UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Aeronautics and Astronautics

Aeronautical Life-Cycle Mission Modelling Framework for Conceptual Design

by

Benjamin Schumann

Thesis for the degree of Doctor of Philosophy

January 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Aeronautics and Astronautics

Thesis for the degree of Doctor of Philosophy

AERONAUTICAL LIFE-CYCLE MISSION MODELLING FRAMEWORK FOR CONCEPTUAL DESIGN

by Benjamin Schumann

This thesis introduces a novel framework for life cycle mission modelling during conceptual aeronautical design. The framework supports object-oriented mission definition using Geographical Information System technology. Design concepts are defined generically, enabling simulation of most aeronautical vessels and many non-aeronautical vehicles. Moreover, the framework enables modelling of entire vessel fleets, business competitors and dynamic operational changes throughout a vessel life cycle. Vessels consist of components deteriorating over time. Vessels carry payload that operates within the vessel environment.

An agent-based simulation model implements most framework features. It is the first use of an agent-based simulation utilising a Geographical Information System during conceptual aeronautical design. Two case studies for unmanned aircraft design apply the simulation.

The first case study explores how the simulation supports conceptual design phase decisions. It simulates four different unmanned aircraft concepts in a search-and-rescue scenario including lifeboats. The goal is to learn which design best improves life cycle search performance. It is shown how operational and geographical impacts influence design decision making by generating novel performance information. The second case study studies the simulation optimisation capability: an existing aircraft design is modified manually based on simulation outputs. First, increasing the fuel tank capacity has a negative effect on life cycle performance due to mission constraints. Therefore, mission definition becomes an optimisation parameter. Changing mission flight speeds during specific segments leads to an overall improved design.

Contents

List of tables

List of figures

Author publications

In the course of developing this thesis, several publications were authored or co-authored. All material in this thesis is believed to be novel unless cited otherwise.

Journal papers:

- Schumann, B., Scanlan, J., Fangohr, H. & Ferraro, M., 2014. Better Design Decisions through Operational Modeling during the Early Design Phases. *Journal of Aerospace Information Systems, AIAA (accepted for publication)*.
- Gorissen, D., Quaranta, E., Ferraro, M., Schumann, B., van Schaik, J., Keane, A.J. & Scanlan, J., 2014. A Value Based Decision Environment: Vision and Application. *Journal of Aircraft, AIAA (accepted for publication)*.
- Surendra, A., Ferraro, M., Schumann, B., van Schaik, J., Daniels, J.J., Gorissen, D., Scanlan, J. & Keane, A.J., 2012. The Challenges of Using Value-Driven Design for Practical Design of UAS. *Journal of Aerospace Operations, 1(1), pp.377-386*.

Conference Papers:

- Schumann, B., Scanlan, J. & Fangohr, H., 2012. Complex Agent Interactions in Operational Simulations for Aerospace Design. In C. Laroque et al., eds. *Proceedings of the 2012 Winter Simulation Conference*. Berlin, Germany: IEEE, pp. 2986–2997.
- Schumann, B., Scanlan, J., Fangohr, H. & Ferraro, M., 2012. A Generic Unifying Ontology for Civil Unmanned Aerial Vehicle Missions. In *Aviation Technology, Integration, and Operations (ATIO) Conferences*. Indianapolis, USA: AIAA.
- Ferraro, M., Gorissen, D., Scanlan, J., Keane, A.J., Quaranta, E., Schumann, B., van Schaik, J., & Bolinches, M., 2012. Toward Value-Driven Design of a Small, Low-Cost UAS. In *53rd AIAA/ASME/ASCE/AHS/AS Structures, Structural Dynamics and Materials Conference*. Honolulu, Hawaii: AIAA.
- Schumann, B., Scanlan, J. & Takeda, K., 2011. A Generic Operational Simulation for Early Design Civil Unmanned Aerial Vehicles. In *SIMUL2011: The Third International Conference on Advances in System Simulation*. Barcelona, Spain: IARIA.
- Schumann, B., Scanlan, J. & Takeda, K., 2011. Evaluating Design Decisions in Real-Time Using Operations Modelling. In R. Curran & S. C. Santema, eds. *Air Transport and Operations Symposium 2011 (ATOS)*. Delft, the Netherlands: Delft University of Technology.

Declaration of authorship

I, Benjamin Schumann

declare that the thesis entitled

AERONAUTICAL LIFE-CYCLE MISSION MODELLING FRAMEWORK FOR CONCEPTUAL DESIGN

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- none of this work has been published before submission.

Signed: ……..

Date:…….

Acknowledgements

Many people have shaped and steered the work that culminated in this thesis. Foremost, I want to thank my supervisors for their continuous support: Prof. Jim Scanlan for fostering freedom while keeping me focussed; Prof. Hans Fangohr for giving very constructive feedback and a different perspective; and Dr. Kenji Takeda for his hands-on positive approach at the start of this journey. Not least, all three fully supported my time off-campus.

Next, I am indebted to my examiners Prof. Mark Price and Prof. Joerg Fliege for spending much time studying and reflecting on this work.

Moreover, I want to thank the team of my doctoral training centre, the “Institute for Complex Systems Simulation” at the University of Southampton: Prof. Seth Bullock and Dr. Jason Noble for outstanding lectures as well as friendly and approachable feedback; and Nicki Lewin for keeping the group together and supporting administrative worries splendidly. Thank you for a very special and social PhD experience.

The case studies of this thesis build on meetings and expert knowledge from several institutions. I want to thank: Richard Norris and the team of the Lee-on-Solent Maritime Coastguard Agency helicopter base for an insightful tour; Simon Hiscock and the team of Folkestone police for sharing their knowledge on Channel police missions; Reinout Gunst from the Port of Rotterdam authority on discussing UAS missions in the harbour area; and the team of the RNLI Calshot lifeboat station for providing operational insights.

My loving and supportive family blessed my time as a research student as well. Foremost my wife Anne, who showed constant interest in my work, provided a healthy portion of criticism and kept me focussed. My daughter Teresa, whose laughter made long simulation runs joyful. My parents with their love and interest for my work. And my brother providing final feedback from the other half of the globe.

I want to thank the students I met in my doctoral training centre cohort for interesting discussions and insights into other research areas. Not least, I am grateful for the conference experiences with inspiring talks and friendly chats.

Finally yet importantly, I thank the EPSRC for generously funding my work through the Doctoral Training Center grant EP/G03690X/1.

Conventions

Several typesets are used throughout this thesis to indicate specific meaning as follows:

- `Code: specific computer code used verbatim in the OSCAR simulation code or in the SQLite database code.`
- CLASS: USED TO DENOTE PREVIOUSLY DEFINED JAVA CLASSES AND OBJECT TYPES.
- `Parameter: used to mark parameters, variables and database column names previously defined.`

- *Definition: used to define terms and concepts.*

Abbreviations

Symbol	Description
3i	Integrated Coastal Zone Management via Increased Situational Awareness through Innovations on Unmanned Aircraft Systems
6DoF	Six degrees of freedom
ATM	Air Traffic Management
CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CoG	Centre of Gravity
COTS	Commercial Off The Shelf
DECODE	Decision Environment for COmplex Design Evaluation
DOC	Direct Operating Cost
ESRI	Environmental Systems Research Institute, Inc. (www.esri.com)
FEA	Finite Element Analysis
FMEA	Failure Modes and Effects Analysis
GB	Gigabyte
GIS	Geographical Information System
GMT	Greenwich Mean Time
GSD	Ground Sample Distance
GUI	Graphical User Interface
ICAO	International Civil Aviation Organisation
IMO	International Maritime Organisation
IPT	Integrated Product Team
IQR	Inter-Quartile Range
MCA	Maritime and Coastguard Agency
OSCAR	Operational Simulation for Conceptual Aeronautical designeRs
PRA	Port of Rotterdam Authority
RNLI	Royal National Lifeboat Institution
rpm	rounds per minute
SESAR	Single European Sky ATM Research
TRL	Technology Readiness Level
UAS	Unmanned Aerial System(s)
USCG	United States Coast Guard
VOM	Value Operations Methodology
VTOL	Vertical Take-Off and Landing

List of symbols

Symbol	meaning	Units
c	coefficient	
C	cost	$\$$
d	data	bytes
D	drag	N
d	total number of cycles for a target	
g	standard gravity	m/s^2
h	height	m
l	camera footprint length	m
m	mass	kg
N	number of cycles across a target	
n	number of replications	
o	distance	m
p	pixel	pixel
P	power	W
r	range	m
R	slant range	m
S	wing area	m^2
T	Student's T distribution sample	
t	time	s
V	velocity	m/s
W	weight	kg
w	width	m
Δ	field of view	$radians$
ζ	propeller efficiency	
μ	Mean	
σ	standard deviation	
Ω	confidence interval	
α	significance level	
β	weibull distribution beta parameter	
γ	detection criteria	
δ	fuel consumption	kg/s
η	propeller rounds per minute	min^{-1}
ρ	density	kg/m^3
ς	target	
τ	number of bytes per pixel	bytes/pixel
φ	quantity	
ϕ	specific fuel consumption	$g/kW.hr$

List of subscripts

Subscript	meaning
a	acquired
avg	average
D	drag
d	drag
dry	dry
H	height
h	horizontal
inst	installed
L	lift
M	Maintenance
max	maximum
Mh	maintenance hour
Mo	maintenance operation
prop	propeller
req	required
v	Vertical
W	width

1. INTRODUCTION

1.1 Motivation

1.1.1 Contemporary aeronautical design

Design of large aeronautical systems consists of trade-offs based on iterative decision-making (Ashok 2013). Most decisions are not straightforward but a trade-off between two or more parameters. These trade-offs are often non-linear, overlap each other and span multiple dimensions as well as scales. Essentially, a small change at one end of the design space can have large consequences for remote aspects of design: This makes aeronautical design a complex undertaking (Alonso et al. 2009; Kroo et al. 1994). Arguably, design complexity grows at a faster pace than the design methods and tools available (Gorissen, Quaranta, Ferraro, Schumann, Schaik, Keane, et al. 2014; Raj 1998). Moreover, ever-growing design complexity exacerbates cost overruns, delivery delays and quality defects (Collopy & Hollingsworth 2009). According to Collopy (2012), “cost overruns are an emergent property of aerospace development programs” inherently rooted in contemporary design methods and tools. With current methods, it is a big challenge to predict the impact of design decisions upon any other part of the design or any aspect of the design operation during its life cycle (Curran et al. 2005; Price et al. 2012). Essentially, engineers need to be able to compare two or more designs at multiple levels of abstraction and obtain decision support as to which option is “better”. However, it is difficult to define formally what “better” means. Value-driven design is a methodology aiming to overcome this difficulty.

1.1.2 Value-driven design

Value-driven design is a framework that aims to improve existing Systems Engineering design processes for large systems. It shifts the design focus from sub-system design optimisation to system level performance (Collopy & Hollingsworth 2011). The latter is quantified by means of a "value", often defined in monetary terms as the ratio of benefit over cost. This value can be used to compare designs quantitatively as it provides a measure of "design goodness" (Cheung et al. 2010). Generating a value requires a holistic perspective of the entire product life cycle including design, production, operations and disposal. Thus, value calculation bases upon a number of models that contribute specific information such as development cost, manufacturing cost or product revenues. A critical component of value-driven design analysis is a model of the product life cycle missions and operations in order to quantify its performance capability (Collopy 2008).

1.1.3 Conceptual design mission modelling

Mission modelling within value-driven design should be used whenever trade-off decisions are being made. Many scientists and engineers agree that the most influential trade-off decisions occur during the very early conceptual design phase because total unit cost strongly depends on such early decisions (Castagne et al. 2009; Cheung et al. 2009; Pidd 1992). Moreover, up to 80% of the total life cycle cost is fixed during the conceptual design phase (Curran et al. 2004; Raj 1998; Saravi et al. 2013; Thokala 2009; Will 1991). However, Jinks (2012) disputes the empirical evidence. Nonetheless, mission modelling within value-driven design should be applied from this earliest design stage onwards. This thesis will introduce a mission model for the earliest stage of design, namely the conceptual design phase.

1.1.4 Explicit mission modelling

A missions and operations model allows quantifying design performance with high fidelity, supporting detailed cost and benefit calculation. However, the fidelity of existing conceptual design phase mission models is low: Typically, it is limited to simulating the performance of one vessel conducting a "typical" mission profile made up of parameterised flight segments (take-off, cruise, land, *etc.*). Quasi-analytical methods based on simple flight physics compute fuel burn for each segment (Torenbeek 2013). Next, designers extrapolate outcomes over the expected vessel life cycle. This approach neglects several important factors:

- Life-cycle variations: A design often conducts different mission types throughout its life cycle. A civil aircraft might become a cargo plane later in life.

- Spatial mission details: It is critical to analyse *where* a design faces operational problems. Does an engine fail near an airport or over the ocean? Do fuel capacity problems occur near an emergency airport?
- Fleet behaviour: A fleet of products behaves differently to a single product instance. Broken aircrafts can be replaced, several aircrafts can work together to accomplish a goal, *etc.*
- Life-cycle impact: Current mission extrapolation trivialises the impact of design changes upon the life cycle.

A mission model that overcomes these deficiencies is called an "explicit" mission model hereafter:

> *Explicit mission model: Defines any operation, life-cycle variation and spatial as well as fleet detail in a specific and unambiguous way.*

Explicit mission modelling takes into account additional information about the design and its operations. This enables estimating the effect of design changes upon the entire life cycle of the product. Moreover, quantifying product value becomes more precise compared to existing mission models. This helps design optimisation and requirements definition during the conceptual design phase. However, implementing an explicit mission-modelling framework causes an initial overhead that must be taken into account.

1.2 Research question

The previous motivation leads to the thesis hypothesis stating that:

> *H_1: Explicit mission modelling during aeronautical conceptual design supports decision-making, trade-off analysis and optimisation for value-driven design.*

More practically, the research question asks:

> *Is it possible to create an explicit, generic life-cycle mission-modelling framework for aeronautical vessels that supports deci-*

sion-making, trade-off analysis and optimisation for value-driven design?

1.3 Objectives & methodology

This thesis aims to answer the research question and confirm or reject the hypothesis. This includes showing that explicit mission modelling can benefit conceptual design phase decision-making and optimisation by processing additional operational intelligence. For this, a mission-modelling *framework* is presented. It is as generic as possible to cover a wide range of vessels and operational scenarios. Vessels include any moving object whose prevalent mission trajectories are parallel to the earth surface. Operations include any mission that can be modelled from a combination of points and paths. In order to be useful beyond theory, the thesis introduces a practical implementation to examine the research question and hypothesis quantitatively. This implementation is an agent-based *simulation* model. It incorporates a Geographical Information System to model the spatial components of the framework. Acknowledging time and resource constraints of the aeronautical conceptual design phase, both the framework and simulation will be as simple as possible while being able to answer the research question. Two case studies apply the simulation model in different settings to demonstrate the capability for decision support and optimisation, respectively. The following methodology applies:

(1) Develop a generic conceptual design phase mission-modelling framework (Chapter 3)
(2) Develop a practical implementation incorporating the framework features (Chapter 4)
(3) Use the practical implementation in two case studies based on existing research projects
    a. Case study – Decision support (Chapter 5)
        i. Demonstrate simulation of four different aircraft designs
        ii. Demonstrate framework generic vessel capability beyond aeronautical vessels by implementing boats into the operational scenario
        iii. Demonstrate decision support capability by comparing aircraft designs and providing a value for each
        iv. Compare impact upon entire life cycle for different designs
    b. Case study – Optimisation (Chapter 6)
        i. Demonstrate conceptual design optimisation and requirement refinement support
        ii. Initial aircraft design performs with specific value
        iii. Manual aircraft parameter variation leads to life-cycle impact and value decrease

iv. Manual mission parameter variation leads to overall value increase demonstrating optimisation capability

(4) Discuss and conclude (Chapter 7)

a. Framework and simulation issues
b. Limitations
c. Applicability for aeronautical design

1.4 Contribution to knowledge

This thesis contributes to scientific knowledge in three ways:

1. It develops an explicit conceptual design phase mission-based life cycle framework that employs agent-based modelling to allow aircraft and component failure in the design loop. This departs from current fault-free single-flight mission modelling.
2. It applies 3D spatial coordinates within a Geographical Information System, moving away from existing distance-based mission profiles.
3. The framework is generic, thus enabling modelling of vessels and scenarios for non-aeronautical and aeronautical applications alike.

1.5 Thesis outline

The acronym OSCAR (*Operational Simulation for Conceptual Aeronautical designeRs)* applies throughout this thesis to refer to the "OSCAR framework" and the "OSCAR simulation".

The rest of this thesis consists of six chapters:

The Literature review in Chapter 2 presents and discusses the current knowledge for the main themes mentioned above. It describes the common aeronautical design phases focussing on the conceptual design phase and its specific challenges. This is followed by a more detailed discussion of value-driven design. Moreover, it discusses the concept of modelling in more detail, distinguishing between conceptual and mathematical modelling. Next, details on simulation modelling include the state of the art in agent-based simulation and spatial Geographical Information System simulation. Last, current methods and tools in aeronautical conceptual design mission modelling are discussed. This includes industrial tools as well as recent scientific research applications.

Chapter 3 ("Framework") details the mission modelling OSCAR framework. First, it derives the four main framework requirements from the literature review, namely genericity, comprehensibility, realism and modularity. Next, the OSCAR framework is divided into scenario-related content and vessel-related content. First, the chapter presents the scenario-related scope

of application and classifies aeronautical operations into three categories. Next, operations are partitioned into modular building blocks called “Segments”, “Tracks” and “Missions”. These are embedded into the spatially explicit Geographical Information System setup unique to this framework. The vessel-related content is presented next: as before, the chapter defines the scope of application and classifies vessels generically. A set of parameters defines each vessel uniquely. Moreover, generic propulsion, fatigue and payload add-ins are described.

Chapter 4 (“Simulation”) describes the OSCAR simulation. After justifying the application of simulation for this model, a functional specification details the objectives and requirements for the OSCAR simulation. After discussing software selection, a brief model overview and model walkthrough introduce the simulation. Subsequently, the chapter describes specific aspects of the simulation model such as data architecture and Geographical Information System support. Moreover, it depicts the classes and modules developed for the simulation. Finally, details are provided on the OSCAR simulation experimentation, including information on how to embed the simulation into contemporary conceptual design phase processes.

Chapter 5 (“Case study – Decision support”) details the first case study conducted to present the conceptual design phase decision support capabilities of the OSCAR framework and simulation. The goal is to quantify the impact of different designs upon an entire life cycle. It describes work conducted as part of a research project where the simulation was used to design, build and fly several UAS (Unmanned Aerial Systems). For this, a fictitious but realistic scenario employs UAS for search-and-rescue operations. The chapter provides an account of the scenario and simulation setup and describes a purpose-build value model. Results compare the UAS quantitatively and comment on UAS selection. Three types of decision support provided by the OSCAR simulation are discussed, namely value-based, cost-based and qualitative decision support.

Chapter 6 (“Case study – Optimisation”) describes the second case study detailing how the OSCAR simulation could be used for conceptual design phase optimisation and requirements refinement. It describes work conducted as part of another research project where several countries collaborate to design a UAS platform for English Channel applications. The structure is similar to the previous case study with a description of a fictitious but realistic scenario in the port of Rotterdam followed by the specific simulation setup. The initial design is simulated, analysing operational as well as performance shortcomings. A first design iteration aims to remedy the shortcomings through manual aircraft parameter changes. This increases problems further due to the mission-specific setup. A second design iteration takes advantage of the more explicit mission modelling and changes mission parameters to improve the situation. Last, all three designs are compared.

Chapter 7 (“Conclusion”) concludes the thesis and details problems faced during the OSCAR framework and simulation development. It lists the inherent limitations that users must be aware

of. Moreover, it provides recommendations for industrial adoption and future work. Last, it comments on the research question and hypothesis answers obtained.

2. LITERATURE REVIEW

This chapter discusses recent developments in related research areas and details how the work of this thesis advances existing knowledge. Section 2.1 describes current aeronautical design processes and methodologies. It details each design phase and focuses on conceptual design phase challenges. Section 2.2 presents value-driven design and how mission modelling is critical for this design paradigm. Section 2.3 explains the term *modelling* and presents conceptual and mathematical modelling in more detail. Next, Section 2.4 introduces *simulation modelling* discussing its advantages and disadvantages. Moreover, it focuses on agent-based simulation and spatially explicit simulation in more detail. Last, Section 2.5 reviews the current practice of mission simulation in conceptual aeronautical design, presenting a number of relevant commercial tools and research initiatives.

2.1 Aeronautical design

Design of artefacts greatly contributes to the artefact final form, cost and reliability. Often, design is referred to as "an art to be learned from experience as opposed to science that can be taught" (Will 1991). However, design of aeronautical systems differs because it includes "science that can be taught". Ideally, this science comprises information and data analysis leading to rational design decisions. In fact, Raj (1998) defines aeronautical design as an "iterative decision-making activity". Price et al. (2006) note that decisions base upon different analysis methods:

> *"The current approach to aircraft (or any) design can be summarised as being based on conventional configurations using empirical methods at the highest level, supplemented by sophisticated multi-disciplinary simulations at more detailed levels."*

Consequently, in his book on aircraft design Corke (2003) states that the goal of aircraft design decision-making is to *balance* the majority of performance-related design aspects while optimising only a few. Curran et al. (2004) observe a gap between high level empirical design and detailed modelling:

> *"Aircraft engineering is not yet integrated as inter-linkage between key variables and parameters has not yet been built into a structured modelling environment".*

The proportion of empirical and analytical modelling varies during the different design phases. Despite numerous schematic representations, the literature generally differentiates between four design phases, namely requirements collection, conceptual design, preliminary design and detailed design (Al-Salka 2001; Bond & Ricci 1992; Price et al. 2006; Tam 2004).

Progressing through the design phases, the number of design options decreases while the number of fixed variables and the number of design parameters increases (Park & Seo 2004). Moreover, the level of design detail and confidence grows (Torenbeek 2013). Below, an account of each design phase follows, emphasising conceptual design phase challenges.

2.1.1 Requirements & specification

Once manufacturers see the need for a new product, they will initiate product requirements capture. Torenbeek (1982) provides a number of reasons for manufacturers to endeavour a new product design. During the initial requirements collection phase, potential customers and manufacturers discuss and define the new product requirements and specifications in detail. Requirements are usually exact numerics based on a specific mission profile (Quinn et al. 2012). Precise and unambiguous requirements are critical to product success (Will 1991). However, the process of obtaining such requirements is difficult. After critically assessing the requirements, they are often decomposed and structured using a morphological matrix or similar methods (Twiss 1992). QFD was proposed to introduce quantifiable metrics into (often subjective and fuzzy) requirement discussions (Chan & Wu 2002). By applying matrix ranking methods, specifications become more rigorous (Hauser & Clausing 1988). The result is a ranked list of design requirements and their relative importance. QFD has several shortcomings: It cannot recognise

and display conflicting customer requirements (Scanlan 2007). Moreover, requirements remain uncertain aspirations due to imperfect understanding of the product missions, life cycle and operating environment at the time of requirements capture (Thokala 2009).

Customers usually provide the majority of requirements (Nilubol 2005). With regards to mission modelling, requirements include a typical mission (or a set of missions) specified by speed, distance and payload for different mission segments like take-off, cruise, etc. (Bond & Ricci 1992). Moreover, specification includes required ranges, landing distances, manoeuvrability metrics, weights, reliability figures and target DOC (Direct Operating Cost).

Collopy (2007) argues that requirements such as weight, costs, reliability and maintainability should be excluded from requirements collection. The reason is that these requirements are a function of sub-system requirements ("extensive attributes"). This causes design teams to design *towards* the sub-system requirement instead of *optimising* their sub-system for the entire system. Cost overrun is the inevitable consequence. Instead, objective functions for extensive attributes should apply as is done within value-driven design. Mavris and Kirby (1999) provide another argument against direct extensive attribute specification: Many extensive attributes are difficult to quantify. During the conceptual design phase, explicit mission modelling helps refining requirements (as demonstrated in the case study in Chapter 6).

2.1.2 Conceptual design phase

2.1.2.1 Overview

The purpose of the conceptual design phase is to obtain a product design that fulfils the functional requirements and the expected missions (Anemaat et al. 2013; Romli 2013). According to Raymer (2006), the conceptual design phase objective is "the development of layouts and assessments of distinct design alternatives addressing a common set of requirements". The length of this phase varies. Mid-size commercial airliner design takes between nine and twelve months while business jet design can be as short as four months (Torenbeek 2013). In fact, Engler (2013) describes the example of the US Army design request for a joint light tactical vehicle: It assigned thirty-three business days to create concept designs.

Engineers usually conduct conceptual design in stages. Saravi et al. (2013) define seven steps, namely (1) customer need clarification, (2) target specification, (3) product concept generation, (4) product concept selection, (5) product concept testing, (6) final specifications and (7) downstream specifications planning. The first step overlaps with the requirements phase and re-emphasizes the importance of good requirements capture. Mavris and Kirby (1999) specifically call step (5) the "Modelling and Simulation" stage, highlighting the importance of computational tools for conceptual design. In order to simplify the process further, Jameson (1999)

lists only three conceptual design phase steps, emphasizing the importance of mission definition (Figure 2-1).

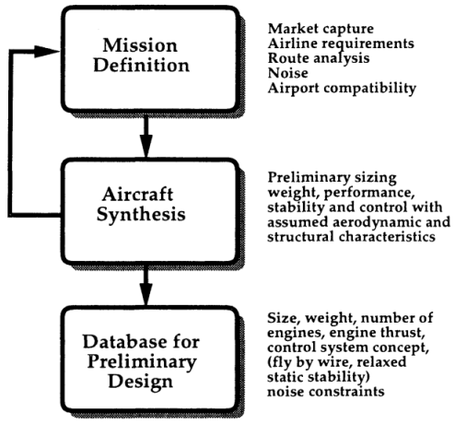


FIGURE 2-1: CONCEPTUAL DESIGN PHASE STAGES. REPRODUCED BY JAMESON (1999).

In practise, designers follow these stages iteratively. Often, several candidates are developed, taking into account that many combinations of sub-systems exist that can satisfy the requirements (Mavris & Kirby 1999; Thokala 2009).

Until the 1980s, engineers conducted conceptual design using legacy data, handbooks, manuals and their experience. There was a lack of purpose-built evaluation tools for fast decision support (Delaurentis et al. 1996; Will 1991). The growth of computational power enabled using simple first order tools and custom spreadsheets. Today, the conceptual design phase is very software intensive as concept models and designs are generally not assembled physically (Glas 2013). A large number of low fidelity computer models is used at the same time, mostly based on empirical legacy data (Nunez & Guenov 2013). Ideally, conceptual design phase tools are highly flexible and fast, accommodating a high degree of uncertainty (Engler 2013). However, in reality engineers often create custom, stationary spread sheets that are hard to validate, integrate and manage (Steinkeller 2011). Alternatively, engineers employ COTS (Commercial of the Shelf) tools such as PACE or Piano for conceptual aircraft design, including mission modelling (see Section 2.5).

The output of the conceptual design phase is the design candidate that best meets the requirements. To find the best candidate, each design is compared against the initial requirements (Quinn et al. 2012). The candidate definition indicates the position of the major components such as wings, engines, undercarriage and doors. Moreover, information exists about expected weight, flight performance, reliability targets and DOC estimates for subsequent analysis (Fielding 1999). Some of this information bases upon conceptual mission modelling.

2.1.2.2 Information challenge

Mavris et al. (1998) describe a "design paradox": During conceptual design, there is little information (or "knowledge") alongside much design freedom but progressing with design, both characteristics reverse as in Figure 2-2.

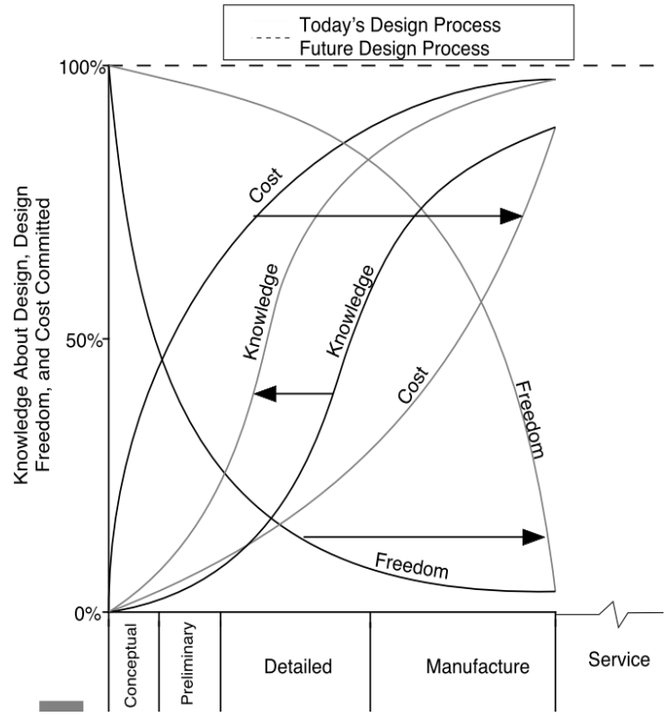


FIGURE 2-2: DESIGN PHASES KNOWLEDGE, FREEDOM AND COST COMMITTED. REPRODUCED WITH PERMISSION FROM MAVRIS ET AL. (1998)

It can be seen that the goal of design method research is to ease this design paradox by processing more information (*i.e.* "knowledge") earlier and increasing design freedom throughout all design stages. To achieve this, design decisions and trade-off requires the right information at the right time (Raj 1998). However, Park and Seo (2004) state that the lack of detailed information during conceptual design is a barrier to rational decision-making. Due to the "incomplete knowledge about the operational environment" (Bandte 2000) and large uncertainties about requirements and product performance, designers are forced to select concepts based on estimates. Therefore, overcoming the lack of detailed information (or "knowledge" as in Figure 2-2) during conceptual design is a key theme in conceptual design research (Abbas-Bayoumi & Becker 2011; Ashok 2013; Kirby 2001; Mavris & Kirby 1999). In fact, there are more reasons for this development.

In the civil aircraft market, engine and aircraft manufacturers lease a growing portion of their production to reduce customer risk (Scanlan 2004; Collins 2012). Since many lease contracts are signed very early in the design process, manufacturers must understand life-cycle behaviour and risks as early as possible (Scanlan & Rao 2006). Military manufacturers face a different

challenge but with the same outcome: As budgets shrink while product requirements focus on increased versatility, understanding the product as early as possible is paramount (Nilubol 2005). Thokala (2009) advocates information shift towards the early design phases in order to replace existing early parametric cost models with more detailed "generative" models. Bandte (2000) suggests to limit communication during conceptual design to quantitative issues using models only. In general, there is a hope to achieve decisions that are more rational during conceptual design by employing information and data that is more detailed (Kirby 2001). Not least, Cassidy (2008) argues that ever-growing computational power enables application of more detailed methods earlier in the design process.

