce dimensionality [105]. In general, an SVD analysis starts with a matrix factorization and decomposition into a general form which is shown in equations 2.1 and 2.2.

$$X = U \cdot W \cdot V^T \tag{2.1}$$

$$\begin{bmatrix} x_{11} & \cdots & x_{1i} \\ \vdots & \ddots & \vdots \\ x_{s1} & \cdots & x_{si} \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \vdots \\ u_{s1} & \cdots & u_{sr} \end{bmatrix} \cdot \begin{bmatrix} w_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & w_{rr} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1i} \\ \vdots & \ddots & \vdots \\ v_{r1} & \cdots & v_{ri} \end{bmatrix}^T \tag{2.2}$$

As seen in equations 2.1 and 2.2, any matrix, or dataset, can be factorized into three different matrices. Essentially, a dataset denoted X contains information about subjects (s) and items (i). For example, subjects can represent different car models, while items could correspond to parameters such as cost, engine size or the number of doors. The U-matrix represents subject vectors, whilst the item vectors are described by the V^T-matrix. The matrices are arranged here so that the elements of the diagonal W-matrix are in descending order. The W-matrix simply corresponds to the singular values, or principal components, of the SVD. The descending order of the W-matrix elements results in a descending order of influence for each row in the U-matrix. Consequently, the singular values, also called SVD variables, of low influence in

the W-matrix can be omitted, thus reducing the overall dimensionality. This results in an estimation model that only requires as many inputs as there are SVD variables left. Naturally, the number of remaining SVD variables becomes a trade-off against accuracy of the estimations. However, since the SVD identifies the principal components and underlying correlations of the dataset used, the required number of SVD variables is typically low compared with the original dimensionality. From here, remaining elements in the U-matrix can be translated back into X through the matrix product. Consequently, the X-matrix is now an approximation of the original dataset, based on just a few SVD variables.

A reduced model obtained from an SVD analysis can be used to estimate characteristics of a system under development. This is done by adjusting the remaining SVD variables so that desired or known characteristics of the system are matched by the model. All other characteristics described in the original dataset are thereby estimated based on only the few SVD variables matched against the desired properties. An SVD analysis like the one just described could be used to give quick estimates of a design based on given requirements, and to expand available information [99].

Engineering Design Optimization and Surrogate Modelling

Genetic algorithms were previously mentioned as one optimization technique related to symbolic regressions. However, a plethora of different optimization methods that can been used in engineering studies exist, ranging from simple gradient-based approaches to more complex particle swarm optimizations and evolutionary algorithms [106], [107]. A formal mathematical definition of an optimization problem is given below in equations 2.3–2.7.

Minimize:

$$F(x) = [f_1(x), f_2(x), ..., f_k(x)] \tag{2.3}$$

subject to:

$$h_i(x) = 0, \qquad i = 1, ..., m_1 \tag{2.4}$$

$$g_j(x) \leq 0, \qquad j = 1, ..., m_2 \tag{2.5}$$

$$x \in X \subseteq R^n \tag{2.6}$$

for:

$$x = (x_1, x_2, ..., x_p) \tag{2.7}$$

All optimization problems typically include objective functions (equation 2.3), constraints (equations 2.4–2.6) and design variables (equation 2.7). An

objective function describes the goal of an optimization, which could be a minimization of a specific variable. Constraints are typically introduced to keep different variables within the optimization in suitable or allowed intervals. Lastly, design variables are the elements that can be changed throughout an optimization. This could, for example, be different parts of a design that a designer can change to impact an objective function. From an aircraft design perspective, an objective function could be a maximization of range, with weight as a constraint and different wing parameters as design variables. Optimizations can also be performed with more than one objective function. Such optimizations are commonly referred to as Multi-Objective Optimizations (MOO). Similarly, optimizations spanning more than one engineering subject, for example aerodynamics, are referred to as Multi-disciplinary Design Optimizations (MDO) [108], [109]. The early design of complex systems and SoSs typically involves multiple design domains and thereby considerations that often result in contradictory requirements. Negotiations and trade-offs must consequently be made to find the most suitable solutions from all design perspectives. Optimization is therefore a powerful tool that can facilitate the design work and aid in decision-making. MOOs and MDOs typically result in different Pareto fronts of optimum solutions [108]. Each design on a Pareto front is an optimum solution with regards to all objectives. As a result, a Pareto front illustrates the available trade space on the set of most suitable design variable combinations.

As an engineering field, MDO additionally deals with sensitivity analyses, information processing and management strategies, to name a few [110]. Additionally, it can be viewed as a methodology where the interactions between involved disciplines and systems are in focus. However, these interactions between different disciplines do introduce additional challenges compared to optimizations on a single discipline. The two main challenges that an MDO approach brings may relate to computational cost and organizational complexity [111]. Expense in terms of computational cost and time can be a problem in all types of optimizations. One remedy for this is to create surrogate or meta models. Individual discipline models in an MDO can here be exchanged for surrogates that approximate the models' outputs and thereby reduces the computational expense. However, certain simulations must be performed in order to create representative surrogate models. One method is to use a Design Of Experiments (DOE) for this purpose and to determine the simulations that should be performed [112]. There are several different kinds of DOE methods that can be used to suggest the simulations to be performed for the creation of a meta model. One of the most common types is, for instance, Latin hypercube sampling. Surrogate models can thereafter be created with a variety of methods, including Singular Value Decomposition (SVD) [99]. Naturally, a surrogate model is an approximation of the model it was created from. Prominent solutions found through optimizations with surrogate models may therefore be subjects for further analysis to reduce uncertainty.

Optimizations and MDO have been utilized in a number of SoS studies. One such study presented in [113] illustrates how design optimization of Unmanned Aerial Vehicle (UAV)s can be performed in an SoS context. Another example is provided in [114], where optimization is used to streamline operations of an SoS and to mitigate prediction errors on emergent behaviours. A methodology for optimizing and evaluating candidate SoS architectures through simulations and different performance metrics is presented in [115]. Finally, [23] illustrates how an optimization framework can be used to architect resilient SoS solutions.

2.5 Visual Analytics

Visual Analytics (VA) can be described as a field that focuses on the translation of data into interactive visual representations that both improve data exploration capabilities and facilitates decision support [116]. The aim of VA is to provide enhanced insight and an ability to obtain useful information from large amounts of data in a human readable way [117]. Analyses and analytical reasoning over large and complex data sets with VA are consequently performed with the "human-in-the-loop". This allows decision-makers to quickly gain insight from the data and possibly discover hidden relationships within it. Both [116] and [117] provide thorough overviews of VA and its applications in different engineering fields. Additionally, [117] explains that a VA process typically follows six steps:

1. Pre-processing of data.
2. Application of algorithmic analysis methods.
3. Visualization of the processed data.
4. Generation of insightful knowledge from user interaction.
5. Interactive knowledge integration with the analysis and visualization.
6. Updated visualization based on user decisions and interactions.

The first three steps listed above involve little focus on user interactions and should be performed in a succeeding order. Steps four to six, on the other hand, are typically done in an iterative manner. These final steps are then repeated until a desired level of insight has been achieved. The visuals can thereafter be used for various purposes, such as enhancing decision-making capabilities. The "human-in-the-loop" idea is heavily focused on and involved throughout steps four to six.

The concepts of design and trade space explorations were described earlier in this chapter. VA is connected to these in several ways and can be used in conjunction with them to enable interactive design and trade space explorations. The influence of different design choices and how their expected delivered values change can, for example, be determined interactively using such

an approach. This allows decision-makers to explore the available design and trade space in a "what-if" manner until the most appropriate solutions have been identified. There are many examples of studies where both VA and Trade Space Explorations (TSE) have been used to approach engineering problems in relation to complex systems and SoSs. One such example is presented in [89], where investigations into the design and trade space for space tug satellite systems are performed in order to assess the delivered value for different system architectures. Another study investigates the design and trade space for supersonic business jet concepts with VA [118]. The VA approach is used here to assist the designer in both navigation and understanding of an extensive number of design alternatives, thus facilitating decision support while reducing the designer's cognitive burden. In this particular case, an Interactive Reconfigurable Matrix of Alternatives (IRMA) is used to provide a concise overview of the available design alternatives. The IRMA consequently allows interactive explorations of the design space for possible concepts. It also allows designers to directly see the impact of different design decisions from perspectives such as dependency and incompatibility. An example of TSE and VA from an SoS perspective is presented in [119], which describes a methodology for experimental design model-based systems engineering. The methodology is illustrated via the conceptual design of a defence system. Additionally, [119] also illustrates how a dynamic dashboard can contribute to an interactive exploration of the design space. Finally, uncertainty in trade space analyses is also an important metric to consider, and it must be both managed and quantified for a design at an early stage to enable elaborate decision-making [120]. VA techniques can be used to provide visualizations of the uncertainty associated with different parts of a design or trade space under exploration. Guidance in, for example, risk mitigation can thereby be facilitated. However, this typically comes at the cost of more dimensions and overall complexity of the visualizations [116].

3

A Holistic Approach for Early Product Development

Based on the five levels introduced in [1], this chapter illustrates a method that has been formed from the theoretical background and literature study presented in chapter 2. The suggested method illustrates how four approaches can be used to realize different parts of the holistic System-of-Systems (SoS) design model from Figure 1.2, and to provide a collective analysis that spans all levels. Figure 3.1 shows an overview of the different approaches that have been used to cover the five levels and to form the foundation of the method presented in this dissertation.

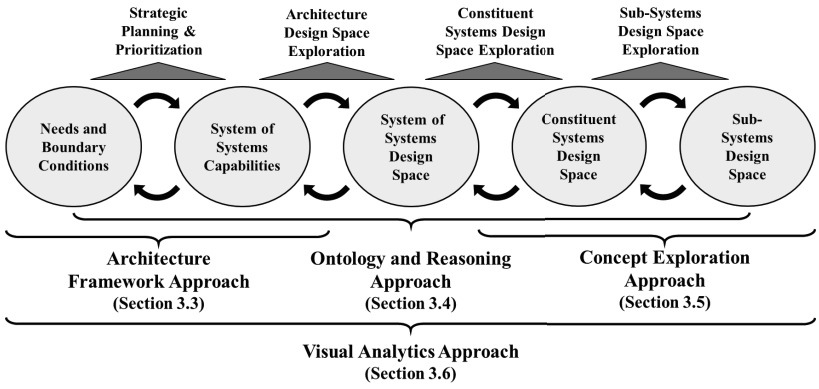


Figure 3.1 *The corresponding levels of interests from [1] for which this dissertation contributes realizations. The figure also illustrates the levels to which the different approaches belong, and in what section of the dissertation they are described.*

Additionally, this chapter shows how the method can be used in early product development and to provide decision support, as well as to illustrate design trade-offs from an SoS perspective. The original versions of the four approaches from Figure 3.1 are described in detail in the appended papers. However, these approaches have been slightly updated in this work to better illustrate their use and to facilitate their interoperability in the presented method.

3.1 Method

The method presented in this section shows the proposed way of traversing the different levels of the holistic SoS design model from [1]. This is shown in Figure 3.2 from a top-down perspective, starting with the SoS needs.

In Figure 3.2, the architecture framework approach (A) is used to break down overarching needs into corresponding capabilities and functions. This particular procedure originates from paper [III]. The outcome from the breakdown is then represented using the ontology approach (B) that incorporates reasoning. The ontology generates a reduced design space of alternatives or functions to be performed. However, the design space can also include existing system solutions here as well. Ontology and reasoning are considered a major part of this dissertation and the presented method. The approach has mainly been based around the work performed in papers [I], [II], [III], [IV] and [VI]. Suggested existing system solutions from the ontology approach can now be used as inputs for simulations of different SoS architectures or compositions. This was first investigated in paper [II] and later used to form the visual analytics approach (D) introduced in paper [V]. The simulations generate performance measures that can be used to identify the most prominent SoSs. However, these simulations can also be used to identify capability gaps and thereby give overarching requirements for new solutions to be developed. The reduced design space of alternatives or functions from the ontology approach can from this point act as inputs for a continued design process with the obtained requirements. The concept exploration approach (C) illustrates how different Constituent System (CS) concepts are explored based on the various design alternatives and requirements. The characteristics of feasible concepts are thereafter approximated using different estimation techniques based on, for example, statistics and optimizations. This way of exploring concepts and their properties was first examined in papers [VI] and [VII]. The simulation process can subsequently be revisited, with the obtained CS estimates added to the set of possible SoS architectures in terms composition. Prominent SoS architectures with reduced design spaces can thereafter be further investigated with more detailed analyses in an iterative way.

Finally, the visual analytics approach (D) can be used to span the workflow performed so far and to give an interactive design platform on which different parameters can be varied to illustrate design trade-offs and the influence of various boundary conditions. This consequently enables interactive trade

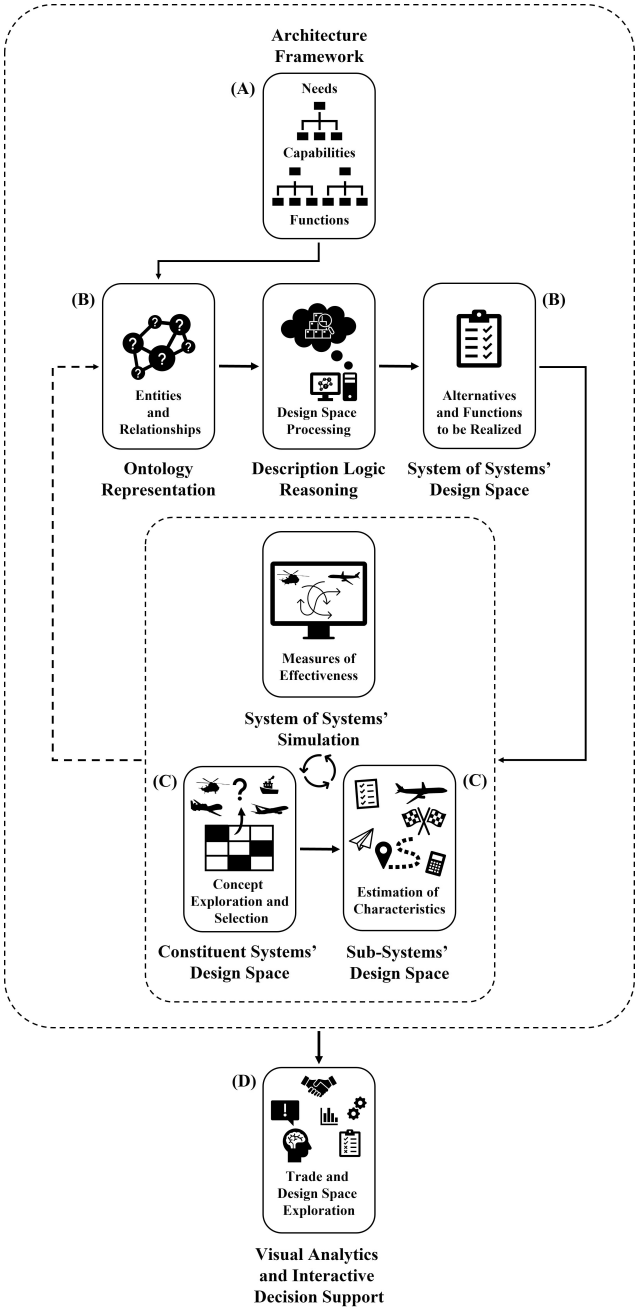


Figure 3.2 *The overall method that has been synthesized from the different approaches to realize a holistic SoS design model.*

and design space exploration capabilities that can be used to provide decision support through "what-if" investigations.

As mentioned in section 1.1.1, the holistic SoS design model from Figure 1.2 has no specific starting point and an analysis can start at any level. The method described in this section shows only one way of combining the different approaches to cover the different levels. The remainder of this chapter will therefore describe the individual parts of the presented method in more detail, and elaborate on how the different approaches can be used outside the method for other aspects of holistic SoS analyses.

3.1.1 Design and Trade Space Definitions

The concepts of design and trade spaces have been mentioned on several occasions so far. The definitions for these throughout this dissertation build upon the content of section 2.4.1. A design space is built upon design variables/parameters which can be of both a discrete and a continuous nature. A specific operational activity may be a discrete design variable from an SoS perspective; the wingspan of an aircraft is an example of a continuous design variable at CS level. In general, design variables are the entities that a designer can change. In contrast, uncertainty parameters represent things that are not precisely known. Uncertainty parameters can be divided into two types, namely aleatoric and epistemic uncertainties [121]. Aleatoric uncertainty represents randomness, such as stochastic variations in an operational environment. Epistemic uncertainty represents uncertainty from limit knowledge, such as the accuracy of low-fidelity models. Unlike aleatoric, epistemic uncertainty can be reduced by, for example, using more accurate models. Together, design variables and uncertainty parameters generate system characteristics, sometimes also called functional characteristics. This can, for example, be the performance of a particular SoS or the resulting range of an airborne CS. As mentioned in section 2.4.1, a design space contains the possible solutions to a given design problem. These solutions are thereby described by their system characteristics obtained from the design variables and uncertainty parameters. Requirements and design constraints act as limiters on design variables and system characteristics.