In order to increase information about the design during the conceptual design phase, methods and tools must adapt. Taguchi (1986) stresses that design improves not through additional quality checks and rework but through improved processes. Instead of focussing on problem solving, design should be pro-active by applying better modelling and simulation tools. Current conceptual design phase models generally focus on performance analysis, neglecting operational life-cycle aspects (Keane & Nair 2005). One of the proposed methods to increase use of modelling and simulation is Integrated Product and Process Development (Ashok 2013). Here, integrated product teams focus their multi-disciplinary effort using digital product models and integrated design tools. By combining concurrent engineering and simulation, manufacturing and operational issues are considered much earlier in the design process (Peters 1995; Ranky 1994). More generally, Kirby (2001) reasons that new modelling and simulation platforms are imperative due to the growing number of design criteria ruling out purely performance-based design. Steinkeller (2011) argues that increasing design knowledge early in the design process requires new conceptual design phase tools with higher fidelities. Price et al. (2006) note a lack of suitable computer models that integrate well into existing Systems Engineering design environments: Existing models would not consider all sub-systems, their interactions and environmental influences, thereby neglecting possible emerging behaviour.

This thesis presents a novel way of processing detailed design information for conceptual mission modelling by employing an agent-based model within a geographical environment. This increases knowledge on operational life cycle performance beyond current mission model outputs because sub-systems, interactions and environmental effects are considered concurrently. Thereby, more rational design decisions and trade-offs become possible.

2.1.3 Preliminary design phase

Emphasizing the importance of the conceptual design phase, Raymer (2006) refers to the early preliminary design phase as the point where “all big questions are answered already”. Taking the conceptual design, each sub-system is now re-considered in more detail using specialist “Integrated Product Teams” (IPT) that encompass several disciplines (Collopy & Poleacovschi 2012). Work force increases along with the number of design parameters. Additional low-fidelity models are used alongside more sophisticated analysis tools. They assess initial estimates on aerodynamics and structures such as flutter, aeroelasticity or engine-wing interactions. Computational requirements on these analysis codes are very high, limiting the number of possible design variations that can be assessed (Keane & Nair 2005).

During the preliminary design phase, engineers also start to consider manufacturing issues. Towards the end, the design is frozen with a complete geometric definition. Often, lack of information, time or analysis capability forces a design freeze despite known defects in order to proceed to the detailed design phase (Keane & Nair 2005). The output of the preliminary design phase includes the system configuration with detailed sub-system specifications such as weights, maintenance elements and software requirements.

2.1.4 Detailed design phase

During the detailed design phase new teams form along part definitions (Price et al. 2006). Each part is specified in detail at a fidelity high enough to enable manufacturing (Nilubol 2005). Computational models are used extensively at high fidelity to test performance, operation and maintainability. Despite the extensive use of computational tools, they have a limited influence upon the design as most features are fixed by now (Keane & Nair 2005). Physical mock-ups and part models are assembled support testing and certification. Mission simulation is conducted at much higher level of fidelity because there is less uncertainty with regards to performance and reliability (Fielding 1999). At the end of the detailed design phase, manufacturing documents and part definitions are released for production (Sadraey 2012).

2.2 Value-driven design

Large engineering systems comprise several layers of sub- systems and thousands of parts (Quinn et al. 2012). During design of such systems, the system-level design intent can become diluted through Systems Engineering requirements flow down, a process whereby system-level requirements are broken into sub-system and component requirements (Collopy 2007; Forsberg & Mooz 1999). Component design teams focus on local optima instead of system capability

(Bandte 2000; Bertoni et al. 2013). Moreover, the Systems Engineering community advocates fixed design requirements. However, Collopy (2007) shows that design for fixed weight, cost, fuel burn, reliability, *etc.* leads to cost overruns. Value-driven design was conceived in order to focus design intent on system-level requirements and specifications throughout the design process. Moreover, it aims to relax fixed design requirements and employ economic principles for design optimisation instead. A common definition is:

> *Value-driven design is an improved design process that uses requirements flexibility, formal optimisation and a mathematical value model to balance performance, cost, schedule, and other measures important to the stakeholders to produce the best possible outcome*[1]*.*

Curran (2010) adds that value-driven design employs a "value function that best describes the value added of a product […]". The goal is to re-focus design efforts on system performance, neglecting "fast and cheap" sub-system optimisation which degrades overall system quality (Curran et al. 2012). By employing a single measure of overall system "fulfilment", i.e. a value, system-wide optimisation can be applied (Bertoni et al. 2013). This system view takes into account traditional performance and cost considerations but is flexible enough to incorporate other stakeholder interests such as the operational environment and life-cycle performance (Cheung et al. 2009). Moreover, value-driven design improves the Systems Engineering requirements flow down process by deriving objective (value) functions for sub-systems and systems (Collopy 2001; Keller & Collopy 2013).

The term "value" is not defined sharply as there is no consensus on its nature and how to determine it generically (Quinn et al. 2012). However, it is used as "the driver for decision-making" because it is a measure of preference of a system compared to another (Bertoni et al. 2013). As such, it lends itself for sub-system and system-wide optimisation (Cheung et al. 2010). Murman et al. (2000) define value as

$$Value = \frac{performance}{cost * \epsilon} \qquad \text{Eq. 2-1}$$

where ϵ is a delay factor relative to the time elapsed in reaching the market. In order to compare systems and to do optimisation, value calculation must be numeric, objective, repeatable

[1] See webpage of the Value-driven Design Institute at http://www.vddi.org/vdd-home.htm, accessed 17/12/2013.

and transparent (Cheung et al. 2010). Smulders et al. (2012) state that value should be relative instead of absolute because obtaining absolute numbers is difficult and not necessary. Relative values suffice for design comparison and optimisation (Keller & Collopy 2013). Most often, value is computed using monetary terms since it is "intuitive, meaningful and comparable" (Collopy & Hollingsworth 2011). Consequently, profit is the most frequent system objective function although Cheung et al. (2009) argues that design attribute objective functions are advantageous. Moreover, Bertoni et al. (2013) state that monetary terms are "largely meaningless" during the conceptual and preliminary design phase due to the high level of uncertainty on cost. Instead, Soban et al. (2011) advocate using scalars because they can include relational aspects on top of functional and physical aspects.

The system value is derived by use of a value model based on economic laws. Keller and Collopy (2013) compare it to an electric field in space that defines a scalar voltage to any point in space. Similarly, a value model assigns a scalar value to the high dimensional attribute space of the product where each point equates to a design. Practically, value models are problem-specific and there is no universal approach for developing a value model for a product.

Models vary in quality and reliability, not least because design information is limited during conceptual design (Mullan et al. 2012). Mission modelling can increase the amount of information during conceptual design, thereby increasing confidence in conceptual design value models. Another inherent value-driven design problem is understanding the relation between value drivers and value objectives during the conceptual design phase. What is the impact of the mean time between failures on product availability? How does profit change if one varies the wingspan? Operational mission modelling is the key tool to quantify component and sub-system impact upon life-cycle performance.

Curran et al. (2012) explicitly include operational aspects by introducing the VOM (Value Operations Methodology): here the focus is on the operational value for customers, enabling a "more realistic operations based performance assessment" (ibid.). VOM helps to understand optimal operations during design concept evaluation because future aeronautical products must be tailored for their intended missions more specifically (ibid). The mission-modelling framework presented in this thesis is a step into that direction.

2.3 Modelling

A concise definition of a model is given by (Leonard 2001):

> *A model is a physical, mathematical or logical representation of a system entity, phenomenon or process.*

Consequently, modelling is the act to create such a model. There are many model classifications: Engler (2013) categorises models by their level of fidelity as in Figure 2-3.

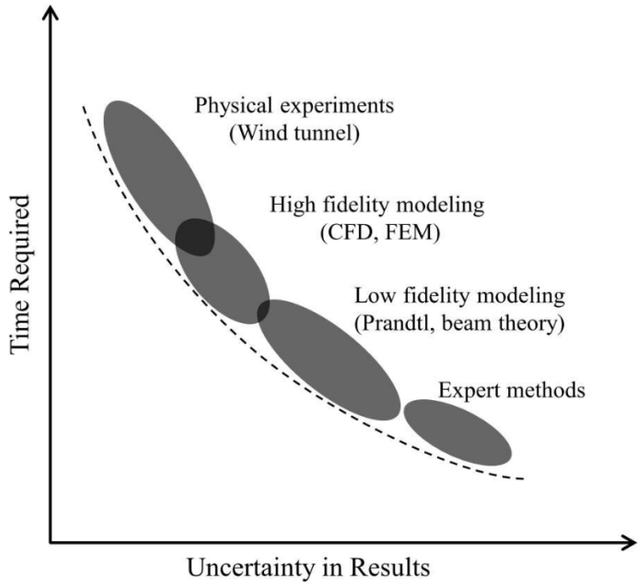


FIGURE 2-3: MODEL CLASSIFICATION BY FIDELITY. REPRODUCED FROM ENGLER (2013).

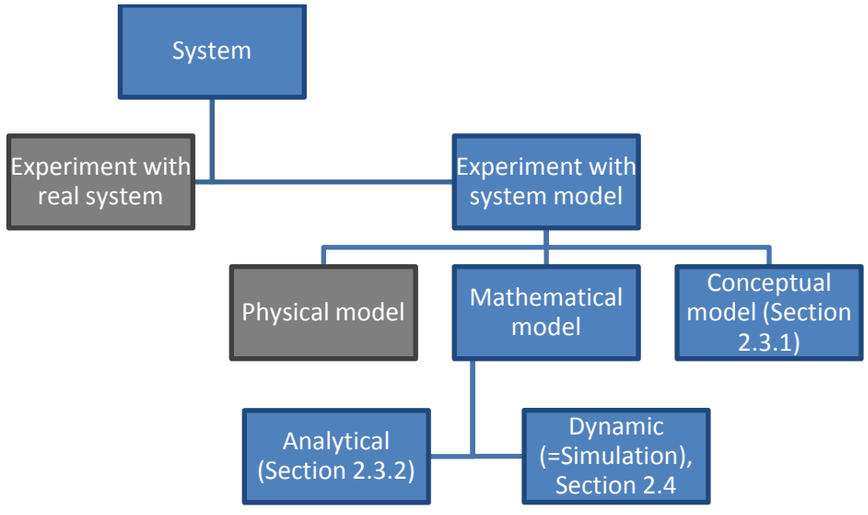


FIGURE 2-4: SYSTEM MODELLING OPTIONS. ADAPTED FROM LAW & KELTON (1997).

There is an inverse relation between result uncertainty and required modelling time: High-fidelity models (including physical experiments as models of very high fidelity) take much time but reward with low uncertainty while low fidelity models (including expert models with very low fidelity) reverse both characteristics. However, such ranking is subjective to some degree because there are no objective measures for computational model fidelity. Law and Kelton

(1997) provide a more objective classification based on rigorous model type boundaries as in Figure 2-4. If modelling a real system, this can be done physically, conceptually or mathematically. Aeronautical conceptual design rarely employs physical models because non-physical modelling provides substantial cost benefits (Backlund 2000; Steinkeller 2011). This section discusses conceptual modelling (Section 2.3.1) and analytical modelling (Section 2.3.2). Subsequently, Section 2.4 presents dynamic modelling (*i.e.* simulation). There is no consensus on the ideal model building process (Prilla et al. 2013). In general, processes aim to match the model with the prevalent paradigm and system problem to reduce cost (Yu 2008). Moreover, most processes recognise the importance of appropriate model fidelity: Ideally, a model is as simple as possible while recreating system characteristics. Fulton et al. (2003) display this idea as the "effectiveness frontier" when relating model articulation (i.e. fidelity) with effectiveness as in Figure 2-5.

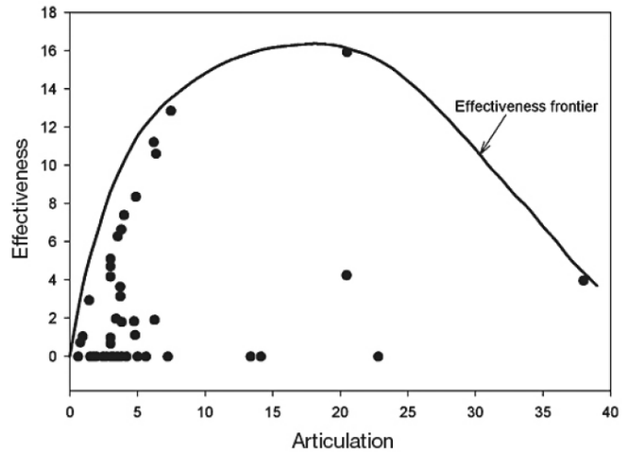


FIGURE 2-5: MODEL FIDELITY VERSUS EFFECTIVENESS. REPRODUCED WITH PERMISSION FROM FULTON ET AL. (2003).

There is an optimum effectiveness for models if they are not too simple and not too detailed. Boehm (1988) developed a comprehensive model development process emphasizing its iterative nature as in Figure 2-6. Each development iteration consists of a requirements phase, evaluation phase, development phase and integration phase until final model implementation. Engler (2013) reviewed and merged Boehms spiral process and several other approaches into a simple five-step procedure shown in Figure 2-7. The development of the framework and simulation in this thesis applies Englers five-step approach due to its ease of implementation and simplicity.

Table 2-1 details where each step is conducted in this thesis. Note that model testing for the OSCAR framework is done throughout framework development and by means of producing the practical implementation, namely the OSCAR simulation.

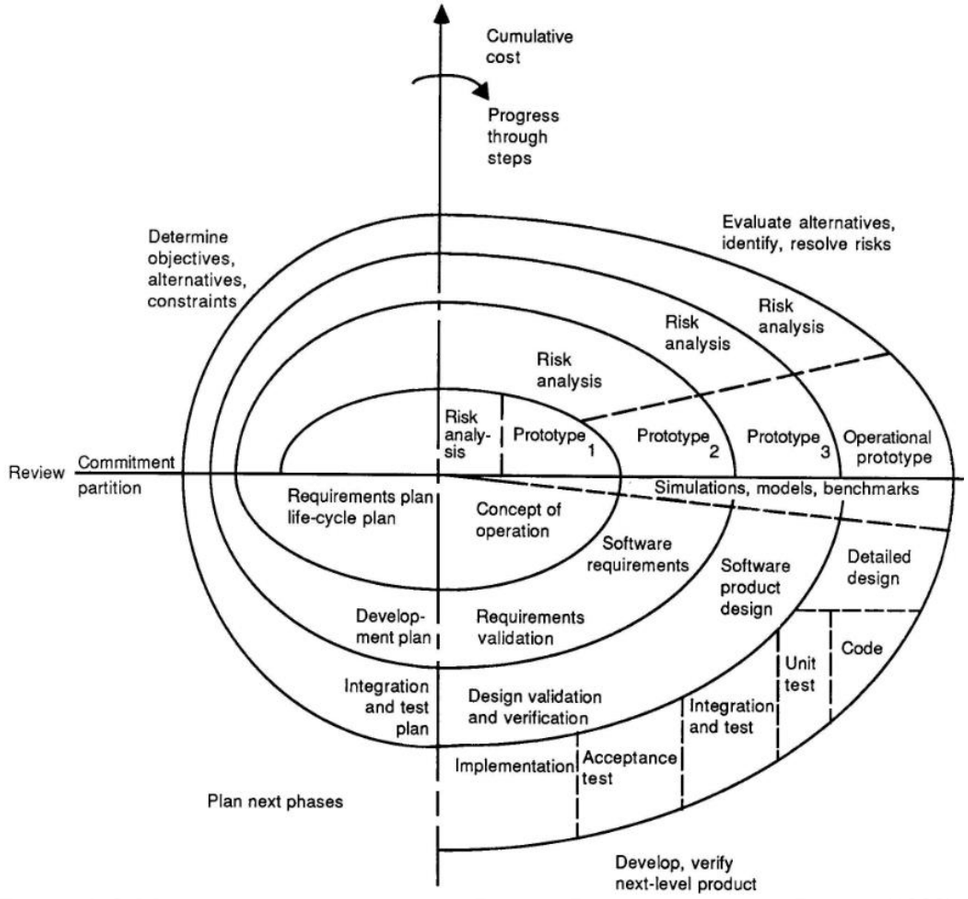


FIGURE 2-6: MODEL BUILDING PROCESS BY BOEHM. REPRODUCED FROM BOEHM (1988).

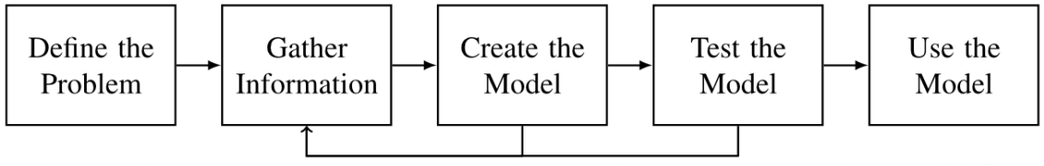


FIGURE 2-7: MODEL BUILDING PROCESS BY ENGLER. REPRODUCED FROM ENGLER (2013).

TABLE 2-1: MODELLING STEPS FOR OSCAR FRAMEWORK AND SIMULATION.

Step	OSCAR framework	OSCAR simulation
Define the problem	Section 3.1	Section 4.1 and 4.2
Gather information	Sections 3.2.1, 3.2.2, 3.3.1 and 3.3.2	Section 4.5
Create the model	Sections 3.2.3 & 3.2.4 and 3.3.3- 3.3.6	Sections 4.6-4.12
Test the model	by means of OSCAR simulation	Section 4.13 and Chapters 5 & 6
Use the model	by means of OSCAR simulation	Chapters 5 & 6

2.3.1 Conceptual modelling

Conceptual modelling is arguably the most basic modelling technique used by anybody every day. Stevenson (2002) distinguishes three types of conceptual models:

- *Mental models*: The brain creates and uses mental models all the time: While driving, it assesses if (and when) another car will hit. Mental models are usually difficult to verbalise.
- *Verbal models* are used early in product design but suffer from ambiguity and ontological barriers (Steinkeller 2011).
- *Schematic models* are also used during early product design to show relationships between entities. An everyday example of a schematic model is a tube map.

Despite their disadvantages, the simplicity and clarity of conceptual models are very useful. They force users to organise and possibly quantify information for later computational models. Moreover, they support problem understanding and offer a systematic approach to problem solving. However, conceptual models can only be a starting point as they lack quantification capabilities. Therefore, more specific models are required.

2.3.2 Analytical modelling

Analytical models apply formulae in order to produce numerical solutions to a well-defined problem (Jinks 2012). However, the main analytical model tool used in aeronautical conceptual design is spread sheet modelling (Keane & Nair 2005). Its apparent flexibility and ease-of-use made it pervasive during the last three decades (Panko 2000). However, most spread sheet modelling is conducted ad-hoc with weak structures, frail scientific unit assignment and misleading graphical front-ends (Scanlan & Rao 2006). Moreover, the error rate in industrial spreadsheets is unacceptably high with some empirical evidence quantifying cell error rates between 1-2% (Panko 2008). Although aeronautical engineering lends itself well to hierarchical applications, hierarchical modelling is difficult with spread sheets (Paine 2000). Even if companies provide best practice guidelines, spreadsheets are inherently hard to comprehend and validate. Knowledge and rationale capture hides in sheets, cells and comments. Moreover, it is difficult to include uncertainty and stochastic simulation within spreadsheet modelling (Streit et al. 2008). Object-oriented hierarchical spreadsheet modelling requires additional effort: Developers must resort to the underlying programming language, creating objects akin to simulation modelling such as queues, event lists, entities and resources (Engler 2013). Therefore, the OSCAR simulation is not based on spreadsheet modelling but employs a purpose-build simulation-modelling tool.

2.4 Simulation

Simulation modelling is widely applied in aeronautical design (Abbas-Bayoumi & Becker 2011). Leonard (2001) describes simulation as the "implementation of a model over time". Conducting a simulation "brings the model to life and shows how a particular phenomenon or object behaves". A more formal definition combining Law and Kelton (1997), Steinkeller (2011) and Robinson (2004) can be given as:

> *Simulation modelling is an analysis method where computers are used to evaluate a model numerically in order to estimate the desired true characteristics of the model. It is a computer experiment performed upon a model of an operational system as it progresses through time.*

Computational simulation models have been used for as long as computers exist. However, before the 1980s, models lacked even basic animation capabilities. Most models were developed for one specific purpose without any reusability(Andersson & Olsson 1998). During the 1980s, simulation systems grew larger, relying on early simulation tools. However, results were prone to error due to a lack of analysis capabilities and useful animation (Steinkeller 2011). The 1990s saw a boost in application because the rise of the PC enabled useful animation outputs, improved analysis and object-oriented work.

According to Rubinstein (2011) there are three main classifications for simulation models:

- *Static or dynamic models*: In static models, nothing changes over simulation virtual time and output is constant if input does not change. The OSCAR simulation, on the other hand, is a dynamic model where variables change over time and output depends on the advance of the simulation virtual time.
- *Deterministic or stochastic models*: In deterministic models, no model component contains stochastic uncertainty and the output is always the same. In stochastic models, one or more components are defined with some uncertainty. Re-running the model with a different random number seed will (in general) produce different results. The OSCAR simulation is a stochastic simulation model because aeronautical design features many uncertainties. OSCAR simulation random number seeds are explained in Section 4.13.2.
- *Time-driven or event-driven models*: In "event-driven" simulations, time advances in discrete steps whenever an event occurs in the virtual simulation environment. The simulation software usually provides a background list of all events with rules on how to handle multiple events at the same time. In "time-driven" simulations, time

advances smoothly, independent of events occurring in the simulation environment. However, the binary nature of computer processing forces such simulations to "slice" time into very small units to give the impression of time passing smoothly. "Time-driven" simulations are inefficient if simulation periods without any events are common. The OSCAR simulation is "event-driven" because it is supposed to span years or decades of vessel operations featuring long periods without events.

Figure 2-8 shows the simulation methods applicable for time-driven and event-driven models.

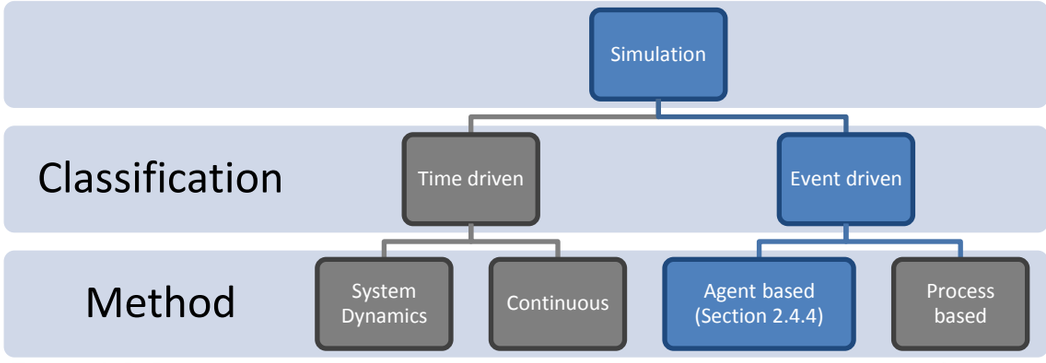


FIGURE 2-8: SIMULATION METHODS FOR TIME-DRIVEN AND EVENT-DRIVEN MODELS. ADAPTED FROM JINKS (2012) AND YU (Yu 2008).

Most "event-driven" simulations apply either agent-based methods or process-based methods. The latter primarily applies in manufacturing and scheduling applications were physical "entities" (products, parts, humans) move through a system with limited processing resources, creating queues and bottlenecks that are of interest to the modeller. The OSCAR simulation, on the other hand, applies the agent-based method for reasons discussed in Section 2.4.4.

Aeronautical conceptual design exhibits many different simulation models, often with overlapping scopes and legacy compliance (Bandte 2000; Glas 2013). According to Steinkeller (2011), good conceptual design phase models should be flexible enough to be used throughout the entire design process to avoid migration losses. However, most models are specific and part of extensive legacy vehicle sizing and synthesis codes. Their level of fidelity and quality varies strongly depending on model development history (Mavris & Kirby 1999).

2.4.1 Advantages

Using a simulation model during conceptual aeronautical design promises a number of advantages. Often, it is infeasible to construct physical prototypes for testing due to time constraints and lack of knowledge. Simulation modelling can reduce costs considerably compared to physical testing (Glas 2013). Moreover, designers can observe life-cycle variables that are impossible to measure in reality, not least because time scales can be extended or shortened.

Simulations are designed to include uncertainty and stochastic analysis methods. Stochastic simulations provide outcome distributions, taking into account extreme values. According to

Mavris et al. (1999) this can be used to maximise the probability of achieving a certain merit ("robust design simulation"). Simulation enables uncertainty at any level of detail for conceptual aeronautical design, namely operational, system and component levels.

Moreover, simulation modelling offers a number of advantages in a business environment. It can foster design creativity because experimentation poses little risk (Yilmaz & Hunt 2011). Designers are encouraged to think through details and interactions. Managers can use design review meetings to understand and question design decisions using simulation outputs and animation.

Animation capabilities have been advocated since the 1970s by Hurrion (1976). During the 1980s, "Visual Interactive Simulation" enabled model users to interact with the simulation during runtime based on visual feedback (Bell & O'Keefe 1987). Today, animation is a core feature among most simulation packages. Animation simplifies verification and fosters some degree of validation (see Section 4.4.2 for definitions of verification and validation). Moreover, it helps in communication between modeller, user and customer (Rohrer 2000).

2.4.2 Disadvantages

Simulation modelling features distinct disadvantages as well. In fact, Pidd (1998) sees simulation as a "last resort". Despite the cost advantage over physical modelling, simulation is expensive due to expert knowledge required (Rubinstein & Kroese 2011). Developing and running simulations is time consuming because stochastic outputs require many runs to produce valid results. This complicates effective optimisation during conceptual design (Robinson 2004). Moreover, results are only valid if the model is a valid representation of the real system. In addition, simulations tend to require much input data, a lot of which is unavailable or inaccessible during conceptual design.

Another common problem with simulations is over-confidence: The output animation and data conceal that all outputs are statistical estimates subject to experiment error (Rubinstein & Kroese 2011). Professional expertise is required to manage user expectations. The opposite effect is also common in conceptual design: Engineers are reluctant to apply a simulation model recognising that all models are wrong but neglecting that some are useful (Box & Draper 1987; Hybertson 2010). Models are rejected because learning is too laborious, company guidelines are missing and validation difficulties foster distrust (SAE 1998).

Many simulation models succeed to inform about *what* happens (*i.e.* produce outputs) but struggle to convey *why* (Karban et al. 2008; Wheeler & Brooks 2007). In addition, objectively assessing model quality is hard because there are very few generic neutral model metrics such as lines of code (Murphy & Collopy 2012; Yu 2008).

Moreover, it is inherently difficult to judge when model development is finished. In other words, it is difficult to specify the "correct" level of fidelity for simulations (Cassidy et al. 2008). However, model fidelity is especially important during the conceptual design phase due to time and design knowledge constraints. Therefore, fidelity definition for the OSCAR framework and simulation is tuned carefully to conceptual design phase requirements (Sections 3.1 and 4.2). Despite these disadvantages, the OSCAR framework advocates use of simulation modelling as discussed in Section 4.1.

2.4.3 Requirements

Applying a simulation model requires certain prerequisites. The simulation must be repeatable. If the same simulation setup runs on two different computers, the same results must be obtained despite stochastic inputs. Therefore, the simulation must feature fixed random number streams, preferably custom-defined as with the OSCAR simulation (Section 4.13.2).

The simulation should allow automatic repetition of the same simulation model with different random numbers from a custom-defined random number stream. The number of repetitions should be user-defined (fixed number, stop upon confidence level reached, *etc.*). Such repetitions are called *replications* in the rest of this thesis.

The simulation should be able to run in batch mode without animation straining computational efforts. Ideally, multiple replications run on different processor cores independently. This speeds up stochastic replication runs dramatically without suffering from the usual problems of parallel computing (multi-threading, synchronisation, *etc.*).

The simulation model should allow clear and transparent simulation structure generation following physical realities. This promotes model validation and user trust.

Furthermore, it is good practice to strictly separate a simulation model from input and output data. This "data-driven generic modelling" (Jinks 2012) is user-friendly because users can amend input data and experiment on outputs without detailed model knowledge. Moreover, separate data allows improved model version control and data post processing independent from the model (ibid). Not least, the model can be used to simulate other structurally similar systems by applying different inputs (Pidd 1992). The OSCAR simulation demonstrates this capability by modelling aeronautical vessels as well as any other moving vessel such as trains, cars or boats.

2.4.4 Agent-based Simulation

This section introduces the agent-based modelling approach and its applicability for the OSCAR framework and simulation. Creating agent-based simulation models is a relatively recent development that has its roots in complex systems research (Weisbuch 1991). Scientists tried to

apply new object-oriented programming techniques to solve long-standing issues in artificial intelligent systems (Jennings et al. 1998).

In agent-based modelling, the modeller specifies autonomous entities (i.e. *agents*) by their behaviour, thus allowing "bottom-up" model development instead of the conventional "top-down" modelling approach (Macal & North 2010). Jennings (2000) defines an agent as

> *Agent: An encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.*

Parunak et al. (1998) add that agents "correspond one-to-one with the individuals being modelled, and their behaviours are analogs of the real behaviours". Wooldridge and Jennings (1995) categorise agents as either *deliberative* or *reactive*:

- *Deliberative agents* reason rationally from environmental perceptions and predict the impact of their action. Such agents can mirror human reasoning based on symbolic artificial intelligence. However, they suffer from the computational effort to translate complex real world information into computational "symbols" used for reasoning (Chapman 1987). Deliberative agents can be used to observe "emerging" phenomena not specifically coded into the model. The social sciences exploit this capability since it lends itself to modelling human behaviour well (Batty et al. 2012; Davidsson 2002).
- *Reactive agents*, on the other hand, respond to simple environmental cues without rational reasoning. Instead, they retrieve pre-programmed behaviour quickly. Here, the modeller must anticipate and specify all problem situations fully during model design. Reactive agents allow intuitive model building for real systems with limited system knowledge. This approach is used for the OSCAR simulation. The reason is that the agents used within OSCAR (aircrafts, lifeboats, trains, components, *etc.*) do not conduct rational reasoning but follow operational rules only.