A trade space, on the other hand, is a result of design spaces spanned by system characteristics. As the name implies, a trade space illustrates the trade-off between system requirements or characteristics. Each point, or design option, in a trade space represents a specific set of system characteristics and thereby specific values for the involved design variables. Ideally, each point represents an optimized design based on a limited number of requirement values. Thus, the number of dimensions for a trade space representation is largely reduced compared to those of a design space. System characteristics may be of different importance in an analysis and can thus be weighted using, for example, a Quality Function Deployment (QFD) approach. Additionally, system characteristics can be put together into value functions, which can further reduce

some of the dimensionality of a trade space [89]. Finally, design and trade space exploration can be performed to increase the overall knowledge of a design at an early stage. Such explorations follow the definitions already listed in section 2.4.1.

3.2 Search and Rescue as a Case Study

Throughout this work, Search and Rescue (SAR) operations have been used as case studies to test the proposed approaches. SAR operations can be considered as typical examples of directed SoSs with a high degree of centralized control over the CSs involved. Furthermore, SAR often includes several types of systems that may be influenced by the operational environment to a large extent. In this sense, the operational environment includes factors such as weather and environmental conditions that may introduce changes in required capabilities and functions to be performed to meet stakeholder needs in a satisfactory way.

The SAR case studies used have mainly been based on available information about SAR systems and SAR operations from the International Aeronautical and Maritime Search and Rescue (IAMSAR) manuals [122]–[124] and the Swedish Maritime Administration (SMA) [125], [126]. Moreover, the focus has mainly been on airborne SAR solutions. Figure 3.3 shows an example of a SAR scenario with different types of SAR assets.

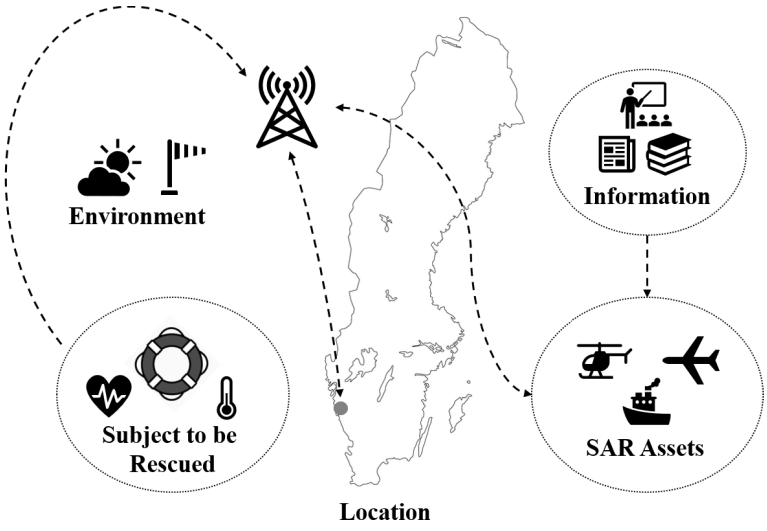


Figure 3.3 *An example of a Search and Rescue (SAR) scenario off the coast of Sweden.*

3.3 Architecture Framework Approach

The architecture framework approach was introduced in paper [III]. This approach shows how an architecture framework can be used to provide a standardized and consistent way of performing a breakdown of SoS needs into required capabilities and subsequent functions to be realized in a continued design process. SoS needs typically correspond to those of the involved stakeholders. In paper [III], four basic needs that any SAR system should meet were identified based on the IAMSAR manuals. These were listed as needs to:

1. Manage information
2. Coordinate search response
3. Coordinate rescue response
4. Provide assistance

These four basic needs were used as a starting point for the breakdown using the architecture framework.

3.3.1 Breakdown of System of Systems Needs Using the Unified Architecture Framework

As mentioned in section 2.2.2, architecture frameworks consist of different viewpoints that are designed to describe the architecture of complex systems and SoSs. They can consequently be used to describe models for areas of interest, such as capabilities and how they are related to operational activities. The Unified Architecture Framework (UAF) was chosen and used in the case study in paper [III], since it builds upon a combination of previous architecture frameworks. The intention of the approach has been to use the taxonomy and description of the UAF to provide a consistent and general way of understanding the hierarchical structure and relationships that lead down from an SoS need to a function to be performed by a CS.

Based on this, different UAF viewpoints were used to create a general procedure for understanding the governing relationships of an SoS needs breakdown. A UAF sample problem, [39], was used as both a guideline and a source of inspiration for utilizing the UAF in the case study in paper [III]. The different UAF viewpoints that were used were as follows:

- **Strategic Taxonomy (St-Tx)**
  Used to describe a taxonomy of available capabilities and their composition.
- **Strategic Structure (St-Sr)**
  Can be used to describe the relationships between different capabilities listed in the St-Tx view.

- **Strategic Traceability (St-Tr)**
  Can show how operational activities support represented capabilities.

- **Operational Taxonomy (Op-Tx)**
  Used to list and describe a taxonomy of all operational activities. Similar to the St-Tx view, but used for operational activities instead of capabilities.

- **Operational Processes (Op-Pr)**
  This view can be used to describe the relationships between operational activities and their corresponding sub-activities.

- **Operational Traceability (Op-Tr)**
  Similarly to the St-Tr view, can be used to describe the mapping between operational activities and capabilities.

- **Resource Taxonomy (Rs-Tx)**
  Can be used to describe a taxonomy of functions and their composition, for example.

- **Resource Processes (Rs-Pr)**
  This view shows the relationships between functions, and can also be used to describe their inputs and outputs.

- **Resource Traceability (Rs-Tr)**
  Describes how the functions implement and map onto the different operational activities.

- **Parameters Environment (Pm-En)**
  This view can be used to describe the environment and relevant environmental conditions of the domain in question.

The different views described above were then synthesized into a diagram describing how needs could be broken down in consecutive steps into capabilities, activities and functions to be performed. This resulting diagram, referred to as the architecture framework approach, can be seen in a UML format in Figure 3.4.

Needs may come from stakeholders or customers, and typically correspond to requests for solutions to problems that can be provided by a service in a defined environment. A stakeholder might be an individual or an organization with an interest in at least one phase of a complex system or an SoS described in the architecture framework. The system must be capable of satisfying the needs and thereby delivering the service. Consequently, the system must possess the capabilities to do so. In this sense, needs must always be associated with capabilities to be fulfilled, and a capability is thus dependent on a corresponding need. A capability is defined as the ability to implement activities through a

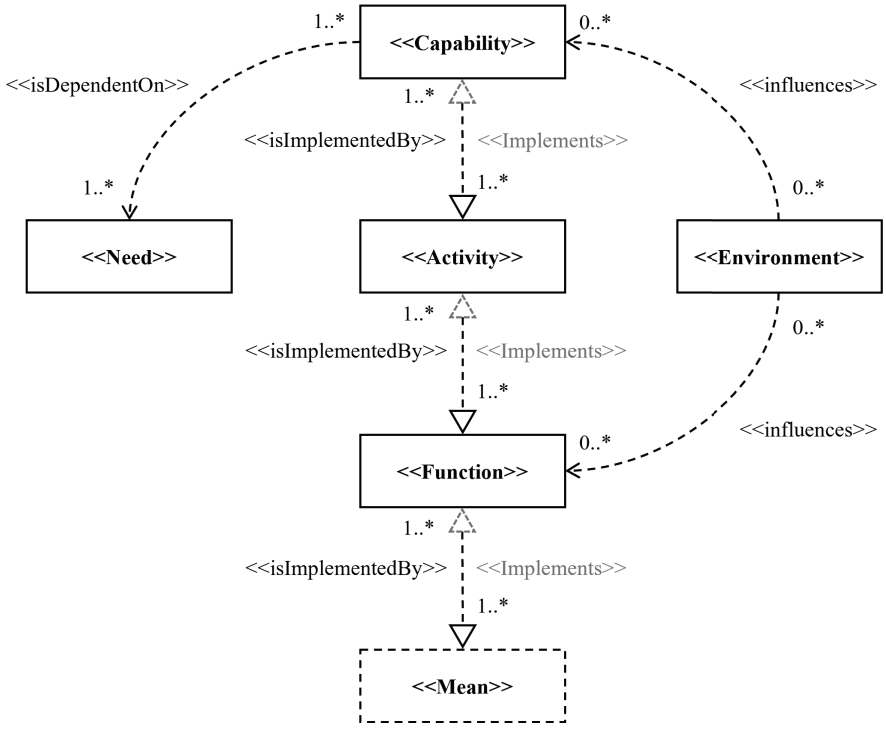


Figure 3.4 *A UML diagram of the architecture framework approach, illustrating the breakdown process and the relationships between needs, capabilities, activities, functions, means and the environment.*

combination of different ways and means. Therefore, activities are implementations, or realizations, of capabilities. As Figure 3.4 shows, a capability can be implemented by at least one activity, and one or more activities can implement at least one capability in the inverse direction. An activity is implemented by one or more functions. The functions are in turn implemented by means or elements that, for example, correspond to CSs and sub-systems. The relationships between functions and means can from this point be further utilized with an F/M tree approach which allows for a continued and more detailed functional breakdown if desired. Finally, the environment has an influence on the required functions and capabilities. An activity might consequently require different functions to successfully realize the intended capability, depending on the surrounding conditions. The definitions listed above are based on the information found in the UAF sample problem and documentation [39], [127].

3.3.2 Outcome and Design Space

The architecture framework approach outlined in section 3.3.1 results in a number of functions that are to be performed to realize the overarching capabilities required to satisfy the customer and stakeholder needs. Consequently, it creates a design space of functions to be performed that can be either realized using existing means or used as inputs for a continued design process. Further details and discussions on how the outcome can be used for this purpose are presented in section 3.5 and chapter 4. Additionally, the outcome and design space of functions from the architecture framework approach can also be further processed by being represented in an ontology, as seen in Figure 3.2.

3.4 Ontology and Reasoning Approach

Ontology and reasoning have been a common topic for almost all the appended papers in this dissertation. Ontology has here been used to represent entities and their relationships, similarly to the way that UML and SysML models are used. In this way, the ontology can correspond to a representation of an available design space for systems and SoSs. Description logic reasoning subsequently allows the design space to be processed, as this section will describe in more detail.

3.4.1 Building an Ontology for an SoS Design Space

The ontology approach referred to in this dissertation was first introduced in paper [I], and has been further refined and utilized in papers [II]–[VI]. Paper [I] also proposed a process for building an ontology that represents an available SoS design space. Figure 3.5 shows a slightly updated version of this process.

The process in Figure 3.5 consists of eight steps, and has been partly based on the guidelines presented in [56] and [65]. The process is intended to be used with the Web Ontology Language (OWL) which supports reasoning, as mentioned in section 2.3. The Protégé ontology editing software, [128], has been used to implement ontologies with the process from Figure 3.5 in all case studies in the appended papers.

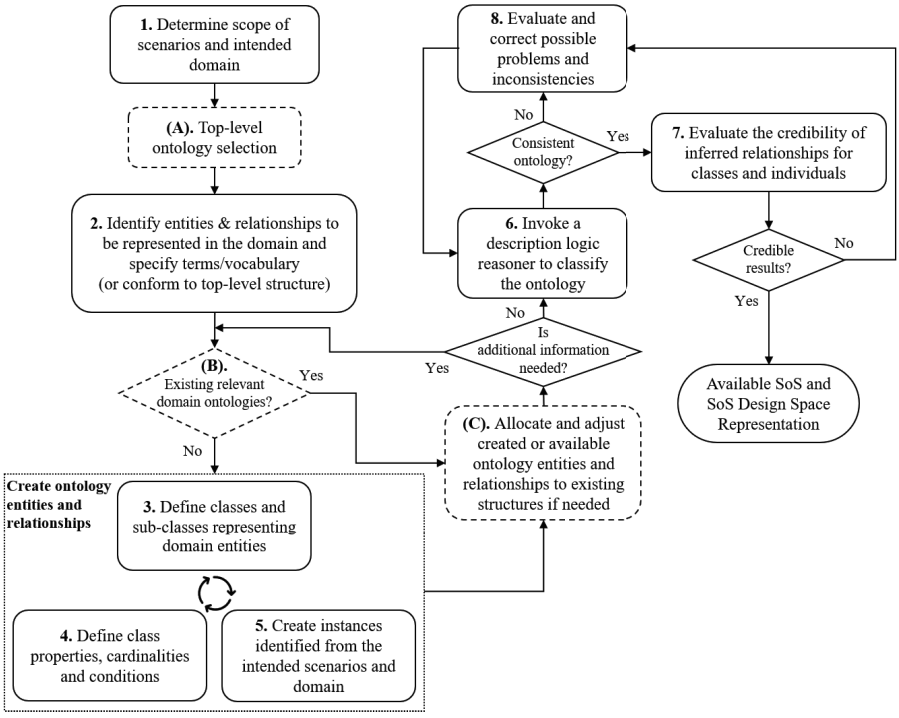


Figure 3.5 *The ontology development process for generating a representation of an SoS design space. Updated from the version first introduced in paper [I].*

Step 1: What Should Be Modelled?

The process in Figure 3.5 starts with the step of determining the scope of the scenarios and the intended domains to be modelled. This could be different SAR operations as shown in the case studies in papers [I] and [II], for example. The scope may also correspond to the outcome of a breakdown with the architecture framework approach as shown in paper [III].

Step (A): Top-Level Ontology Selection

An optional step before step two allows for the incorporation of a top-level ontology structure. As explained in [129] and briefly mentioned in section 2.3.1, top-level ontologies can bring several benefits, such as improved interoperability between different domain ontologies. Top-level ontologies, such as the Basic Formal Ontology (BFO) [65], [129], offer a predefined structure and hierarchy for representing domain knowledge. The case study in paper [IV] illustrates how BFO can be used as a structure for merging different domain-specific ontologies. An OWL variant of BFO ([130]) was here utilized to give a predefined

class structure that the other ontologies could conform to. Figure 3.6 shows a reduced representation of the BFO class hierarchy.

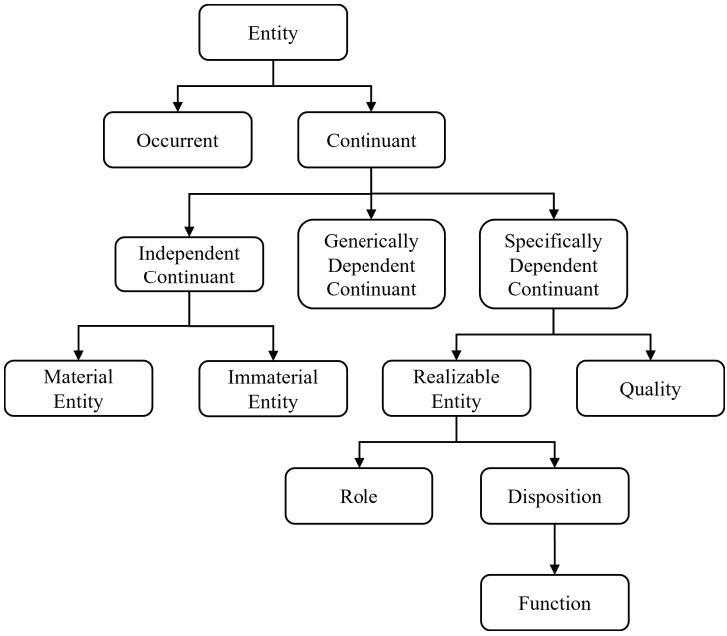


Figure 3.6 *A small part of the Basic Formal Ontology (BFO) class hierarchy.*

The different classes in the BFO class hierarchy are intended to be domain neutral so that knowledge from any domain can be represented. Additionally, the relationships between classes in BFO build on an *"is a"* structure upwards in the hierarchy. This is explained in further detail under **Step 3** in this section. The BFO specification and user's guide ([65]) gives guidance on what entities may fit under the different classes. Functions such as the ones discussed in relation to the architecture framework approach, for example, are placed under the *Function* class in Figure 3.6. Aerospace systems, such as aeroplanes and helicopters, are suitably placed as sub-classes and individuals under the *Material Entity* class in the BFO hierarchy.

Step 2: Identify Relevant Entities and Relationships

The second step of the process involves identifying the relevant entities and relationships that are to be represented in the ontology. From a SAR perspective, entities may include available assets, weather conditions, capabilities, regulations and more. However, the scope can also be on the design of new SAR vehicles from an SoS perspective. Entities to be modelled can in such a case correspond to overall functions and the means by which to implement

them, together with their respective design alternatives, as shown in the case study in paper [IV]. Relevant entities to be represented can also be identified via holistic analyses of possible scenarios for the intended domain and scope, for example. Terms and vocabularies in the ontology should be specified in a coherent way so that ambiguities in definitions are avoided. Top-level ontologies may include existing terms and vocabularies in their definitions, and the knowledge to be represented should conform with these to facilitate reusability if such an ontology structure is included.

Step (B): Include Existing Domain Ontologies?

The option of including existing domain ontologies is introduced before step 3. The reason for this is that ontologies can suffer from a "reinvent-the-wheel" syndrome, where several ontologies are created for the same purpose [129]. Therefore, it is desirable to consider using existing ontologies in the process to avoid creating a new and redundant one [56]. If existing ontologies are to be used, they should be adjusted to fit the intended formalism and structure of the ontology under development, as explained in **Step (C)**. The case study in paper [IV] once again provides an example of how this can be done. In this particular case, an aircraft design ontology from [68] is incorporated to provide means/design alternatives for some of the required SAR functions. This ontology was also adjusted to fit the BFO formalism in accordance with **Step (C)** before being incorporated to facilitate the merging process.

Step 3: Create Classes

Step 3 of the process involves the creation of ontology classes from the entities identified in **Step 2**. Ontology classes are implemented in a hierarchical structure with classes that can have sub-classes, which in turn have their own sub-classes and so on. The relationships between classes in the hierarchy structure are typically of an *"is a"* nature. This means that a sub-class is a specialization of its super-class, and that the sub-class consequently inherits the properties of the super-class. For example, an aeroplane *is an* airborne vehicle, which in turn *is a* vehicle that *is a* system that *is an* entity. Saying that all vehicles have a mass property, for example, implies that all aeroplanes must also have mass properties. Figure 3.7 shows an illustration of an *is a* hierarchy.

Different approaches can be used when establishing a class hierarchy, as described in [56]. Due to the open world and non-unique naming assumption in OWL, it is also important that classes that are not equal to each other are defined as disjoint for the subsequent invocation of the reasoner. A helicopter class should, for example, be disjoint with a weather condition class, since a helicopter cannot possibly be a type of weather phenomenon.

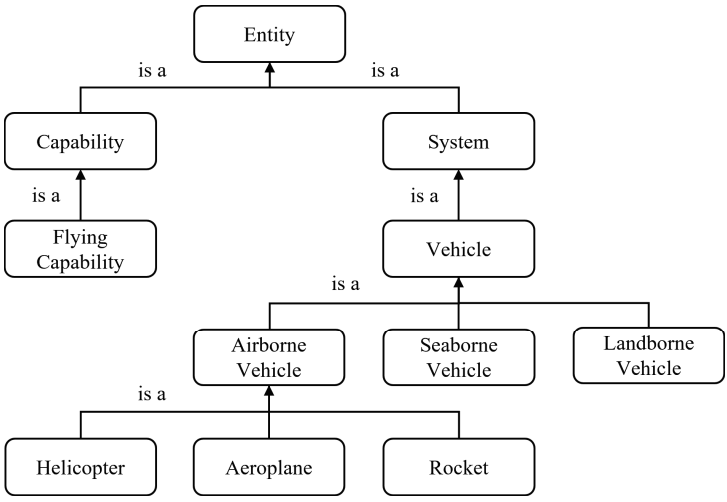


Figure 3.7 *A class hierarchy illustrating the "is a" relationship.*

Step 4: Create Relationships

This step is used to define all relationships that are to be represented in the ontology. Ontology relationships describe how classes and individuals are related to each other, but can also describe structures, properties and data values, for example. Relationships are described using *object* and *data properties* in OWL. *Object properties* are used to describe relationships between ontology classes and instances, while *data properties* are used to describe how classes and instances relate to data, such as numbers and strings. Cardinality specifications can be used to describe the number of relationships that classes have in a min., max. or exact logic. OWL includes many more options for defining relationships using *object* and *data properties*, and further explanations and details are found in [56] and [60].

Additionally, ontology classes can be defined as either *primitive* or *defined*. The differences between these two types are what OWL refers to as *necessary* and *sufficient conditions*. A *primitive* class only contains *necessary conditions*. These may, for example, explicitly describe how an entity is built up in terms of relationships within the ontology. A *defined* class contains at least one *necessary* and *sufficient condition*. In short, *defined classes* allow for automated classification and processing of implicit relationships between classes and individuals in the ontology using a reasoner. More information about this, as well as examples, can be found in [60] and in the case studies in papers [I], [II], [III], [IV], and [VI].

Step 5: Create Instances

Individuals, also sometimes referred to as instances, are defined as *"the most specific concepts represented in a knowledge base"* [56]. Consequently, individuals represent the lowest granularity or level of detail in the modelled domain of the ontology. Classes can be seen as collections of corresponding individuals. Relationships between classes thereby describe general relationships between their individuals. Individuals can also be members of more than one class. In a SAR context, an individual can be a helicopter of a specific type that is a member of the helicopter class in Figure 3.7, for example.

As just shown, **Steps 3** to **5** are used to create the entities and relationships in the ontology model. These three steps have no specific starting point, and can typically be performed in an iterative manner until a desired ontology representation of the domain is achieved.

Step (C): Adjust Entities and Relationships

This optional step is to be utilized if top-level or existing domain ontologies are being used. The previously represented entities and relationships from **Steps 3-5** should here be allocated and adjusted into the existing ontology structure provided by any top-level or domain ontologies. **Steps 3-5**, **(B)** and **(C)** can then be revisited and iterated if additional information is needed to model the domain in the intended way and to establish a desired knowledge representation.

Paper [IV] provides an illustration of this step by adjusting three different ontologies to the BFO hierarchy. More specifically, the case study shows how the classes and instances of the SAR ontology from [III] and the aircraft design ontology described in [68] can be allocated to suitable places within the BFO hierarchy. An aircraft function ontology is also created directly in the BFO hierarchy in the case study of paper [IV]. This aircraft function ontology is implemented using **Steps 3-5** and is based on an analysis of core aircraft functions described in [131]. The three ontologies are thereafter merged and mapped to each other with **Step (C)** under the BFO hierarchy. The end result of this is a large ontology that represents the available design space for a search aircraft to be developed from an SoS perspective.

Step 6: Invoke the Reasoner

Once a desired knowledge representation has been obtained, and the relevant entities and their relationships have been modelled, a reasoner can be invoked. The reasoner will classify and check the implemented ontology for inconsistencies, but it will also build up a new *inferred* ontology. The unclassified ontology is referred to as the *asserted* ontology, and corresponds to the outcomes from **Steps 3-5** that have not been processed by the reasoner. The *inferred* ontology

will be built up based on the defined relationships, properties and conditions in the *asserted* ontology model. *Defined* ontology classes will be populated with classes and instances that fulfil the *necessary* and *sufficient conditions* to be regarded as members of the *defined* class. The reasoner can consequently draw conclusions about implicit relationships in the ontology and restructure it accordingly. **Step 8** of the process should be performed if the reasoner classifies the ontology model as inconsistent.

Step 7: Assess Inferences

Step 7 should be performed to assess and evaluate the inferences made in the ontology model and its *defined* classes. The ontology may prove to be consistent but include unwanted or unreasonable results. The reasoner will make the inferences based on the modelled domain and its implemented relationships. Consequently, any modelling errors from **Steps 3-5** might result in a consistent ontology but with erroneous inferences from the modeller's perspective. If such inferences are found, or if the ontology is classified as inconsistent, **Step 8** must be performed.

Step 8: Correct Model Errors or Inconsistencies if Needed

Step 8 of the process should only be performed if the ontology is classified as inconsistent or if unwanted and erroneous inferences have been made according to the ontology modeller. Corrections can be made by revisiting **Steps 3-5** and consequently adjusting the relationships, properties and conditions of the implemented classes and individuals. Unwanted, erroneous or simply unreasonable results can be further investigated using description logic querying. The inferred ontology is here used to answer "questions" about the modelled domain and its classes and instances [79]. Consequently, description logic querying can give the modeller an indication and an explanation of why such inferences have been made by the reasoner. Inconsistencies can be hard to identify, and different ontology debuggers for OWL may be useful for finding the source of any inconsistencies in the ontology [132].

Ontology Process Outcome

The outcome from the ontology process in Figure 3.5 is a knowledge representation and a model of the entities and relationships that exist in the chosen domain. This knowledge representation can correspond to an initial design space representation for an SoS with, for example, available SAR assets to perform a mission, as the case studies in papers [I] and [II] show. However, it can also correspond to a design space of functions to be performed to meet overarching capabilities and needs with available means and design alternatives, as described in papers [III] and [IV]. Additionally, the case studies in papers [II]

and [IV] have shown how the proposed process can be used to extend and incorporate an existing ontology structure. These case studies have also illustrated how the process can be reiterated to include more information than initially specified.

A design space representation generated via the process can from here be further processed using the reasoner, as the next section will explain in more detail.

3.4.2 Reasoning and Design Space Processing

Reasoning has thus far mainly been used to classify and check the ontology representation for inconsistencies in the process from Figure 3.5. Another important capability that a reasoner provides is that it can infer implicit relationships in the ontology model, as briefly mentioned above. Consequently, reasoning can be used to further process the design space generated using the ontology development process. Reasoning has been used for this purpose in most of the papers appended to this dissertation.

Design Space Reduction

The general approach for processing the design space has been established by modelling *defined* classes with *necessary* and *sufficient conditions* in the ontology asking for user-defined needs, requirements and desired performances, for example. Reasoning has then been used to automatically populate the *defined* classes with the classes and instances that are able to fulfil the *necessary* and *sufficient conditions* described in them. This approach thereby filters the available design space and results in a reduced design space representation. The conditions in the *defined* classes can here ask for entities with relationships to certain capabilities or entities with certain data properties, for example, such as a rescue capacity above a specified number as shown in paper [I]. The reasoner can consequently filter out irrelevant solutions and thereby parts of the design space representation in the *defined* classes. However, this does not remove information from the overall design space, and the conditions in the *defined* classes can always be adjusted to open up or narrow down the size of the design space reduction. Figure 3.8 shows an illustration of a design space reduction using *defined* ontology classes.

The *defined* classes marked with grey backgrounds in Figure 3.8 create an intersection which represents the area where included instances and classes are able to fulfil the conditions of the two *defined* classes. Consequently, the available reduced design space is represented by the inferred sub-classes and individuals of the *defined* classes that have been used to query the ontology. More information and illustrative examples of how *defined* ontology classes and their *necessary* and *sufficient conditions* are created to perform design space reductions can be found in the case studies in papers [I], [II], [III], [IV] and [VI]. Additionally, paper [III] elaborates on the possibility of not finding any

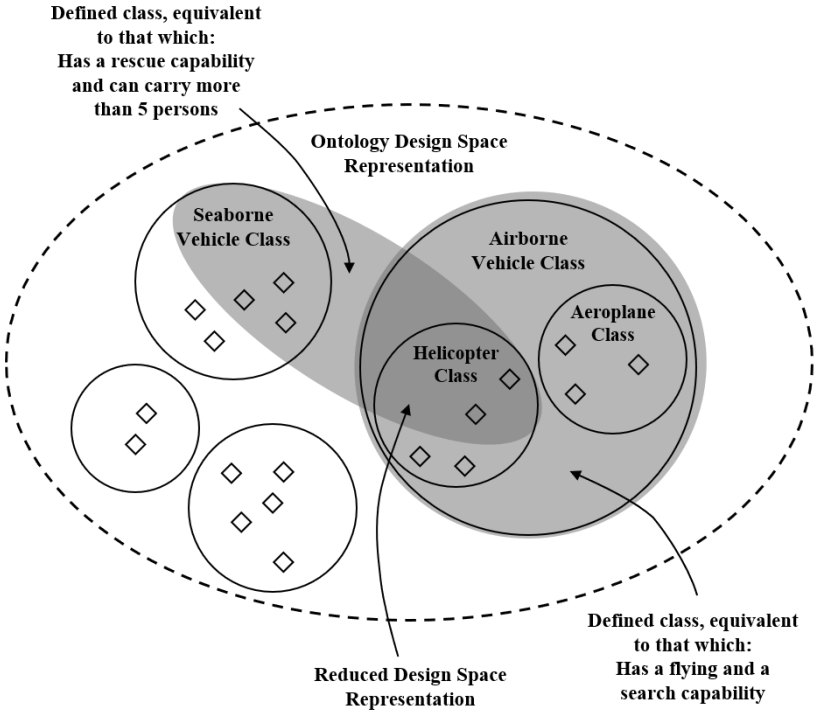


Figure 3.8 *An illustration of an ontology represented design space with defined classes marked with grey backgrounds. Classes are represented by circles and individuals by diamonds.*

solutions for the conditions in the *defined* classes, and how these conditions can consequently be used as inputs for a continued design process. This is also further discussed in section 3.4.3.

Design Space Expansion

The reasoner can also be used to expand the available design space using implicit relationships in the ontology representation. Examples of this have been both shown and discussed in the case studies of appended papers [I], [II], [III], [IV] and [VI]. For example, a new entity can be added to the ontology with only one property stating that it has the capability to fly. The reasoner can then infer that this new entity must be some sort of airborne vehicle, since these have the capability to fly. The new entity can consequently be regarded as belonging to the airborne vehicle class and will thereby inherit the airborne vehicle class properties. These may be properties stating that all airborne vehicles have a weight, cruise speed, average fuel consumption, fulfilment of certain functions,

and more. The reasoner can thereby infer and add more information to the newly added entity automatically. Consequently, the newly added entity can also be regarded as a solution to the conditions of *defined* classes in a design space reduction context. However, using the reasoner to expand the ontology and infer implicit relationships adds to the cost in terms of computational time, as described in section 2.3.

The case study in paper [III] also shows how an initial ontology representation created from the architecture framework approach can be expanded with additional information before a design space reduction is performed. Implicit relationships are also used here to sort ontology classes into the modelled class hierarchy. Finally, paper [VI] describes and illustrates how *defined* ontology classes can be used to represent aircraft concepts as selections of different design alternatives to required functions within the ontology. Reasoning has thereafter been used to automatically identify suitable datasets of similar existing solutions based on the specific concept selection. These available datasets have here been represented as ontology individuals that the reasoner has subsequently related to suitable concept individuals during reasoning. This has resulted in additional information, and a suggestion for a continued estimation procedure, about the aircraft concept in question. More specific details about this are found in the case study of paper [VI]. Additionally, section 3.5.1 further explains how this information can be used to expand and initially estimate aircraft concept characteristics outside the ontology representation.

3.4.3 Summary and Ontology Approach Outcome

Overall, the ontology and reasoning approach presented in this section results in a reduced design space of alternatives and functions to be performed in order to meet overarching needs and capabilities. The approach has been used to give suggestions for both suitable systems and SoSs. The case studies in papers [II] and [V] have shown this by integrating the ontology approach with Agent-Based Simulations (ABS) to generate, reduce and evaluate an SoS design space. In paper [II], different combinations of existing SAR assets were generated from the ontology and then simulated to give the different SoS performances in terms of overall mission time and cost to find a rescue subject. Visual Analytics (VA) was thereafter used in paper [V] to create an interactive dashboard from the previous results which were used to explore various aspects of the simulated SoSs and environmental conditions. Figure 3.9 shows an overview of the process used in papers [II] and [V]. More details about this process are found in the methods and case studies of the corresponding papers.

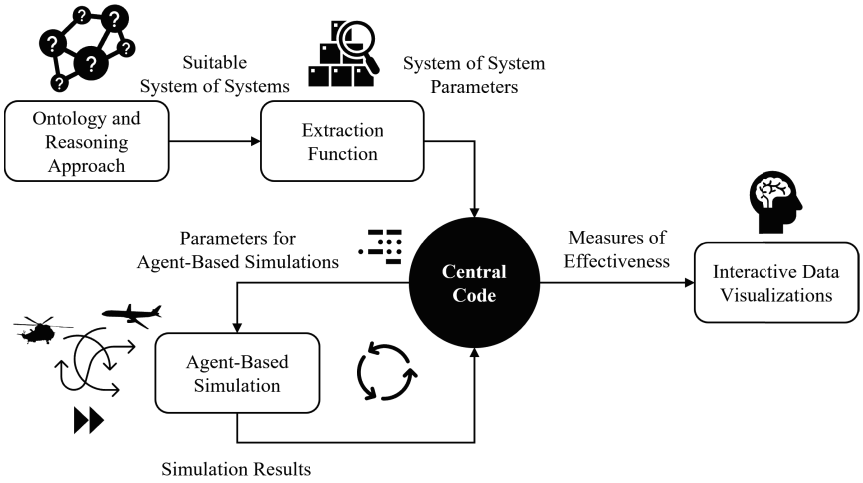


Figure 3.9 *The process used to connect ontology with Agent-Based Simulations (ABS) in paper [II]. The interactive data visualizations step was added in paper [V] to also include Visual Analytics (VA) in the process.*

Finally, Paper [III] shows how the ontology approach is used to give a reduced design space of functions to be performed by new systems to be developed. The suggested functions from the ontology approach are here inputs for a continued design process that is used to create new systems from an SoS perspective. This has been the main focus for papers [IV], [VI] and [VII] that also have been the foundation of the concept exploration and estimation approach described in the next section.