According to Macal and North (2010), the typical structure of an agent-based model consists of three items:

- There is a set of agent classes and agent instances with attributes and behaviour.
- Agents relate to each other through connections and interaction methods.
- Agents act within and interact with an environment.

Macal and North (2010) identify five agent environment types, namely cellular automata, Euclidian 2D/3D space, networked, non-spatial "soup" and Geographical Information System environments. Applying a Geographical Information System environment allows influencing and controlling agent behaviour (see next Section 2.4.5). Since aeronautical products react and in-

teract with their physical environment in many ways, the OSCAR framework and simulation incorporate a Geographical Information System environment for agents (Sections 3.2.4 and 4.6).

Agent-based modelling has the highest possible TRL (Technology Readiness Level) of nine (Yu 2008) and it is applied more and more frequently in engineering (Chen et al. 2012). Bussmann et al. (2004) describe manufacturing optimisation for car engine cylinder heads using agents. In military applications, agent-based modelling is a tool for battlefield simulation in order to support tactical and strategic decisions (Cioppa et al. 2004). During previous years, agent-based modelling is used more frequently for ATM (Air Traffic Management) research because human decision-making dictates operations within a technical environment. As such, Niedringhaus (2004) describes the "Jet:Wise" model where agents are used to model the US airspace system at high fidelity. It predicts future airline decisions based on air space policy changes. Wieland and Satapathy (2010) continue the effort describing the substantial effort behind the agent-based architecture. Bosse et al. (2013) present an agent-based model for hazards modelling in commercial air traffic control as part of the SESAR (Single European Sky Air Traffic Management Research) development. In aeronautical engineering, the commercial tool "aerogility" [2] applies agent-based modelling to aftermarket and operational support services. Aerogility can model airline operations from warehouse level to business processes by applying custom-build "intelligent" agents. However, there is no research on applying agent-based modelling using a Geographical Information System environment in conceptual aeronautical design.

Agent-based modelling offers several advantages for conceptual aeronautical design. The bottom-up modelling approach enables model building with limited system information and model amendment from gradual information increase (Macal & North 2010). Moreover, Yu (2008) demonstrated that for complex models with many entities, an agent-based model runs faster than a similar process-driven model in a conceptual design phase setting.

However, agent-based modelling features disadvantages as well. Model validation can become more difficult than for process simulations because "emergent" behaviour can occur without apparent explanation. This contrasts with engineering expectations of "predictable" model results providing quantitative indications but no new qualitative insight. Some agent-based modelling tools define a separate computing thread to each agent (multi-threading). This can cause undesired behaviour if threads are not synchronised (Yu 2008).

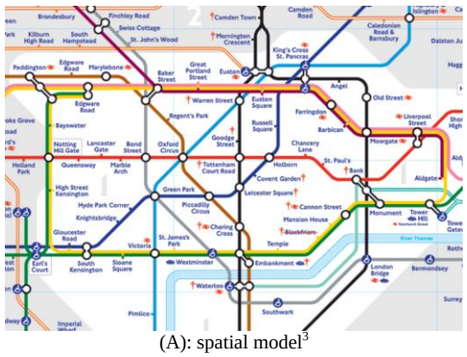
This thesis is the first work studying agent-based modelling for mission simulation in conceptual aeronautical design.

See [2] http://www.aerogility.com/, accessed 19/12/2013.

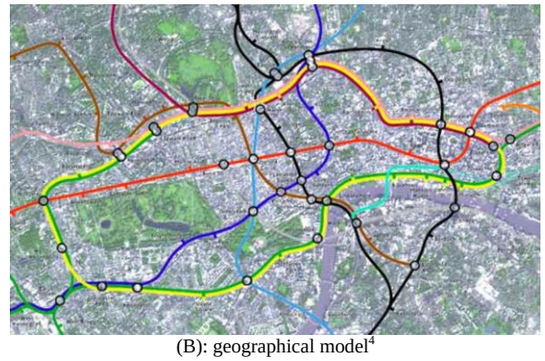
2.4.5 Spatially explicit simulation

Spatially explicit simulation is a specific type of simulation. It emphasises spatial details as important features for simulation output. There are two types of spatially explicit models:

> *Spatial models: order components and actions spatially* relative *to each other without geographical reference (see Figure 2-9 (A)). This allows analysing spatial relations for which geography is not important.*
>
> *Geographical models: order components and actions spatially with absolute geographical reference as in Figure 2-9 (B).*



(A): spatial model[3]



(B): geographical model[4]

FIGURE 2-9: SPATIAL AND GEOGRAPHICAL MODELS.

Computer-based analysis of spatial patterns dates back to the 1960s. Until the 1990s, spatially explicit models focussed on aggregate information lacking specific detail (Batty et al. 2012). Agent-based modelling offers more detailed spatially explicit simulation results because agents can interact with spatial details directly. Moreover, the agent-based modelling community became interested in spatially explicit models because traditional agent models are limited to grid-based spatial patterns only (Birkin & Wu 2012; Crooks 2008). However, spatially explicit model users do not usually work with agent modelling tools while most agent modellers are unfamiliar with spatially explicit modelling tools (Rand et al. 2005). Therefore, there are few agent-based models where agents interact with a spatially explicit model directly. In order to overcome the disparity, Brown et al. (2005) suggest to develop a middleware because both paradigms feature demanding software packages that are difficult to master simultaneously. However, they admit that this may not be the best long-term solution.

[3] Source: http://www.tfl.gov.uk/assets/images/general/standard-tube-map.gif, accessed 23/10/2013
[4] Source: (http://www.steveprentice.net/tube/TfLSillyMaps/tubegeo.jpg, accessed 23/10/2013

Today, spatially explicit agent-based models often evade to drawing spatial patterns manually within the simulation package. Adelantado (2004) describes an agent-based airport simulation where the airport layout is drawn manually. For the OSCAR simulation, a manual approach would increase the setup and preparation time for users considerably while reducing genericity and flexibility (Section 3.1). Moreover, model re-use becomes more difficult because spatial objects are internal to the simulation model. The separation of model and data would not hold anymore.

Zhu and Sala-Diakanda (2007) describe an alternative method using the AnyLogic software also applied in this thesis (Section 4.3.2). Here, an agent-based model is linked dynamically to a geographical modelling program. Agents send their spatial position to the program and receive information about their spatial status (in this case their depth under water). Based on the feed-back, agents decide future actions. However, this approach is computationally very expensive and not useful for large-scale life-cycle simulations as conceived for this thesis. Therefore, this thesis develops a unique method of geographical model implementation into AnyLogic as described in Sections 3.2.4 and 4.6.

2.5 Mission Simulation

This section discusses developments in aeronautical conceptual design mission simulations. Testing designs performing their intended missions in a virtual environment is a very important factor in contemporary aeronautical design. The number of complex interdependent factors and variables is very high even for low fidelity mission models, easily exceeding human analysis capability (Heilala & Maantila 2010; McLean & Leong 2001). Kirby (2001) sets an "unconstrained mission analysis" capability as a very high priority for modern simulation environments because it would provide cost savings, faster design lead times and improved product quality. Price et al. (2006) states that the biggest challenge of design today is to account for the "wider system" in which aircraft operate, i.e. ATM, regulations and missions. This "wider system" is commonly viewed as a design constraint. However, it could be treated as a design parameter instead. In order to use mission definitions as design parameters, they need to be modelled realistically. Chapter 6 demonstrates the use of mission definitions as design parameters using the OSCAR simulation.

The "Probabilistic System of Systems Design Methodology" developed by Soban (2001) sees aircraft as a system that is part of a larger system that includes the operation, "campaign" and the entire life cycle (Soban 2001; Soban & Mavris 2000a; Soban & Mavris 2000b). Applying an object-oriented approach, aircraft objects are used as part of mission-objects that make up a "theatre" of (life-cycle) operations. This allows measuring the "goodness" (similar to "value" in

value-driven design) of the aircraft earlier in the design process. However, the level of mission fidelity is low because the focus of the work is to include probabilistic variables into the design process.

This thesis applies a similar “system of systems” approach where vessel objects act on mission objects that make up a life cycle. However, the focus is on explicit mission modelling and generic vessel definitions.

2.5.1 Current practice

Operational scenarios and missions are defined broadly at the start of the conceptual design phase (Amirreze et al. 2013). The exact mission definition is critical in determining if the design is feasible and viable (Cassidy et al. 2008; Delaurentis et al. 1996). Typically, mission scenarios are defined deterministically without any uncertainties despite the variable real environment with weather, fuel price and, policy uncertainties (Frangopol & Maute 2003). Moreover, mission definitions are not detailed during conceptual design because it is deemed inappropriate and fast model responses are required (Duquette 2009). While mission parameters are fixed, vessel parameters vary for sizing optimisation runs (Bond & Ricci 1992). More specifically, a point-mass object follows a mission profile until given requirements such as fuel burn and thrust are met within given tolerances. This is repeated with different vessel parameters until a satisfactory result occurs (Kirby 2001). The point-mass dynamics base on standard textbook equations such as Raymer (2006) and Torenbeek (2013).

Current conceptual design phase mission-modelling techniques suffer from several disadvantages. Mission profiles are often the sum of simplistic parametric modules such as “take-off”, “cruise” or “loiter”. These modules recreate average missions without specific detail. Often, the chosen mission profile determines the output design trade-offs (Keane & Nair 2005). Conceptual design phase optimisation uses mission simulation to tweak aircraft design parameters such as weight, aerodynamics and structural performance. However, it neglects optimising mission profiles themselves (Scanlan & Rao 2006). Moreover, the choice of modelling tools and methods is largely based on company tradition: This leads to suboptimal model allocation with high fidelity models and low fidelity models mixed and matched as seen fit (Krus & Jouannet 2010).

The rest of this section introduces a number of tools (commercial and non-commercial) as well as recent research on conceptual design phase mission simulation.

2.5.2 Tools

2.5.2.1 Pacelab Mission Suite

Pace[5] is a software development company for the aircraft industry. It offers knowledge-based engineering COTS tools for all design phases as well as aircraft operations support (Glas 2013). One product on offer is the "Pacelab Mission Suite". Customers can setup simple or more complex mission scenarios including routes, fuel policies or ETOPS requirements. Aircraft manufacturers use the tool to demonstrate product capabilities to potential airline customers. Missions can be defined geographically or schematically while aircraft performance is computed using first order or second order principles. The suite provides sophisticated plug-ins to simulate realistic operations: it can apply real airport departure and arrival routes, average weather conditions and real flight routes.

However, the tool is designed for use by sales teams at the end of the design process or during production. It is not integrated into an aircraft design environment, although it could be used for such purposes (and current business plans work towards that goal). In addition, only one aircraft instance can be analysed at a time, although linear fleet extrapolation is possible. Moreover, all Pacelab products are specific to commercial, subsonic, civil transport aircraft and do not offer a large degree of flexibility to include a wider variety of aeronautical vessels. Not least, the Pacelab mission suite does not support stochastic inputs because uncertainty is irrelevant to sales teams.

Despite its target group, components of the Pacelab mission suite are used in conceptual aircraft design: It is one of the most advanced mission simulation tools on the market.

2.5.2.2 PIANO

Piano[6] ("Project Interactive Analysis and Optimisation") is a COTS aircraft synthesis tool for conceptual and preliminary design developed by an individual over the past 20 years. It has been used by large aerospace companies such Boeing and Airbus. Piano contains all relevant modules for conceptual aircraft design such as geometry definition, mass estimation, aerodynamics, emissions and engine modelling. Moreover, it provides a "range & mission performance" module that computes performance from first order principles. User can specify generic parameterised mission blocks such as climb, cruise and descent. As with most conceptual design phase tools, spatial details are neglected and only one aircraft instance can be analysed at any one time (i.e. no fleets, no competitors, etc.). Moreover, Piano is designed for conventional,

[5] See www.pace.de, accessed 20/12/2013.
[6] See http://www.lissys.demon.co.uk/, accessed 20/12/2013.

commercial and subsonic transport aircraft, lacking genericity and flexibility to include other designs easily.

2.5.2.3 ACS

The "AirCraft Synthesis tool"[7] (also abbreviated ACSYNT) has evolved over the past 40 years from Fortran-based NASA research at the Ames Research Center (Cassidy et al. 2008). It is similar to Piano in that geometry, aerodynamics, propulsion and weight modules enable design analysis and synthesis. Users define parameterised mission building blocks such as take-off, climb or cruise. The software computes performance at every point for a single aircraft instance. Unlike Piano, ACS allows wider flexibility concerning aircraft designs, including support for non-conventional configurations such as flying wings, supersonic aircraft or UAS.

2.5.2.4 RDS

Developed by D. Raymer based on his well-known book "Aircraft Design: A conceptual approach" (Raymer 2006), RDS[8] is a conceptual aircraft design tool. It is very similar on scope and functionality to ACS, allowing flexible configurations but limiting mission simulation to parameterised building blocks.

2.5.2.5 FLAMES

The "FLexible Analysis Modelling and Exercise System" is a COTS generic simulation framework developed by Ternion[9]. FLAMES is able to model a wide variety of moving systems such as aircraft, helicopters, humans, cars, etc. It aims to model complex military battlefield simulation applications. Unlike Piano, ACS or RDS, it applies object-oriented principles to create truly flexible and geographical mission scenarios. However, FLAMES does not provide support for aeronautical design activity but for tactical and strategic operational decisions. Cassidy et al. (2008) developed a work-around to use FLAMES (together with ACS) for design of a single military aircraft using a single mission. However, integration is not straightforward and complex 3D environments are beyond conceptual design phase requirements both computationally and functionally.

2.5.2.6 FLOPS

The "FLight OPtimisation System" is a public-domain multi-disciplinary sizing tool for the conceptual and early preliminary design phase (McCullers 1995). It was developed by the NASA Langley Research Center and includes nine modules: weight, aerodynamics, engine cy-

[7] See http://spinoff.nasa.gov/spinoff1997/ct11.html, accessed 20/12/2013.
[8] See http://www.aircraftdesign.com/rds.shtml, accessed 20/12/2013.
[9] See http://ternion.com/, accessed 20/12/2013.

cle analysis, propulsion scaling, take-off & landing, noise, cost and mission performance. Most modules are based on low fidelity empirical methods conforming to conceptual design phase practice. Mission definition is similar to Piano, ACS and RDS in that parameterised non-spatial mission blocks are combined. FLOPS supports vehicle design optimisation but does not allow mission definition changes, i.e. it cannot automatically vary mission parameters for optimisation (Delaurentis et al. 1996). The FLOPS code is easily extensible and applied in many research projects.

2.5.2.7 Other tools

There are a number of other tools available for conceptual aircraft design. CEASIOM[10] emphasises aircraft stability and control issues during conceptual design, providing enough detail to use 6DoF (six degrees of freedom) flight simulations. The "Advanced Aircraft Analysis" tool by DARcorporation[11] also focuses on stability and control estimation. The "j2" tool kit by j2 Aircraft Dynamics Ltd.[12] Allows to investigate several design candidates in parallel.

These tools do not provide purpose-build mission modelling modules. Instead, they provide plugins to link designs to COTS 6DoF flight simulators. This approach enables very detailed simulation results for one specific flight. However, it is difficult to set up a number of missions for a fleet of aircraft to model an entire life cycle.

2.5.3 Research

This section reviews recent scientific advances in early design phase mission simulation. Nalepka and Duquette (2003) developed a human-in-the-loop-oriented programming mission simulator to explore operational issues for military UAS. Their simulation framework includes five core performance modules, namely move, sense, communication, shoot and interfere. However, the simulation is used for short-term tactical mission analysis rather than UAS design.

Nilubol (2005) created a similar framework for combat aircraft but was first in including maintenance, vulnerability, performance and cost into a unified conceptual design phase tool. At the heart of the framework is the "Operation Mission Simulation" (Figure 2-10). However, mission simulation is not generic: instead, users can choose from five mission types with fixed parameters. The simulation is process-based and does not allow modelling of several aircraft instances. Moreover, missions have no waypoints or spatial distribution.

[10] See http://93.88.249.84/index.php, accessed 16/01/2014.
[11] See http://www.darcorp.com/Software/AAA/, accessed 16/01/2014.
[12] See http://www.j2aircraft.com/application-benefits/conceptual-and-preliminary-design-2/, accessed 16/01/2014.

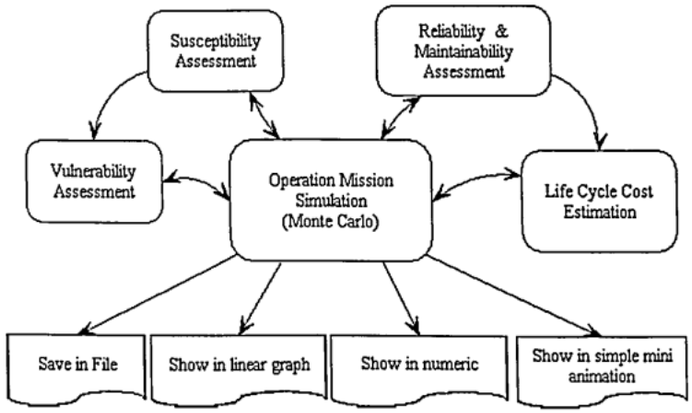


FIGURE 2-10: COMBAT AIRCRAFT SIMULATION FRAMEWORK FOCUSSING ON OPERATIONAL MISSION MODEL. REPRODUCED FROM NILUBOL (2005).

In order to test UAS control algorithms, Duquette (2009) developed the "AMASE" mission simulator. It can model a fleet of UAS but approximates flight performance with second order look-up tables. It includes simplified climb, descent and turn performance as well as detailed loiter options (orbit, figure-eight, racetrack, etc.). "AMASE" focuses on spatially explicit modelling but excludes geography because UAS control algorithm testing does not require geography.

Thokala (2009) developed a process-centric discrete event mission simulation to compute improved life-cycle cost estimates during the conceptual design phase. Here, missions are limited to a number of pre-defined mission types. Aircraft performance applies standard flight dynamics equations and atmospheric tables. In addition, it applies simplified CFD (Computational Fluid Dynamics) analysis for conventional designs and limits propulsion to turbojet engines. As the focus is on life-cycle analysis, the model allows defining an entire aircraft life cycle. However, only one aircraft instance is modelled and missions cannot change dynamically (e.g. cancel due to fuel shortage, *etc.*).

Krus and Jouannet developed a more detailed mission simulation (Krus 2011; Krus & Jouannet 2010). It focuses primarily on preliminary design phase applications employing a parallelised mission simulator with 6DoF aerodynamics. The goal was to create more design knowledge during the early design phases by including concurrent sub-system design through detailed simulation. However, to keep runtimes acceptable, sub-system models are computed in parallel on different cores. Moreover, time-compression algorithms reduce runtime by using simplified performance models in steady flight. Missions occur spatially distributed but require manual drawing by the user. The idea of parallel mission simulation execution could be useful for detailed conceptual design phase simulators. However, module split will be more difficult because sub-systems are not clearly defined during conceptual design.

Cassidy et al. (2008) used simple operational simulations to determine (but not optimise) critical mission parameters such as range and speed. They recognised the importance of spatial factors on vessel design but the mission simulation was not flexible enough to test and compare different mission scenarios. However, it is one of the few approaches identifying the effect of design changes on mission effectiveness.

Mission modelling has been applied for different purposes beyond aeronautical design as well. Royo et al. (2013) coupled the commercial flight simulator X-Plane with an ATM simulation framework developed by EuroControl in order to test UAS operations within non-segregated airspace. Here, the UAS model includes component deterioration and interactions to observe possible UAS performance influence upon the ATM environment and vice versa. However, this approach requires much computing power for 3D animation.

In summary, there are many existing commercial and scientific approaches to mission simulation. However, scope and fidelity vary widely. No tool allows to model life cycles using a Geographical Information System environment, several aircraft instances (fleets, backup aircraft, *etc.*) and a largely generic definition of aircraft vessels. The rest of this thesis will introduce the OSCAR framework and simulation aiming to fill this gap.

3. FRAMEWORK

This chapter presents the theoretical OSCAR framework developed to support conceptual design phase decision making and optimisation. It spans the intellectual process converting real vessels and scenarios into framework building blocks and constructs. Essentially, the framework is a set of simplifications, schemas, ontologies and recommendations. Users can apply and implement it according to their needs. They can adopt parts of the framework that enhance existing processes or decide to implement the whole framework based on existing computer system requirements. The OSCAR framework is the theoretical foundation for the sample implementation ("OSCAR simulation") presented in Chapter 4.

A core part the OSCAR framework is the organisation of knowledge. There are several existing schemas for conceptual aircraft design. Boehnke et al. (2012) define the Common Parametric Aircraft Configuration Schema (CPACS), a hierarchical schema describing characteristics of aircraft, rotorcraft, engines, fleets and missions. However, definitions target conventional civil airliner design (Glas 2013). Moreover, CPACS follows a top-down approach where most detail is specified at high-level design constructs (Deshpande et al. 2013). However, OSCAR requires more flexibility concerning vessels as well as more detailed low-level specifications. Deshpande et al. (2013) present the "Aircraft Design Markup Language" (ADML) that specifies low-level design constructs in more detail. However, its mission representation is based on typical segments such as take-off, cruise or landing, neglecting spatial information. Therefore, a unique schema is presented here that incorporates the requirements of geographical modelling and generic vessel representation but still overlaps to a large degree with existing schemas.

Section 3.1 presents the requirements for a useful design framework posed by conceptual design phase procedures and processes. The actual OSCAR framework consists of two branches: Section 3.2 details the *mission*-related (or *scenario*-related) framework components while Section 3.3 presents the *vessel*-related framework elements.

3.1 Requirements

Current aeronautical design processes evolved over decades and are well established in the industry (Bond & Ricci 1992). Any procedural change disrupts the conduct of design: hence, he benefits of change must be larger than the cost of implementation. While most of this thesis discusses the benefits of change, this section covers the cost of implementation. To minimise the cost of implementation, Nurminen et al. (2003) identified several key characteristics for engineering expert systems to be successful:

- Systems should support experts instead of trying to do their work and replace them.
- Usability is more important than automation.
- Technical knowledge capture works best using object-oriented principles.
- Pure spread sheet modelling becomes expensive in the long run because "[…] it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail" (Maslow 2002).

Engler (2013) argues that conceptual design phase tools must be flexible and fast to help reduce the vast design space efficiently. Based on these insights, this section identifies four requirements that the conceptual design phase process demands from a mission-modelling framework (Figure 3-1).

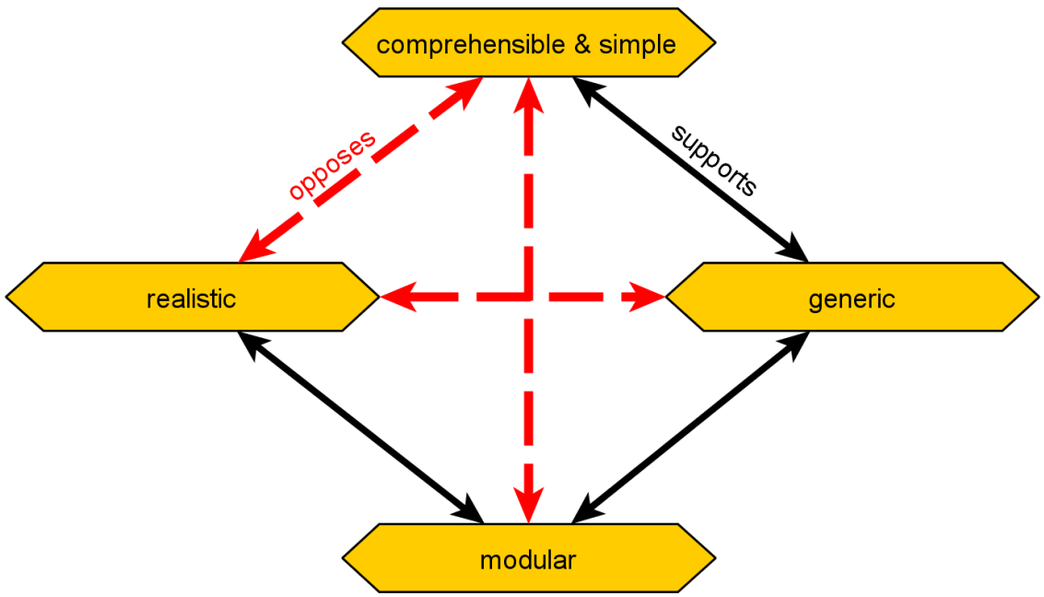


FIGURE 3-1: OSCAR FRAMEWORK REQUIREMENTS RELATIONSHIPS.

First, the framework should be as *generic* as possible to accommodate the largest variety of designs (Section 3.1.1). Second, the framework components and interactions must be *simple* and *comprehensible* to allow quick setup and easy application (Section 3.1.2). Third, applying the framework should return *realistic* results to merit its use and convince decision makers (Section 3.1.3). Last, the framework should be *modular* to allow easy adaptation and extension (Section

3.1.4). A perfectly realistic model is hard to comprehend and highly specific (i.e. not generic). However, perfect realism can be achieved through a modular approach. A (completely) modular framework is hard to comprehend but it can be generic. Similarly, a fully generic framework can be comprehensible and simple. A discussion of each of the four requirements follows below.

3.1.1 Genericity

The aeronautical conceptual design phase is characterised by synthesis and analysis of different design concepts. By definition, this requires a certain degree of generic capabilities to accommodate all design ideas. However, the OSCAR framework is not limited to one manufacturers set of conceptual design ideas. Instead, the framework intends to support the entire aeronautical industry. Moreover, its level of genericity should allow related transport industries to apply the framework (automotive, marine, railway). Therefore, it must be able to recreate the majority of transport vessels and missions through a generic ontology. This ontology should be able to map complex real products and missions into simplified framework components using a small set of concepts and relationships.

A generic framework allows creating any aeronautical product variation and sharing it with other designers. Moreover, even seemingly unrelated or widely different aeronautical products permit easy comparison. Reuse of past products allows performance comparison with new designs. Obviously, a generic product definition suffers from lack of detail that can void useful comparisons. Therefore, translating an existing design idea into the generic framework design requires care.

Generic scenario definition provides similar advantages: easy comparison and exchange of scenarios becomes possible. Designers can create a library of existing scenarios and re-use parts or entire scenarios for future products, reducing development time. However, scenario definition details may get lost if careless translation occurs.

In order to facilitate easy translation from products and scenarios to framework concepts, the ontology set of concepts must be comprehensible yet rich enough to enable capture of most products and scenarios.

3.1.2 Comprehensibility

Often, change in industry fails because users do not understand the structure and value of new methods (Jones et al. 2005). Therefore, in order to implement a new framework in aeronautical conceptual design, it must be comprehensible for its users. Simple inspection should allow anybody working with the framework to understand and believe it.

Moreover, a comprehensible framework facilitates translation between real world products and scenarios into framework components. Ideally, it avoids ambiguities by using exclusive terms that do not overlap. A comprehensible framework is simple in the sense that it allows quick understanding. However, a simple framework should not be too simple; otherwise, it becomes trivial and inhibits a realistic and generic framework.

The comprehensible framework needs to employ industry standard terms, concepts and vocabulary. Moreover, concept boundaries should be defined such that the relationships between them are obvious. A "mission" consisting of an "origin" and a "destination" allows intuitive understanding of those terms and their relationship.

3.1.3 Realism

In order to be of any use, the framework must be able to produce realistic results. However, lack of information and time during the conceptual design phase will force the level of realism below that attained with detailed design phase tools (Scanlan & Rao 2006). One of the reasons is the limited knowledge about the product and the scenarios during conceptual design. Initially, only the customer requirements and legacy knowledge are available. The required level of framework realism should reflect the level of knowledge available during conceptual design.

Moreover, the selection of framework concepts should reflect the most prominent product and scenario characteristics. For example, if a framework is good at capturing lifting device performance, this might not be useful for initial design considerations when it is not even clear if lifting devices are needed. However, some simple representation of lifting device performance might be desirable.

Therefore, it is essential to find a trade-off between realism, computing capabilities, available knowledge and useful concept application.

3.1.4 Modularity

For a conceptual design phase framework to be useful, it should not be static, fixed and closed. Instead, it should be editable, extensible and adaptable. A modular framework approach allows users to edit and adapt the framework for their specific needs. It can become less generic in some areas to allow for more realism if required by the users. Alternatively, users can add functionality where the current framework offers no solution. The modular approach mimics object-oriented programming used in computer science. Module choice allows easy and intuitive understanding of the module content while module interactions become obvious from the module boundaries. Therefore, it is useful to choose module boundaries based upon physical boundaries. Krus (2010) suggests to reflect physical system structures in simulation modules to

enable possible parallel execution. Consider a payload module: it should contain all payload items (which can be sub-modules themselves) but it should not include fuel tank components.

After establishing the framework requirements, the next section will describe the first part of the OSCAR framework on scenario modelling.

3.2 Scenario framework

This section describes the scenario-related framework components. This thesis defines the term "SCENARIO" as follows:

> SCENARIO*: The fully defined life cycle of a vessel system, including all operations, repairs, overhauls and idle times.*

Section 3.2.1 considers the extent and boundaries of the framework. Which scenarios are possible within OSCAR and what real applications can be covered? Section 3.2.2 assesses the OSCAR operations coverage to arrive at the first insight: each aeronautical operation can be classified into one of three operational goals. Next, Section 3.2.3 covers the topic of modularisation: How should scenario-related modules be organised according to the requirements described above? It includes a detailed description of all scenario-related objects such as MISSIONS, TRACKS and SEGMENTS. Section 3.2.4 introduces the unique OSCAR approach of spatially explicit geographical scenario definitions.

3.2.1 Scope of application

This section describes the operative range for the OSCAR scenarios. It starts out by excluding two major application areas: First, military applications are precluded because they feature distinct characteristics that are difficult to capture with OSCAR (Section 3.2.1.1). Second, OSCAR excludes space operations due to their fundamentally different flight profiles (Section 3.2.1.2). Last, this section defines the remaining scope of application (Section 3.2.1.3).

3.2.1.1 Military exclusion

Aeronautical vessels can be classified by their use for either civil or military applications. Civil applications tend to be relatively homogeneous because operational goals are very similar: civil aeronautical vessels are used either to transport objects[1] from A to B or to witness assets[2]

[1] "Object" refers to people or goods here.
[2] "Asset" refers to buildings, areas, roads, field, etc.

(this neglects use of aeronautical vessels for pleasure). On the other hand, military operations are more heterogeneous as the number of unique goals is higher. Many military scenarios require a fleet of aeronautical (and other) vessels working together to achieve complex goals like defending a specific area while striking enemy targets. Most civil applications do not require several vessels to achieve a target (an exception would be scientific UAS swarms analysing the atmosphere). Battlefield interactions are more complex as soldiers and aircraft interact with commanders, AI-driven equipment and –not least– the enemy.