3.5 Concept Exploration Approach

There are several ways of continuing the design process and exploring system concepts from the functions and alternatives suggested by the ontology and reasoning approach from the previous section. Qualitative system representations may, for example, be chosen and created directly in the ontology as individuals by selecting design alternatives for each required function in their definition. However, this becomes more difficult when the number of functions, alternatives and interdependencies starts to grow. Ontologies are also limited in terms of numerical calculation capabilities, as mentioned in section 2.3.2. The estimations that can be performed on the characteristics of new Constituent Systems (CS) properties are therefore also limited. A temporary transition outside the ontology representation is consequently necessary in order to enable further analyses for new CS, such as ABS. Knowledge extraction from ontology representations has been investigated in several of the appended

papers. As mentioned before, paper [II] has shown how the knowledge in an ontology can be used to define agents for ABS based on existing solutions. Papers [IV] and [VI] have, on the other hand, illustrated how new CS solutions can be explored, and how their characteristics can subsequently be estimated. Figure 3.10 shows an updated version of the concept exploration and estimation approach first introduced in paper [VI]. Figure 3.10 thereby also shows a more detailed view on the *concept exploration and selection* and *estimation of characteristics* steps from the overall method in Figure 3.2.

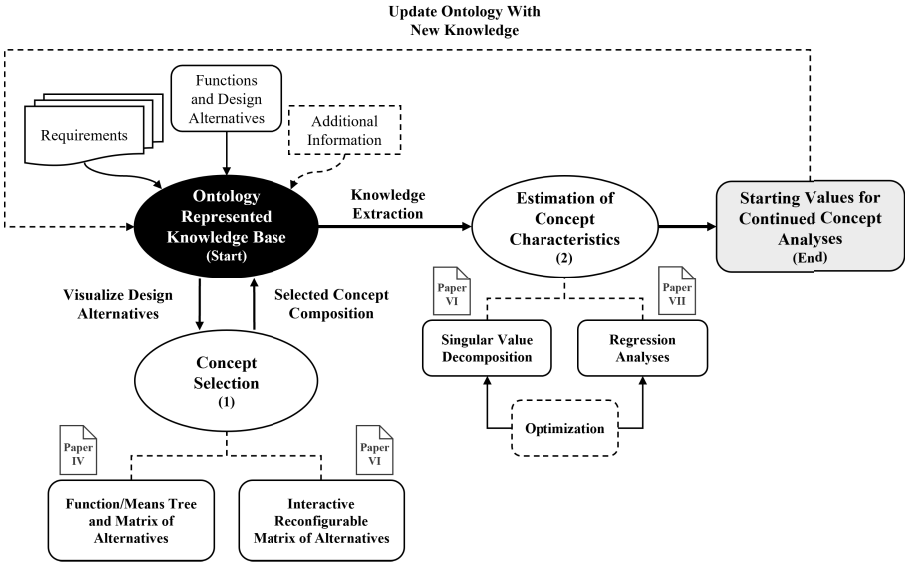


Figure 3.10 *An approach to explore concepts from an ontology represented knowledge base. The figure also shows how different selection and estimation techniques have been used in the appended papers.*

As Figure 3.10 illustrates, the approach starts with an ontology that is used as an overarching knowledge base of represented entities and relationships. These are, for example, the functions to be fulfilled and the different design alternatives to do so. However, the ontology can also include information about relevant requirements and additional considerations, such as environmental aspects or details about suitable continued design processes. Reasoning can from this point be performed to narrow down the design space as shown in the previous section. The resulting design space for concept selection can be visualized in different ways. Paper [VI] utilized an Interactive Reconfigurable Matrix of Alternatives (IRMA) for this step while paper [IV] showed a more traditional approach using an F/M tree and a conventional matrix of alternatives. Selected concepts are thereafter reintroduced and defined in the ontology as compositions of design alternatives for each function. A temporary transition

outside the ontology must subsequently be performed to estimate quantitative characteristics of the selected concepts. Knowledge, in terms of concept compositions, requirements and suggested estimation techniques, is extracted and used for further analysis. The quantitative estimations of concept characteristics have been based on statistics of existing solutions in this dissertation. Papers [VI] and [VII] have illustrated this using different techniques as seen in Figure 3.10.

Ultimately, the concept exploration and estimation approach ends in initial approximations of constituent system solutions that have been calculated outside the ontology representation. The approximations can thereby be seen as starting values for more detailed continued concept analyses, such as an initial sizing. These first approximations can from this point also be reintroduced to the ontology representation again, thereby expanding the original knowledge base with additional details. Further investigations, such as additional ABS, can thereafter be performed with the newly estimated concepts in an SoS context. This could, for example, be done with a reiteration of the process in Figure 3.9. The remainder of this section will elaborate more on the approach in Figure 3.10 and how the parts for concept selection and estimation of concept characteristics are used.

3.5.1 Concept Selection

A variety of methods can be used in concept selection for new system solutions. At this stage, a selected concept would correspond to a composition of design alternatives without any quantitative characteristics, such as performance metrics. There are different ways of representing the design space for concept selection in terms of design alternatives for required functions.

Function/Means Tree and Matrix of Alternatives

A traditional approach in early product development is to visualize the different functions to be fulfilled with the available alternatives in an F/M tree. This creates a hierarchical structure of functions and alternatives that can subsequently be used for concept selection in a morphological matrix, or matrix of alternatives. The case study in paper [IV] illustrates how functions and alternatives from an ontology can be visualized in an F/M tree for facilitated decision support. The F/M tree has also been used for a continued functional breakdown in the case of paper [IV]. Reasoning was here used to suggest suitable vehicle system types for implementing a search function. Airborne vehicles were inferred as one means for implementing the search function, which consequently implied that such vehicles had to fulfil the main functions described in [131] for airborne vehicles. Each function in the large ontology from paper [IV] was mapped against suitable alternatives from the included aircraft design ontology and the SAR ontology that it was originally based on. This mapping was done based on information found in the International Aeronautical and

Maritime Search and Rescue (IAMSAR) manuals, [122]–[124], and Daniel P Raymer's book "Aircraft Design – A Conceptual Approach", [106]. The mapping was then continued until the lowest level aircraft functions were assigned corresponding means. Consequently, both constituent system and sub-system level means were included. Each means can have one or more design alternatives. This was, for example, illustrated in paper [IV] with a *provide power* function that could be fulfilled by several different power plant configurations. Alternatives for the mounting location of the power plant configurations were also included based on the guidelines in [106]. The functional breakdown and F/M tree represented design space were finally used to create a matrix of alternatives. The matrix represented a small part of the overall design space and illustrated how different functions, means and their design alternatives could be visualized for facilitated concept selection. More details and visualization on the F/M tree and matrix of alternatives are found in the case study of paper [IV].

The concept selection methods discussed so far have solely been visualizations of the knowledge represented from an ontology model in order to give a better overview of the available design space. However, these more traditional product development methods provide little information on, for example, compatibilities and fulfilment of requirements. More advanced methods are therefore favourable in the vast design spaces that SoSs span.

Interactive Reconfigurable Matrix of Alternatives

A more elaborate method in concept selection is to use an Interactive Reconfigurable Matrix of Alternatives (IRMA). As described in section 2.4.2, an IRMA builds upon a matrix of alternatives but includes additional aspects such as compatibilities between design alternatives and requirements. An IRMA is also interactive, which allows designers to investigate and visualize how different combinations of design alternatives reconfigure and prune the available design space. Concept selection using an IRMA was carried out in the case study of paper [VI]. The IRMA was, in this case, generated from knowledge in an ontology model which described both functions, means and design alternatives with corresponding compatibilities for airborne vehicles. The matrix itself was generated based on the implementation work described in [133]. Concept selections could thereafter be performed in the IRMA by interactively selecting means or alternatives for each represented function. Incompatibilities between alternatives were actively highlighted depending on the designer's decisions. An additional possibility with an IRMA is to show, for example, relationships between design alternatives and requirements. Studies such as [97] illustrate how certain engine types for an aircraft concept, for example, are incompatible with cruise speeds above certain Mach numbers. However, such requirements were not included in the case study and IRMA of paper [VI]. Instead, the matrix only acted as a way of facilitating concept selection with the functions and alternatives. Based on the interactive visualization of the design space

in the IRMA, different concept selections could be performed. The selected concepts could then be represented as individuals within the ontology. Consequently, these individuals included relationships to all required functions and their selected design alternatives from the IRMA. Moreover, some relationships to overarching requirements were also specified in the definition of the individuals, which in the particular case of paper [VI] were requirements on range and the number of passengers to be carried by airborne vehicles. From this point, the process in Figure 3.10 may be continued with quantitative approximations of the various concept individuals generated.

3.5.2 Estimation of Concept Characteristics

The concept individuals created with the process in Figure 3.10 so far only consist of a set of design alternatives and basic information, such as requirements, represented in the ontology model. However, further estimations must be performed to enable evaluations of these concept individuals in an SoS context. As mentioned earlier, the description of concept individuals needs to be extracted from the ontology for quantitative estimations and numerical calculations. Selected concepts with estimated characteristics can thereafter first be used in, for example, ABS to obtain a measure of their performances in an SoS context. Extraction of information from an ontology can be done in several ways. OWL ontologies can, for example, be navigated as files in an Extensible Markup Language (XML) format. Relevant instances representing existing systems or concepts can be associated with a dedicated class so that XML navigation is facilitated. Papers [II], [V] and [XVI] have all shown how information from an ontology can be extracted using different tools and methods.

Estimations of concept characteristics or properties can be made in several ways. However, the techniques shown in this dissertation have solely built upon statistical data of existing solutions. This has been done to achieve "ballpark" estimates, which are usually sufficient at the very beginning of a design process and in the evaluation of possible initial concepts in an SoS context. From an analysis perspective, such low-fidelity models provide cheap estimations from a computational point of view but at the cost of reduced accuracy, as mentioned in section 2.2.3.

Singular Value Decomposition

The case study in paper [VI] illustrated how a Singular Value Decomposition (SVD) analysis could be performed on a dataset of existing passenger aircraft to estimate characteristics. Ontology and reasoning were again leveraged to automatically sort generated concept individuals into different statistical categories within the ontology. This was done using *defined* classes that implicitly described the needed design alternatives for a specific category. For example, the dataset of existing passenger aircraft was represented as an ontology individual that could be inferred as a suitable dataset for continued analysis

if certain design conditions were fulfilled by a concept individual. The case study in paper [VI] showed this by automatically classifying an aircraft concept individual with a relationship to the passenger aircraft dataset. A similar classification was also done for a helicopter dataset introduced in paper [VII], which implied that individuals corresponding to a rotorcraft would be inferred with this as a suggested dataset for additional investigations. The SVD analysis in paper [VI] was performed in an Excel-implemented macro that made the matrix factorizations and ultimately gave an SVD model with determined SVD variables. As described in section 2.4.3, the number of SVD variables can be reduced from here so that only the most influential ones remain with respect to a desired uncertainty, against a reference from the statistical dataset used. This allows the model to estimate all parameters within a dataset based on only a few inputs. The number of SVD variables in the case study of paper [VI] was reduced so that the previously mentioned requirements on range and passengers could be used in combination with a minimization of the Maximum Take-off Weight (MTOW). These three inputs could therefore be used to estimate all other characteristics represented in the dataset for the aircraft concept individual, such as wing area, thrust and take-off distance. As discussed in paper [VI], the results from the SVD analysis can act as inputs for more detailed sizing procedures, for example the ones suggested in [106] or [134]. However, the results can also be used to generate a first geometry of the aircraft concept, as also shown in the case study of paper [VI]. Overall, the SVD analysis results in expanded information about a selected concept individual's properties. From here, the resulting estimations can be used for further analysis or be reintroduced to add more information to the original ontology model and knowledge base, as illustrated in Figure 3.10. Naturally, more details and illustrations of the SVD analysis and its connected parts are found in paper [VI].

Regression Analyses

A different, and perhaps more detailed, analysis for concept characteristic estimations was investigated in paper [VII]. This particular paper focused on early design estimates of helicopters based on a statistical dataset. Ontologies were not included in the work, but the concept selection and dataset suggestion techniques shown previously still apply in this case as well. The case study also included the same requirement types of range and passengers as in paper [VI]. The purpose of the case study in paper [VII] was to illustrate how relatively little detail about a design and domain could be used to create optimization frameworks for initial concept estimations based on models derived from statistics. The study showed how different methods, such as linear and exponential regressions, could be used to obtain an estimation model for determining required parameters. Symbolic regressions with genetic algorithms were also investigated in paper [VII]. This was used as a means of performing regression analyses between variables that did not have a clear correlation. The regression analyses finally resulted in a number of estimation models for

different helicopter design disciplines, such as aerodynamics and propulsion. The obtained discipline models could thereafter be interconnected in an optimization framework to enable design space explorations and trade-off analyses between, for example, requirements.

Optimization

Optimization was used in the case studies of both papers [VI] and [VII]. In paper [VI], optimization was used to adjust the SVD variables so that the desired characteristics, or requirements, for the aircraft concept individual were met. This optimization was performed using a Generalized Reduced Gradient (GRG) method, where the objective was to minimize MTOW while keeping the range and passenger capacity at the values given by the requirements. The design parameters were consequently the used SVD variables. Paper [VII] utilized optimization in the case study at two different points. First, optimization was used in the symbolic regression analysis to obtain expressions for the relationships between different helicopter variables. As described in section 2.4.3, a genetic algorithm is a suitable optimization method for symbolic regression. It was consequently also used to perform all symbolic regressions in the case study of paper [VII]. Finally, optimizations were used to obtain a design space of different helicopter designs calculated from the various discipline models. This was done through the optimization framework mentioned before which was created based on a process outlined in paper [VII]. Figure 3.11 illustrates an updated version of this process for creating optimization frameworks for early design studies.

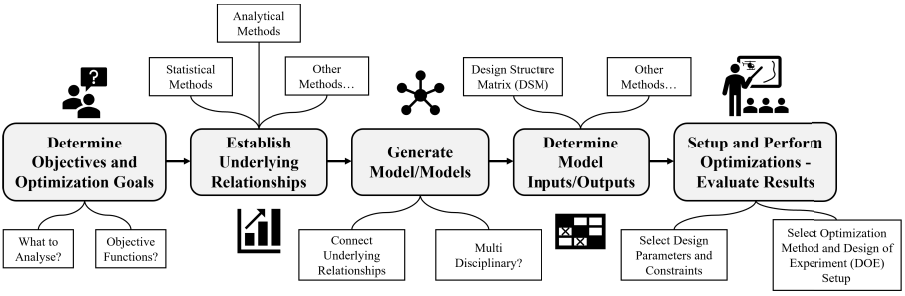


Figure 3.11 *The process and workflow for creating optimization frameworks which was introduced and followed in the case study of paper [VII].*

The process in Figure 3.11 essentially starts with determining the goal of the analysis and deciding on the desired objectives for the subsequent optimization. Once initial goals and objectives are determined, underlying relationships of the domain in question need to be established. This can be done using statistical data from existing solutions or with other methods such as well-established analytical formulas for the specific design field in question. The overall purpose of

the step is to define the relationships between the optimization objectives, system characteristics, and the possible design variables to be used. The obtained relationships can thereafter be connected to each other in different calculation models. One possibility is to divide the generated models based on different disciplines within the field of study. This is illustrated with a "multi-disciplinary" option in Figure 3.11. A benefit of dividing models into different disciplines can be that some can be exchanged for higher fidelity ones in future calculations if needed or desired. Each generated model will have required inputs and outputs. The connections between model inputs and outputs must be determined in a Multi-disciplinary Design Optimizations (MDO) case. Structuring techniques, such as a Design Structure Matrix (DSM), can help in this specific task by providing an overview of the different models' inputs and outputs. However, this can also be done using other methods, as shown in Figure 3.11. The case study in paper [VII] illustrates how both an input/output matrix and a DSM can be used to illustrate the required connections between involved models. Once a model structure is in place, design variables and constraints can be determined based on the previously established objectives and goals. The optimization itself can thereafter be set up and performed with desired optimization and design of experiment, or design space sampling, methods. Additionally, the steps of the process just described are not necessarily intended to be followed in a strict order. Certain steps might need to be revisited throughout the process as more knowledge about a design is gained. Already determined design parameters and constraints can also, for example, be added at the very beginning of the process. It is consequently also possible to define additional objective functions from the system characteristics throughout the process.

The workflow outlined in Figure 3.11 was used to create an optimization framework for early helicopter design in the case study of paper [VII]. It thereby illustrates how optimizations and relatively few details can be used to make early estimates of characteristics for investigated concepts. Finally, the case study also shows and elaborates on how different visualization techniques can be used to illustrate the obtained Pareto fronts between optimization objectives and thereby also the available design space and existing design trade-offs. For example, a sensitivity matrix between design parameters and objective functions was used to illustrate the relationships within the obtained design space. Similarly, a symmetric system characteristics correlation matrix was used to display the dependencies and available trade space for the different optimization objectives. These, and more design space visualizations, are found in the case study of appended paper [VII].

3.5.3 Outcome and Summary

Overall, the approach in Figure 3.10 results in estimated concepts with calculated characteristics or properties. These can form here be used as starting values for continued concept analyses. The outcome may also be reintroduced to the underlying ontology model for further processing with, for example, rea-

soning as described earlier. Investigations similar to those conducted in papers [I] and [II] can thereafter be performed to give suggestions on suitable SoS architectures that include the newly estimated CS concepts. This is also illustrated in the overall method from Figure 3.2. Consequently, ABS can be used to evaluate the performance of each such suggested SoS architecture. The most prominent SoS will thereafter be used for further analyses, such as a continued and more detailed sizing of the involved CS concepts.

The concept exploration and estimation approach described in this section illustrates how different aspects of the design and trade spaces for the constituent system and sub-system levels in Figure 3.1 can be explored. However, the complexity of these explorations increases once they are considered together with the remaining levels from Figure 3.1. The next part of this section will describe how such aspects can be visualized to facilitate decision support and understanding in holistic design and trade space explorations over the different levels in Figure 3.1.

3.6 Visual Analytics Approach

The final part of the overall method in Figure 3.2 illustrates how Visual Analytics (VA) and interactive decision support can be used as an overarching scope on all involved steps. This can consequently also allow for interactive design and trade space explorations over several aspects of the holistic SoS design model from Figure 1.2. VA was first introduced in paper [V] as a means for exploring the results of different Agent-Based Simulations (ABS) interactively. The case study was based on the SAR investigations from paper [II], and allowed the influence of factors such as environmental conditions and different search tactics to be interactively explored. Additionally, the level of fidelity for an involved sensor model could be altered in order to interactively explore the influence of the chosen sensor model fidelity level. Paper [V] also introduced a suggested workflow for creating interactive visual representations for enhanced decision support in data explorations. This process is reproduced in its original form in Figure 3.12.

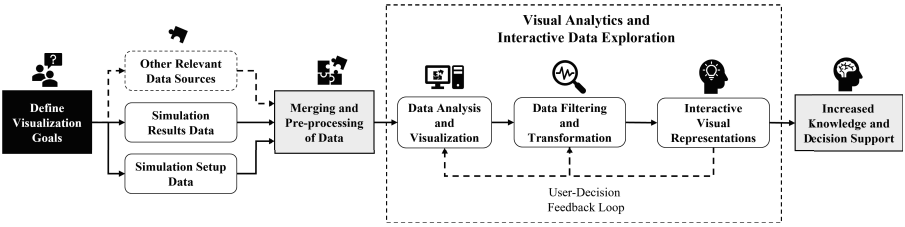


Figure 3.12 *The overall Visual Analytics (VA) approach and workflow for creating interactive visualizations for enhanced decision support. First introduced in paper [V].*

The workflow in Figure 3.12 is based around the guidelines and methodologies for VA described in [116], [117] and [118]. Overall, the workflow starts with the definition of visualization goals and what questions the visualizations are intended to answer. A data pre-processing step is thereafter performed in order to prepare the underlying data to be visualized. As illustrated in Figure 3.12, the data to be visualized can come from different simulation parameters and results. However, this can just as well come from other data sources, such as knowledge from an ontology or an architecture framework. Regardless of the data source, certain pre-processing procedures might be required. These may include cleaning, merging/integration and transforming procedures. Once the data to be visualized has been prepared, the initial data analysis and visualization can be performed. Naturally, this is closely related to the initially determined visualization goals. Different visualization techniques might be of interest depending on the specified goal. This in turn implies that different types of data analyses are required to achieve the intended visualization. As a general aim with VA is to facilitate human reasoning and decision support, a data filtering and transformation step is hereafter performed. This allows the underlying data to be interactively filtered and reduced to a comprehensive level which, for example, fits the current exploration and analysis objectives. However, such filtering should never alter the underlying pre-processed data. It must therefore always be possible to revert back to the original data representation and visualization if desired. As seen in Figure 3.12, the interactive elements of the approach result in a *User-Decision Feedback-Loop* which essentially means that the resulting visualizations are updated based on, for example, different filtering decisions. The end results of the workflow outlined in Figure 3.12 are enhanced knowledge and decision support capabilities, through the interactive data explorations that have been used to meet the goals of the intended visualizations. More details of the approach in Figure 3.12 can be found in paper [V].

3.6.1 Design Space Visualizations and Interactive Dashboards

So far in this dissertation, an Interactive Reconfigurable Matrix of Alternatives (IRMA) has been mentioned as one interactive visualization technique for illustrating an available design space in paper [VI]. The interactive elements of the matrix allow designers to explore different alternatives for implementing required functions. In this particular case, an ontology has acted as the underlying data source from which relevant information has been extracted for the visualization. Individual interactive visualizations can facilitate decision support and exploration of data. However, such visualizations can also be bundled and represented in a collective manner through an interactive dashboard. The case study in paper [V] illustrates how such a dashboard, with many interactive elements, can be created using the process from Figure 3.12. Several visualization goals were specified here and later realized by merging data from numerous ABS runs on different SoSs under various environmental conditions and oper-

ational scenarios. The dashboard included result graphs, charts and filters, all of which could be used in an interactive manner. As mentioned previously, the dashboard was, among many other things, used to explore the influence of a sensor model fidelity. An interactive dashboard can consequently act as a result exploration platform that can facilitate decision-making and the understanding of a specific topic, such as an available design space. Filters may be included and used to temporarily focus an analysis on specific parts. However, the underlying data remains unchanged and new analyses can be performed at any time.

3.6.2 Interactive Design and Trade Space Explorations

VA with, for example, interactive dashboards is an enabler of facilitated design and trade space explorations on the various levels of an SoS design model. A VA-based approach is consequently an overarching and common denominator, as seen in Figure 3.1. Analyses such as "what-if" investigations can give designers and decision makers an overview of various aspects that influence and traverse the different levels. Paper [V] has so far illustrated how an interactive dashboard can aid in an analysis and exploration at an SoS design space level from ABS results. However, aspects from all levels can be represented in a dashboard to allow for more holistic analyses. This also applies for all other parts of the method in Figure 3.2. Figure 3.13 illustrates an example of a dashboard based on an underlying ontology representation of SoS architecture components, such as different SAR assets.

The dashboard in Figure 3.13 builds upon an updated version of the ontology from paper [II] where the corresponding ABS results have been added to the knowledge base. The Microsoft Power Business Intelligence, or Power BI [135], software was used to perform the different steps of the VA process from Figure 3.12 and to create the dashboard in this case. The dashboard itself can be used for a variety of analyses. Capabilities can, for example, be filtered so that only solutions that meet them are displayed. Peaks in performance graphs can also be interacted with and used as filters to quickly identify the corresponding solutions. This can thereby also highlight suitable SoSs with their involved Constituent Systems (CS). An average water temperature slider can be interacted with and used to filter out solutions that are unable to detect a rescue subject within the expected survival time. Finally, the leftmost network graphs allow decision makers to trace the relationships between ontology classes and individuals in an interactive manner. This can, for example, be used to illustrate what CS a specific SoS is composed of, and in turn what sub-systems each CS have. Dashboards like the one in Figure 3.13 can also include aspects from architecture frameworks and optimizations at CS and sub-system levels. An available design and trade space spanned by an optimization, similar to the one in paper [VII], could be represented in an interactive dashboard, for example. More discussions on how VA could be used to explore additional SoS aspects are found in chapter 4.

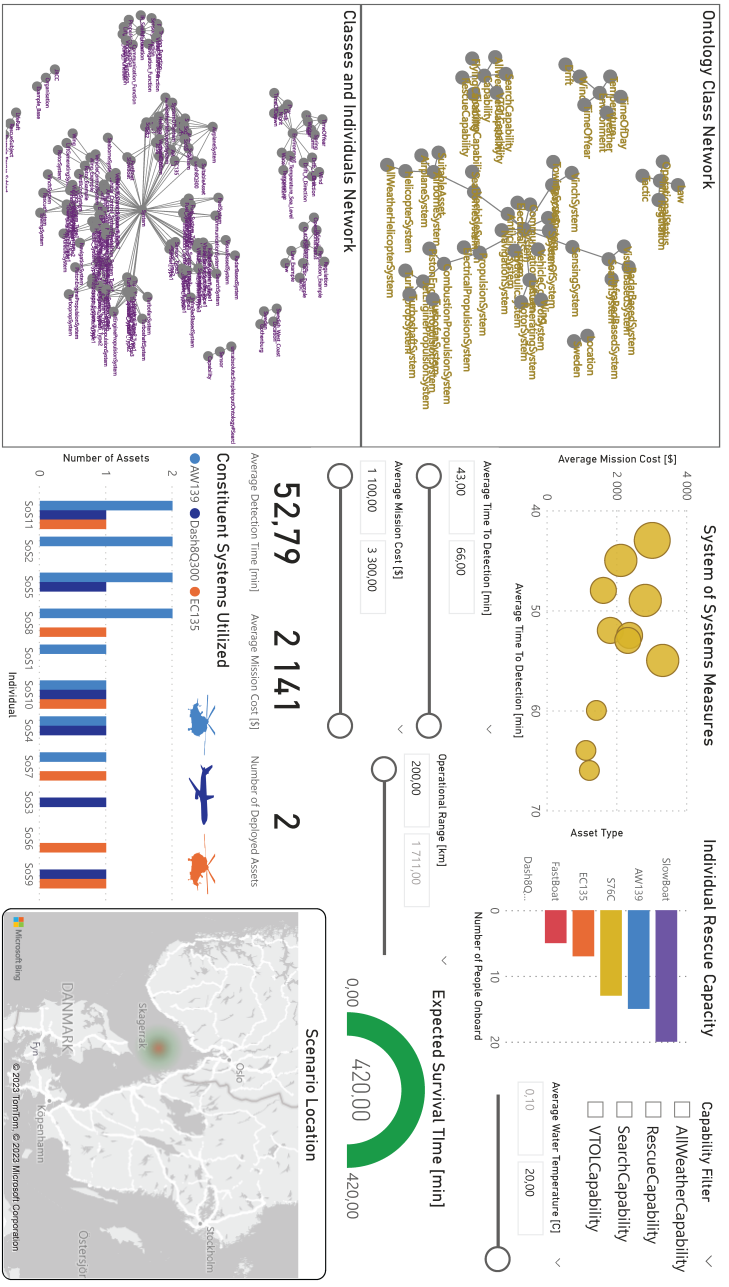


Figure 3.13 *An interactive dashboard based on an underlying ontology and knowledge base.*

4
Discussion

Ontologies have been a central part of the performed work. The case studies of the appended papers have shown that ontologies with description logic reasoning capabilities are suitable options for generating and processing the design spaces at different levels of a holistic SoS analysis. The ontology and reasoning approach is consequently applicable at all levels of the holistic design model in Figure 1.2. Ontologies act as overarching knowledge representations and can thereby be used to represent any aspects of an SoS. These may be different operational details, environmental conditions and time aspects as shown in paper [I]. However, the ontology can also be used to represent functions and design alternatives for both SoSs, Constituent Systems (CS) and sub-systems, as illustrated by the case studies in most of the appended papers. Reasoning can be leveraged to process an available design space in terms of both reductions and expansions. This dissertation and the appended papers have also shown many additional examples of how reasoning can be used to, for instance, give suggestions for suitable SoS solutions, and data sets to be used for continued approximations of selected CS concepts characteristics. However, these investigations have only scratched the surface of the many possibilities that reasoning can bring in a design process.

Initial design space explorations can be performed using reasoning and the ontology approach as well. This is not explicitly shown in the case studies in the appended papers. However, paper [I] both shows and provides discussions on how several *defined* ontology classes, describing different conditions and user-defined needs, can be created for the purpose of finding persistent system solutions. The different *defined* classes will consequently result in various reduced design space representations where, for example, instances with the highest reoccurrence can be identified. It is thereby possible to dynamically investigate how changes in the *defined* ontology classes, such as varying requirements or environmental conditions, will affect the solution space provided by the reasoner. The same idea can be applied to CS represented in the ontology model. Additional functions can be added to different CS that subsequently

might be inferred as new viable solutions to queries in *defined* classes that they could not fulfil earlier. This can therefore be used to explore the design space represented in the ontology and also give a view of the sensitivity and connectivity between modelled entities and relationships.

Parallels can also be drawn between set-based design and the ontology and reasoning approach. *Defined* ontology classes can represent a set of possible solutions in a reduced design space that can always be adjusted if desired or needed. Consequently, ontologies allow for a flexible representation of a design space, and more knowledge about design solutions and options can be added at any time due to the underlying *open world* and *non-unique naming assumptions*. As mentioned in section 2.4.1, design space explorations are performed to find the best solution to a given design problem. Consequently, the design space explorations that can be performed with the ontology approach are labelled as initial, since more considerations, such as numerics, must be included. Currently, the approach only gives suggestions for suitable solutions that fulfil described conditions, and not specifically for single optimal solutions. A continued evaluation process using simulation techniques like Agent-Based Simulations (ABS) is one possible option for indicating the effectiveness of different SoS solutions, as shown in papers [II] and [V].

Moreover, paper [I] explains how the ontology approach can be used on three different SoS types from a time perspective. The first is a near-term SoS perspective, where only existing solutions and how best to use them are investigated. Investigations of existing solutions have been the main focus of papers [I] and [II]. A mixed SoS perspective includes both existing and new solutions to be developed. Paper [III] shows how the approach is used to give a reduced design space of functions to be performed. Paper [III] also mentions how existing solutions can be modified to meet certain functions and then collaborate with other existing solutions to create a valid SoS. In this sense, a new system does not have to be created. A long-term SoS perspective corresponds to one where the solutions are yet to be developed. Here, the functions to be performed must be populated with different design alternatives as shown in papers [IV] and [VI]. Possible concepts can thereafter be selected to give a qualitative representation. However, estimations of quantitative concept properties are performed on extracted knowledge in the presented work due to the limited numerical calculation capabilities of ontology representations, as established in section 2.3.2. It is possible that some initial calculations can be performed directly in an ontology through rule languages such as the Semantic Web Rule Language (SWRL) and the Semantic Query-Enhanced Web Rule Language (SQWRL). This could allow the overall cost of an SoS with existing solutions to be calculated at an early stage, which could be used to rule out overly expensive solutions through reasoning. Surrogate models could possibly also be represented with rules which would allow for early explorations of performances for different system solutions. Meta-ontology structures were mentioned in section 2.3.1, together with top-level ontologies. A meta-ontology structure, where

relevant ontologies can be dynamically activated, would be beneficial from a scalability perspective for the design space processing using the reasoner. The design space could consequently be kept at a manageable level, where irrelevant information does not have to be processed by the reasoner. This could reduce the computational effort in large ontology models. Additionally, the theory of information entropy might be a convenient way of quantifying the size of a design space [87]. This could be used in design space processing with a reasoner to give numbers on the performed reduction and expansion operations. Information entropy is also seen as a possible complement in quantification of knowledge in ontology models, whose sizes typically are measured through the number of axioms present.

The ontology process in Figure 3.5 and section 3.4.1 has only been used with the Protégé ontology editing software ([128]) in the case studies of the appended papers. However, the ontology process and approach should be applicable in any tool that supports OWL and description logic reasoning, as discussed in paper [I]. There are also more methods for building ontologies that would result in similar knowledge representations to those generated by the approach in this dissertation. It is possible that the outcome of the ontology process in Figure 3.5 can be obtained with UML or SysML models. However, the benefit of having the knowledge represented in an ontology is, for example, that a reasoner can be used to automatically handle the complex relationships that may exist in even relatively small and low-fidelity representations, as discussed in paper [III].

The architecture framework approach was used to break down a set of SoS needs into subsequent functions to be performed. It has thereby been used to traverse the two leftmost levels of the holistic design model in Figure 3.1. However, the approach has only incorporated a selection of viewpoints from the Unified Architecture Framework (UAF). It is possible that more levels can be covered with the approach if additional viewpoints are included. The UAF, and other architecture frameworks, include many more considerations and viewpoints than those described in section 3.3.1 and more widely in this dissertation, such as the use of existing resources and how best to use them from a Concept of Operations (CONOPS) perspective. However, the goal of the architecture framework approach has, as previously mentioned, been to provide a consistent and standardized way of breaking down the needs into required capabilities and functions. Consequently, the viewpoints used have been deemed sufficient for this purpose. It is also possible to represent an architecture framework through an ontology. This would thereby allow it to be processed by a reasoner which could facilitate navigation in the large network spanned by the different viewpoints. A knowledge representation like that could also be used to create interactive visual elements similar to the Interactive Reconfigurable Matrix of Alternatives (IRMA) in paper [VI]. Such a matrix could, for example, interactively show the dependencies between the various viewpoints of an architecture framework.