While civil aeronautical operation outcomes are generally predictable, many military operation outcomes are not. They depend upon the capability of the aeronautical vessel, its allied forces and that of the enemy.

Moreover, military aeronautical vessels usually conduct fewer operations over their life cycle compared to civil products. Operators use most military products on demand while civil products (usually) operate on a regular basis. Therefore, the predictive capability of an operations-based conceptual design phase framework like OSCAR is limited for military vessels.

For these reasons, the OSCAR framework design focuses upon civil aeronautical applications. Nonetheless, OSCAR can model many military applications. However, a complex battlefield simulation with realistic agent interactions is beyond the scope of OSCAR in its current state.

3.2.1.2 Space exclusion

The aerospace industry combines aeronautical and space products because they share key challenges like propulsion, lift and 6DoF manoeuvrability. However, OSCAR is an operations-based design framework and operational characteristics between aeronautical (i.e. atmospheric) and space (i.e. non-atmospheric) flight differ.

Most aeronautical operations keep constant flight altitudes relative to the earth surface for most of the flight (except during climb/descent and inflight altitude changes). Therefore, a flat earth assumption is valid, thereby reducing the altitude dimension to a simple number (climb/descent can be modelled without an explicit third dimension). Essentially, a 2D flat-earth assumption suffices for aeronautical operations. Space operation profiles, on the other hand, cannot follow this assumption: their flight altitudes constantly vary with respect to earth surface, both in earth-bound missions (such as satellites following elliptical orbits) and in inter-planetary missions.

Moreover, contemporary space operations are not repetitive and –usually– unique. An exception is presented by Keller and Collopy (2013) applying a simple mission simulation for calculating the value of a *reusable* space launch system. Still, the life cycle of most satellites or rockets consists of one mission only (*i.e.* fly to Mars or orbit the earth for a couple of years). Afterwards, the product is discarded. Space vessel design hinges upon the characteristics of this one

mission to a very large degree. However, the strength of OSCAR is to analyse the impact of a wide variety of mission types repeated over a life cycle. Space products usually require only one mission profile that is followed once during their lifetime. Specialised space operations analysis software (e.g. GMAT[3]) will produce results that are more accurate.

Due to these fundamental differences, the OSCAR framework excludes space operations from its target capabilities. OSCAR assumes a 2D flat earth surface, modelling altitude as a separate mission parameter. Note that the flat earth assumption does not prevent great circle routes using explicit mission modelling (see Sections 3.2.4 and 4.6).

3.2.1.3 Remaining scope

After excluding military and space operations from the OSCAR framework, this section describes the remaining scope of application, namely civil aeronautical operations. OSCAR allows modelling of most civil aeronautical operations such as passenger transport, cargo transport, public authority, private and unmanned aviation missions.

In addition, OSCAR SCENARIOS can include non-aeronautical operations such as railway, automotive or maritime missions. Essentially, OSCAR can model any vessel moving at specific speed and height (even under ground or water) along specific paths. The only requirement is that vessel altitude change is not the most constitutive characteristic of its operations. Subsequent descriptions will demonstrate this by including non-aeronautical examples. However, the focus of OSCAR is aeronautical product design.

OSCAR SCENARIOS can last between seconds and centuries, depending on user requirements. SCENARIOS can combine any number of different operation types or just repeat the same operation. Operations can accommodate any number of operation segments, allowing any level of operational complexity.

3.2.2 Operations classification

Having established the scope of application for OSCAR SCENARIOS, this section will classify the wide array of possible SCENARIO operations. This thesis defines an "OPERATION" as follows:

> *Operation: Comprises any action undertaken in order to complete a user-defined transportation goal for a moving system. Since goal definition depends on user requirements and industry context, op-*

[3] "General Mission Analysis Tool", open source tool by NASA. See http://gmatcentral.org/, accessed 27/11/2013.

erations can vary between different users, even for the same moving system.

A typical passenger transport OPERATION is flying from airport A to airport B. The airline as the user defines the operational goal as transporting passengers from A to B. However, a different airline using a similar aircraft might define an OPERATION as going from A to B, then from B to C and from C back to A (a "round trip"). Here, the operational goal is to move the aircraft back to A via B and C. The definition of OPERATIONS is context-dependent allowing users to setup OSCAR around existing procedures.

Since OSCAR is a spatially explicit framework (see Section 3.2.4), OPERATIONS are categorised by the spatial character of an OPERATION. There are three fundamental geographical OPERATIONS, namely point, path or area OPERATIONS (or any combination).

3.2.2.1 Point operations

The goal of a point OPERATION is to reach one or more geographical points. Here, "point" is defined as:

POINT: *2D coordinate specified through longitude and latitude.*

Figure 3-2 depicts a generic POINT OPERATION with three spatially explicit POINTS.

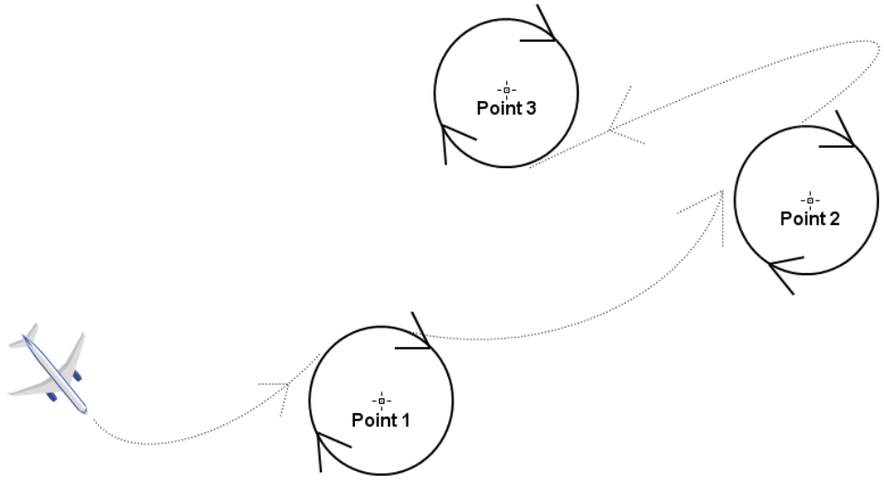


FIGURE 3-2: OSCAR FRAMEWORK POINT OPERATION EXAMPLE.

The vessel moves from its initial position towards POINT 1. Upon arrival, it may or may not loiter for a specified time (note that rotary-wing, floating and ground-based vessels can loiter stationary while fixed-wing vessels would conduct loiter patterns similar to Figure 3-2). Subsequently, the vessel moves towards POINT 2, and so on. The sequence as well as the spatial distribution of POINTS is critical to define a POINT OPERATION. A typical POINT OPERATION is an

aircraft gathering scientific data at pre-defined positions. Another example is a police helicopter transporting workers between offshore oilrigs. Civil airliner designers could simplify airliner OPERATIONs into POINT OPERATIONS by omitting airway information: the airliner simply flies from airport to airport (POINT to POINT).

3.2.2.2 Path operations

A path OPERATION goal is to move along one or more geographical paths. Here, "Path" is defined as:

> PATH: *a 2D line consisting of a start point, any number of corners in a specific sequence and an end point. Start point, corners and end point are 2D coordinates specified through longitude and latitude.*

Figure 3-3 depicts a generic PATH OPERATION consisting of three PATHS. PATH 1 consists of five corners, PATH 2 has 13 corners while PATH 3 comprises zero corners (i.e. a straight line). The vessel starts to fly towards the start point of PATH 1. It moves along the PATH and may or may not loiter upon reaching the end of PATH 1. Subsequently, it moves towards the start point of PATH 2, and so on. Note that start and end points of PATHS may or may not have the same location.

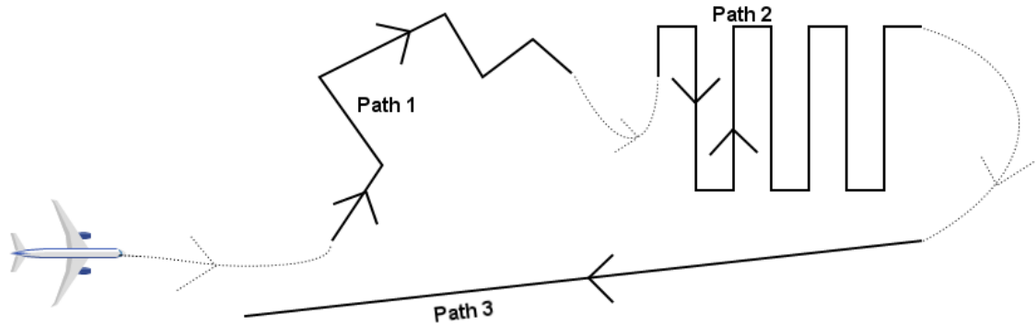


FIGURE 3-3: OSCAR FRAMEWORK PATH OPERATION EXAMPLE.

PATH OPERATIONS include airliners flying along one or several airways, a traffic-monitoring aircraft following specific motorways or border patrol and pipeline monitoring.

3.2.2.3 Aerial operations

The goal of an area OPERATION is to cover a specific geographical area. Here, "Area" is defined as:

> AREA: *a closed 2D polyline enclosing a specified area. The polyline consists of any number of corners (minimum: three).*

Figure 3-4 depicts a typical Area OPERATION consisting of two areas (grey regions). The vessel follows specific routes to cover both AREAS as required by the user. Initially, it moves towards the start point of the first route and moves along. Upon arrival at the first route end point, it may or may not loiter. Subsequently, it moves towards the start point of the second area route to cover the second AREA region.

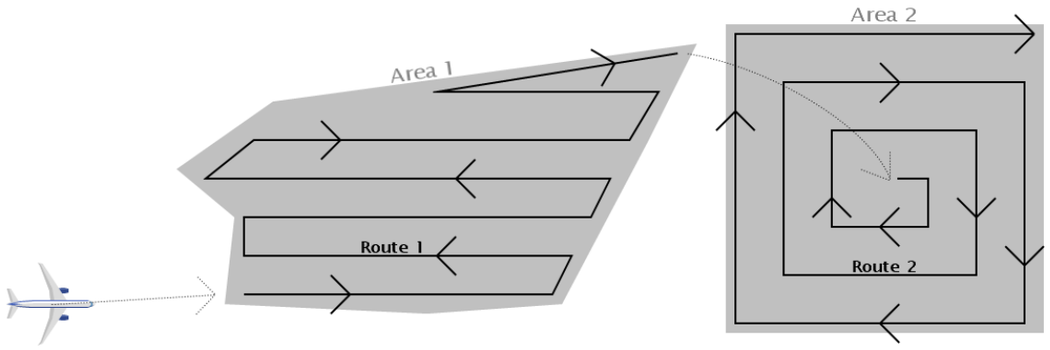


FIGURE 3-4: OSCAR FRAMEWORK AREA OPERATION EXAMPLE.

Note that the route to cover an AREA can vary in shape depending on user requirements. In fact, a vessel must follow a specific route to cover any AREA. Within OSCAR, such routes can be defined as PATHS. Therefore, OSCAR treats any AREA OPERATION as a PATH OPERATION. It is the user's responsibility to define PATHS such that the required AREA is covered. The user needs to take into account vessel flight altitude (if any), sensor capabilities (or human eye skills), visibility and possible sensor footprint overlap (how often should a point within the AREA be scanned?). However, these factors may be unknown during conceptual design phase or they may be part of the design investigation.

A superior way to deal with AREA OPERATIONS would define an AREA geographically and employ an algorithm to calculate an automated PATH for the vessel movement. Ideally, the algorithm would take into account parameters such as sensor capability, visibility and flight altitude. Goerzen et al. (2010) present a review of motion planning algorithms for UAS applications based on sensor capability. However, it is beyond the scope of this thesis to develop such an approach. The rest of this thesis treats "AREAS" as specific PATHS, neglecting the AREA definition.

Typical AREA missions include crop spraying, mapping and reconnaissance airborne missions as well as normal agricultural tractor operations or offshore fishing activities.

3.2.2.4 Combinations

Many real aeronautical OPERATIONS are more intricate and do not exactly match the definition of POINT, PATH or AREA OPERATIONS above. A border patrol aircraft may fly along the border PATH but upon spotting illegal activity starts covering the AREA in question. A highway-monitoring helicopter may be required to stop at a car crash POINT, interrupting its road-

following PATH duty for some time. Therefore, the OSCAR framework allows mixed OPERATIONS.

Through modularisation (see Section 3.2.3), it is possible to create combined OPERATIONS as in Figure 3-5. Upon arrival at POINT 1, the vessel proceeds towards PATH 1 (effectively covering an AREA as well). OSCAR can combine any number and combination of OPERATION types.

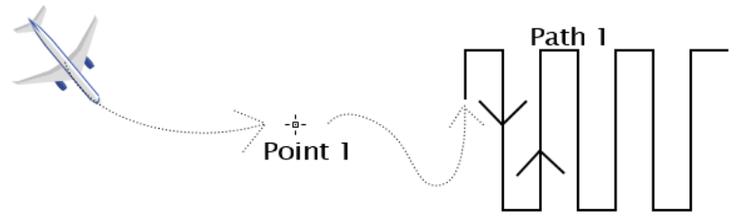


FIGURE 3-5: OSCAR FRAMEWORK COMBINED POINT AND PATH OPERATION.

3.2.3 Modularisation

The previous section distinguished between three OPERATION types, acknowledging that many civil aeronautical OPERATIONS (and, in fact, most moving vessel OPERATIONS) are a combination of those. However, work was conceptual until now. This section will formalise the insights obtained so far.

As discussed above, the life cycle of a moving vessel consists of OPERATIONS, repairs, maintenance operations and idle time, together forming a SCENARIO in OSCAR. Neglecting repair, maintenance and idle time, this thesis defines the "OPERATIONAL SCENARIO" of a vessel as:

> OPERATIONAL SCENARIO: *The sum of all moving* OPERATIONS *of an (aeronautical) vessel over its lifetime, omitting repairs, maintenance operations and down time.*

Note that adding repair, maintenance and idle time to an OPERATIONAL SCENARIO returns a SCENARIO (see definition on page 41). As shown in Figure 3-6, OSCAR defines an OPERATIONAL SCENARIO to consist of "MISSIONS", "TRACKS" and "SEGMENTS".

A "SEGMENT" is the equivalent to the OPERATION goals defined conceptually in Section 3.2.2 above. It can be either a POINT or a PATH. Section 3.2.3.1 describes a "SEGMENT" as the smallest building block to create an OPERATIONAL SCENARIO. Adding "SEGMENTS" together, OSCAR creates a concept called "TRACK", as described in Section 3.2.3.2. By combining "TRACKS", OSCAR forms a "MISSION" as defined in Section 3.2.3.3. Combining all "MISSIONS" forms the OPERATIONAL SCENARIO defined above.

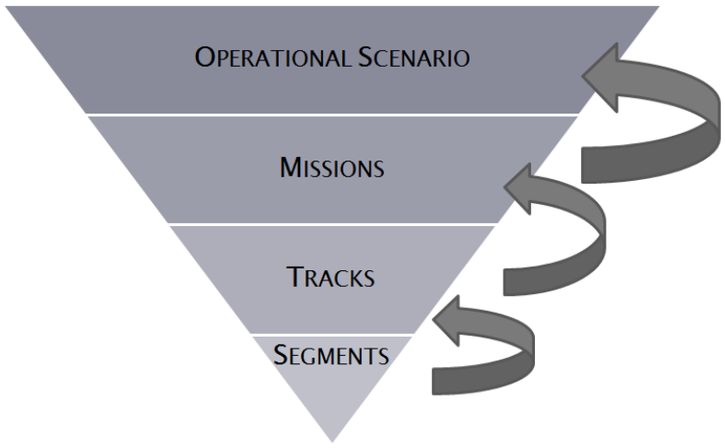


FIGURE 3-6: OSCAR OPERATIONAL SCENARIO PYRAMID.

3.2.3.1 Segments

A SEGMENT is the smallest building block to create an OPERATIONAL SCENARIO. It can be either a POINT or a PATH (Section 3.2.2). Each SEGMENT has a specific spatial position (latitude and longitude). If the SEGMENT is of type PATH, each corner and the start and end point must be defined through spatial coordinates in the correct sequence. Section 3.2.4 describes the implementation of SEGMENTS into Geographical Information System shapefiles. Beside the coordinates, eleven parameters uniquely define each SEGMENT namely `Time`, `Origin`, `Destination`, `UponArrival`, `Type`, `TargetHeight`, `TargetWidth`, `DetectionCriteria`, `Loiter`, `Height` and `Speed`. Appendix 1 details each parameter in more detail.

3.2.3.2 Tracks

A TRACK is an aggregation of any number and types of SEGMENTS. Moreover, it encloses a set of SEGMENTS with a dash and return segment linking them to a "Base" to a "Destination" (see Figure 3-7).

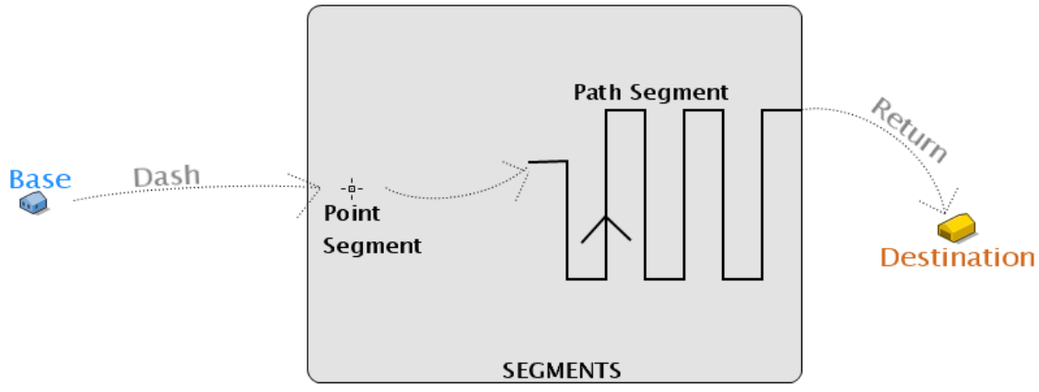


FIGURE 3-7: OSCAR TRACK SCHEMATIC.

As described above, the definition of a TRACK can vary depending on user requirements and mission definitions. However, one fundamental characteristic of a TRACK is its enclosure into a

"Base" and "Destination". Every vessel within the scope of OSCAR requires operational breaks, be it for rest (humans, animals), for repair or for operational reasons (aircraft schedules, ferries used only during the day, *etc.*). Therefore, TRACK allows the user to model these interruptions easily. Depending on the vessel type, "Base" and "Destination" vary in their meaning: for airborne vessels, they can represent airports or take-off and landing sites. For ground-based vehicles, it can be parking lots while for maritime vessels, harbours or anchorage sites are the major applications.

A TRACK is defined by twelve parameters, namely `Vessel_IDs`, `Base`, `Track`, `Track-Fragmented`, `Destination`, `Time`, `Repetition`, `Priority`, `DashHeight`, `DashSpeed`, `ReturnHeight` and `ReturnSpeed`. Appendix 2 describes each parameter in detail.

3.2.3.3 Missions

A MISSION pools any number of TRACKS together in a sequential manner to form an arbitrary sub-section or the total of the product life cycle. This is helpful to keep the life cycle organised. The user defines MISSIONS in a way that is most applicable for product operations. Aircraft designers may pool all TRACKS concerning passenger services of their new aircraft in one MISSION while keeping all late-life cargo missions in another. Alternatively, it might be useful to pool all tracks by week, month or year. Car designers may pool all TRACKS of European drivers into one MISSION and those of Asian drivers into another.

Essentially, the concept of the MISSION is a categorisation tool to avoid thousands of TRACKS in a product life cycle SCENARIO. It helps designers categorising their product TRACKS in a useful way, allowing simulation runs of selected MISSIONS only. Hence, it is easy to simulate only early-life TRACKS or only TRACKS of a specific region or time. Alternatively, MISSIONS can sort TRACKS by the product type if several different products are tested simultaneously.

Finally, the life cycle of a product (i.e. the OPERATIONAL SCENARIO) is the sum of its MISSIONS (one or many).

3.2.4 Spatially explicit setup

One of the unique features of OSCAR is its focus on spatially explicit geographical modelling where entities interact within a spatially explicit geographical environment. This Section describes the motivation to include spatially explicit geographical modelling (Section 3.2.4.1), how it is implemented into the OSCAR framework (Section 3.2.4.2) as well as the assumptions and simplifications applied (Section 3.2.4.3).

3.2.4.1 Motivation

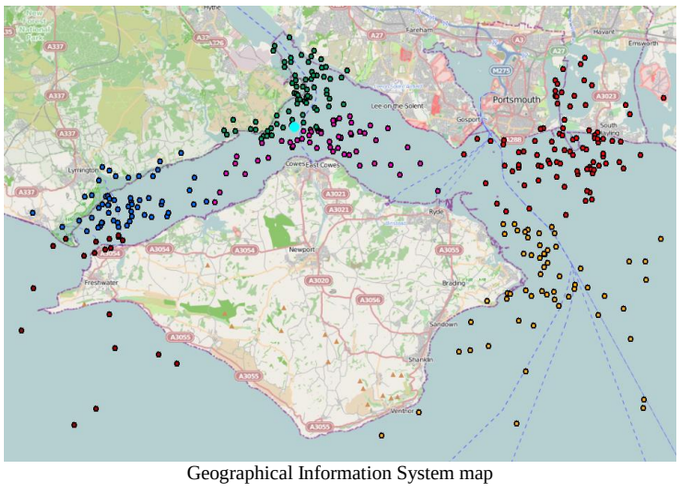
As discussed in Section 2.4.5, conceptual aeronautical design models do not apply spatial explicit modelling. Arguably, adding this level of complexity to the conceptual design phase adds

cost and development time that is needed elsewhere without obvious benefit. However, this section introduces benefits that can possibly outweigh the disadvantages.

During conceptual aeronautical design, geographical operational knowledge exists implicitly or explicitly because customers and designers usually know the operational applications for their product already. Designing a typical civil airliner, market analysis reveals potential routes and airlines contribute "significant input" (Raymer 2006), including intended routes and operations. Car manufacturers, on the other hand, usually use customer surveys with non-geographical information such as average driving distance or time (Otto & Wood 2001). In such cases, spatial modelling without geographical information suffices (see Section 3.2.4.2).

In order to model geography, the OSCAR framework uses the well-established shapefile format established by the Environmental System Research Institute, Inc. (ESRI 1998). Any geographical modelling software is able to interpret and create shapefiles. If customers or manufacturers own spatial or geographical information, it will most likely use the shapefile format, making it easy to adapt and import into OSCAR. The modular approach of OSCAR allows users to store SCENARIOS, MISSIONS and TRACKS in relational databases. Therefore, geographical knowledge needs to be created and stored only once and can be reused easily.

Another benefit of using geographical knowledge during the conceptual design phase is its ability to convey a lot of information visually through mapping (see Figure 3-8). This supports conceptual design activities because critical operational bottlenecks can be spotted earlier. Consider the case of search-and-rescue incidents as in Figure 3-8: most incidents occur near shores and around harbours, helping designers to estimate required vessel ranges easily. It would be more difficult to extract such information from pure data tables.



Geographical Information System map

Point	Latitude	Longitude
1	52.37	-1.17
2	51.83	-0.51
3	52.17	-1.11
4	52.51	-0.43
5	52.68	-0.41
6	52.15	-0.86
…	…	…

Same information in a table.

FIGURE 3-8: GEOGRAPHICAL INFORMATION SYSTEM MAP VERSUS TABLE.

Moreover, geographical data often encodes non-trivial information regarding design decisions. Consider the following though experiment displayed in Figure 3-9: an aircraft is supposed to patrol the shore waters during daytime. However, its fuel capacity requires refuelling during the

patrol. Operationally, it is critically important to determine where the aircraft most often runs out of fuel.

FIGURE 3-9: GEOGRAPHICAL IMPORTANCE OF FUEL: RUNNING OUT OF FUEL AT POINT A OR B?

If the aircraft runs out of fuel at Point A, returning to the airfield for refuel affects the patrol operation to a certain degree (due to the delay). However, if the aircraft runs out of fuel at Point B, the impact is disproportionally larger because the time to reach the airfield is much larger and the duration off patrol increases. A non-spatial analysis could not capture this discrepancy easily while it becomes obvious using spatial modelling.

There are many similar situations arising from operational scenarios in geographical models: Does oil tanker storage fail in open waters or near environmentally critical shorelines? How does a jet engine compare operating in dusty Arabian countries compared to wet South-East Asian countries? Can one optimise parcel delivery truck routes by only choosing right-turn routes[4] ? New insights into design decisions and optimisation strategies arise by considering the spatial and geographical component of these operations early in the design process.

3.2.4.2 Implementation

This section presents how geographical modelling fits into the OSCAR framework. Section 4.6 discusses the practical realisation within the OSCAR simulation.

As indicated above, OSCAR supports a geographical modelling approach by implementing geographical maps (compare Section 2.4.5). However, spatial modelling without geographical information is also possible within OSCAR by projecting the spatial information upon an empty background map. This approach ensures greatest flexibility for designers.

[4] Compare http://compass.ups.com/UPS-driver-avoid-left-turns/, accessed 23/10/2013.

Geographical modelling within OSCAR rests upon a 2D map of the entire world, the "base map". Users can employ different base maps to allow categorising POINTS and PATHS by their interaction with the base map: are they upon water or land, city or countryside, country A or B? Projected onto the base map are the two geographical SCENARIO components, namely POINTS (Section 3.2.2.1) and PATHS (Section 3.2.2.2). Each POINT and each point of a PATH feature a longitude and latitude coordinate defining the position on the base map. In addition, this thesis assigns the following additional geographical information:

- Search-and-rescue incident position uncertainty data (Section 5.4.1.2)
- POINT and PATH end-point object height and width (Section 3.2.3.1)
- Loiter times, flight profiles and behaviour upon vessel arrival (Section 3.2.3.1)

However, designers can extend the OSCAR framework to include custom geographical effects such as distributions of temperatures, dust levels, wind speeds or visibilities. By extending the OSCAR simulation logic, this enables rapid analysis of different operational influences upon the design.

3.2.4.3 Assumptions & Simplifications

Including geographical modelling into the conceptual design phase increases the level of detail considerably. However, there are a number of assumptions and simplifications to keep workload and computation times reasonable following conceptual design phase requirements.

Foremost, all mapping is two-dimensional only. This has two effects upon product design: First, the base map lacks elevation. There are no mountains, buildings or barriers of any kind. This does not prevent OPERATIONAL SCENARIOS to include altitude profiles (see Section 3.2.3.1) but altitude profiles do not map with elevation data. If a car has `altitudeMax=0` and is supposed to travel along a mountain road, it will keep an altitude value of zero. However, it is possible to model the mountain road elevation profile by adjusting the road PATH `Height` values and increasing the `altitudeMax` parameter for the car (see Section 3.3.3). As stated earlier, the OSCAR framework is most applicable for moving vessels where altitude change is not the most constitutive characteristic of its operations.

The second effect of 2D base maps for designers is that the earth surface is flat. Therefore, users must model great circle routes (for ships or aircraft) manually by adjusting the PATH between two points following great circle rules. However, this is straightforward with any modern geographical modelling software.

Another assumption for geographical modelling within OSCAR neglects curves between SEGMENTS and PATH edges. Any directional vessel turn is immediate as in Figure 3-10.

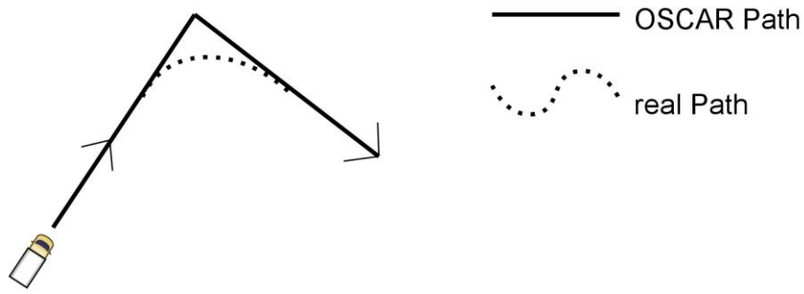


FIGURE 3-10: OSCAR VESSEL TURNING PERFORMANCE.

Turning performance is difficult to predict generically because it varies strongly between vessel types and operations. For aircraft, turn performance depends on speed, altitude, wind, bank angle, g-force limits and acceptable overshoot. For ground-based vessels, different parameters such as road surface, steering wheel limits and driving speed dictate curve performance. With maritime vessels, yet another set of parameters is required, namely rudder size, relative speed and ship mass. It is beyond the scope of this research to find a unifying curve performance algorithm. However, the user must be aware that vessel performance accuracy decreases linearly with the number of directional changes in the OPERATIONAL SCENARIO. However, for conceptual design, the loss in accuracy is acceptable because straight lines usually dominate aeronautical vessel operations. Exceptions include search operations and crop spraying which base upon frequent directional changes.

Similar to neglecting curves, geographical modelling within OSCAR also neglects climb and descent performance (Figure 3-11). Any change in altitude between SEGMENTS occurs instantaneously. This affects airborne vessels changing flight altitudes, submerged vessels changing diving depths as well as ground-based vessels following elevation profiles (i.e. crossing a mountain).

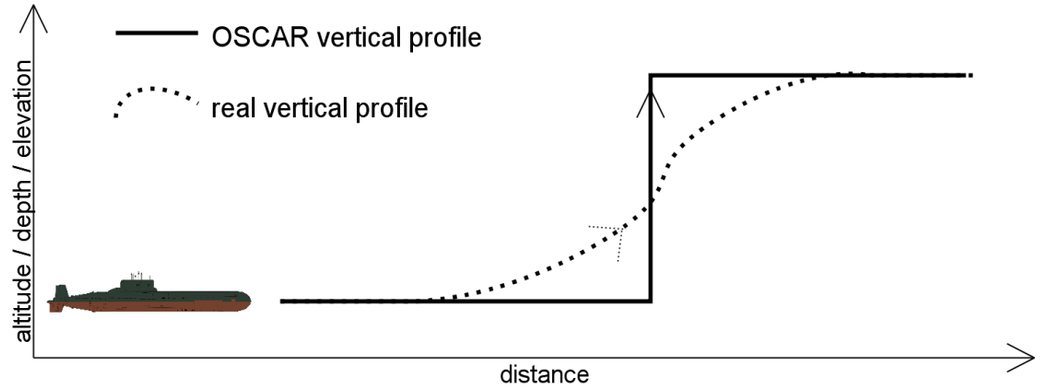


FIGURE 3-11: OSCAR VERTICAL PROFILE.