As mentioned in section 3.3.2, the outcome of the architecture framework approach results in a design space of functions to be performed that can be used as inputs for a continued design process. The overall structure of the approach in Figure 3.4 is similar to the one used in a Function/Means tree (F/M tree). A continued functional breakdown like the one presented in paper [IV] is therefore a suitable continued step. Additionally, capabilities can be seen as special cases of functions, and can therefore be treated as the top-level, or main, functions in an F/M tree. In this case, the activities are the means by which to implement the capabilities. Another important aspect in a breakdown using the architecture framework approach is the different scenarios in which a complex system or SoS is expected to be involved. Different scenarios can have new or different stakeholders and corresponding needs, which in turn might require additional capabilities and functions to be fulfilled. The design space of functions to be performed could be explored by investigating different possible scenarios using the approach. Reoccurring capabilities and functions can thereby be identified and considered as the most important ones to fulfil in order to cover a range of different possible scenarios. Investigations of CONOPS were beyond the scope of this dissertation. However, another interesting topic would be to investigate capabilities and corresponding operational activities through an F/M tree. This could illustrate a design space of different CONOPS architectures, which could be used to fulfil the required capabilities by combining operational activities in different ways. This can be regarded as a future project, on which chapter 6 will elaborate.

The case studies of papers [IV], [VI] and [VII] have shown various ways of selecting new system concepts to be developed and how their characteristics can later be estimated. However, these estimations have only been performed based on statistical analyses of existing solutions. More novel solutions with little or no available data might therefore be harder to approximate. The statistical analyses performed have also only resulted in starting values for continued evaluations. One possible option from here would be to continue with more sophisticated estimation techniques and simulations for an individual system, such as Computational Fluid Dynamics (CFD), to estimate more details and parameters. An MDO framework could, for instance, thereafter be used to connect different discipline models and ultimately span an updated design space of solutions for a new concept using optimization. Visual Analytics (VA) techniques, such as the dashboards presented in this dissertation and paper [V], could then be used to provide decision support on the optimization results through interactive design and trade space exploration capabilities. Additionally, the estimation models in an optimization or Multi-disciplinary Design Optimizations (MDO) framework can also be exchanged for models with other fidelity levels if needed. This could, for example, be done using standards such as the Functional Mock-up Interface (FMI) and System Structure and Parameterization (SSP). Such substitutions possibly also apply to ABS, where individual agents could be represented by a corresponding co-simulation model.

This could consequently allow for ABS with mix-fidelity models of the involved CS or scenario entities. This would however increase the computational cost in terms of time, especially when considering the high dimensionality of SoS problems. As illustrated by papers [VI] and [VII], data-fit models of low-fidelity can be obtained through estimation such as SVD or symbolic regression. Consequently, surrogates of high-fidelity models can also be obtained through the techniques just listed, for example to reduce the computation effort. Neural networks and machine learning algorithms could also be used for this purpose. However, these have been outside the scope of this dissertation and are generally less transparent than the aforementioned techniques.

The visual analytics approach has illustrated how interactive investigations can be performed on various aspects of the holistic SoS design model. VA is in general considered as an overarching scope for all aspects, but this has not been shown in this dissertation or the appended papers. As discussed earlier, the VA approach and interactive visualizations, such as an IRMA, could be used on architecture frameworks to facilitate understanding of the various viewpoints. Interactive dashboards that include elements from all parts of the method in Figure 3.2 would enable the full holistic analyses envisioned in [1].

To conclude, the overall method in Figure 3.2 shows one way of approaching the holistic design model from Figure 1.2. However, this model has no specific starting point and the analysis order does not necessarily have to follow the one illustrated in this dissertation. An analysis could, for example, start with the inclusion of a new available sub-system. This could in turn be used to estimate the performance of a CS that incorporates the new sub-system. The CS can subsequently be evaluated through ABS to investigate whether, for example, additional capabilities can be obtained and so on. The case studies of the appended papers have also only been based around Search and Rescue (SAR) and this has been a delimitation of the dissertation, as described in section 1.3. The approaches presented are, however, applicable to any SoS and have also been kept general for this purpose. Another SoS case study would generate a dissimilar set of needs than those listed in section 3.3. The architecture framework approach would, however, be the same, as the identified relationships between needs, capabilities and more are not case-study-specific. There are also more approaches and initiatives for analysing SoSs that may be used to approach the aim and research questions of this dissertation that have not been investigated so far. The presented method does, however, provide some insight into how early product development can be approached from an SoS perspective, and how the corresponding research questions can be answered.

5
Conclusions

This dissertation has shown how early product development for individual system solutions can be approached from a System-of-Systems (SoS) perspective. The theoretical background studies have been used to gather and synthesise knowledge from literature and similar research initiatives. This has subsequently been used to propose a method with corresponding parts that have been tested in the Search and Rescue (SAR) related case studies of the appended papers. Overall, the method has been based on architecture frameworks, ontologies, visual analytics and various concept exploration and estimation techniques. Ontology with description logic reasoning has, however, been the main approach and common denominator with which the problem, aim and research questions outlined in the introduction chapter have been addressed. The results from the performed case studies in the appended papers have shown that the holistic SoS design model from [1] can be covered using the method and corresponding parts as illustrated in Figure 3.1. Consequently, this dissertation contributes to the realization of the holistic SoS design model.

Based on the results from the appended papers and the highlighted approaches in this dissertation, the research questions can be answered as follows:

- **RQ1: How can needs from a system-of-systems be broken down into required capabilities and subsequently functions to be performed by constituent systems in a standardized and consistent way?**
  This dissertation and the case study in paper [III] have shown that an architecture framework, namely the Unified Architecture Framework (UAF), can be used to provide a structured, standardized and consistent way of understanding the relationships that connect SoS needs to capabilities, activities, functions and means. Consequently, these governing relationships can be used to perform a breakdown of identified SoS needs. The functions and outcome of the architecture framework approach can subsequently be further processed, allocated to suitable means or Constituent Systems (CS), or used as inputs for a continued design process.

- **RQ2: How can a design space for system-of-systems be represented in a flexible manner that also allows for traceable processing?**
  Ontologies with description logic reasoning capabilities can be used to represent an SoS design space. An ontology representation allows for the incorporation of more knowledge at any time from different stakeholder perspectives, which results in a flexible, interoperable and scalable knowledge representation. This dissertation and the case studies of papers [I], [II], [III], [IV] and [VI] have, for example, shown that an ontology with reasoning capabilities can create an SoS design space through the suggested ontology approach and related process. SoS scenarios with involved entities and their relationships are used to model the ontology and design space representation. Design space processing in terms of reductions and expansions can thereafter be performed using the reasoning capabilities of the ontology. The original design space representation is, however, always retained, which results in facilitated traceability throughout such processing.

- **RQ3: In what way can an overarching system-of-systems perspective generate requirements and be used in the exploration of new constituent system designs?**
  Both papers [II] and [V] have illustrated how Agent-Based Simulations (ABS) can be used to evaluate the performance of different SoS solutions. This can consequently generate overall requirements for more detailed analyses of the most prominent SoS solutions found. However, ABS can also be used to identify, for example, capability gaps in current available solutions. This, in combination with obtained functional requirements from the architecture framework approach in [III], can act as a basis for the estimation of new system solutions to be used in an SoS context to fill such gaps. The case studies of papers [IV] and [VI] have shown how such required functions and requirements can be populated with different design alternatives through ontologies and reasoning. CS concepts can thereafter be explored and estimated using techniques such as Singular Value Decomposition (SVD), symbolic regressions and optimizations, as illustrated in papers [VI] and [VII]. Estimated new CS concepts can finally be evaluated in an SoS context through ABS, as illustrated by the proposed method of this dissertation.

- **RQ4: How can decision support be facilitated in the context of system-of-systems in early product development?**
  Decision support can be facilitated through many means, however, Visual Analytics (VA) with interactive visualizations and dashboards has been suggested as an appropriate way of visualizing different aspects of early product development in the context of SoS in this dissertation. Paper [VI] has illustrated how an Interactive Reconfigurable Matrix of Alternatives (IRMA) can facilitate decision support in a design space of functions and their alternatives by interactively highlighting incompatibilities in a concept selection process. Furthermore, Figure 3.13 and the case study in paper [V] shows how interactive dashboards can be used to perform explorations of various aspects of an SoS and its operational environment from both ABS and ontology results. In general, VA is considered as an enabler of interactive explorations and enhanced decision support on all aspects of the method and related parts presented in this dissertation.

6
Outlook

Chapter 4 touched upon some important topics that have not been considered in the performed work. Consequently, this short outlook chapter highlights some prominent topics and areas for future endeavours.

One important future consideration is Concept of Operations (CONOPS) for System-of-Systems (SoS) and how this can be used to, for example, assign roles within an SoS architecture to be simulated. Naturally, this implies that more focus needs to be put on SoS simulations, such as Agent-Based Simulations (ABS), and behaviour modelling in the future. A Function/Means tree (F/M tree) approach applied to capabilities and activities could lead to a selection of different operational elements that together span a design space for CONOPS. Each concept is then a specific selection of activities that form a process to be performed by the Constituent Systems (CS) of an SoS to meet scenario goals. The different parts of the overall method in Figure 3.2 can possibly be used here as well. For example, the ontology and reasoning approach can still act as an overarching knowledge base, while additionally giving suggestions for suitable operational alternatives through reasoning. The represented knowledge can subsequently be extracted and visualized in an Interactive Reconfigurable Matrix of Alternatives (IRMA), where the available design space consists of different activities to meet the required capabilities. ABS can thereafter be used to give the Measures of Effectiveness (MOE) for different SoS solutions that also have different CONOPS included in their evaluation. Finally, the ABS results could be used to create surrogate models with, for example, Singular Value Decomposition (SVD) to be used as quick estimators for SoS performances.

The addition of an optional rule language step to the ontology and reasoning approach is a salient possibility and future topic to be investigated as well. This would possibly allow for quantitative estimations directly in an ontology representation through reasoning. Design and trade space explorations could thereby be facilitated through "what-if" questioning using a reasoner.

Another topic for future investigation is the inclusion of Visual Analytics (VA) in all aspects of the holistic SoS design model from Figure 1.2 simulta-

neously. This could thereby lead to a comprehensive design and trade space exploration platform which would facilitate decision support throughout all levels. Visualization of uncertainties is also an important consideration that has not been included in the VA case studies shown throughout this work. Dedicated visual elements in interactive dashboards that can illustrate both aleatoric and epistemic uncertainty are thereby desired additions. Such visualizations combined with interactive surrogate models would be powerful features in a dashboard for early holistic design studies. The possibility of starting new simulations directly from such a dashboard is also an intriguing thought.

Finally, Artificial Intelligence (AI) will most likely have a significant impact on many topics addressed throughout this dissertation in the future. One example is natural language processing with Large Language Models (LLM) and deep learning algorithms that allow computers to interpret and understand human languages. Such algorithms can already now build up conceptual knowledge graphs from written text. It is, for instance, probably just a matter of time until written requirements, architecture framework documentations and other SoS considerations can be processed to accurate representations within an ontology automatically. A paradigm shift is perhaps around the corner.

7
Review of Papers

Paper I

An Ontological Approach to System-of-Systems Engineering in Product Development

This paper shows how an ontology with description logic reasoning capabilities can be used to generate and reduce a design space representation for System-of-Systems (SoS). The paper also introduces a first version of the ontology development process seen in this dissertation and illustrates how it can be used to create an ontology based on SoS scenarios. The proposed approach and process are then tested in a case study based on the Search and Rescue (SAR) operations of the Swedish Maritime Administration (SMA). The results from the case study show how a reasoner can assist in design space processing for SoSs and give suggestions on suitable solutions based on user defined queries. The work performed in this paper has acted as a foundation for papers [II], [III], [IV] and [VI].

Paper II

A System of Systems Approach for Search and Rescue Missions

Based on the work presented in paper [I], this paper shows how the results from the ontology approach can be extracted and used to perform Agent-Based Simulations (ABS) to investigate different SoS constellations of SAR assets. Consequently, this paper shows how an approach including ontology and ABS can be used to generate, reduce and evaluate an SoS design space. Additionally, the performed case study shows how the ontology from paper [I] can be used as an existing domain ontology and how it can subsequently be expanded with more information using the ontology development process.

Paper III

A Breakdown of System of Systems Needs Using Architecture Frameworks, Ontologies and Description Logic Reasoning

This paper presents an approach for breaking down SoS needs to required capabilities and their corresponding functions. The breakdown approach, also referred to as the architecture framework approach in this dissertation, has been derived from the Unified Architecture Framework (UAF). Different UAF viewpoints have been used to provide a standardized and consistent way of understanding the relationships that exist between needs, capabilities, activities, functions, means and the environment. In this paper, a SAR case study is used to perform a breakdown of four basic needs with the proposed approach. The breakdown results in an initial design space of functions to be performed to fulfil the overarching capabilities and needs. Additionally, this paper shows how the outcome of the breakdown can be used as inputs for the ontology development process from paper [I] to allow for additional expressiveness. The outcome consequently corresponds to the domain of the ontology to be modelled. In a similar way to the case study in paper [II], the ontology model representing the breakdown is subsequently expanded with additional information, such as available SAR assets. Design space reductions are then performed using user defined queries and the reasoner, as done in papers [I] and [II]. This finally results in a reduced design space of alternatives or functions to be performed to meet the overarching needs of an SoS.

Paper IV

Ontology-Represented Design Space Processing

The work presented in papers [I] and [III] is leveraged and further built on in this article. Overall, the paper illustrates how different ontologies can be merged under a top-level ontology using the ontology development and integration process first introduced in paper [I]. The resulting ontology from paper [III] is used as a starting point and later merged with other ontologies to ultimately span an available design space for a new search aircraft intended for SAR operations. The design space is in this case represented as various functions with alternative means by which to implement them. Additionally, this paper illustrates how description logic reasoning can facilitate navigation and overall awareness in a relatively large ontology. The paper also shows and discusses how ontology represented knowledge can be visualized to facilitate concept selection using, for example, a Function/Means tree (F/M tree) or a morphological matrix.

Paper V

Exploring the Impact of Model Fidelity Through Interactive Visualizations for System of Systems

Paper [V] is a continuation of the work presented in paper [II]. As the title implies, the paper investigates how scenario parameters and model fidelity influence the Measures of Effectiveness (MOE) for different SoSs with Agent-Based Simulations (ABS). A Visual Analytics (VA) approach is introduced as a means to create an interactive dashboard, where the results from the various simulations can be explored to find overall trends or simply answer "what-if" questioning. The method in this paper thereby first illustrates a whole chain from an ontology representation of an SoS design space, to an evaluation in ABS, that in turn leads to interactive data visualizations facilitating results exploration as well as decision-making.

Paper VI

Ontology-Assisted Aircraft Concept Generation

The purpose of this paper is to illustrate how knowledge from an ontology can be extracted, expanded and used to generate a geometry for an aircraft concept. The paper can mainly be seen as a continuation of paper [IV], since it describes how ontology represented functions and means can be used to generate suggestions for a concept. An Interactive Reconfigurable Matrix of Alternatives (IRMA) is created from the ontology model in this paper, and is used to give guidance in concept selection and to illustrate incompatibilities between available design alternatives. Specified requirements from the ontology are then used in conjunction with concepts selected from the IRMA and description logic reasoning. This consequently categorizes chosen concepts into classes describing suitable statistical datasets to be used for subsequent estimations of design characteristics. The indicated statistical dataset for a chosen concept is thereafter used with described requirements from the ontology in a Singular Value Decomposition (SVD) analysis. This allows the ontology information about the concept to be expanded with estimates for the concept's main characteristics. This is then used for a continued geometrical sizing procedure, where more parameters are calculated and eventually used to generate a first sized version of the chosen concept. The obtained information from the sized concept is finally reintroduced into the ontology in order to expand the original knowledge base and to allow for further processing capabilities with, for example, description logic reasoning.

Paper VII

Optimization Framework for Early Conceptual Design of Helicopters

While not directly addressing SoSs, this paper contributes a method for creating optimization frameworks for early system design studies. As with the work in previous papers, SAR is used as a case study. The scope is, however, limited to the early design of helicopters only. The paper can consequently be seen as an alternate continuation of paper [VI], where it illustrates how characteristics of systems other than fixed-wing aeroplanes can be approximated. In general, the paper illustrates how statistics of existing solutions can be used to create basic models for different disciplines within the intended design domain. Statistical estimation methods, such as regressions, are mainly used to derive the models. Symbolic regressions with genetic algorithms are also introduced as an additional approach for creating estimation models. The obtained models are thereafter connected in an optimization framework, where objective functions, design parameters and constraints can be defined based on intended system requirements. Ultimately, this spans an available design and trade space of different optimal system solutions that can be explored.

Bibliography

[1] I. Staack, K. Amadori, and C. Jouannet, "A Holistic Engineering Approach to Aeronautical Product Development," *The Aeronautical Journal*, vol. 123, no. 1268, pp. 1545–1560, 2019. DOI: 10.1017/aer.2019.51.

[2] INCOSE, *What is Systems Engineering?* [Online]. Available from: https://www.incose.org/about-systems-engineering, [Accessed 22 April 2023].

[3] INCOSE, *Systems Engineering HandBook: A Guide for System Life Cycle Processes and Activities*, 4th. San Diego, CA: International Council on Systems Engineering (INCOSE), 2015.

[4] M. W. Maier, "Architecting Principles for Systems-of-Systems," *Systems Engineering: The Journal of the International Council on Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[5] A. Kossiakoff, W. N. Sweet, S. J. Seymour, and S. M. Biemer, *Systems Engineering Principles and Practice*, 2nd. Hoboken, New Jersey: John Wiley & Sons, 2011.

[6] M. Jamshidi, *System of Systems Engineering Principles and Applications*, 1st. Boca Raton: CRC Press, Taylor & Francis Group, 2009.

[7] R. Nola and H. Sankey, "The Hypothetico-Deductive Method," in *Theories of Scientific Method: An introduction*, London. UK: Acumen, 2007. Chapter 7.

[8] N. Leveson, "The Drawbacks in Using The Term "System of Systems"," *Biomedical Instrumentation & Technology, AAMI*, vol. 47, no. 2, pp. 115–118, 2013.

[9] J. Boardman and B. Sauser, "System of Systems – The Meaning of Of," in *Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering*, vol. 1, Los Angeles, CA: IEEE, 2006, pp. 118–123.

[10] INCOSE, "INCOSE Systems of Systems Primer," *INCOSE-TP-2018-003-01.0*, 2018.

[11] J. S. Dahmann and K. J. Baldwin, "Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering," in *Proceedings of SysCon 2008 – IEEE International Systems Conference*, Montreal, Canada: IEEE, 2008.

[12] J. Axelsson, "A Systematic Mapping of the Research Literature on System-of-Systems Engineering," in *Proceedings of the 10th System of Systems Engineering Conference (SoSE)*, San Antonio, Texas, USA, 2015, pp. 18–23. DOI: 10.1109/SYSOSE.2015.7151918.

[13] N. B. Shah, D. H. Rhodes, and D. F. Hastings, *Systems of Systems and Emergent System Context*, [Online]. Available from: http://web.mit.edu/adamross/www/SHAH_CSER07.pdf, 2007, [Accessed 19 March 2023].

[14] MITRE, *Systems Engineering Guide – Collected Wisdom From MITRE's Systems Engineering Experts*. The MITRE Corporation, 2014, ISBN: 978-0-615-97442-2.

[15] J. Dahmann, J. A. Lane, G. Rebovich, and R. Lowry, "Systems of Systems Test and Evaluation Challenges," in *2010 5th International Conference on System of Systems Engineering*, 2010, pp. 1–6. DOI: 10.1109/SYSOSE.2010.5543979.

[16] C. E. Dickerson and D. N. Mavris, *Architecture and Principles of Systems Engineering*, 1st ed. Boca Raton: CRC Press, 2010.

[17] P. T. Biltgen and M. D. N., "Capability-Based Quantitative Technology Evaluation for Systems-of-Systems," in *IEEE International Conference on System of Systems Engineering*, 2007.

[18] B. Bagdatli, K. Griendling, D. Kalpakchian, *et al.*, "A Method for Examining the Impact of Interoperability on Mission Performance in a System-of-Systems," 2010 IEEE Aerospace Conference, Apr. 2010, pp. 1–15. DOI: 10.1109/AERO.2010.5446884.

[19] Z. Fang, K. Moolchandani, H. Chao, and D. DeLaurentis, "A Method for Emission Allowances Allocation in Air Transportation Systems From a System-of-Systems Perspective," *Journal of Cleaner Production*, vol. 226, pp. 419–431, 2019, ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.04.083.

[20] A. K. Raz, C. R. Kenley, and D. A. DeLaurentis, "A System-of-Systems Perspective for Information Fusion System Design and Evaluation," *Information Fusion*, vol. 35, pp. 148–165, 2017, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2016.10.002.

[21] S. Y. Han and D. DeLaurentis, "Development Interdependency Modeling for System-of-Systems (SoS) using Bayesian Networks: SoS Management Strategy Planning," *Procedia Computer Science*, vol. 16, pp. 698–707, 2013, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.01.073.

[22] Z. Fang, D. DeLaurentis, and N. Davendralingam, "An Approach to Facilitate Decision Making on Architecture Evolution Strategies," *Procedia Computer Science*, vol. 16, pp. 275–282, 2013, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.01.029.

[23] N. Davendralingam and D. DeLaurentis, "A Robust Optimization Framework to Architecting System of Systems," *Procedia Computer Science*, vol. 16, pp. 255–264, 2013, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.01.027.

[24] Office of the Deputy Under Secretary of Defence for Acquisition and Technology, Systems and Software Engineering, *Systems Engineering Guide for System of Systems. Version 1.0. Washington, DC: ODUSD(A&T)SSE*, 2008.

[25] J. A. Lane, "What is a System of Systems and Why Should I Care?" In *USC-CSSE-2013-001*, University of Southern California, 2013, ISBN: 978-3-642-11828-9.

[26] C. Keating, R. Rogers, R. Unal, *et al.*, "System of Systems Engineering," *EMJ – Engineering Management Journal*, vol. 15, no. 3, pp. 36–45, 2003, ISSN: 10429247. DOI: 10.1080/10429247.2003.11415214.

[27] A. Gorod, B. Sauser, and J. Boardman, "System-of-Systems Engineering Management: A Review of Modern History and a Path Forward," *IEEE Systems Journal*, vol. 2, no. 4, pp. 484–499, 2008. DOI: 10.1109/JSYST.2008.2007163.

[28] CAE, *Capability Engineering, 2019*, [Online]. Available from: https://www.cae.com/media/documents/Defence_Security/Services_-_Documents/datasheet.capability.engineering.pdf, [Accessed 22 April 2023].

[29] J. A. Lane, "System of Systems Capability to Requirements Engineering," *Proceedings of the 9th International Conference on System of Systems Engineering: The Socio-Technical Perspective, SoSE 2014*, pp. 91–96, 2014. DOI: 10.1109/SYSOSE.2014.6892469.

[30] Office of the Deputy Director for Engineering, *Mission Engineering Guide. Washington, DC*, 2020.

[31] J. A. Lane and T. Bohn, "Using SysML to Evolve Systems of Systems," INCOSE, Tech. Rep., 2010.

[32] R. Williamson, *INCOSE (MBSE) Model Based System Engineering (SoS) System of Systems/Enterprise Activity Introduction*, 2012.

[33] M. Mori, A. Ceccarelli, P. Lollini, *et al.*, "Systems-of-Systems Modeling Using a Comprehensive Viewpoint-Based SysML Profile," *Journal of Software: Evolution and Process*, vol. 30, no. 3, pp. 1–20, 2018, ISSN: 20477481. DOI: 10.1002/smr.1878.

[34] Z. Li, C. Sim, and M. Y. H. Low, "A Survey of Emergent Behavior and Its Impacts in Agent-based Systems," in *Proceedings of the IEEE International Conference on Industrial Informatics*, Sep. 2006, pp. 1295–1300. DOI: 10.1109/INDIN.2006.275846.

[35] D. A. DeLaurentis, K. Moolchandani, and C. Guariniello, *System of Systems Modeling and Analysis*, 1st ed. Boca Raton: CRC Press, 2022, ISBN: 9781032138305. DOI: https://doi.org/10.1201/9781003231011.

[36] Object Management Group (OMG), *Unified Architecture Framework*, [Online]. Available from: https://www.omg.org/uaf/index.htm, [Accessed 22 April 2023].

[37] Object Management Group (OMG), *About the Unified Architecture Framework Specification Version 1.1*, [Online]. Available from: https://www.omg.org/spec/UAF/About-UAF/, [Accessed 22 April 2023].

[38] Object Management Group (OMG), *Unified Architecture Framework (UAF) Domain Metamodel, Version 1.1*, 2020.

[39] Object Management Group (OMG), "Unified Architecture Framework (UAF) Sample Problem (Informative)," OMG Unified Architecture Framework, Tech. Rep., 2018.

[40] J. Bankauskaite and A. Morkevicius, "Towards an Automated UAF-based Trade Study Process for System of Systems Architecture," in *30th Annual INCOSE International Symposium, Virtual Event*, Jul. 2020.

[41] D. H. Meadows, *Thinking in Systems: A Primer*, 1st. Chelsea: Chelsea Green Publishing, 2008.

[42] L. B. Rainey and A. Tolk, *Modeling and Simulation Support for System of Systems Engineering Applications*, 1st. New Jersey: Wiley, 2015.

[43] "IEEE Standard for Application and Management of the Systems Engineering Process," Institute for Electrical and Electronics Engineers (IEEE), New York, NY, USA, Standard, Dec. 2021.

[44] G. An, Q. Mi, J. Dutta-Moscato, and Y. Vodovotz, "Agent-Based Models in Translational Systems Biology," *WIREs Systems Biology and Medicine*, vol. 1, no. 2, pp. 159–171, 2009. DOI: https://doi.org/10.1002/wsbm.45. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wsbm.45.

[45] D. Klein, J. Marx, and K. Fischbach, "Agent-Based Modeling in Social Science, History, and Philosophy: An Introduction," *Historical Social Research*, vol. 43, no. 1, pp. 7–27, 2018, ISSN: 0172-6404. DOI: https://doi.org/10.12759/hsr.43.2018.1.7-27.

[46] W. Roberts, K. Griendling, A. Gray, and D. N. Mavris, "Unmanned Vehicle Collaboration Research Environment for Maritime Search and Rescue," in *International Council of the Aeronautical Sciences*, 2016.

[47] A. Papageorgiou, K. Amadori, C. Jouannet, and J. Ölvander, "A Multidisciplinary and Multifidelity Framework for Evaluating System-of-Systems Capabilities of Unmanned Aircraft," *Journal of Aircraft, Accepted, In-print*, 2019.

[48] M. Giselle Fernández-Godino, C. Park, N. H. Kim, and R. T. Haftka, "Issues in Deciding Whether to Use Multifidelity Surrogates," *AIAA Journal*, vol. 57, no. 5, pp. 2039–2054, 2019. DOI: 10.2514/1.J057750. eprint: https://doi.org/10.2514/1.J057750.

[49] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," *SIAM Review*, vol. 60, no. 3, pp. 550–591, 2018. DOI: 10.1137/16M1082469. eprint: https://doi.org/10.1137/16M1082469.

[50] M. A. Gallagher, D. V. Hackman, and A. A. Lad, "Better Analysis Using the Models and Simulations Hierarchy," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 15, pp. 1–10, Jan. 2018. DOI: 10.1177/1548512917752086.

[51] Modeling and Simulation Coordination Office 1901 N. Beauregard St., Suite 500, *Modeling and Simulation (M&S) Glossary. Alexandria, VA 22311*, 2011.

[52] FMI Development Group, *FMI Functional Mock-up Interface*, [Online]. Available from: https://fmi-standard.org/, [Accessed 7 February 2023].

[53] Modelica Association Project System Structure and Parametrization, *System Structure and Parameterization*, [Online]. Available from: https://ssp-standard.org/, [Accessed 7 February 2023].

[54] FMI Development Group, *Functional Mock-up Interface for Model Exchange and Co-Simulation*, Report 2.0. Modelica Association, July 25, 2014.

[55] A. Oprea, R. Hällqvist, I. Staack, *et al.*, "Model Fidelity and Its Influence on Aircraft Conceptual Design," in *Proceedings of the 33rd Congress of the International Council of the Aeronautical Sciences*, Stockholm, Sweden, 2022, pp. 1–14.

[56] N. F. Noy and D. L. McGuinness, *Ontology Development 101: A Guide to Creating Your first ontology*. Stanford University, 2000.

[57] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[58] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL*, 2nd. Waltham, Boston: Elsevier, 2011.

[59] D. Kalibatiene and O. Vasilecas, "Survey on Ontology Languages," Lecture Notes in Business Information Processing, 2011, ISBN: 978-3-642-33280-7. DOI: 10.1007/978-3-642-33281-4.

[60] M. Horridge, H. Knublauch, A. Rector, *et al.*, "A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3," The University of Manchester, Tech. Rep., 2011.

[61] S. Abburu, "A Survey on Ontology Reasoners and Comparison," *International Journal of Computer Applications*, vol. 57, no. 17, pp. 33–39, 2012. DOI: 10.5120/9208-3748.

[62] T. Lampoltshammer and S. Wiegand, "Improving the Computational Performance of Ontology-Based Classification Using Graph Databases," *Remote Sensing*, vol. 7, pp. 9473–9491, Jul. 2015. DOI: 10.3390/rs70709473.

[63] V. Haarslev and R. Möller, "On the Scalability of Description Logic Instance Retrieval," *Journal of Automated Reasoning*, vol. 41, pp. 99–142, Aug. 2008. DOI: 10.1007/s10817-008-9104-7.

[64] K. Stock, D. G. Leibovici, L. Delazari, and R. Santos, "Discovering Order in Chaos: Using a Heuristic Ontology to Derive Spatio-Temporal Sequences for Cadastral Data," *Spatial Cognition & Computation*, vol. 15, pp. 115–141, 2015.

[65] B. Smith, *Basic Formal Ontology 2.0 Specification and User's Guide*, 2015.

[66] Construction Innovation Hub, *A Survey of Top-level Ontologies: To Inform the Ontological Choices for a Foundation Data Model*, 2020.

[67] V. Mascardi, V. Cordì, and P. Rosso, "A Comparison of Upper Ontologies," in *8th AI*IA/TABOO Joint Workshop "From Objects to Agents": Agents and Industry: Technological Applications of Software Agents*, 2007, pp. 55–64.

[68] M. Ast, M. Glas, and T. Roehm, "Creating an Ontology for Aircraft Design, an Experience Report About Development Process and the Resulting Ontology," in *Deutscher Luft-und Raumfahrtkongress*, 2013.

[69] P. Morosoff, R. Rudnicki, J. Bryant, *et al.*, "Joint Doctrine Ontology: A Benchmark for Military Information Systems Interoperability," in *CEUR Workshop Proceedings*, vol. 1523, 2015, pp. 2–9.

[70] H. Dogan, M. J. Henshaw, and J. Johnson, "An Incremental Hybridisation of Heterogeneous Case Studies to Develop an Ontology for Capability Engineering," in *INCOSE International Symposium*, vol. 22, Rome: INCOSE, 2012, pp. 956–971.

[71] J. S. Osmundson, T. V. Huynh, and P. Shaw, "Developing Ontologies for Interoperability of Systems of Systems," in *Conference on Systems Engineering Research*, 2006.

[72] L. C. Van Ruijven, "Ontology for Systems Engineering," in *Conference on Systems Engineering Research, CSER'13*, Procedia Computer Science 16, 2012, pp. 383–392.

[73] C. Hennig, A. Viehl, B. Kämpgen, and H. Eisenmann, "Ontology-Based Design of Space Systems," in *International Semantic Web Conference*, ISWC, 2016, pp. 308–324.

[74] G. Langford and T. Langford, "The Making of a System of Systems: Ontology Reveals the True Nature of Emergence," in *12th System of Systems Engineering Conference (SoSE)*, IEEE, 2017, pp. 1–5.

[75] D. A. Wagner, M. B. Bennett, R. Karban, *et al.*, "An Ontology for State Analysis: Formalizing the Mapping to SysML," in *2012 IEEE Aerospace Conference*, IEEE, 2012, pp. 1–16.

[76] S. Jenkins, *Introduction to System Modeling and Ontologies*, [PowerPoint presentation]. Available from: https://ntrs.nasa.gov/citations/20120009873, 2011, [Accessed 22 April 2023].

[77] H. Wardhana, A. Ashari, and A. K. Sari, "Transformation of SysML Requirement Diagram into OWL Ontologies," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 4, pp. 106–114, 2020. DOI: 10.14569/IJACSA.2020.0110415.

[78] L. Yang, K. Cormican, and M. Yu, "Ontology-Based Systems Engineering: A State-Of-The-Art Review," *Computers in Industry*, vol. 111, pp. 148–171, 2019. DOI: https://doi.org/10.1016/j.compind.2019.05.003.

[79] R. Stevens, M. Stevens, N. Matentzoglu, and S. Jupp, "Manchester Family History Advanced OWL Tutorial," The University of Manchester, Manchester, Tech. Rep., 2015.

[80] L. Iannone and A. Rector, "Calculations in OWL," in *Proceedings of the 5th OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008)*, Karlsruhe, Germany, 2008, pp. 1–5.

[81] M. DeBellis, *A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins*, Edition 3.2, October 8, 2021.

[82] M. O'Connor, *The Semantic Web Rule Language*, "Stanford Center for Biomedical Informatics Research, Stanford University. 2009. [Online Presentation]. Available from: https://protege.stanford.edu/conference/2009/slides/SWRL2009ProtegeConference.pdf", [Accessed 14 February 2023].