If necessary, it is possible to model smooth altitude changes through discretisation by using a large number of SEGMENTS with slightly different altitudes each. The same argument as for directional changes applies: altitude change performance is difficult to model generically for

many different vessel types. Moreover, most vessel TRACKS occur at constant altitudes for most of the TRACK duration. Even long-distance airliners increase their altitude due to depleting fuel only in near-discrete steps every couple of hours. Therefore, this simplification is acceptable for operations where altitude changes are not the main characteristic. The more altitude changes are characteristic to vessel operations, the less accurate results will be.

3.3 Vessel framework

The previous section described the SCENARIO-related framework details. However, in order to model the life cycle of aeronautical, automotive, maritime or railway products, it is necessary to provide a generic framework for modelling the vessels themselves. Section 3.3.1 starts out by outlining the scope of application of the vessel framework. In Section 3.3.2, a vessel classification system refines the scope of application. Section 3.3.3 presents the generic parameter set defining vessels within OSCAR. Subsequently, Section 3.3.4 outlines the generic vessel performance calculation method that can be used to calculate energy consumption of any vessel. Section 3.3.5 introduces the capability of specifying vessel components, including component parameter definitions. This allows defining deterioration performance and maintenance. Lastly, Section 3.3.6 describes the optional payload module add-in for vessels.

3.3.1 Scope of application

OSCAR aims to enable modelling of any object that can change its position based on its own initiative. Therefore, define:

> VESSEL*: Any physical object that is capable to some degree to change its position in a controlled way using its own means. Control is exercised through the* VESSEL *itself or through a* VESSEL *user.*

This definition includes any type of automotive, ships, submarines, aircraft, helicopters and UAS. It also includes non-obvious entities such as humans and most animals. However, some entities do not count as a VESSEL: Hot-air balloons, for instance, are not capable to control their flight path to a large extend, although they can control their altitude well. The OSCAR framework requires user to input specific operation PATHS, but for balloons, these are unknown beforehand. However, users can define typical balloon PATHS, neglecting the specific influence of winds upon performance.

As discussed in Section 3.2.1, military and space OPERATIONAL SCENARIOS are not supported within OSCAR. Consequently, OSCAR does not support military and space vehicles within the VESSEL framework, although they may fall under the VESSEL-definition above.

Based upon the object-oriented modelling approach, OSCAR can model any number and any type of VESSEL in parallel.

3.3.2 Classification

A large number of VESSELS exist that follow the VESSEL definition above. The OSCAR framework provides a VESSEL classification in order to group VESSEL types of similar application together.

The OSCAR VESSEL classification assigns two characteristics to each VESSEL: category and type. This helps users to distinguish VESSEL agents quickly but it also allows adding functional behaviour to specific categories or types only. The categories and types supported within the OSCAR framework are shown in Figure 3-12.

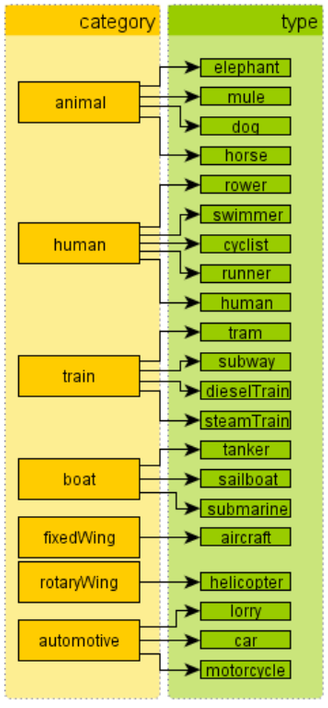


FIGURE 3-12: OSCAR SIMULATION VESSEL `CATEGORIES` AND `TYPES` (`NOT EXHAUSTIVE`).

Note that neither category nor type entries are exhaustive: users can add their own categories and types based on requirements.

Within the OSCAR simulation, types are used to sanity check altitude data: submarines and ships cannot cruise above `Height=0` while aircraft cannot fly below `Height=0`. Beyond this, neither categories nor types have any influence on VESSEL performance. Users can exploit

the classification further to include more complex operational group behaviours: for example, fixed-wing aircraft cannot fly at a speed of zero, even if asked to do. Automotive VESSELS must not cruise on water. Fixed wing VESSELS cannot loiter stationary, etc. However, more detailed classification behaviours are beyond the scope of this thesis.

3.3.3 Parameters

Although there are many different VESSEL types, the research goal of OSCAR is to find a generic set of unifying parameters that describes each VESSEL equally well. In addition to the `category` and `type` parameters described above, there are twelve parameters required to define a VESSEL, namely `performanceModel`, `fuelType`, `occupants`, `speedMax`, `speedMin`, `speedTypical`, `altitudeMax`, `altitudeMin`, `altitudeTypical`, `useTypicalSetup`, `weightDry` and `weightFuel`. Appendix 3 describes each parameter in more detail.

3.3.4 Propulsion performance

Having defined the core characteristics of VESSELS above, this section continues by introducing the generic performance model developed for the OSCAR framework. The various VESSEL types supported by the OSCAR framework feature very different propulsion systems: some burn petrol or diesel (cars, Lorries, *etc.*), other have electric drives (some UAS, e-bikes, e-cars), yet others need food (animals, humans) or nuclear fuel elements (some submarines). However, all propulsion systems are similar in that they consume energy at different rates.

Therefore, the generic propulsion performance model within the OSCAR framework computes energy consumption, independent of the propulsion system used by the VESSEL. For this, each VESSEL features a table linking energy consumption with velocity. Users can specify complex energy consumption profiles in a simple way. For example, aircraft use flaps below certain speeds, changing fuel burn (and thereby energy consumption). Similarly, ships or cars have distinctive fuel consumptions if stationary (with their engines turned on). Figure 3-13 presents several energy consumption profiles. This allows modelling and comparing very different VESSEL types and propulsion systems. The advantage of this modelling approach is that energy consumption data is easily available for most VESSELS (*e.g.* Cullinane and Khanna (1998) developed a set of equations for large container ships based on power consumption). Moreover, conceptual designers have access to energy consumption profiles for the product in question. In some cases, simple unit conversion is required (*i.e.* aircraft designers work with specific fuel consumption).

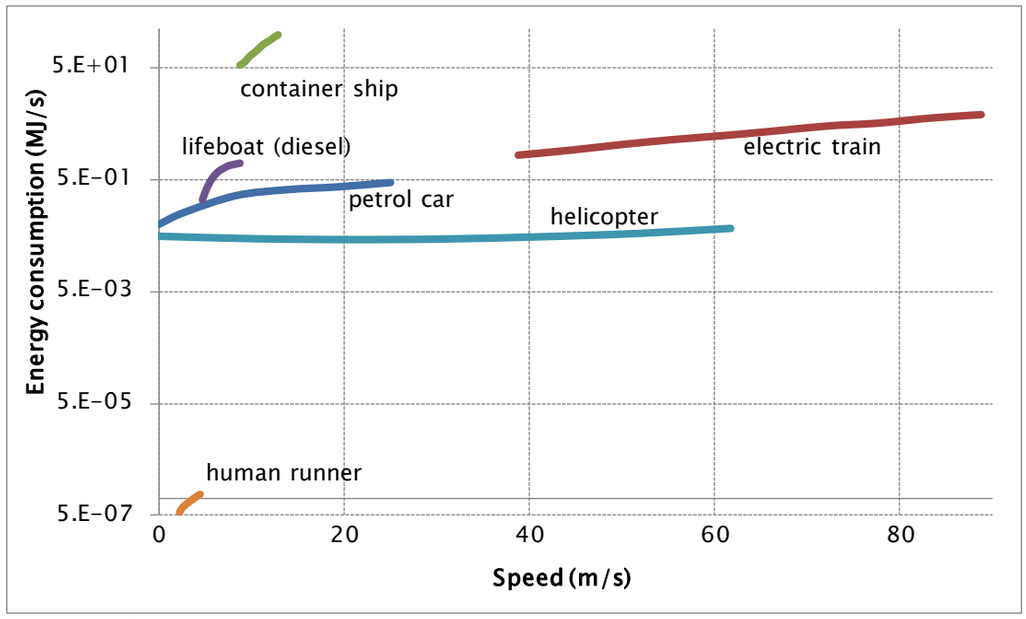


FIGURE 3-13: ENERGY CONSUMPTION FOR VARIOUS VESSEL TYPES (VERTICAL AXIS HAS LOGARITHMIC SCALE WITH BASE 2)[5].

However, a number of assumptions and simplifications underlie this approach. In reality, energy consumption depends on more factors than VESSEL speed. Often, ambient conditions such as temperature, pressure and wind alter energy consumptions. Moreover, friction or drag varies with the product design shapes. Some of these factors can be factored in the OSCAR generic propulsion performance model (i.e. drag varies with VESSEL speed). Other factors cancel each other out over the VESSEL life cycle. Here, average expected values can be factored into the OSCAR model (*i.e.* temperature, pressure or wind cancel each other out to varying degrees). More importantly, any VESSEL energy consumption rate varies with VESSEL weight. Neglecting the influence of total VESSEL weight is reasonable for VESSELS that have a small (or zero) maximum `weightFuel`/`weightDry` ratios (see Section 3.3.3). Here, diminishing fuel weight has no big impact on total weight. However, some VESSELS such as aircraft, container ships or helicopters have a larger maximum `weightFuel`/`weightDry`. In this case, neglecting the impact of total VESSEL weight on energy consumption may not be justifiable anymore: Designers should embed custom energy consumption models taking into account weight variation such as the custom aircraft propulsion model described in Appendix 9.

[5] Sources: human runner → http://www.brianmac.co.uk/energyexp.htm, accessed 30/10/2013; Petrol car → (Sturm & Hausberger 2005); lifeboat → Police of Kent; container ship → http://people.hofstra.edu/geotrans/eng/ch8en/conc8en/fuel_consumption_containerships.html, access 30/10/2013; helicopter → (Mabus 2008); electric train → (Garcia 2010);

3.3.5 Fatigue

3.3.5.1 Components

During conceptual design, it is important to model not only the basic VESSEL behaviour and its energy consumption. Any VESSEL deteriorates in service and is subject to planned or unplanned maintenance. Current conceptual design phase procedures apply empirical relations derived from historical data: based on design parameters such as wingspan, mass or thrust these relations return the expected defects and maintenance hour per flight hour (Fielding 1999). However, object-oriented methods enable more detailed component-based modelling. Therefore, OSCAR enables each VESSEL to comprise any number of "COMPONENTS" that deteriorate over time:

> COMPONENT*: A physical object being part of a* VESSEL *that deteriorates in some way during* VESSEL *operations.*

A VESSEL can contain no COMPONENTS at all, if required. In that case, it acts as a fault-free agent that never fails. This can be useful in order to create VESSELS that interact with a conceptual VESSEL design but are not themselves part of the investigation, especially if their rate of failure is much lower than that of the VESSEL under investigation (*i.e.* lifeboats in chapter 4.14).

Seven parameters define each COMPONENT, namely `weibullLifeMeasure`, `weibullEta`, `weibullBeta`, `LossProbabilityFromFailure`, `unplannedMaintenanceDuration`, `quantityOnboard` and `robustnessScalingFactor`. Appendix 4 describes each parameter in detail.

Upon VESSEL creation, each COMPONENT is assigned a time-to-failure (or cycles-to-failure if `weibullLifeMeasure=cycles`) randomly drawn from the specified weibull distribution and adjusted by the `robustnessScalingFactor`. Upon planned and unplanned maintenance, the value is re-calculated in the same way. This introduces an element of randomness imitating real variations in time-to-failure.

3.3.5.2 Deterioration and failure

During VESSEL operations, each COMPONENT deteriorates as defined above (i.e. it ages by the duration the VESSEL operated or by its cycles). The COMPONENT will fail when it operated for longer than the time-to-failure allows (or had more cycles than cycles-to-failure). Upon failure, the VESSEL checks for possible redundant COMPONENTS that can take over the workload. If `quantityOnboard>1`, the failed COMPONENT is shut off and unplanned maintenance is scheduled after the current TRACK. If `quantityOnboard=1`, the VESSEL will be lost with `LossProbabilityFromFailure`. If the COMPONENT does not cause the VESSEL to be lost, it

schedules unplanned maintenance after the current TRACK. If the COMPONENT does cause VESSEL loss, it does not finish its current SEGMENT, TRACK and MISSION but is lost immediately.

The deterioration and maintenance model bases upon the following assumptions. COMPONENTS deteriorate by one mechanism only. In reality, components usually deteriorate by several mechanisms: an aircraft wing structure fatigues during flight (`weibullLifeMeasure=duration`) but also upon landings (`weibullLifeMeasure=cycles`). However, most COMPONENTS deteriorate from one primary mechanism and other mechanisms have second-order effects only. Therefore, it is reasonable to neglect multiple fatigue mechanisms for conceptual design.

Another simplification assumes that COMPONENT fatigue and failures are independent of each other. In reality, failure of one COMPONENT can create secondary failures on other COMPONENTS. Within OSCAR, ripple-on effects are only modelled for redundant COMPONENTS: if a COMPONENT fails and has `quantityOnboard>1`, the redundant COMPONENT'S time-to-failure (or cycles-to-failure) is reduced by $\frac{1}{\#redundant\ components} * weibullBeta * 0.1$. The more redundant COMPONENTS are available, the less reduction in time-to-failure occurs for each COMPONENT as they are assumed to share the load of the failed COMPONENT. The factor "0.1" is arbitrarily chosen to ensure that no COMPONENT suffers more than 10% reduction in time-to-failure. However, this simplification neglects that failing COMPONENTS can influence different COMPONENTS: it is not possible to model complex COMPONENT interactions like that that leading to the space shuttle Columbia crash[6].

Another assumption in the OSCAR deterioration model is that COMPONENT time-to-failure (or cycle-to-failure) distributions are weibull-shaped. Due to flexibility of the weibull distribution, it is used widely in reliability engineering. Future work may comprise a number of distributions to choose from by users.

So far, all COMPONENT deterioration focussed on mechanical fatigue due to operations. However, modern VESSELS usually feature electronic components that also suffer from programming bugs and electric sensibilities. Although electrical failures can be factored into the weibull distribution for an electronic COMPONENT, it is difficult to account for programming bugs. Modern industry algorithms feature between 15-50 errors per 1000 lines of code (McConnell 2004). Modern car software components run up to 100 million lines of code[7]. Although not every defect causes erroneous behaviour in a VESSEL, future work could provide facilities to simulate electronic component failures due to programming bugs.

[6] During launch, a foam insulation piece came off and struck the left wing. The wing damage caused the shuttle's disintegration upon re-entering, see http://www.nasa.gov/columbia/home/CAIB_Vol1.html, access 30/10/2013.

[7] See Charette (2009).

3.3.5.3 Planned maintenance

A recent review in aircraft maintenance operations lists 21 different types of maintenance operations out of which only three are "unscheduled or non-routine" (Bergh et al. 2013). So far, the OSCAR deterioration and maintenance model described unplanned maintenance only. However, most industrial VESSEL designs are not maintained *reactively*, *i.e.* once a problem occurs during operations. Instead, technicians maintain equipment *actively*, that is through planned maintenance checks at specific intervals (Gao et al. 2009). Currently, the OSCAR framework supports planned maintenance only indirectly: When a COMPONENT does not fail before `weibullBeta`, it schedules a planned maintenance for the next available slot between TRACKS.

Future work may add an additional parameter `plannedMaintenanceInterval` indicating in seconds how often the COMPONENT should be checked for problems. This would not recreate complex maintenance schedules seen in contemporary aircraft components but would suffice for conceptual design purposes.

3.3.6 Payload

Austin (2010) defines payload in the context of aircraft as

> *Payload: the part of the aircraft, which is specifically carried to achieve the mission.*

This research applies the definition not only to aircraft but also to all VESSELS. The operational purpose of many VESSELS is to transport payload for various applications. Civil airliners transport passengers or goods from A to B. Ships, trains and cars often fulfil the same purpose. Another application is dispensing payload during operation. Examples include crop-spraying or humanitarian air aid. A third application is transporting sensory payload (sensors, animals or humans) to specific locations for any kind of remote sensing (mapping, environmental monitoring, spying…). In fact, beyond scientific and industrial prototypes, there are no VESSELS designed for carrying no payload at all. A major aspect of conceptual design decision support is to assess the suitability of a VESSEL design concerning payload performance. How good is the VESSEL at transporting payload from A to B? Alternatively, how well does payload sensors perform during operations due to VESSEL design? To answer these questions, the OSCAR framework supports a generic payload model allowing payload performance quantification. Moreover, users can add custom payload models to refine performance calculation.

As discussed above, VESSELS either transport payload from A to B ("inactive payload") or use it to gather intelligence ("active payload"). Note that dispensable payload can be categorized as "inactive payload". Each payload type is described in more detail below.

3.3.6.1 Inactive payload

VESSELS transport inactive payload with the goal of transporting. In order to measure transport performance, the following parameters are used upon defining each VESSEL:

> `PayloadItems`*: Integer value indicating how many items the* VESSEL *can transport. This neglects varying capacities in different operating conditions (airliners accept fewer passengers for very long ranges, etc.). During operations, the* VESSEL *always transports at full capacity.*
>
> *PayloadWeight: Double value in kilograms indicating the weight of each payload item. This assumes constant weight of each payload item. Operations with varying payload item weights (parcel delivery, cargo ships…) must use average values.*

Note that this model does not distinguish between animate (humans, animals) and inanimate (parcels, *etc.*) payload. It also ignores replenishing items such as food. Each time a VESSEL completes a TRACK (or a repeated TRACK), it logs the number of payload items and their total weight as "on time", "delayed" or "cancelled". Items delivered "on time" arrive at the TRACK's `Destination` with no disruptions during the TRACK. Items delivered "delayed" faced a disruption during conducting the TRACK. Disruptions include refuelling due to fuel shortage or moving at speeds slower than scheduled (because the VESSEL's `speedMax` is not large enough). Items not delivered are marked as "cancelled": here, the VESSEL did not complete a TRACK because it crashed (airborne), sunk (maritime) or broke (land-based) due to COMPONENT failure.

Upon post-processing, users can amend the data with monetary information such as price paid per payload item delivered on time. This allows computing VESSEL profits based on payload transport and VESSEL performance.

Note that inactive payload metrics are not implemented into the OSCAR simulation.

3.3.6.2 Active payload

VESSELS use active payload (i.e. sensors) to gather any form of intelligence about the area around the VESSEL TRACK. Active payload comprises mechanical devices that are capable to sense specific physical quantities. Examples include cameras of any light range (visible, infrared, *etc.*), microphones, magnetometers, particle sensors or chemical sensors. Active payload must be positioned in space to point towards a given target. This can include an area, volume or point of interest. Moreover, any active payload sensor has a field-of-view that depends on the sensor design.

Most commercial and scientific VESSEL missions carry electro-optical sensors (i.e. cameras) of any type and light range (Duquette 2009). The rest of this section presents a custom add-in that models electro-optical sensors in detail. Other active payload sensors can be modelled with the add-in as they are based upon positioning the sensor in space towards a target. However, the detection probability algorithm (described in Section 4.10.2) is restricted to electro-optical sensors only.

VESSELS employ electro-optical sensors for mapping and for spotting targets. For mapping missions, camera performance hinges on the image quality, the area covered and image size. For target missions, camera performance depends upon the same factors plus the capability to spot the target.

3.3.6.2.1 Payload parameters

Duquette (2009) specifies three important characteristics of electro-optical sensors, namely pixel array size, field-of-view and sensor orientation relative to the aircraft. Herpel et al. (2008) define similar characteristics for automotive pedestrian-avoidance sensors. The OSCAR simulation applies these characteristics using six active payload parameters for each VESSEL, as described in Table 3-1.

TABLE 3-1: REQUIRED TO DEFINE AN ELECTRO-OPTICAL SENSOR ACTIVE PAYLOAD ITEM ONBOARD A VESSEL.

Active payload parameters	Description
sensorFOVhor	Double value between 0 and π in radians specifying the horizontal field-of-view of the sensor.
sensorFOVver	Double value between 0 and π in radians specifying the vertical field-of-view of the sensor.
sensorPixelshor	Integer value larger than zero specifying the number of horizontal pixels of the sensor.
sensorPixelsver	Integer value larger than zero specifying the number of vertical pixels of the sensor.
sensorTiltAngle	Double value between $-\pi / 2$ and $\pi / 2$ specifying the sensor tilt angle relative to the *VESSEL* roll axis. If *sensorTiltAngle=0*, the sensor is aligned with the *VESSEL* roll axis. For an aircraft flying in level flight, if *sensorTiltAngle=π/2*, the sensor looks vertically down to earth.
sensorRecognitionFactor	Double value larger than zero amending the likelihood of spotting a target. If `sensorRecognitionFactor=0`, the target is never spotted. If `sensorRecognitionFactor=1`, the likelihood is unchanged. If `sensorRecognitionFactor=1.1`, the likelihood increases by 10%. This parameter is used to validate payload performance against real data, if required.

Computing sensor detection probabilities is highly specific to the sensor application. Section 4.10.2 details the algorithms used for detection probability calculation of electro-optical sensors.

The target detection algorithm applies only if the target is actually within the camera footprint. Appendix 5 details the computation of the camera footprint based on VESSEL operational characteristics.

3.3.6.2.2 Limitations

The active payload electro-optical sensor add-in described above allows estimating VESSEL performance based on sensor performance. However, it is only applicable to missions that scan a 2D surface such as the earth (for airborne and ground-based VESSELS) or the ocean surface (for submerged VESSELS or ships). Some MISSIONS require sensors to scan 3D spaces or track targets within space. Examples include volcano ash cloud monitoring or tracking whales in the ocean from a boat. The OSCAR framework does not support scanning of 3D spaces as this is deemed too complex for conceptual design studies.

Another limitation of the add-in is its focus on electro-optical sensors only. It can be used for the visible and infrared light range (although the user should be aware of the significant changes required with `sensorFOVhor` and `sensorFOVver`, see Appendix 5). However, it is not possible to implement different sensor types such as microphones, radar or Geiger counters.

Moreover, the add-in cannot easily model human eye performance. It is simple to adapt the model for human eye use if the payload performance is measured by quantities like scanned area. However, measuring performance based on spotting targets is much more difficult. There is very little literature on human eye scanning performance, especially for simple models as required here. The GEorgia Tech Vision (GTV) model developed by Doll et al. (1998) is too complex for conceptual design purposes as it simulates performance cell-wise. There is no simple model linking target size and distance to the probability of spotting from a human eye. However, humans operate many active payload-carrying VESSELS and their vision often plays an important role in achieving MISSION success. Examples include any kind of search-and-rescue missions, human traffic monitoring or border patrols. Therefore, users must be aware that performance outputs will be skewed for such missions.

4. SIMULATION

The previous chapter presented the OSCAR *framework*, i.e. the theoretical foundation used to answer the research questions posed in Section 1.2. This chapter details the OSCAR *simulation*, *i.e.* one possible practical implementation of the OSCAR *framework*.

Section 4.1 starts out by justifying the use of computational simulation as the best means to achieve the OSCAR requirements. Section 4.2 provides the functional specification for the OSCAR simulation. Section 4.3 details the process of software selection to find the most suitable software package for creating the OSCAR simulation. The remaining sections cover the actual OSCAR simulation: Section 4.4 provides a short overview of the software structure, including a toy model walkthrough to promote understanding of the user workflow. Then, Section 4.5 goes into more detail describing the data structure employed for the OSCAR simulation as well as the practical aspects of data handling. Section 4.6 characterises the Geographical Information System implementation that is critical to the OSCAR framework and simulation. Subsequently, Section 4.7 depicts the Base module used within the OSCAR simulation. Section 4.8 specifies the VESSEL module, including details about the characteristic parameters and the state chart behaviour. VESSEL performance implementation is discussed in Section 4.9, split into the generic performance module and the custom aircraft performance module developed for the thesis case studies. The next Section 4.10 details the payload module derived for the OSCAR simulation. It is limited to “active” payload equipment that is used to scan 2D surfaces with electro-optical sensors only. Section 4.11 focuses on a specific VESSEL application, namely Search-and-Rescue TRACK patterns. Section 4.12 describes the VESSEL COMPONENT module implementation and functionality. Last, Section 4.13 wraps up the chapter by detailing the OSCAR experimental setups: the “Customer” experiment can be used for detailed one-time analysis while the

“Freeform” experiment allows a top-level view on system performance by running many random replications.

4.1 Justification

Following the general justification for simulation in Section 2.4, this Section argues specifically why the OSCAR framework is best implemented using simulation.

The life cycle of any VESSEL consists of MISSIONS, maintenance and idle time (see Section 3.2.3). Thereby, a VESSEL life cycle complies with the four system characteristics that suggest simulation modelling, namely variability, dynamics, interconnectedness and complexity (Sterman 2000). Each of the following characteristics on its own can be handled with other tools beyond simulation. However, combining all four is best handled using simulation modelling.

- VESSEL operations are *variable* because many aspects are inherently stochastic. Aircraft passenger boarding, oil tanker loading or car component maintenance all vary in duration in reality. Variability is often caused by (minor) disruptions or human actions that are beyond the model scope but must be captured nonetheless.
- VESSEL operations are *dynamic* because the core aspect of modelling the life cycle of a VESSEL is to follow it through time. The VESSEL operates and performs over time. Any action or interaction of any agent occurs in time. Although computers can simulate dynamic processes in a quasi-dynamic fashion only, this is sufficient for conceptual design phase requirements.
- VESSEL operations are *interconnected* because VESSELS interact with their environment. This includes the physical surrounding (road surface, water temperature, air density), the operational environment (MISSION profiles), their own COMPONENTS and other VESSELS. Change in one VESSEL agent can cause change in its physical environment, MISSION profile, any of its own COMPONENTS, or in another VESSEL. It is one of the major advantages of OSCAR to be able to simulate the life cycle of a fleet of products (not just one individual VESSEL). Hence, realistic interaction with competitor VESSELS in the same environment is possible only through *interconnectedness*.
- Vessel operations are *complex*, both combinatorial and dynamically. Combinatorial complexity can arise due to the number of components in a VESSEL and the number of VESSELS employed throughout the life cycle (i.e. the VESSEL fleet). In fact, the level of *interconnectedness* correlates with the level of combinatorial complexity. Dynamic complexity arises from component interactions over time. Dynamic systems like VESSEL life cycles, physical environment, VESSEL performance, VESSEL COMPONENTS and

other VESSELS all interact with each other dynamically. As dynamic complexity is hard to predict analytically, it is prudent to use simulation for VESSEL life cycle modelling.

Another reason to apply simulation modelling is that simulation allows managers to understand the model easily due to visualisation and interaction. Moreover, virtual experimentation stimulates a less risk-averse design process because non-intuitive "what-if" scenarios are possible at low cost. Hence, managers can probe design concepts in more detail and challenge existing paradigms with little business risks. A simulation usually fosters more communication as design decisions and interactions must be thought through in more detail (Robinson 2004). Without simulation, it is more difficult to create and follow thought experiments and arrive at useful conclusions about the VESSEL design.

The alternatives to simulation are inferior regarding implementation of the OSCAR framework. Spreadsheets offer rudimentary capabilities to integrate dynamics and variability while advanced spreadsheet functionality requires additional coding. Moreover, presentation and animation capabilities are inferior to simulation. Not least, spreadsheets cannot recreate interconnectedness between entities easily as they do not support object-oriented interactions.

Analytical and mathematical models can include variability. However, they require far more assumptions to create the model. Analytical models are not flexible in the sense that their solution is usually just focussed on a single problem. Not least, most VESSEL life cycles are too complex to be modelled analytically.

Another alternative to simulation is using a real prototype. However, for conceptual design this approach is not acceptable. First, building prototypes is expensive and time consuming. Second, detailed design information is unknown, ruling prototype test results void. Last, it is not practical to test prototype performance throughout its life cycle.

Although simulation is the superior modelling approach to implement the OSCAR framework, it features inherent disadvantages as discussed in Section 2.4.2 such as *inaccurate results*, *expensive* and *time-consuming* development, *lack of data* and *over-confidence*. However, concerning implementing the OSCAR framework, these disadvantages are acceptable:

- *Inaccurate results* caused by stochastic simulation runs require a large number of simulation runs to increase accuracy. The actual number depends on the quantity of random numbers and their functionality. The OSCAR simulation allows for parallel computing, reducing runtime on HPC clusters and multi-core computers. As HPC clusters are commonplace in industry these days, enough simulation runs are possible even during conceptual design phase development to achieve accurate results with stochastic inputs.
- Simulation development is usually *expensive* due to the time required by simulation experts to create the simulation model. However, the OSCAR simulation only re-

quires model integration and adaption. This can be achieved at a fraction of the cost of full model development.

- This also reduces the *time* required to integrate the OSCAR simulation. Once implemented into conceptual design procedures, overhead time reduces to defining new VESSEL parameters and MISSIONS. Due to the object-oriented nature of all OSCAR components (geographical maps, VESSELS, COMPONENTS, *etc.*), object re-use is encouraged, minimising future time overheads.
- Most simulation projects suffer from *lack of data* to create realistic outputs. The OSCAR framework aims to avoid that by including only data that is usually available during conceptual design. Designers know about VESSEL and COMPONENT parameters, albeit with uncertainties and estimations. Customers usually know the operational applications they intend to use the VESSEL for. Although this may not be in geographical shapefiles, conversion is required only once and can be re-used from then on. Moreover, OSCAR allows to add additional data if it exists (*i.e.* about environment, other VESSELS, *etc.*).
- Due to the concrete nature and the persuasive animations, simulation results often suffer from over-confidence of their users. However, the OSCAR simulation is supposed to be used by expert design engineers that are aware of this common misconception. Moreover, outputs allow including confidence intervals or distributed outputs, further underlying the uncertain nature of simulation results.

4.2 Functional specification

According to Cusumano et al. (2003), creating and following a functional specification greatly increases the chance of product success. This section details the functional specification for the OSCAR simulation project.

4.2.1 Objectives

The OSCAR simulation should be capable of showing that an operational simulation can be applied during conceptual aeronautical vessel design. The simulation should be able to accommodate the OSCAR framework, namely the SCENARIO and VESSEL framework. The simulation must allow detailed, geographical VESSEL life cycle modelling. The OSCAR framework requirements also apply: the simulation should be generic, comprehensible, realistic and modular throughout.

4.2.2 Data requirements

Following good simulation modelling guidelines, the OSCAR simulation should keep data and model entirely separate. The OSCAR simulation must be capable of reading geographical maps using the standard shapefile format. It is acceptable to pose specific constraints upon the shapefile structure. Each shapefile feature must be characterized specifically in a database, conforming to the OSCAR framework definition of MISSIONS, TRACKS and SEGMENTS. VESSELS must be able to interact with shapefile features. All VESSEL, COMPONENT and Base parameters can be specified in a separate database. For simplicity, the input and output data should reside in one database file in order to allow storing simulation setup and results together. This allows checking previous simulation runs and results easily without re-running the simulation.

4.2.3 Level of Detail

In general, the simulation should be as detailed as necessary and as simple as possible. However, OSCAR puts focus on some aspects of VESSEL life cycle performance while neglecting others. Therefore, the LoD varies throughout the simulation model.

The exact nature of geographical shapefiles requires a high LoD for geographical implementation. Each shapefile feature has an exact location. MISSION, TRACK and SEGMENT definition include exact height and speed profiles. Shapefiles must be 2D, neglecting elevation.

VESSELS must be generic enough to model most existing moving vessels. They must be able to follow geographical features at specific height and speed profiles. However, VESSELS are not capable to turn, accelerate and climb realistically. Instead, step changes replace realistic behaviour. VESSEL performance must capture energy consumption based on speed. Other influences are neglected. However, a plug-in enables adding more detailed performance modules.

COMPONENTS fail based on a specific weibull distribution only. Each COMPONENT has a specific probability to cause VESSEL loss upon failure. COMPONENTS are maintained through unplanned maintenance and can be replaced during operation by redundant COMPONENTS onboard. There is no interaction between COMPONENTS beyond redundancy. Only physical COMPONENT deterioration can be modelled, software bugs are not included.

4.2.4 Animation

Model animation is as important to the model developer as to all other stakeholders (Banks & Gibson 2009). Simulations usually produce a large amount of data making it difficult for managers to access relevant information (Heilala & Maantila 2010). OSCAR simulation animation must enable model validation and verification (see Section 4.4.2) as well as promote client understanding and comprehensibility. All geographical shapefile features must be visible on a 2D

map of the world as MISSIONS, TRACKS and SEGMENTS. VESSEL operations must show clearly on the map, including details about the VESSEL performance and MISSION. Each VESSEL and COMPONENT agent state chart must be accessible during runtime to observe specific behaviour. All simulation parameters and variables must be accessible during runtime. A user-friendly GUI (Graphical User Interface) enables easy navigation and interaction. It allows dynamic update of database parameters before simulation start. Last, users must be able to turn off all animations to run the model in fast mode.

4.2.5 Outputs

Each simulation run must output all data generated on MISSIONS, TRACKS, SEGMENTS, VESSELS and COMPONENTS. The simulation only stores raw data without post-processing. Users can post-process outputs as desired. Data is stored in the same database file as the simulation input data. Output data is divided into operational performance data and failure data. Output is object-oriented: operational performance objects refer to each SEGMENT. Failure objects refer to each failure or VESSEL loss.

4.3 Software selection

There are many simulation software packages available to prospective modellers. They vary in price, capabilities, available support and learning curve. However, it is not possible to survey all simulation software and arrive at the ideal candidate based on modeller requirements. Many software features cannot be judged objectively and vary in importance based on application needs (Law & Kelton 1997).

Still, this section attempts to follow the software selection process briefly. Section 4.3.1 presents the software requirements to implement the OSCAR framework. Section 4.3.2 introduces AnyLogic, the tool of choice. Last, Section 4.3.3 lists a number of alternative packages and their distinct disadvantages regarding the OSCAR simulation.

4.3.1 Requirements

Based on the OSCAR framework, there are a number of specific requirements for simulation software to be able to create the OSCAR simulation. Initially, the decision to employ purpose-build simulation software avoids coding all model capabilities from scratch. Based on that decision, the single most important requirement is the capability to employ *agents* due to the object-oriented character of VESSEL life cycles (Section 3.1.4). There are a number of additional requirements that the simulation software must comply with:

- Animation: the simulation software should feature extensive and customisable animation capabilities including geographical map support to allow creation of a user-friendly model.
- Extensibility and external plug-ins: The simulation software should allow program extensions for every aspect of the model (agents, environments, experiments, *etc.*). Moreover, it must not be self-contained, i.e. it should allow and support external plug-ins and communication to external data sources.
- Integration: An open software structure allows importing the model into larger software constructs. Communication with other software should be possible and easy. Ideally, the software is capable to run as a stand-alone program without the need for a license. It should allow to be called externally.
- Parallel computing: The software should allow running model instances on multiple cores in parallel. This reduces the computing time for a large number of simulation runs to achieve statistically sound outputs.
- Random numbers: The software must allow full access on random number generations, including custom random number streams for each random variable. Moreover, independent replication runs must be supported. This ensures reliable and unbiased outputs.

4.3.2 AnyLogic

Based on the requirement set above, the software of choice is AnyLogic by "The AnyLogic Company" (formerly "xjTek"; www.anylogic.com). It supports agent-based modelling and geographical shapefile integration explicitly. It has extensive and flexible animation capabilities and is Java-based. This ensures open communication with external software or data because the standard Java-libraries can be used. Moreover, any Java package can be imported and used within AnyLogic, reducing development time to write specific functions. Moreover, Java standalone applets and applications can be created from the simulation model. These can be used on any machine, independent of the operating system and without an AnyLogic license. Parallel simulation runs can be executed on multi-core processors or on HPC clusters. Last, AnyLogic allows full control on random numbers.

4.3.3 Alternatives

There are a number of alternative packages capable of implementing the OSCAR framework. However, each posed one or more distinctive disadvantages:

- *Arena* is a discrete-event simulation tool offering similar capabilities to AnyLogic (www.arenasimulation.com). However, it is not truly object-oriented, making it difficult to create agent models easily. Moreover, Arena cannot use geographical shapefiles.
- *Simio* (www.simio.com) is a direct competitor to AnyLogic because it offers object-oriented simulation capabilities to industrial users. It was created by Arena developers trying to improve on Arena's inherent disadvantages. Simio focuses on production management and intelligent scheduling. However, it lacks direct agent support (although rudimentary agents are available) and cannot integrate shapefiles.
- *FlexSim* (www.flexsim.com) is very similar to Simio capabilities and lacks shapefile and agent support.
- *ExtendSim* (www.extendsim.com) is similar to Arena in its focus on classical discrete-event simulation and lack of object-oriented principles.
- *Repast Symphony* (http://repast.sourceforge.net/) is an open-source simulation platform focussing on agent-based modelling. It is Java-based, supports parallel simulation runs and allows import of geographical shapefiles. However, agents cannot readily interact with shapefile features. Instead, features must be drawn manually within the Repast IDE. Repast is difficult to extend and integrate into other software as it employs its own programming language.
- *Flames* (www.ternion.com) is an expensive COTS simulation framework that complies well with the plug-in approach postulated by the OSCAR framework. Users develop sub-models simulating sub-systems and plug them into the Flames application. The software is highly extensible and provides only basic library items. However, it features its own programming language and requires specific runtime licenses to run models on client computers. Flames supports agents interacting with shapefile features. However, the fidelity of scenario definitions and physical models is very high: it is designed primarily for military tactic and strategic battlefield decision support. Despite multi-core support, single scenario runtimes can be very long.
- *Presagis* (www.presagis.com/) is similar in capabilities to Flames. It has the same target application of military decision support, requiring high fidelity models of geographical 3D environments and physical processes. The tool is too sophisticated for conceptual design phase processes and very expensive.

4.4 Overview

Before describing each part of the OSCAR simulation in more detail, this section provides a brief overview on the model structure. Subsequently, it presents a walkthrough of a simple sample model to increase understanding of model handling.

4.4.1 Model structure

The OSCAR simulation model consists of three distinct components, namely the Geographical Information System, the database and the simulation model. Figure 4-1 displays the relation of the three components in a typical application workflow.

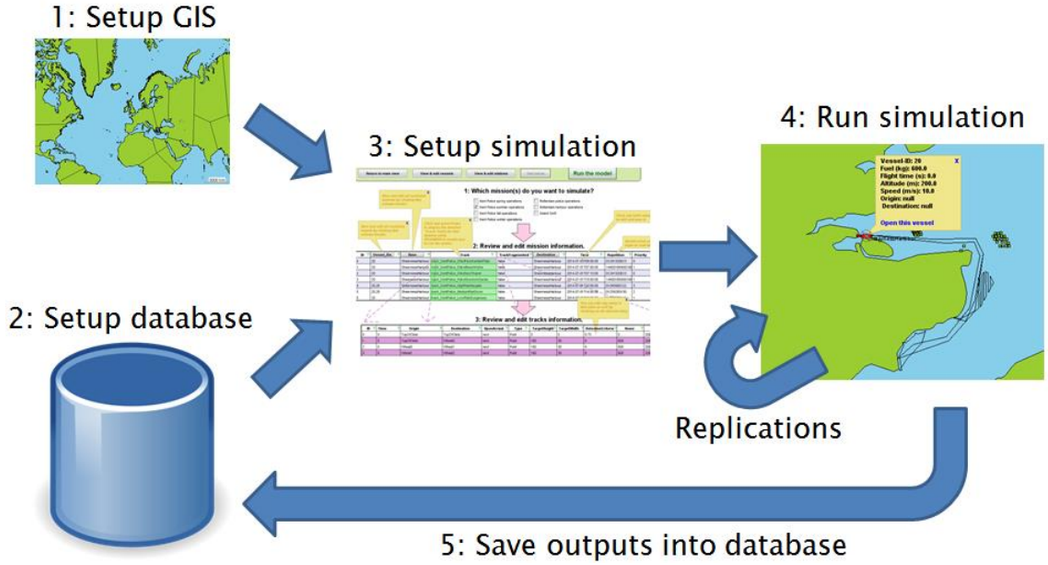


FIGURE 4-1: OSCAR SIMULATION STRUCTURE AND WORKFLOW.

First, the user constructs the geographical shapefiles (Section 4.6 and Appendix 10) for the TRACKS as required. Second, he generates the database inputs for the given SCENARIO (Section 4.5). Upon starting the simulation model, it loads the Geographical Information System and database inputs. The user can opt to refine inputs on the SCENARIO and simulation setup (Section 4.13). Subsequently, the simulation conducts the SCENARIO replications (one or many) according to the simulation setup. It saves the outputs of each replication into the database. Once all replications finish, the simulation model closes and the user can analyse output data in the database. He can choose to refine database inputs or the geographical setup and rerun the cycle.

The simulation model itself consists of a number of Java active objects as in Figure 4-2. The user can choose to load one of two experiments that load specific simulation setups (Section 4.13). Once the simulation model starts, it creates a “Main” active object. It loads the geographical map and defines it as the agent environment. All agents “live” in this geographical map environment. The “Main” active object accommodates two Java active object classes itself. The

"Base" class holds instances of agents representing bases (Section 4.7). The "Vessel" class holds instances of agents representing VESSELS (Section 4.8). Within the VESSEL class, each VESSEL instance uses a specific performance module (Section 4.9), a specific payload module (Section 4.10) and a search-and-rescue module, if applicable to the VESSEL MISSION (Section 4.11). Note that these modules are not Java classes. Instead, they are separate modules that employ existing code within OSCAR or load external module code. The "Vessel" class comprises a sub-class "Components" which holds instances of agents representing COMPONENTS (Section 4.12).

Appendix 6 details a model walkthrough in order to comprehend how a geographical map is created, inputs are specified and the simulation is set up and run.

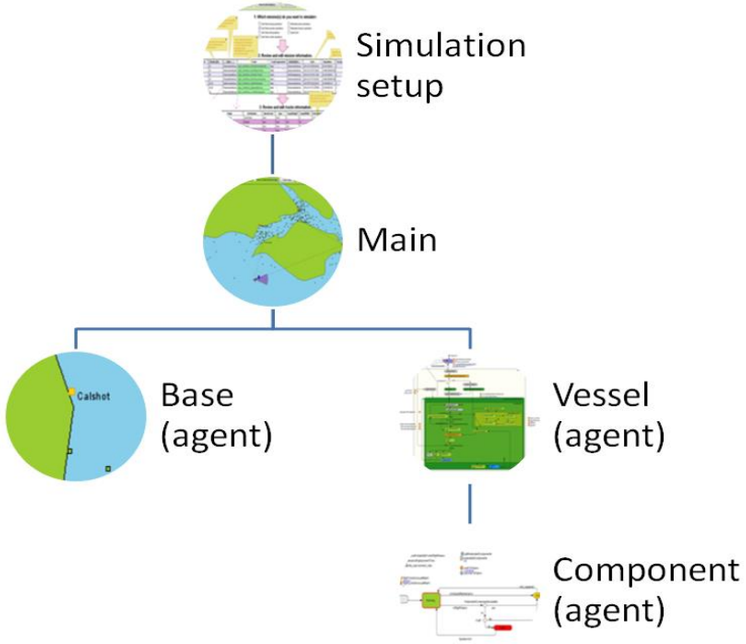


FIGURE 4-2: OSCAR SIMULATION MODEL STRUCTURE HIRARCHY.

4.4.2 Verification & Validation

The Project Management Institute (PMI 2013) defines verification and validation as:

> *Verification: The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process.*
>
> *Validation: The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.*

More informally, verification is often associated with the question "Are you building the thing right?" whereas validation asks "Are you building the right thing?". Verification is independent of the modelling context, ensuring that a fault-free tool was used, programming was conducted correctly and that the programming tool features were applied correctly (Sargent 2007). OSCAR simulation verification was conducted internally throughout model development applying standard software engineering techniques such as unit testing, black-box testing and walk-throughs (Banks & Gibson 2009). Sawyer and Brann (2009) describe the difficulty of implementing the widely used JUnit[1] unit testing tool into AnyLogic. Following their recommendation, the AnyLogic IDE was used to create custom unit tests instead.

With regards to model validation, Steinkeller (2011) and Glas (2013) note that model validation in aeronautical conceptual design phase cannot be conducted due to the lack of experimental data and physical testing. Similarly, Sterman (2000) argues that the term is misleading as one can never formally prove that a model represents reality. Instead, validation should be used to ensure a model is convincing and useful. In order to make the OSCAR simulation model convincing and *useful*, rigorous model building processes were followed:

The OSCAR simulation model development reduced components and features to the minimum complexity required to achieve the desired functionality. From the start, clients were involved closely: Initially, the DECODE ("Decision Environment for COmplex Design Evaluation") research team (Section 5.2) represented the clients. Later, the RNLI (Royal National Lifeboat Institution, see Section 5.3.1) and the PRA (Port of Rotterdam Authority, see Section 6.3.1) adopted the client role. Another factor for successful model validation is useful documentation: AnyLogic incorporates a "Description" tab for any model feature: this was used extensively. Moreover, code documentation follows best practices and the external database items are documented as well.

For the case study in Chapter 5, traditional model validation was conducted by comparing the baseline scenario (Section 5.3.1) to available operational metrics. This was not possible with the second case study (Chapter 6) as the scenario did not exist in reality.

4.5 Data

This section describes the data management within the OSCAR simulation. One database file stores all input and output data. This allows complex database queries connecting input and output data tables. The database engine of choice is SQLite, the "most-widely deployed SQL data-

[1] Available at www.junit.org, accessed 05/12/2013.

base engine in the world"[2]. Appendix 7 describes each table type within the database in more detail. The remainder of this section portraits the application of the database within the OSCAR simulation, including reading in data, runtime data handling and storing output data.

The database inputs are loaded into the OSCAR simulation at different points, depending on the chosen experiment (Section 4.13). For the single run experiment, all required MISSION tables are loaded upon creating the "Main" active object on start-up (Figure 4-2). Moreover, all BASES, VESSELS and SEGMENTS required for the chosen experiment are loaded from the database. Users cannot influence the data during the start-up period. For the interactive single run experiment, all database input tables are loaded into the experiment setup page. Users can interact and amend data as required. Upon starting the experiment, the amended user data is loaded into the simulation. For the "RunFast" experiment, database information is loaded on creating the "Main" object, similar to the "Simulation" experiment. Users can only amend data beforehand by editing the database directly.

During simulation runtime, no input data is loaded into the simulation. Moreover, output data is stored within the simulation model until simulation termination.

Data output upon simulation termination depends on the selected experiment type. For the single run experiment, no outputs are copied into the database. For the interactive single run experiment, SEGMENT and maintenance outputs are copied into the respective tables after the simulation run finished. For the "RunFast" experiment, Segment and maintenance outputs are copied into the respective tables after the *every* simulation iteration finished.

4.6 Geographical modelling

As discussed in Section 3.2.4, one of the novel advances of OSCAR is its implementation of geographical modelling into conceptual aeronautical design. This section explains the practical application of geographical modelling within the OSCAR simulation. The simulation software AnyLogic (Section 4.3.2) offers some basic build-in capabilities to display geographical shapefiles. However, enabling agents to interact with shapefile features required considerable manual extensions because AnyLogic shapefile support is limited to displaying shapefiles as background maps without interactive capabilities. Appendix 8 details the essential structure required to import and use shapefile data within the OSCAR simulation. Moreover, it describes the actual data import and how the data is turned into objects for AnyLogic processing. The rest of this section presents how the simulation applies shapefile data for realistic VESSEL operations.

[2] According to www.sqlite.org, accessed 08/11/2013.

Moreover, it discusses how future work should employ the object-oriented paradigm of Java to turn shapefile features into Java objects.

4.6.1 Application

After the OSCAR simulation loaded and consolidated shapefile and database SEGMENT data (Appendix 8), this section describes how the data is processed within the simulation.

VESSELS move on the geographical map visually but they do not interact with map features directly. Instead, interaction occurs with GISPOSITIONFULL objects (GIS abbreviates Geographical Information System). Figure 4-3 displays when VESSELS read and follow GISPOSITIONFULL objects during operations (See Section 4.8 for full description of VESSEL state chart).

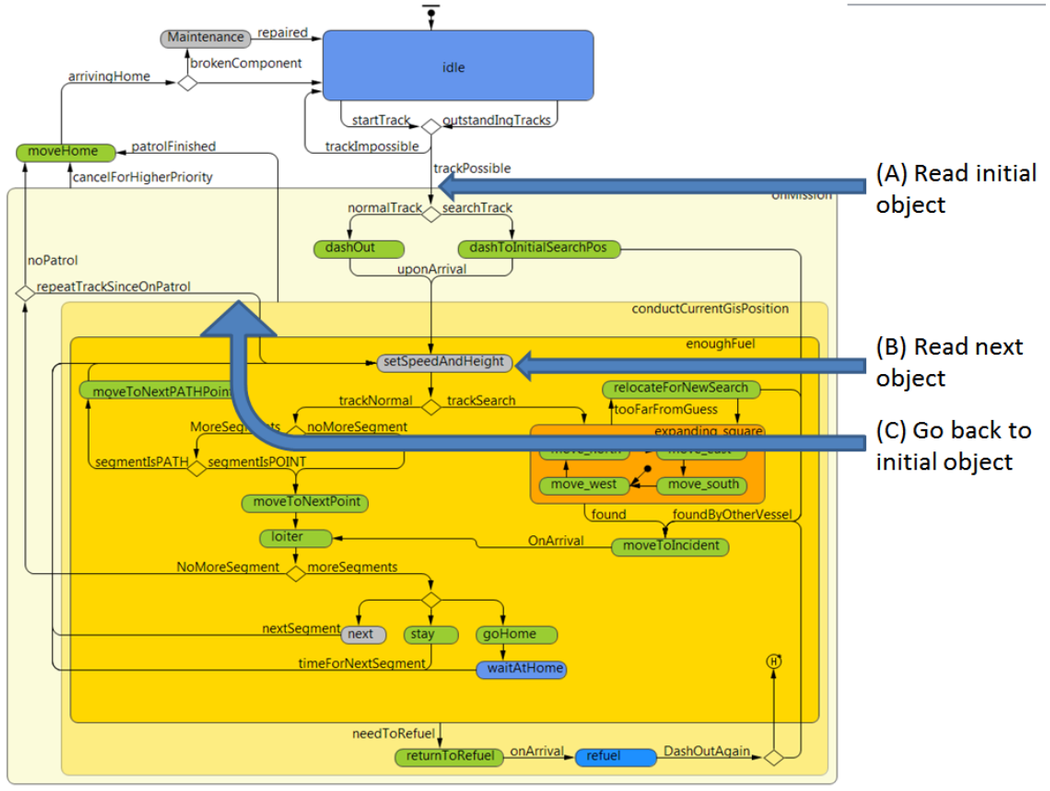


FIGURE 4-3: GISPOSITIONFULL APPLICATION DURING VESSEL OPERATIONS. SEE 4.8 FOR FULL DESCRIPTION OF VESSEL STATE CHART.

The initial GISPOSITIONFULL object is read before the VESSEL starts to dash out to the initial TRACK SEGMENT (A). After finishing the dash phase, the VESSEL moves to each GISPOSITIONFULL object in turn (B), reading in the next GISPOSITIONFULL object. If the VESSEL is patrolling, it will re-load the initial GISPOSITIONFULL object when re-starting the patrol (C). Otherwise, the TRACK is finished when the last SEGMENT and the last GISPOSITIONFULL is reached.

4.6.2 Future work

AnyLogic can only display geographical maps without support for agent interaction. Therefore, the OSCAR simulation employs a custom approach consolidating shapefile data with external database SEGMENT data to create the arbitrary GISPOSITIONFULL objects (Appendix 8). However, two improvements would simplify geographical-agent interactions without the need to revert to specialist software.

First, Segment data (`Height`, `Speed`, *etc.*, see Appendix 1) could be stored in the shapefiles directly using the dbf-tables associated with every shapefile. All TRACK tables in the current database would become void because the data is stored with the shapefile features directly. This would simplify data handling and data consolidation becomes redundant. However, custom algorithms reading all data from shapefiles would be required.

A second approach to simplify data consolidation employs the agent capabilities of AnyLogic. Here, every MISSION, TRACK and SEGMENT becomes an agent-object featuring the parameters that correspond to the database table columns. MISSION agents contain any number of TRACK agents, themselves containing any number of SEGMENT agents. This setup also simplifies a number of algorithms aimed to manage current MISSIONS, TRACKS and SEGMENTS. TRACK agents, for example, can launch themselves when it is time, asking for the required VESSEL. SEGMENT agents consist of coordinate objects (one for Point SEGMENTS, several for Path SEGMENTS) and collect statistics on their specific performance individually. However, VESSEL life cycles spanning decades including thousands of MISSIONS may create computer memory problems due to the large number of SEGMENT agents involved. Intelligent agent creation and destruction algorithms would be required.

4.7 Base class

Any VESSEL operation starts at a specific location and ends at a specific location (unless the VESSEL was lost during the operation). Aircraft start and land at airports, ships dock at harbours, cars park, trains stop at stations, *etc*. The BASE class within the OSCAR simulation provides a generic platform to model start and end points of operations.

Based on the `equipment_Bases` table in the database (Appendix 7), the OSCAR simulation creates BASE class instances for any BASE mentioned in the `Base` or `Destination` column of any MISSION table loaded for the current experiment. `StationName`, `StationID` and geographical coordinates specify each BASE instance. VESSELS use BASE instances to move to for refuel, breaks or upon the end of a TRACK. Figure 4-4 shows the visual representation of a BASE instance within the OSCAR simulation.

FIGURE 4-4: BASE INSTANCE REPRESENTATION: YELLOW SQUARE WITH STATIONNAME.

4.8 Vessel class

The VESSEL class is a major component of the OSCAR simulation, providing a framework to simulate a large variety of vessels using one Java class only. This section describes the mode of operation of this class in more detail.

Each VESSEL instance is defined by a unique set of parameters loaded from the database. Appendix 3 summarises all standard VESSEL parameters. Moreover, optional parameters exist to feed the add-on performance model for fixed wing aircraft VESSELS (Appendix 9). These are only applicable to VESSELS of the correct category and type. During VESSEL operations, the agent refers to its parameters to calculate specific information. Parameters do not change during a simulation run.

Figure 4-5 shows the generic VESSEL state chart used within the OSCAR simulation for every VESSEL agent.

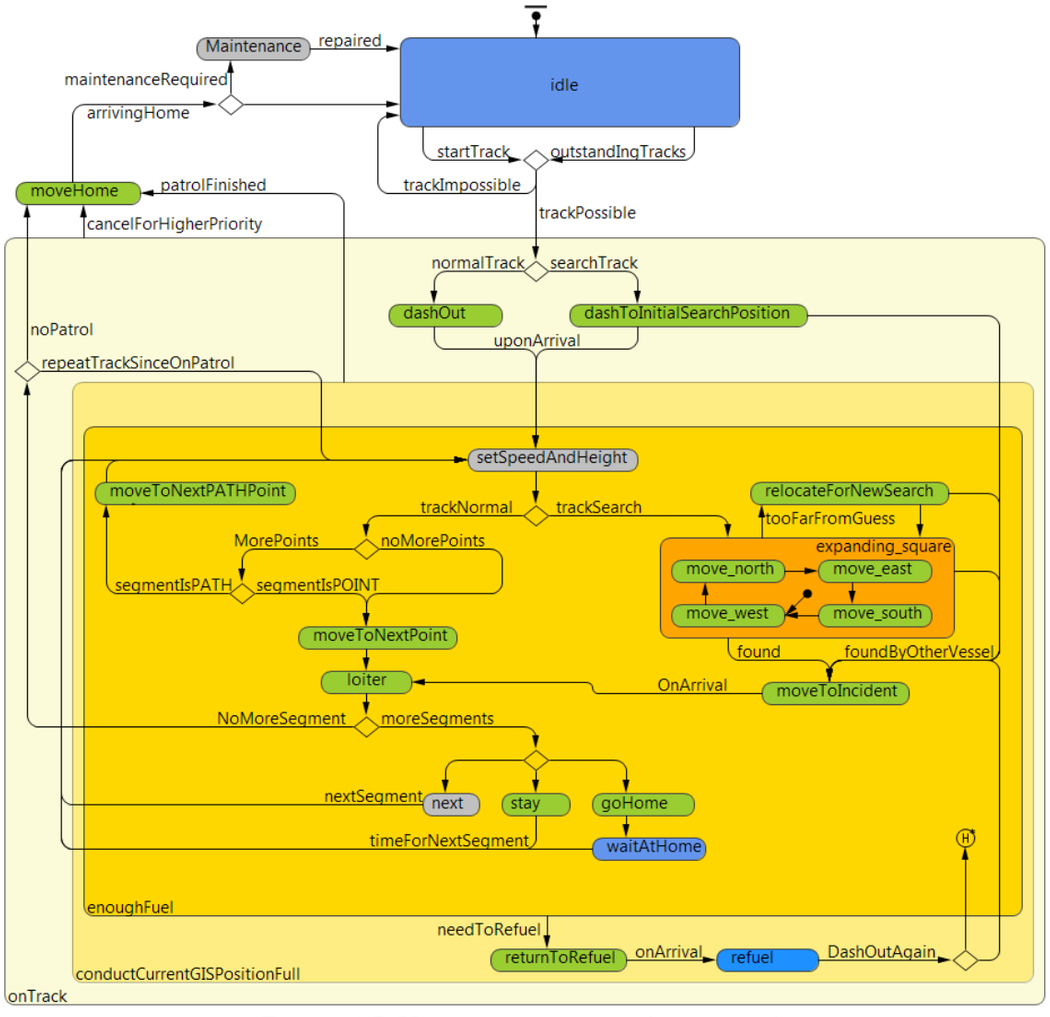


FIGURE 4-5: VESSEL STATE CHART (SIMPLIFIED).

Upon VESSEL creation, the agent rests in the "idle" state. A TRACK operation commences either if a TRACK'S `Time` parameter triggers its start or if the VESSEL has outstanding TRACKS it could not commence on time (due to delays). The VESSEL checks if it can conduct the entire TRACK as requested concerning its energy capacity. This check is not done for nuclear powered VESSELS (assuming infinite energy) and food powered VESSELS (a hungry human can still walk). All other VESSELS conduct the requested TRACK virtually (i.e. before the agent actually moves in the simulation), including dash, return, possible scheduled refuels and loitering. If it detects running out of energy at any point, the VESSEL cancels the entire TRACK. Outputs will be marked with "cancelled due to fuel" (compare output Table A-6).

If the TRACK is possible, the VESSEL dashes out to the initial POINT or the start point of the initial PATH at the `DashHeight` and `DashSpeed` specified in the TRACK table. However, if the TRACK is a search-and-rescue mission, the Vessel dashes to the "initial search position" described in Section 4.11. Upon completing the dash phase, the VESSEL changes speed and alti-

tude to match the upcoming SEGMENT. Once more, a distinction between normal and search-and-rescue TRACK is made.

For normal TRACKS, the VESSEL checks if the current SEGMENT has more points (only applicable for PATH SEGMENTS) or if this is the last point of the SEGMENT (always true for POINT SEGMENTS). In the latter case, the VESSEL simply moves towards the last point of the SEGMENT. In the former case, another check occurs to see if the current SEGMENT is of type PATH or POINT. If it is a PATH SEGMENT, all points of the PATH are conducted in sequence repeating the loop "moveToNextPATHPoint". If the last point of a PATH is reached or if the SEGMENT is a POINT SEGMENT, the VESSEL moves to this point. Upon arrival, it loiters for the duration specified in the SEGMENT characteristic `Loiter`.

For search-and-rescue TRACKS, SEGMENT treatment is fundamentally different, thus requiring an additional state chart branch. The VESSEL follows a user-specified pattern (expanding square pattern in this example) described in more detail in Section 4.11. Upon discovering the search-and-rescue incident, the VESSEL also loiters as with normal TRACKS.

Once the loiter `duration` is over, the VESSEL checks if the current TRACK has more SEGMENTS for operation. If so, the current SEGMENT'S `uponArrival` determines subsequent behaviour. If `uponArrival=next`, the VESSEL adjusts its speed and altitude and conducts the next SEGMENT as above. If `uponArrival=stay`, the VESSEL will keep loitering until the subsequent SEGMENT `Time` dictates to start the SEGMENT. If that `Time` already passed, the VESSEL will commence immediately. If `uponArrival=home`, the VESSEL will move to the current TRACK'S `Destination` (not `Base`). On arrival, it waits until the subsequent TRACK'S `Time` passes, upon which it will depart for the next TRACK'S initial point. If the `Time` already passed, it will commence from the `Destination` immediately.

If the VESSEL found "NoMoreSegments" after loiter, it will attempt to move home to the TRACK'S `Destination`. However, if the current TRACK is an active patrol as defined by the TRACK'S `Repetition` characteristic (Appendix 2), the VESSEL repeats the entire TRACK. Once the patrol duration is over, the VESSEL stops the current TRACK (independent of the current SEGMENT) and moves to the `Destination`.