[83] M. O'Connor and A. Das, "SQWRL: a Query Language for OWL," in *Proceedings of OWL: Experiences and Directions 2009 (OWLED 2009)*, Stanford, CA: Stanford Center for Biomedical Informatics Research, 2009, pp. 1–8.

[84] P. Krus, "An Information Theoretical Perspective on Design," in *Proceedings of the International Conference on Engineering Design (ICED'07)*, Paris, France: ICED'07, 2007.

[85] M. Rosinger, M. Buker, and R. Weber, "An Approach to Guide the System Engineer during the Design Space Exploration Process," in *Proceedings of CEUR Workshop – Software Engineering Workshops (SEWS)*, Mar. 2015, pp. 81–90.

[86] E. Kang, E. Jackson, and W. Schulte, "An Approach for Effective Design Space Exploration," in *Calinescu R., Jackson E. (eds) Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems. Monterey Workshop 2010. Lecture Notes in Computer Science*, vol. 6662, Springer, Berlin, Heidelberg, 2011. DOI: 10.1007/978-3-642-21292-5_3.

[87] P. Krus, "Information Entropy in the Design Process," in *Proceedings of the International Conference on Research into Design (ICoRD'13)*, Chennai, India: ICoRD'13, 2013.

[88] N. Aeronautics and S. A. (NASA), *NASA Systems Engineering Handbook*. Military Bookshop, 2016, ISBN: 9781680920895.

[89] A. Ross and D. Hastings, "The Tradespace Exploration Paradigm," *INCOSE International Symposium*, vol. 15, Jul. 2005. DOI: 10.1002/j.2334-5837.2005.tb00783.x.

[90] M. E. Fitzgerald and A. M. Ross, *Recommendations for Framing Multi-Stakeholder Tradespace Exploration*, [Online]. Available from: http://seari.mit.edu/documents/presentations/INCOSE16_Fitzgerald_MIT.pdf, [Accessed 22 April 2023].

[91] Scaled Agile, Inc., *Set-Based Design*, [Online]. Available from: https://www.scaledagileframework.com/set-based-design/, [Accessed 22 April 2023].

[92] M. M. Andreasen, *Machine Design Methods based on Systematic Approach – Contribution to Design Theory*. Doctoral Thesis, Lund University, Sweden, 1980.

[93] A. J. Robotham, "The Use of Function/Means Trees for Modelling Technical, Semantic and Business Functions," *Journal of Engineering Design*, pp. 243–251, 2002. DOI: 10.1080/09544820110108944.

[94] B. O'Sullivan, "Interactive Constraint-Aided Conceptual Design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 16, no. 4, pp. 303–328, 2002. DOI: 10.1017/S0890060402164043.

[95] S. D. Eppinger and T. R. Browning, *Design Structure Matrix Methods and Applications*. Cambridge: MIT Press, 2012, ISBN: 9780262301428.

[96] T. Ritchey, "Fritz Zwicky, Morphologie and Policy Analysis," in *16th EURO conference on Operational Analysis*, Brussels, 1998.

[97] M. Schmit, S. Briceno, K. Collins, *et al.*, "Semantic Design Space Refinement for Model-Based Systems Engineering," in *10th Annual International Systems Conference, SysCon 2016 – Proceedings*, 2016, ISBN: 9781467395182. DOI: 10.1109/SYSCON.2016.7490579.

[98] K. Lynch, R. Ramsey, G. Ball, *et al.*, "Conceptual Design Acceleration for Cyber-Physical Systems," in *Annual IEEE International Systems Conference (SysCon)*, Apr. 2017, pp. 1–6.

[99] P. Krus, "Models Based on Singular Value Decomposition for Aircraft Design," in *Proceedings of the Aerospace Technology Congress*, Stockholm: Swedish Society of Aeronautics and Astronautics, Oct. 2016, pp. 1–8.

[100] J. Frost, *Curve Fitting using Linear and Nonlinear Regression, Statistics By Jim, 2022*, [Online]. Available from: https://perma.cc/44Z5-4FXC [Accessed 22 April 2023].

[101] D. Freedman, *Statistical Models: Theory and Practice*. Cambridge, UK: Cambridge University Press, Jan. 2005, Available online: https://doi.org/10.1017/CBO9781139165495 [Accessed 22 April 2023].

[102] C. Ford, *Interpreting Log Transformations in a Linear Model. University of Virginia Library, 2018*, [Online]. Available from: https://perma.cc/T7AK-Q6AJ [Accessed 22 April 2023].

[103] R. Ruggiero, *Symbolic Regression: The Forgotten Machine Learning Method. Towards Data Science, 2020*, [Online]. Available from: https://perma.cc/7UM6-Q3D3 [Accessed 22 April 2023].

[104] J. Mandel, "Use of the Singular Value Decomposition in Regression Analysis," *The American Statistician*, vol. 36, no. 1, pp. 15–24, 1982, ISSN: 00031305. DOI: 10.2307/2684086.

[105] Z. Jaadi, *A Step-by-Step Explanation of Principal Component Analysis (PCA)*, Built In. [Online]. Available from: https://builtin.com/data-science/step-step-explanation-principal-component-analysis, [Accessed 15 March 2023].

[106] D. P. Raymer, *Aircraft Design: A Conceptual Approach*, 6th. Reston, Virginia: American Institute of Aeronautics and Astronautics, Inc., 2018.

[107] F. Rothlauf, *Optimization Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 45–102, ISBN: 978-3-540-72962-4, Available online: https://doi.org/10.1007/978-3-540-72962-4_3 [Accessed 22 April 2023].

[108] K. Miettinen, *Nonlinear Multiobjective Optimization.* (International Series in Operations Research & Management Science). New York, USA: Springer Science + Business Media, LLC, 2004, ISBN: 9781461375449, Available online: https://doi.org/10.1007/978-1-4615-5563-6 [Accessed 22 April 2023].

[109] M. D. Watson, B. L. Mesmer, and P. A. Farrington, *Engineering Elegant Systems: Theory of Systems Engineering. National Aeronautics and Space Administration. NASA/TP-20205003644, 2020*, [Online]. Available from: https://perma.cc/R78L-RSWF [Accessed 22 April 2023].

[110] P. M. Zadeh and M. A. S. Shirazi, "5 – Multidisciplinary Design and Optimization Methods," in *Metaheuristic Applications in Structures and Infrastructures*, A. H. Gandomi, X. Yang, S. Talatahari, and A. H. Alavi, Eds., Oxford: Elsevier, 2013, pp. 103–127, ISBN: 978-0-12-398364-0, Available online: https://doi.org/10.1016/B978-0-12-398364-0.00005-X [Accessed 22 April 2023].

[111] J. Sobieszczanski-Sobieski and R. Haftka, "Multidisciplinary Aerospace Design Optimization – Survey of Recent Developments," in *34th Aerospace Sciences Meeting and Exhibit*, Reno, NV: American Institute of Aeronautics and Astronautics, 1996, Available online: https://doi.org/10.2514/6.1996-711 [Accessed 22 April 2023].

[112] G. Wang and S. Shan, "Review of Metamodeling Techniques in Support of Engineering Design Optimization," *Journal of Mechanical Design – J MECH DESIGN*, vol. 129, Apr. 2007, Available online: https://doi.org/10.1115/1.2429697 [Accessed 22 April 2023].

[113] A. Papageorgiou, *Design Optimization of Unmanned Aerial Vehicles - A System of Systems Approach.* Linköping, Sweden: Linköping University, PhD Dissertation No. 2018, 2019, ISBN: 978-91-7519-001-3.

[114] M. Antal, C. Pop, T. Cioara, *et al.*, "A System of Systems Approach for Data Centers Optimization and Integration Into Smart Energy Grids," *Future Generation Computer Systems*, vol. 105, pp. 948–963, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2017.05.021.

[115] C. G. Palmer, "Optimizing Multi-Domain System-of-Systems Using Model-Based Systems Engineering," Air Force Institute of Technology, Wright-Patterson AFB, United States, Tech. Rep. AD1054113, 2016.

[116] M. Jändel, P. Bivall, P. Hammar, *et al.*, *Visual Analytics - Perspectives on the Field of Interactive Visualization.* Swedish Defence Research Agency (FOI), 2016, ISBN: 1650-1942.

[117] W. Cui, "Visual Analytics: A Comprehensive Overview," *IEEE Access*, vol. 7, pp. 81 555–81 573, Jun. 2019. DOI: 10.1109/ACCESS.2019.2923736.

[118] D. N. Mavris, O. J. Pinon, and D. C. Fullmer, "Systems Design and Modeling: A Visual Analytics Approach," in *Proceedings of the 27th International Congress of the Aeronautical Sciences (ICAS)*, Sep. 2010, pp. 1–28.

[119] A. D. MacCalman, H. Kwak, M. L. McDonald, *et al.*, "Illuminating Tradespace Decisions Using Efficient Experimental Space-Filling Designs for the Engineered Resilient System Architecture," Operations Research Center (ORCEN), Tech. Rep., 2015.

[120] S. Briceno and D. Mavris, "Implementation of a Physics-Based Decision-Making Framework for Evaluation of the Multidisciplinary Aircraft Uncertainty," in *Proceedings of the World Aviation Congress & Exposition*, Sep. 2003. DOI: 10.4271/2003-01-3055.

[121] E. Hüllermeier, *Aleatoric and Epistemic Uncertainty in Machine Learning*, [Online]. Available from: https://www.gdsd.statistik.uni-muenchen.de/2021/gdsd_huellermeier.pdf, [Accessed 14 March 2023].

[122] ICAO/IMO, *IAMSAR Manual – International Aeronautical and Maritime Search and Rescue Manual – Volume 1 Organization and Management*, 10th. Montréal, Quebec: International Civil Aviation Organization and International Maritime Organization, 2016, ISBN: 978-92-9258-057-5.

[123] ICAO/IMO, *IAMSAR Manual – International Aeronautical and Maritime Search and Rescue Manual – Volume 2 Mission Co-ordination*, 7th. Montréal, Quebec: International Civil Aviation Organization and International Maritime Organization, 2016, ISBN: 978-92-9258-058-2.

[124] ICAO/IMO, *IAMSAR Manual – International Aeronautical and Maritime Search and Rescue Manual – Volume 3 Mobile Facilities*, 10th. Montréal, Quebec: International Civil Aviation Organization and International Maritime Organization, 2016, ISBN: 978-92-9258-059-9.

[125] Sjöfartsverket, *Sjö- och flygräddning*, [Online]. Available from: https://www.sjofartsverket.se/Sjofart/Sjo--och-flygraddning/, [Accessed 22 April 2023].

[126] M. Halléhn, *Svenskt Program för Sjö- och Flygräddning. In Sjöfartsverket Document Nr: RADDALIV-6-182 (in Swedish)*, 2019.

[127] No Magic, Inc, *UAF Plugin 19.0 LTR Documentation*, [Online]. Available from: https://docs.nomagic.com/display/UAFP190/UAF+elements, [Accessed 22 April 2023].

[128] M. A. Musen, "The Protégé Project. A Look Back and and a Look Forward," *AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence*, vol. 1, no. 4, 2015. DOI: 10.1145/2557001.25757003.

[129] R. Arp, B. Smith, and A. D. Spear, *Building Ontologies with Basic Formal Ontology*. The MIT Press, 2015, ISBN: 0262527812, 9780262527811.

[130] J. Zheng, *Basic Formal Ontology (BFO)*, [Online]. Available from: https://github.com/bfo-ontology/BFO/wiki, [Accessed 22 April 2023].

[131] G. Esdras and S. Liscouët-Hanke, "Development of Core Functions for Aircraft Conceptual Design: Methodology and Results," in *Proceedings of the 62nd CASI Aeronautics Conference and AGM 3rd GARDN Conference*, Montreal, QC, Canada: Canadian Aeronautics and Space Institute, 2015.

[132] K. Schekotihin, P. Rodler, and W. Schmid, "OntoDebug: Interactive Ontology Debugging Plug-in for Protégé," Alpen-Adria-Universität, Klagenfurt, Tech. Rep., 2018. DOI: 10.1007/978-3-642-11829-6.

[133] A. Karpur, "Ontology Information Processing to Matrix-Based Approaches for Conceptual Design," in *Master Thesis Report LIU-IEI-TEK-A-21/04217-SE*, Linköping University, 2021.

[134] E. Torenbeek, *Advanced Aircraft Design: Conceptual Design, Analysis and Optimization of Subsonic Civil Airplanes* (Aerospace Series). Wiley, 2013, ISBN: 9781118568095.

[135] Microsoft, *Microsoft Power BI*, [Online]. Available from: https://powerbi.microsoft.com/en-us/, [Accessed 22 April 2023].

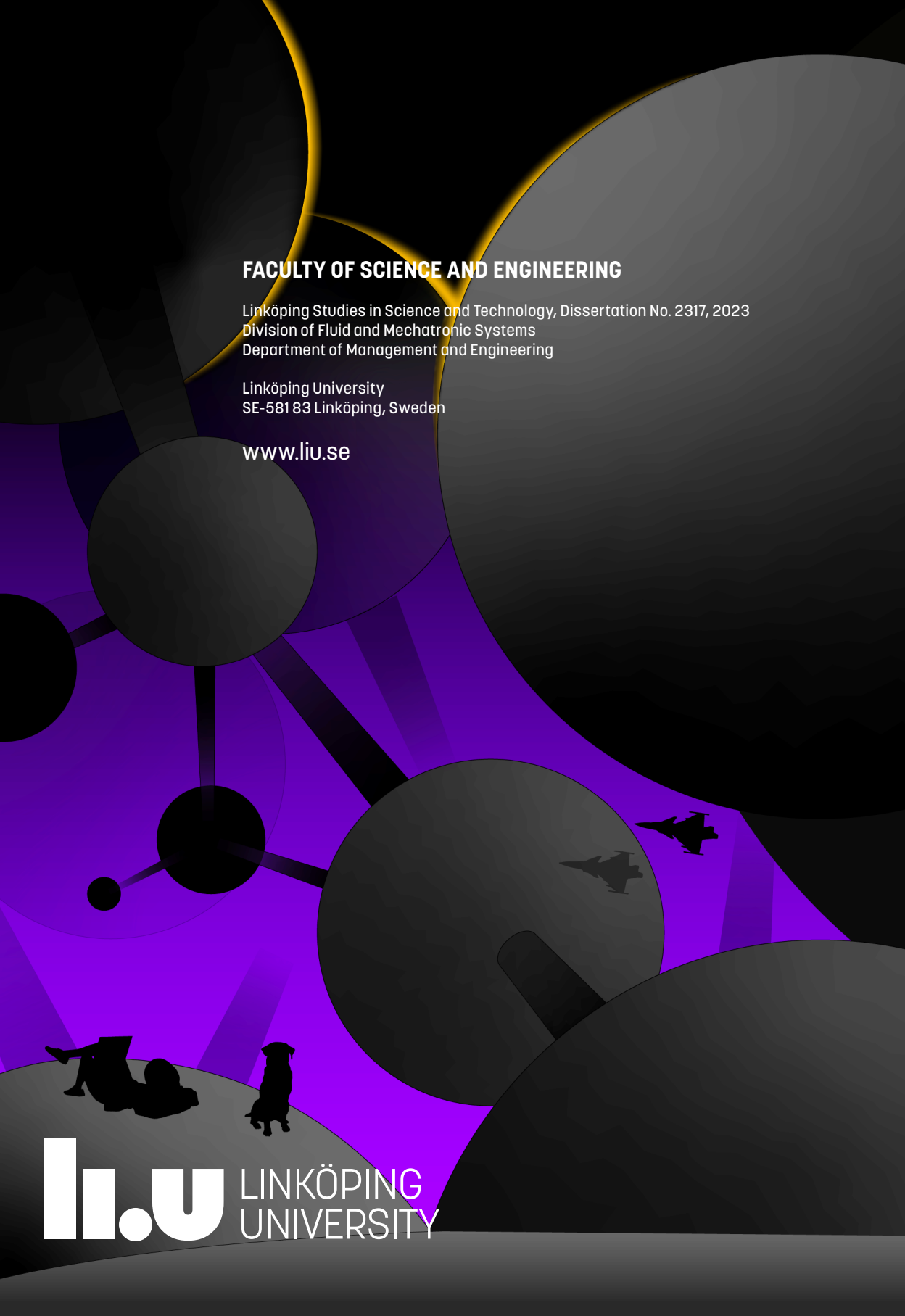
Papers

The papers associated with this thesis have been removed for copyright reasons. For more details about these see:

https://doi.org/10.3384/9789180751667

You know, I'm something of a scientist myself

Norman Osborn



FACULTY OF SCIENCE AND ENGINEERING

Linköping Studies in Science and Technology, Dissertation No. 2317, 2023
Division of Fluid and Mechatronic Systems
Department of Management and Engineering

Linköping University
SE-581 83 Linköping, Sweden

www.liu.se

LINKÖPING UNIVERSITY