Upon arriving at the `Destination`, the VESSEL checks if any of its COMPONENTS broke during the operation or if any COMPONENT scheduled a planned maintenance (see Section 4.12). In such a case, maintenance is conducted on the COMPONENT for the duration specified in the COMPONENT'S `MaintenanceReplacementTime` characteristic. Subsequently, the VESSEL becomes "idle" again.

While the VESSEL conducts a TRACK, it continuously monitors its remaining energy (*i.e.* fuel) level (except for nuclear powered and food powered VESSELS). If there is not sufficient fuel to reach the closest BASE without breaking a 10% energy reserve, the VESSEL leaves the "enough-

Fuel" composite state to return to the closest BASE for refuelling. After 15 minutes of refuelling at the BASE, the VESSEL re-commences its TRACK operation where it left off.

4.9 Propulsion performance module

This section describes two approaches to propulsion performance modelling within the OSCAR simulation. The OSCAR simulation can load different propulsion models for each VESSEL agent. The VESSEL parameter `performanceModel` (Appendix 3) defines the propulsion module required. If `performanceModel=powerAgainstSpeed`, the generic propulsion model in Section 4.9.1 will be used. To load different modules, the naming convention "`category_type_fuelType`" must be used. Hence, an aircraft VESSEL with fixed wings burning petrol will have `performanceModel=fixedWing_aircraft_petrol`. Section 4.9.2 introduces a propulsion add-in for aircraft, discussing the structure and requirements for external plug-ins to be used within the OSCAR simulation.

4.9.1 Generic propulsion model

4.9.1.1 Inputs

Based on the discussion in Section 3.3.4, the generic propulsion model uses a VESSEL-specific relationship between speed and energy consumption to compute the operational energy consumption over the VESSEL life cycle. Hence, each VESSEL features two parameters `speedValues` and `powerValues` describing the relationship (see Table A-2). Figure 4-6 shows two sample database entries for an Audi A3 and a lifeboat featuring different number of entries and range of speeds.

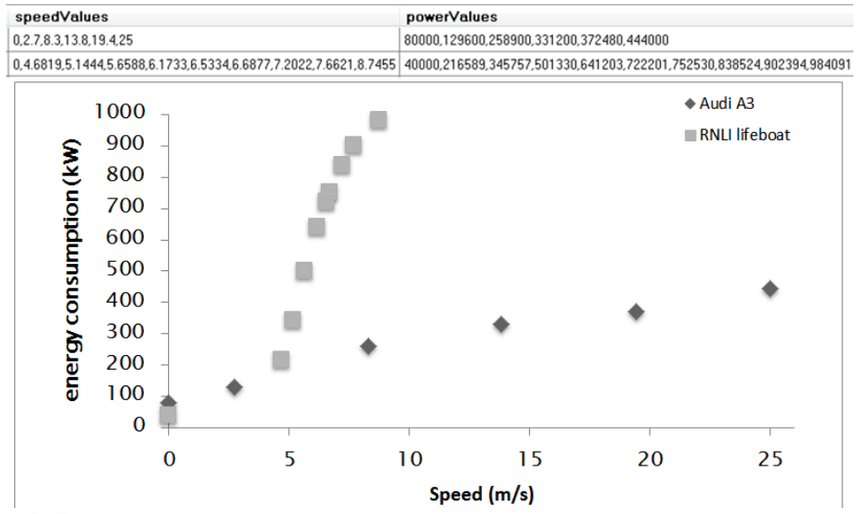


FIGURE 4-6: GENERIC PROPULSION MODEL INPUTS AS DATABASE INPUTS (UPPER SECTION) AND VISUAL REPRESENTATION (LOWER GRAPH).

Upon VESSEL agent creation, the OSCAR simulation loads both parameters into a table function.

4.9.1.2 Processing

During simulation runtime, the VESSEL agent consumes energy based on the inputs described above. While the VESSEL operates (i.e. not at a BASE or on maintenance), it updates its energy consumption after every SEGMENT. Using linear interpolation, the agent queries the speed-power table function providing the speed used during the previous SEGMENT. The returned power value is multiplied with the duration it took to complete the SEGMENT to obtain the total SEGMENT energy consumed. Simultaneously, the consumed energy is converted into fuel used by dividing by the VESSEL'S calorific value. This, in turn, depends on the VESSEL'S `fuelType` as follows:

TABLE 4-1: VESSEL FUELTYPE AND CORRESPONDING FUEL CALORIFIC VALUE.

`fuelType`	calorific value (in MJ/kg)
`petrol`	44.4
`diesel`	41.1
`coal`	20
`nuclear`	infinity[3]
`food`	0.003
`electric`	infinity[4]

[3] Assuming that nuclear material holds vastly more energy per unit weight than conventional fuel types.

[4] Assuming that electric batteries do not lose weight while depleting energy. An infinite calorific value ensures that the output $fuelUsed = 0$ while $energyUsed \neq 0$.

4.9.1.3 Outputs

The energy used is stored in the database output table `output_Segments` (see Table A-6) with the appropriate SEGMENT entry in the column `energyUsed`.

4.9.2 Custom aircraft performance model

The OSCAR simulation enables users to provide individual performance modules for more realistic performance calculations. This chapter describes the steps required to implement a custom performance module for any VESSEL type. In order to implement a new performance module, the following steps are required:

1. Create and code the model according to user needs
2. Convert it to Java code and compile it into a Java archive (.jar) file.
3. Copy the.jar file into the model folder and name it following the naming convention "`performance_category_type_fuelType`" depending on the applicable VESSEL `category`, `type` and `fuelType`. If the module covers several `categories`, `types` or `fuelTypes`, copy the.jar file and rename it accordingly.
4. Add the.jar file to the model dependencies
5. Adjust the model:
    a. Add the additional required VESSEL parameters.
    b. Add a new variable named like the.jar file, *i.e.* "`performance_category_type_fuelType`"
    c. Load the module in the VESSEL function `createPerformance()` using existing if-function structure (Figure 4-7).
    d. Adjust energy calculation code in functions `getSegmentEnergyUsed()` and `getSegmentFuelUsed()` using existing if-function structure (Figure 4-7).

```
if (performanceModel.equals(fixedWing_aircraft_petrol))
    {
    // do something with fixedWing_aircraft_petrol
    }
else if (performanceModel.equals(powerAgainstSpeed))
    {
    // do something with powerAgainstSpeed
    }
```

FIGURE 4-7: NEW PERFORMANCE MODULES IMPLEMENTATION THROUGH IF-STATEMENTS.

Appendix 9 describes a sample custom performance module used for aircraft fuel burn calculations. This sample module is applied in the case studies in Chapters 5 and 6.

4.10 Payload module

This section describes the implementation of the payload module add-on for active payload as introduced conceptually in Section 3.3.6.2. This add-on creates a realistic representation of electro-optical sensors of the visual and infrared light spectrum.

4.10.1 Inputs

The OSCAR simulation loads each of the six parameters described in Chapter 3.3.6.2.1 as VESSEL parameter objects from the `equipment_Vessels` database table (Table A-2).

4.10.2 Processing

While a VESSEL operates on a SEGMENT (*i.e.* not on a dash or return, refuel or "stay", compare Figure 4-5), it monitors its environment using the on-board camera defined in the camera parameters above. It records one image every time its camera footprint (Appendix 5) covers an entire new area (for airborne VESSELS) or every 10 seconds (for ground-based and submerged VESSELS). Figure 4-8 schematically compares the two approaches.

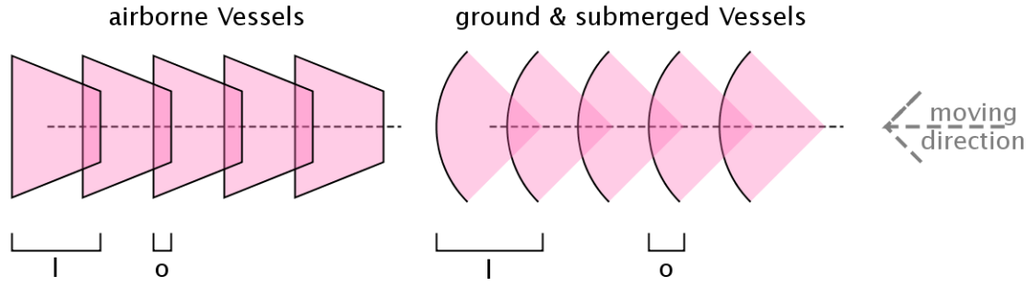


FIGURE 4-8: CAMERA FOOTPRINT OVERLAPPING SCHEMATIC.

Appendix 5 explains calculation of the footprint area for all VESSELS. The overlap *o* for ground and submerged VESSELS varies depending on VESSEL speed as

$$o = V * t \qquad \text{Eq. 4-1}$$

where *V* is the VESSEL speed and *t* is the overlap time of 10 seconds. Every time the VESSEL records an image, the net new area scanned (equivalent to the light pink areas in Figure 4-8) is added to the output `areaScanned`. Moreover, the `imagesTaken` is updated and the `dataAcquired` changes as

$$d_a = p_h \times p_v \times \tau^2 \qquad \text{Eq. 4-2}$$

Where d_a is the acquired data in bytes, p_h are the horizontal camera pixels (OSCAR parameter `cameraPixelshor`), p_v are the vertical camera pixels (OSCAR parameter `cameraPixelsVer`) and τ is the number of bytes per pixel. Assuming the common pixel format RGBA32, $\tau = 4$. After converting to Megabytes, the value is added to the SEGMENT output `dataAcquired`. Eq. 4-2 assumes no file compression (i.e. raw picture file) and neglects RGBA32 header data.

Every time the VESSEL records a camera image, it checks if the current SEGMENT target requires detection (if not, `targetWidth=targetHeight=0`, see Appendix 1). If the target requires detection, the next check tests if the target position is within the camera footprint. If so, the detection probability algorithm determines the likelihood of the camera detecting the target using the following discussion based on work carried out by Amrith Surendra, drawing on Leachtenauer and Driggers (2001) and Gundlach (2012).

Digital imaging cameras use a collection of individual detectors (i.e. pixels) to form an image. The focal plane array arranges the individual pixels in a plane. The field-of-view at the focal plane array is the angular view of the focal plane. In general, sensor designers determine the field-of-view characteristics (Leachtenauer & Driggers 2001). One of the fundamental parameters that govern image quality is GSD (Ground Sample Distance) as shown in Figure 4-9.

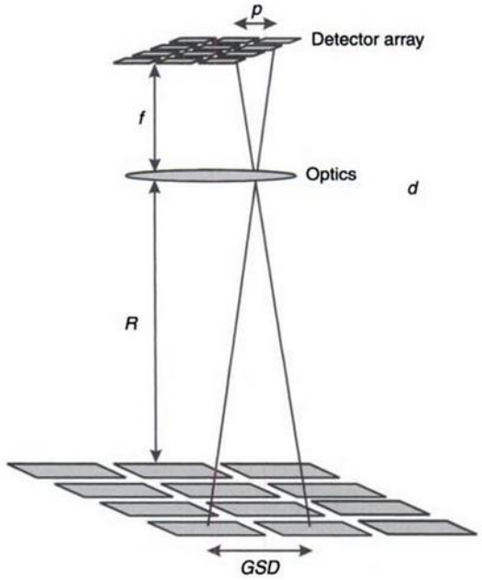


FIGURE 4-9: GROUND SAMPLE DISTANCE (GSD) DEFINITION. REPRODUCED FROM LEACHTENAUER AND DRIGGERS (2001).

GSD is a function of the focal plane array, optics, and collection geometry. The horizontal GSD definition is

$$GSD_h = 2 \times \tan\left(\frac{\Delta_h}{2 \times p_h}\right) \times R \qquad \text{Eq. 4-3}$$

where Δ_h is the horizontal camera field of view (OSCAR parameter `cameraFOVhor`), p_h is the number of horizontal pixels (OSCAR parameter `cameraPixelshor`) and R is the slant range. The vertical GSD definition is

$$GSD_v = \frac{2 \times \tan(0.5 \times \Delta_v \times p_v)}{\cos(\theta_{look})} \times R \qquad \text{Eq. 4-4}$$

Where Δ_h is the vertical camera field of view (`cameraFOVver`), p_v is the number of vertical camera pixels (`cameraPixelsVer`) and θ_{look} is the look angle as defined in Figure 4-10.

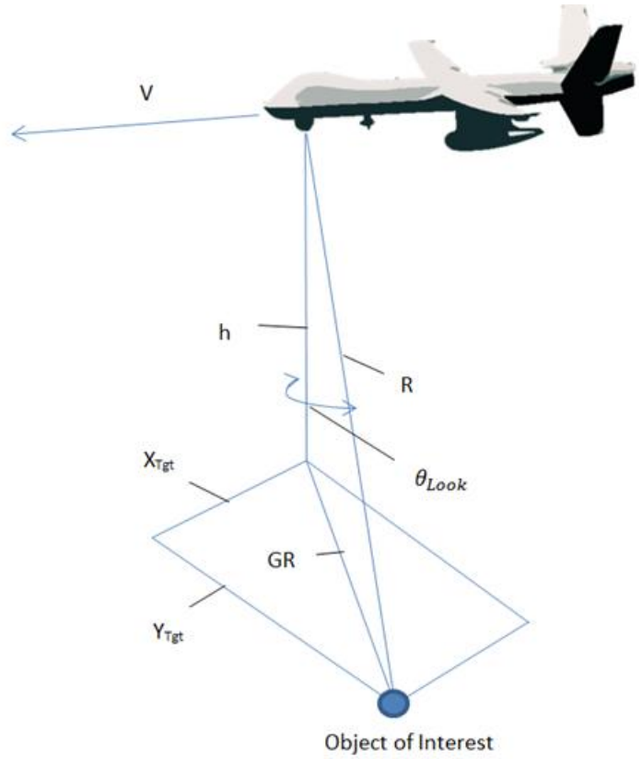


FIGURE 4-10: CAMERA GEOMETRY CONVENTIONS. REPRODUCED FROM GUNDLACH (2012).

The slant range *R* is defined as

$$R = \sqrt{h^2 + GR^2} \qquad \text{Eq. 4-5}$$

where *h* is the VESSEL `height` and *GR* is the ground range between the Vessel and the target.

The GSD acts as metric to identify the performance of the camera. However, GSD is not a metric for image quality. To determine if image quality is sufficient for target detection, empirical approaches are required. In this research, the "Johnson criterion" defines three levels of object discrimination, namely detection, recognition, and identification (Leachtenauer 2003). Detection occurs if an imagery feature is recognized to be part of a general group (i.e. vehicle, ship, aircraft…). Recognition is the discrimination of the target class (i.e. car, SUV, truck…). Identification is the discrimination of the target type (i.e. BMW, Mercedes, Porsche…). The sensor resolution (i.e. pixels per inch) determines the probability of detection, recognition, and identification. Using the Johnson criteria, targets are replaced by black and white stripes each constituting a cycle. The total number of cycles for a target of given dimensions is

$$d = \sqrt{\varsigma_H \times \varsigma_W} \qquad \text{Eq. 4-6}$$

where ς_H is the target height (`targetHeight`) and ς_W is the target width (`targetWidth`). `Both` are parameters of each SEGMENT end point (Section 3.2.3.1). The number of cycles across the target is

$$N = \frac{d}{2 \times GSD_{avg}} \qquad \text{Eq. 4-7}$$

Hence, the probability of detection is defined as

$$P(N) = \frac{\left({}^{N}/_{\gamma}\right)^{2.7+0.7\times(N/\gamma)}}{1 + \left({}^{N}/_{\gamma}\right)^{2.7+0.7\times(N/\gamma)}} \qquad \text{Eq. 4-8}$$

where γ is the detection criterion `(detectionCriteria) which` defines the 50% probability of successfully performing detection. For detection, $\gamma = 0.75$, for recognition $\gamma = 3.0$ and for identification $\gamma = 6.0$. As discussed in Section 3.2.3.1, `detectionCriteria` is a parameter of any SEGMENT.

Each time the payload camera records an image, OSCAR checks if the camera footprint (Appendix 5) contains the SEGMENT end point. If so, it calculates the probability of detecting the current SEGMENT end point based on the given inputs. This allows estimating the camera operational performance based on VESSEL performance. The case study in Chapter 6 demonstrates how camera performance figures influence overall VESSEL performance and value.

4.10.3 Outputs

The SEGMENT outputs `imagesTaken`, `areaScanned` and `dataAcquired` are updated during SEGMENT operation. Upon SEGMENT completion, these are saved into the database.

If a target is detected during SEGMENT operation, the SEGMENT output `timeOfSpotting` is updated accordingly. Moreover, the target is internally marked as "detected" to avoid subsequent camera searching (and re-detection).

4.11 Search-and-rescue module

Most real-life vessel operations can be modelled easily by applying the generic OSCAR framework using POINT and PATH elements. However, some vessel operations depend on dynamic environment variables, i.e. their mission path changes depending on factors unknown before departure. One example is search-and-rescue were vessels do not know the exact position of their target incident. Instead, they move along specific patterns until they find the incident or give up. Since the OSCAR simulation was used to assess maritime UAS (see Chapters 5 and 6), search-and-rescue operations featured prominently. Therefore, a custom search-and-rescue module improved usability for value-driven design because performance measurement became more precise. This section introduces the search-and-rescue add-in and demonstrates how additional modelling capabilities can be added to the OSCAR simulation.

Sub-section 4.11.1 describes search-and-rescue procedures and execution in the real word. Sub-section 4.11.2 presents the OSCAR simulation search-and-rescue module and how it is applied during runtime. Sub-section 4.11.3 describes the VESSEL state chart amendment for search-and-rescue operations. Last, sub-section 4.11.4 introduces specific SEGMENT targets used during OSCAR search operations, namely "Incidents".

4.11.1 Search-and-rescue in reality

The UK Maritime and Coastguard Agency (MCA 2008) defines search-and-rescue as

> *Search-and-Rescue: The undertaking of locating and recovering persons either in distress or missing, recovering them and transporting them to a save location.*

Search-and-rescue typically occurs in three distinct application areas, namely

- Maritime search-and-rescue uses maritime VESSELS to search for inshore, near-shore and offshore incidents.

- Aeronautical search-and-rescue uses airborne VESSELS (aircraft, helicopters and most recently UAS) to look for incidents over water and land.
- Inland search-and-rescue uses people, dogs and cars to find and recover persons in distress on land.

The OSCAR search-and-rescue add-in limits operations to maritime and aeronautical search-and-rescue operations because inland search-and-rescue patterns depend on local geography and infrastructure. Maritime and aeronautical search-and-rescue patterns are generic as they are used around the world.

Each country is responsible for search-and-rescue organisation and most countries signed various treaties guaranteeing publicly funded conduct of search-and-rescue to any person in distress. However, most countries feature several civil agencies, NGOs and military units working together to conduct search-and-rescue operations. Still, search vessels follow similar policies defined by the IMO (International Maritime Organisation) and ICAO (International Civil Aviation Organisation) in the International Aeronautical and Maritime Search-And-Rescue manual (IAMSAR 2007). Depending on the incident type, position, daylight, weather and the number as well as type of available search-and-rescue vessels, it recommends specific search patterns. Moreover, it provides search stop conditions if "all reasonable hope of rescuing survivors has passed" (ibid).

One specific pattern recommended for maritime search of persons or small boats is the "Expanding Square" pattern shown in Figure 4-11 (A).

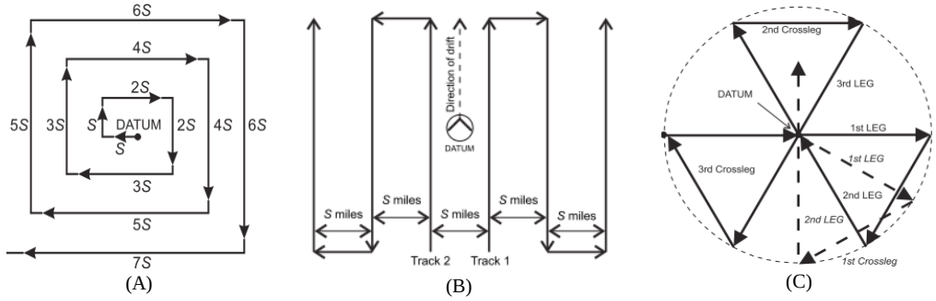


FIGURE 4-11: TYPICAL SEARCH-AND-RESCUE PATTERNS SHOWING EXPANDING SQUARE (A), PARALLEL TRACK SEARCH (B), AND SINGLE-UNIT SECTOR PATTERN (C). REPRODUCED FROM IAMSAR (2007).

The UK search operators RNLI and MCA use the expanding square pattern most frequently for near-shore incidents[5]. Therefore, it is implemented into the OSCAR search add-in. The

[5] Pattern usage is not recorded by the RNLI or MCA but interviews with RNLI Calshot lifeboat station staff and MCA Lee-on-Solent helicopter pilots affirmed this claim.

IAMSAR manual (IAMSAR 2007) provides specific values for the track spacing value *S* depending on search vessel and incident type. However, the OSCAR add-in assigns *S* such that the camera footprints overlap each other slightly (compare Figure 4-8).

A typical search-and-rescue operation is conducted as follows: When a search-and-rescue incident occurs (person falling off a ship, ship lost, *etc.*), it is reported to some search-and-rescue authority agency that delegates search-and-rescue vessels to the nearest known incident position, the "initial search position". Here, vessels initiate the most appropriate search pattern until the incident is found or abandoned. Upon detection, vessels move to the incident and rescue it. Here, rescuing refers to different actions such as dragging a person out of the ocean but also giving technical assistance to a drifting ship. Eventually, the search-and-rescue vessels return to their home base.

4.11.2 Search

The OSCAR simulation triggers a search-and-rescue operation as soon as a VESSEL starts a TRACK with the first SEGMENT featuring `uponArrival=search`. The VESSEL does not dash out to the initial SEGMENT coordinates but creates a specific "initial search position" near the incident (Figure 4-12).

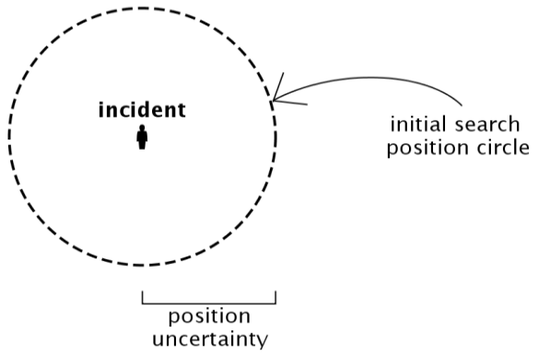


FIGURE 4-12: SEARCH-AND-RESCUE INCIDENT INITIAL SEARCH POSITION.

This "initial search position" appears on an imaginary circle with its radius defined by the incident SEGMENT `type` parameter (Appendix 1). The exact position on the circle is defined through a random function, *i.e.* it is different for every search. Upon arrival at the initial search position, the VESSEL initiates an expanding square pattern as in the schematic in Figure 4-13. This pattern neglects any incident drifting.

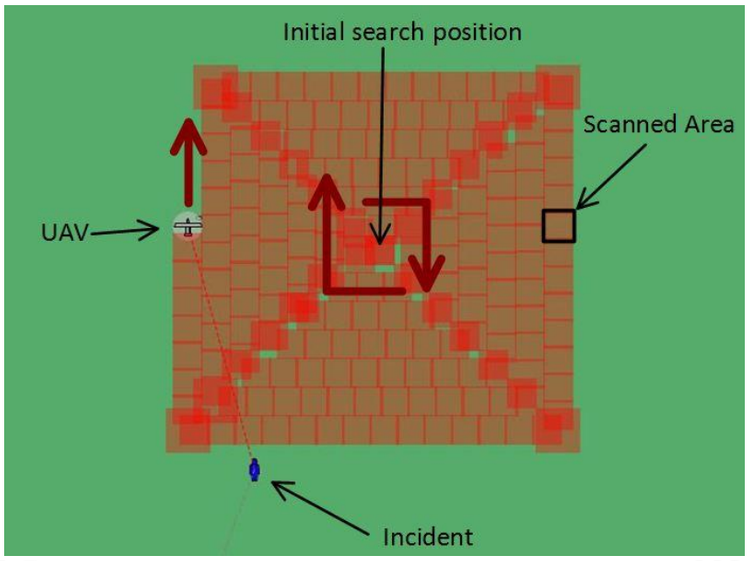


FIGURE 4-13: EXPANDING SQUARE SEARCH-AND-RESCUE PATTERN WITHIN OSCAR SIMULATION. VIEW AREAS ARE RECTANGULAR BECAUSE THE CAMERA POINTS VERTICALLY DOWNWARDS.

The VESSEL starts to fly a schematic expanding square pattern where the size of the track space *S* depends on the camera footprint size. Footprints overlay by about 5%. Additional overlay naturally occurs at the square corners. Note that some minor areas around the corners are not covered. This area can be reduced by increasing the footprint overlay. However, in reality the camera system would also miss minor areas during direction changes (if it has a fixed camera and footprint overlay is low). The missed area can also be minimized by changing the `cameraTiltAngle` parameter. Note that this pattern is an ideal case that might be amended in adverse conditions in reality.

4.11.3 State Chart

Figure 4-5 presented the generic VESSEL state chart defining VESSEL operations. However, it omitted the search-and-rescue add-on states required to simulate realistic search-and-rescue operations. The search-and-rescue operation states are shown in a more detailed state chart cutout in Figure 4-14. The first deviation from the generic state chart occurs upon dash out: the VESSEL moves to the initial search position, which is *near* the GISPOSITIONFULL coordinate defining the search-and-rescue incident. Upon arrival, it initiates an expanding square pattern as described above. It alternates between moving west, north, east and south in ever-increasing squares as in Figure 4-13.

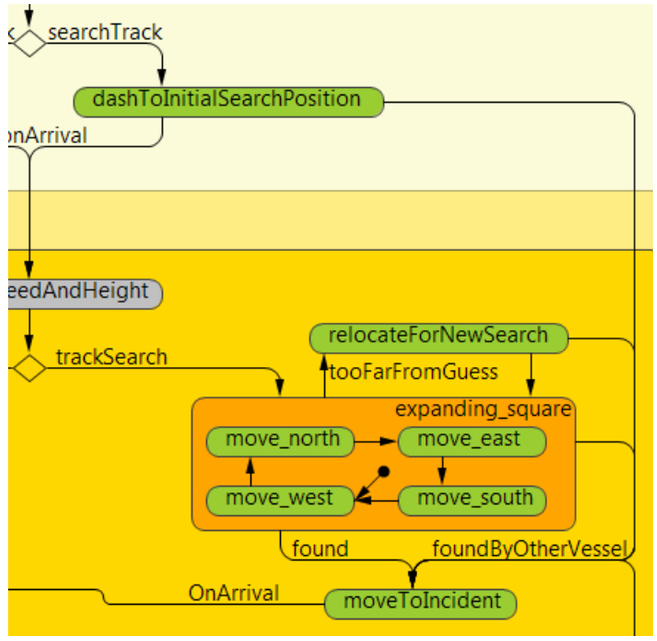


FIGURE 4-14: VESSEL STATE CHART SEARCH-AND-RESCUE DETAILS (CUTOUT FROM FIGURE 4-5).

Searching stops if any of the following four events happen:

- The VESSEL runs out of energy. If the incident is not found by other VESSELS during refuel, the VESSEL will re-commence search where it left off. Alternatively, it will move to the discovered incident.
- The VESSEL discovers the incident. For this, the camera footprint must include the incident (Appendix 5) *and* the detection probability must return true (Section 4.10.2). The `expanding_square` composite state is left through the transition "`found`". The VESSEL moves to the incident position for rescue and notifies other VESSELS also searching for the current incident.
- Another VESSEL discovers the incident. In this case, the current VESSEL receives a detection message from the detecting VESSEL. The `expanding_square` composite state is left through the transition "`foundByOtherVessel`" and the VESSEL moves to the incident for rescue.
- The `expanding_square` pattern grows so large that the VESSEL distance from the initial search position is larger than five times[6] the incident position uncertainty specified in the SEGMENT'S `type` column. The `expanding_square` composite state is left through the transition "`tooFarFromGuess`". The VESSEL randomly draws a new initial search position on the imaginary initial search position circle (Figure 4-12), moves towards it and initialises a new expanding square search.

[6] This is an estimate based on interviews with RNLI staff at Calshot lifeboat station.

Users can implement different search-and-rescue patterns by adding new states next to the `expanding_square` composite state.

4.11.4 Incidents

A SEGMENT becomes a search-and-rescue incident if it has `uponArrival=search`. In that case, a new agent of class INCIDENT is created. This agent class is only used for search-and-rescue incidents and is not generic. An INCIDENT represents a human floating in water (helped by a life vest) and waiting for help. The INCIDENT is positioned at the SEGMENT coordinates. Once created, it follows the INCIDENT state chart in Figure 4-15.

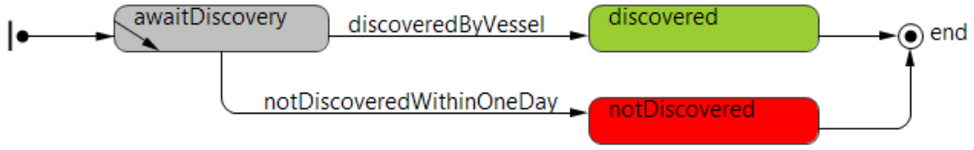


FIGURE 4-15: INCIDENT CLASS STATE CHART.

Initially, the Incident awaits its detection by a Vessel. During that time, the probability of survival decreases. The initial survival probability is 0.85, assuming accidental immersion in cold water at a temperature of 14ºC (Wissler 2003). Subsequently, the probability of survival decreases with time as in Figure 4-16.

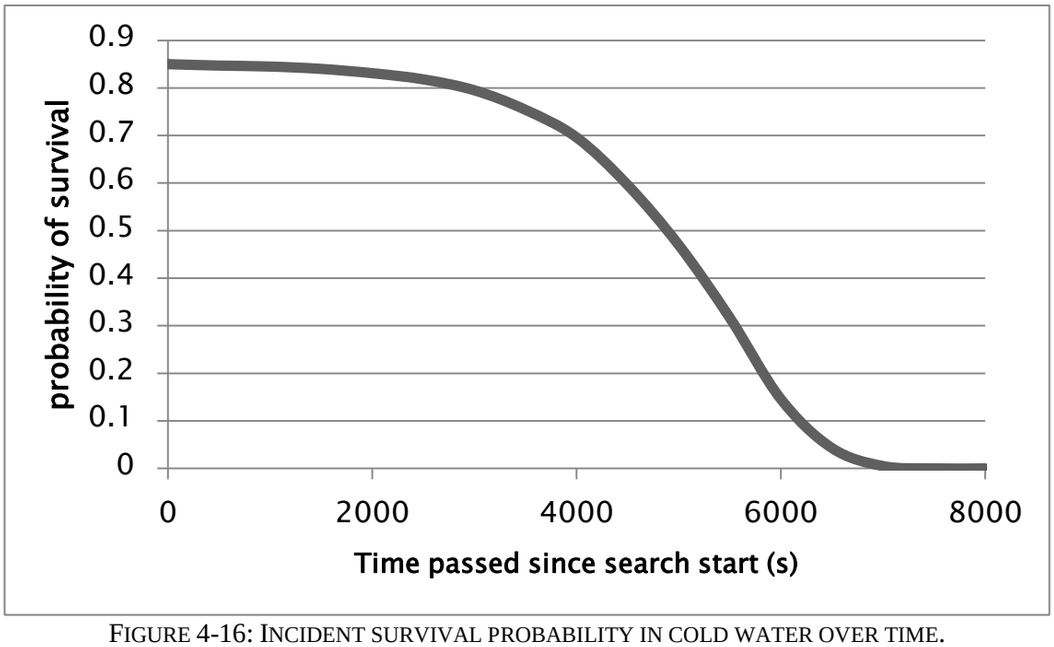


FIGURE 4-16: INCIDENT SURVIVAL PROBABILITY IN COLD WATER OVER TIME.

The survival probability is averaged over data from several experimental studies (Brooks 2001; Glickman-Weiss et al. 1997; Golden & Tipton 1987; Keatinge 1961; Tipton et al. 1999). There is a time window of less than two hours for VESSELS to find INCIDENTS alive. This is in

line with the more complex Cold Exposure Survival Model developed by the Canadian Defence Research authority (Tikuisis & Keefe 2005).

Once a VESSEL detects an INCIDENT at its position, the current INCIDENT survival probability is stored in the SEGMENT output `segmentMeasure` (value can be zero, *i.e.* the incident is dead). Note that the OSCAR simulation is only concerned with survival probabilities instead of Boolean "alive/dead" values. Although this approach is less realistic from a simulation perspective, it reduces the number of required simulation replications because less noise is generated.

If no VESSEL finds the Incident within 24 hours, the simulation assumes that all hope of rescuing the INCIDENT alive has passed. All searching VESSELS stop and return to their BASE. The INCIDENT agent is destroyed. The output `timeOfSpotting` is set to `null`.

4.12 Component class

If the database specifies components linked to a VESSEL agent of given `category`, `type` and `fuelType`, the VESSEL creates sub-agents of the OSCAR simulation class COMPONENTS. The number of COMPONENT sub-agents depends on the number of entries in the respective component database table. Appendix 4 describes the seven parameters defining COMPONENTS. Each COMPONENT agent follows a generic state chart displayed in Figure 4-17.

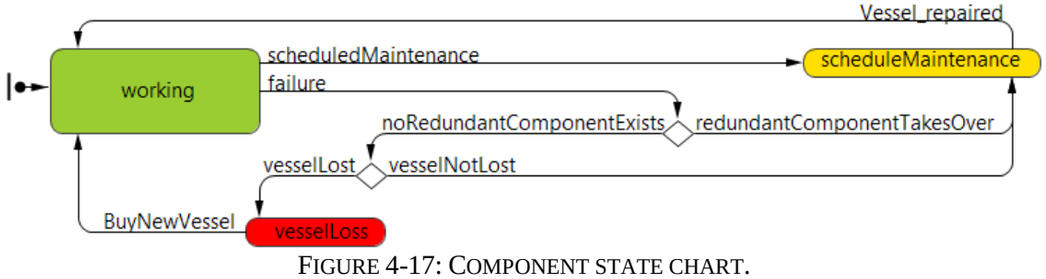


FIGURE 4-17: COMPONENT STATE CHART.

Upon initialisation, any COMPONENT is functional. While the parent VESSEL operates, its COMPONENTS accumulate operating time (if `WeibullLifeMeasure=duration`) or operating cycles (if `WeibullLifeMeasure=cycles`).

By default, each COMPONENT schedules maintenance every `WeibullBeta` of operating time (if `WeibullLifeMeasure=duration`) or of operating cycles (if `WeibullLifeMeasure=cycles`). However, users could add a new database column "scheduledMaintenance" and enter a simple maintenance interval manually. If the COMPONENT schedules maintenance, its parent VESSEL will conduct maintenance upon finishing its current TRACK.

However, the COMPONENT may experience failure before scheduled maintenance. Upon COMPONENT initialisation, the "time to failure" (or "cycles to failure" if `WeibullLifeMeasure=cycles`) is sampled from the COMPONENT weibull distribution (defined by `weibul-`

`lEta` and `weibullBeta`) and multiplied with `1+robustnessScalingFactor`. A higher `robustnessScalingFactor` increases the COMPONENT robustness because time to failure increases. When the COMPONENT operating time (or operating cycles) becomes larger than the time to failure (or cycles to failure), the COMPONENT fails. If the VESSEL contains redundant COMPONENTS (Section 3.3.5.1), they take over the workload. The VESSEL will conduct maintenance upon finishing its current TRACK. The redundant Components reduce their "time to failure" (or "cycles to failure") by

$$t_i = t_i - \left(\frac{1}{N} \times \beta \times 0.1\right) \qquad \text{Eq. 4-9}$$

Where N is the number of redundant COMPONENTS, β is the OSCAR parameter `weibullBeta` and t_i is the i^{th} redundant COMPONENT time to failure (or cycles to failure). If there are few redundant COMPONENTS they are stressed proportionally more, thereby increasing their own reduction in fault free operation time. If more redundant COMPONENTS exist, they are assumed to share the load of the failed COMPONENT. Arbitrarily, the maximum reduction in t_i is limited to 10% of `weibullBeta`.

If the failed COMPONENT has no redundant COMPONENTS (`quantityOnboard=0`), the aircraft is lost with probability `LossProbabilityFromInflightFailure`. If the random sample returns no VESSEL loss, the COMPONENT schedules maintenance and the VESSEL returns to its BASE immediately for repair. If the random sample returns VESSEL loss, the VESSEL is lost immediately. However, for model simplicity, a new VESSEL agent immediately replaces the lost VESSEL and continues the current TRACK. This assumption neglects operational performance drop due to cancelled TRACKS.

4.13 Experimentation

After describing the OSCAR simulation components in detail, this section presents the simulation element that actually runs the simulation: the specific simulation experiments. Each OSCAR experiment is defined by two experiment parameters explained in Section 4.13.1. Section 4.13.2 details the randomisation for the OSCAR simulation. Appendix 10 describes each of the three OSCAR simulation experiments in more detail, namely "Single Run", "Interactive single run" and "Run fast".

4.13.1 Parameters

Each simulation experiment is defined by three parameters as below:

`RandomSeed`: *An integer value indicating the random seed to be used for this experiment.*

`missions`: *A Java array of strings. Each string specifies a* MISSION *table to be loaded. The parameter can include one or many* MISSION *strings. Each string must equal the respective* MISSION *table name in the database.*

4.13.2 Randomisation

One of the core characteristics of good simulation modelling is to use different random number seeds for *each* model part applying random sampling (Robinson 2004). The OSCAR simulation features five random numbers:

- COMPONENT units to failure: Upon creation of a new COMPONENT agent, its time-to-failure (if `WeibullLifeMeasure=duration`) or cycles-to-failure (`weibullLifeMeasure=cycles`) is drawn randomly from a weibull distribution (Section 4.12).
- VESSEL loss during operation: Whenever a COMPONENT with no redundant COMPONENTS fails during operation, it causes complete VESSEL loss with probability `LossProbabilityFromFailure` (Section 4.12).
- Target detection: If a VESSEL uses its active payload (Section Payload module4.10) to look for a target, it can spot the target only if it is within the camera footprint (Appendix 5). In that case, the VESSEL spots the target with a probability depending on target size, camera quality and distance to the target (Section 4.10.2). A random sample is generated from this probability.
- Initial incident search position: For the search-and-rescue add-on (Section 4.11), VESSELS start a search at a random point on an imaginary circle around the actual INCIDENT position. To define the random point, a random uniform sample is drawn between 0 and $2 \times \pi$. The initial search position is placed on the imaginary circle using the random value as an angle starting from the 12 o'clock position.
- VESSEL arrival: Upon arriving at the current TRACK `destination`, the VESSEL might be lost if it applies the landing loss plug-in (see Section 5.4.4).

4.14 Embedding

In order to apply the OSCAR framework or simulation effectively, it needs to be embedded into a design workflow. This section describes how the OSCAR simulation can be implemented into a larger conceptual design phase tool.

The easiest way to embed the OSCAR simulation is to create a Java applet from the AnyLogic IDE. This way, users do not need an AnyLogic license (subject to the AnyLogic version licensing statement). Java applets run on any machine, independent of the operating system. A vessel design software creates VESSEL parameters upstream of running the OSCAR simulation. It updates the database VESSEL (and COMPONENTS) tables accordingly. The conceptual design phase tool automatically starts the OSCAR simulation Java applet through a batch file. Upon simulation close, the OSCAR outputs are written to the database. The Java applet closes automatically (if using the "RunFast" experiment). The conceptual design phase tool can now start a downstream tool to post-process the outputs. Section 5.2.2 describes a similar embedding approach practically applied for this thesis.

5. CASE STUDY – DECISION SUPPORT

This chapter applies the OSCAR simulation in a value-driven conceptual design phase setting. It demonstrates its suitability for design decision support. The case study is based upon the DECODE (Decision Environment for COmplex Design Evaluation) research project (Section 5.2) that designed, built and flew four UAS using a value-driven design approach. This case study investigates which UAS design is most suited for a particular operational scenario, namely search-and-rescue support around southern UK waters (Section 5.3). For this, the OSCAR simulation is setup to model the scenario as closely as possible (Section 5.4). Results are analysed in Section 5.5, disclosing two of the four designs to be well suited for the given tasks. Section 5.6 discusses the results in the light of surplus value, added costs, qualitative OSCAR outcomes and the applicability of the work within a real conceptual design phase environment. Last, Section 5.7 summarises findings from this chapter.

5.1 Background

This section outlines the preconceived case study scenario presenting the OSCAR simulation in a value-driven conceptual design phase setting. The recent rise of military UAS applications

has led to considerations about civil applications of these devices (Cox et al. 2004; Dalamagkidis et al. 2011; Herrick 2000). One of the most promising applications with regards to medium-term feasibility, certification and practical value is that of search-and-rescue support (Austin 2010; Keeter 2008). In 2008, the USCG (United States Coastguard) started investigating the use of UAS for maritime search-and-rescue support, using a MQ-9 Guardian Predator UAS in Florida (Egan 2011). In 2013, USCG operated a ScanEagle UAS for 90 hours, aiding the interdiction of nearly 600kg of cocaine[1]. Also in 2013, Iran developed a UAS designed to help people drowning in near-shore areas[2]. In the UK, research into UAS for search-and-rescue operations does not exist.

In this case study, a leading UK search-and-rescue operator, the RNLI, seeks to examine the usefulness of implementing UAS into their ground-based fleet of lifeboats. The case study assumes that RNLI managers are not yet confident that UAS acquisition is useful to their service. Therefore, they instruct a UAS manufacturer to suggest various initial design ideas and report upon their value within the RNLI operational environment. The UAS manufacturer develops four design candidates. Candidates differ strongly in order to find the best design idea for search-and-rescue support.

The OSCAR simulation is used by the manufacturer during this conceptual design phase to find out if:

- UAS is adding any value to RNLI operations at all?
- If so, should the RNLI purchase UAS?
- If so, which of the design candidates is most promising for more detailed development?

5.2 DECODE project

The preconceived scenario described above bases upon real work with the RNLI as part of the DECODE research project concluded in 2012 at the University of Southampton. DECODE stands for "*Decision Environment for COmplex Design Environments*" and this section describes the DECODE research goals, the unique design software suite developed and the four UAS designs built using the DECODE software suite.

[1] See http://www.uscg.mil/acquisition/uas/default.asp, accessed 16/07/2013.
[2] See http://www.wired.co.uk/news/archive/2013-03/27/iranian-rescue-robot, accessed 16/07/2013.

5.2.1 Research goals

The DECODE research was motivated by the insight that today's aeronautical companies are not primarily engineering constructors anymore but have become information-processing structures. In order to deliver their products on time, in budget and according to specification, these companies must adapt their processes, tools and knowledge management. Many companies still use relatively simple, ad-hoc design tools, especially within the conceptual design phase (Nunez & Guenov 2013).

DECODE aimed to alleviate these shortcomings by investigating integrated design tools allowing holistic optimisation at the system level through value-driven design approaches. Moreover, tools should enable active design exploration of system level trade-offs between performance, unit cost and life cycle costs. A demonstrator toolkit was developed, linking conceptual design aircraft spread sheets with CFD and CAD (Computer Aided Design) analysis of varying fidelities (for conceptual and more detailed analysis). Uniquely, design should incorporate the impact of operations of the products so the OSCAR simulation was incorporated into the toolkit. Moreover, a value model quantified cost and benefits of manufacturing and operations to allow a value-driven design approach. Section 5.2.2 describes the DECODE design suite in more detail.

During the three-year project (2009-2012), three different aircraft for different applications were designed. A fourth aircraft was designed after the projected ended. All aircraft were UAS because this allowed not only using the toolkit theoretically but also actually building and flying the designs to improve the toolkit over time. The four designs are described in Section 5.2.3.

5.2.2 DECODE design suite

DECODE aims to enable computational tools to have a *direct* influence on design decisions, reducing (but by no means eliminating) human intervention with the goal to reduce design time (Keane & Nair 2005). Avoiding the traditional design iterative spiral, DECODE follows a more agile design approach: Essentially, preliminary design is integrated into the conceptual design phase as all design variables are considered conceptually (Gorissen, Quaranta, Ferraro, Schumann, Schaik, Bolinches I Gisbert, et al. 2014). Similarly, manufacturing considerations are closely integrated into the DECODE design process as are maintenance and deterioration (through OSCAR).The DECODE suite ties together different functional modules of existing and new software through a data management layer, a logic layer and a presentation layer (ibid). The core module is a spreadsheet concept design tool developed from basic principles. It forwards basic geometry into a CAD model to develop detailed geometry. Moreover, it informs a CFD tool to perform aeroelastic and fluid flow analysis. FEA (Finite Element Analysis) is used for structural analysis. These steps can be skipped for low-fidelity runs using empirical formu-

lae. The design spreadsheet also provides inputs to the OSCAR simulation using a simple text file (Figure 5-1). The DECODE suite starts the OSCAR simulation as a Java applet and sets an additional experiment parameter `externalTextfile=true`.

```
opsim.a_wing=1.4
opsim.cl_max=1.57356
opsim.d_prop=0.4572
opsim.k1_a=0.043676
opsim.k1_b=0.0
opsim.k1_c=0.0
opsim.k2_a=0.0
opsim.k2_b=0.0
opsim.k2_c=0.0
opsim.k3_a=0.04207
opsim.k3_b=0.0
opsim.k3_c=0.0
opsim.p_inst=3400.0
opsim.rpm=7000.0
opsim.sfc_a=1339.784
opsim.sfc_b=-970.149
opsim.sfc_c=297.7281
```

FIGURE 5-1: EXTERNAL TEXT FILE SPECIFYING VESSEL PARAMETERS.

This enables the Java applet to load and apply the text file parameters for UAS type VESSELS. After the experiment ends, OSCAR writes outputs to the database but also to a specific output text file. The output text file contains processed outputs (*i.e.* the OSCAR simulation processed the raw data already) such as averages and standard deviations of specific measures. The DECODE suite uses these measures for the downstream value model that is similar but more detailed than the one applied in this case study (Section 5.4.3).

5.2.3 UAS design

The DECODE research produced three UAS designs. A fourth design was created after the project ended using the same methodology and toolkit. Below is a brief description of each design, focussing on the characteristics important for OSCAR. The rest of this study refers to each design by the following acronyms: DECODE, BBC, SULSA and 3i. Appendix 12 compares all four designs by their performance, camera and COMPONENT inputs used for the OSCAR simulation.

5.2.3.1 Design 1 – DECODE

The first aircraft designed using the DECODE suite was a twin-boom pusher configuration UAS with a non-retractable undercarriage. It was designed focussing on modularity and easy handling, allowing operators to carry a disassembled UAS in a car, assemble it in a few minutes and get it into the air (Figure 5-2).

FIGURE 5-2: DECODE UAS SHOWING MODULAR APPROACH. REPRODUCED WITH PERMISSION FROM ANDY J. KEANE[3].

This aircraft was designed with search-and-rescue operational capability in mind to allow for long endurance. However, the early development stage of the DECODE software suite (including OSCAR) at the time prevented targeted search-and-rescue development. Instead, this design was a proof-of-concept aircraft demonstrating aircraft development using agile and integrated design tools (Gorissen, Quaranta, Ferraro, Schumann, Schaik, Bolinches I Gisbert, et al. 2014). The UAS has an empty weight of 8.8 kg with a 1.8 kg fuel capacity. It is made of carbon spars and has a twin-cylinder four-stroke engine.

5.2.3.2 Design 2 – BBC

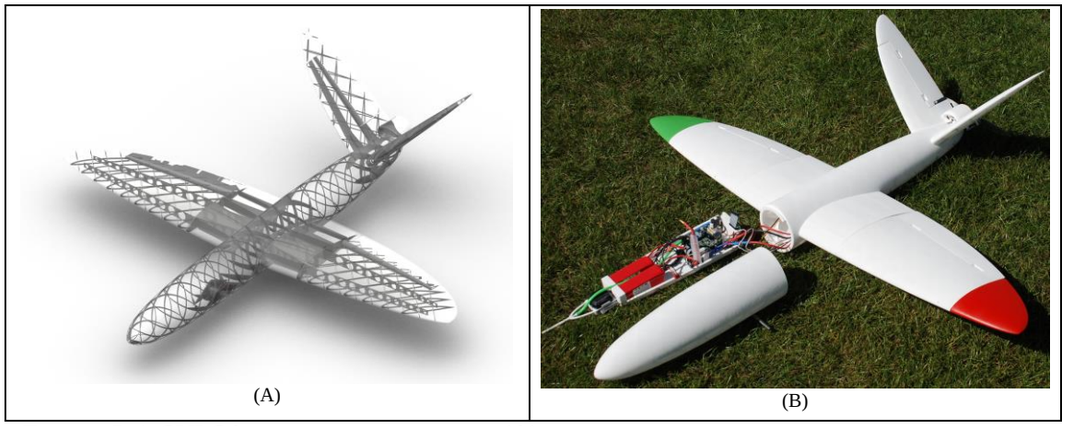
FIGURE 5-3: BBC UAS VIRTUAL DESIGN (A) AND ASSEMBLED AIRCRAFT INFLIGHT (B). REPRODUCED WITH PERMISSION FROM JEROEN VAN SCHAIK[4] AND ANDY J. KEANE[5].

This UAS was developed based on collaboration between the University of Southampton and the BBC. It investigated the use of UAS for communication and live video broadcasting of ma-

[3] E-Mail: andy.keane@soton.ac.uk
[4] E-Mail: jeroenrob@gmail.com
[5] E-Mail: andy.keane@soton.ac.uk

jor events. The UAS is based upon the DECODE design but has larger dimensions at 4 m wing span, 2.5 m length and almost 23 kg take-off weight. A larger twin-cylinder four-stroke engine allows much higher flight speeds compared to DECODE. Moreover, the fuel tank is larger, increasing flight endurance. The final design is a monoplane with a pusher 3-blade propeller and a twin-boom "inverted V" tail as in Figure 5-3. The internal structure was designed for rapid prototyping and aerodynamic surfaces use ultra-light foam.

5.2.3.3 Design 3 – SULSA



(A) (B)

FIGURE 5-4: SULSA INTERNAL STRUCTURE (A) AND THE DISASSEMBLED AIRCRAFT SHOWING INSTRUMENTS TRAY (B). REPRODUCED WITH PERMISSION FROM JEROEN VAN SCHAIK[6] AND ANDY J. KEANE[7].

SULSA (*Southampton University Laser-Sintered Aircraft*) is the smallest design and was created to explore the DECODE rapid prototyping capabilities (Figure 5-4). It was designed, built and tested in less than two weeks. Moreover, it was the "world's first *printed* aircraft"[8] as its entire structure, including wings, control surfaces and hatches were printed layer by layer on a nylon laser sintering machine. SULSA's dry weight is 0.208 kg only with a wing area of 0.24 m^2.As SULSA is the only design in this case study that is propelled by an electric engine, the custom performance model (Section 4.9.2) is not applicable. Instead, the generic performance model (Section 4.9.1) links flight speed to power consumption as in Figure 5-5. Appendix 11 details the rationale.

[6] E-Mail: jeroenrob@gmail.com
[7] E-Mail: andy.keane@soton.ac.uk
[8] See http://www.newscientist.com/article/dn20737-3d-printing-the-worlds-first-printed-plane.html#.UfU-D43vh8E, accessed 29/07/2013.

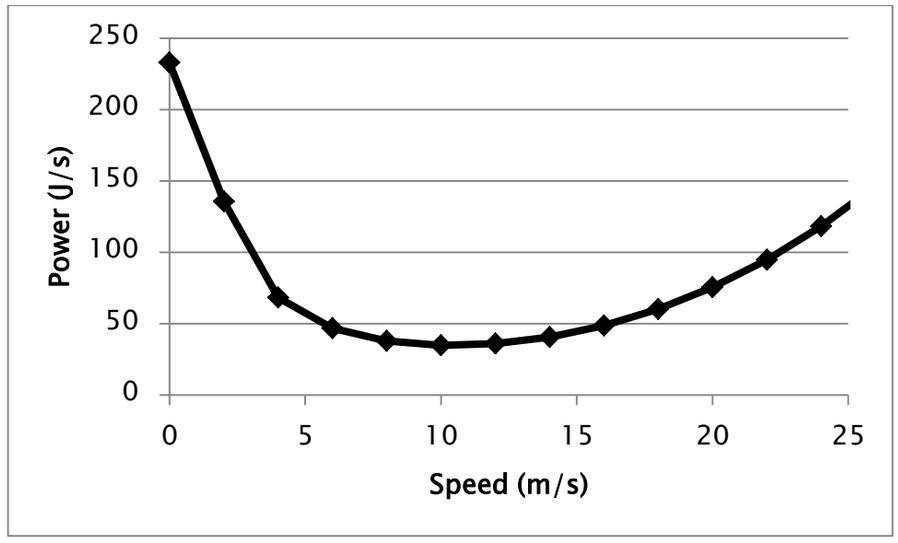


FIGURE 5-5: SULSA POWER-TO-SPEED RELATIONSHIP.

5.2.3.4 Design 4 - 3i

FIGURE 5-6: 3I AIRPLANE OVERVIEW. REPRODUCED WITH PERMISSION FROM JEROEN VAN SCHAIK[9].

This design is the largest and heaviest of all four UAS. It was developed as part of the follow-on research project after DECODE, namely the 3i project ("Integrated Coastal Zone Management via Increased Situational Awareness through Innovations on Unmanned Aircraft Systems" described in Section 6.2). Design focussed upon flight safety and component redundancy as the product is intended to be used in real operations after the project end. A FMEA (Failure Modes and Effects Analysis) indicated that using two engines and a duplex redundant flight control system was required to achieve high levels of flight reliability. About 70% of the 3i UAS are laser-sintered, reducing production errors and performance loss. The design configuration is a

[9] E-Mail: jeroenrob@gmail.com

twin-engine, twin-boom monoplane, featuring a versatile payload pod near the CoG (Centre of Gravity) with an unobstructed forward view (Figure 5-6). The aircraft is the heaviest design at 24.2 kg but also the fastest, capable of flying at 45 m/s. If one engine fails, it can safely return to base using the remaining engine.

5.3 Scenario

This section describes the situation of search-and-rescue operations as it is found today around the Solent region at the south coast of the UK. Moreover, it discusses how adding one or more UAS would change real search-and-rescue execution for stakeholders. The remainder of this chapter uses the keyword “baseline” to refer to the former case (Section 5.3.1) while “revised” alludes to the latter (Section 5.3.2).

5.3.1 Baseline scenario

FIGURE 5-7: SOUTH COAST OF THE UK INDICATING RNLI LIFEBOAT STATIONS IN THAT REGION.

UK search-and-rescue operations are conducted by several authorities that follow specific guidelines (Section 4.11.1). One of them is the RNLI employing maritime vessels exclusively. The area chosen for this case study covers part of the south coast of the UK, ranging from Lyme Regis in the West to Portsmouth in the East as seen in Figure 5-7.

The area includes the Solent, the water strait between the Isle of Wight and the City of Southampton. The Solent is one of the busiest shipping routes of the UK and features a very high level of leisure activities, ranging from sailing and diving to water-skiing and surfing. Ten RNLI lifeboat stations conduct search-and-rescue in the chosen area. Each harbours between one and two lifeboats of varying types. Lifeboat services include missions due to vessel machinery failures, troubled vessels, persons in distress/missing, capsizes, fires or collisions. About one third of all missions can be classified as *searches* because the incident position is unknown (RNLI 2009). In this case study, only missions including unknown incident positions are included as these merit additional search tools.

Generally, search missions consist of four stages: initially, the responsible lifeboat station will dispatch the appropriate lifeboat towards the suspected incident position. Next, a coordinated search begins, using fixed search patterns until the incident is found or searchers give up. One or more lifeboats can search for an incident, depending on weather, incident type, available equipment and location. Upon spotting the incident, rescuing is performed based on the incident's requirements. Lastly, lifeboats return to their base.

5.3.2 Revised scenario

Current search-and-rescue operations combine very expensive equipment (lifeboats, helicopters, aircraft) with limited human performance as pilots and lookouts can conduct searches for no more than several hours on end. Moreover, equipment and operators can be severely restricted by weather and night time operations. This leads to very high fixed costs with acceptable (yet improvable) search performance. The use of UAS lends itself ideally to support search-and-rescue operations because UAS are not confined to the same limits. Using autonomous technologies, UAS can operate much longer than humans can. Given the right payload, they can autonomously scan and analyse large areas for incidents, even in bad weather or during night.

Once the technology has matured, UAS could be easily implemented into current search-and-rescue operations, weather permitting. They could be launched from lifeboat station managers after lifeboats dispatched. Most stations have a strip of grass long enough to house medium sized UAS as those presented in this case study. Alternatively, UAS could be launched by lifeboats themselves while searching, broadcasting a live view from the air into the lifeboat or to its base. Combining live search image analysis and human search potentially increases the chance to find incidents earlier at little additional cost.

5.4 Simulation setup

The simulation runtime of all scenarios is one virtual year. This is less than the entire life cycle of a search-and-rescue UAS but suffices to for the purpose of this case study, *i.e.* to compare and decide upon the best UAS design. The input data described in this section is applicable over the course of one year only. Therefore, longer runtimes would have extrapolated that data without new model insights.

This section presents how current search-and-rescue operations and the anticipated UAS integration were translated into the OSCAR simulation. Section 5.4.1 describes the baseline scenario as found at the south coast of the UK today. Section 5.4.2 presents the setup for the revised scenario that includes a UAS vessel. Section 5.4.3 details the value model developed to compare both scenarios quantitatively. Last, section 5.4.4 introduces an additional OSCAR simulation add-in used for modelling the landing of UAS in more detail. Appendix 16 details the rationale behind setting the number of simulation replications for this case study.

5.4.1 Baseline scenario

The baseline scenario is used to judge the performance of each UAS design against a datum. It represents the current status of search-and-rescue operations around the southern coast of UK waters as described in chapter 5.3.1 within the OSCAR simulation.

5.4.1.1 Lifeboat stations

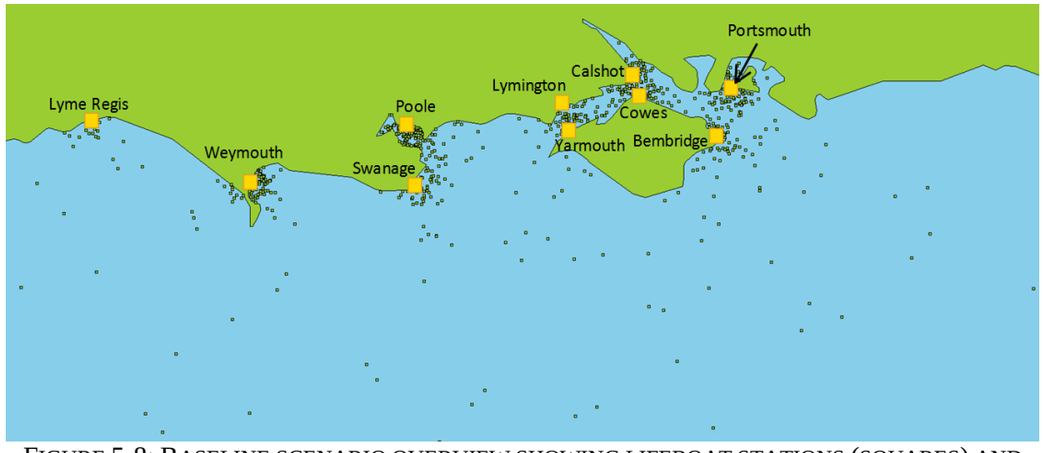


FIGURE 5-8: BASELINE SCENARIO OVERVIEW SHOWING LIFEBOAT STATIONS (SQUARES) AND INCIDENT POSITIONS (DOTS).

This case study models 10 RNLI lifeboat stations along the south coast of the UK as shown in Figure 5-8. Each station is an agent incident of the BASE class (Section 4.7) with the respective coordinates.

The geographical region was chosen such that model validation could be supported by local RNLI personnel. The geographical extent of this case study was chosen based on initial range and endurance estimates for the UAS such that they could support all incidents from an edge location. Each lifeboat station is assigned incidents and lifeboat vessels as described below.

5.4.1.2 Incidents

Figure 5-8 shows the geographic distribution of incidents. Each incident is an agent of the non-generic INCIDENT class (Section 4.11.4). The majority of INCIDENTS occur very close to the shore and in proximity to lifeboat stations, most acutely within the Solent region between the Isle of Wight and mainland UK. Here, leisure activities such as sailing, surfing and swimming are predominant. Further out at sea, incidents involve fishing vessel capsize or staff washed overboard (RNLI 2009; RNLI 2008). Each station features an average number of incidents occurring at that station per year as in Figure 5-9. This neglects seasonal differences.

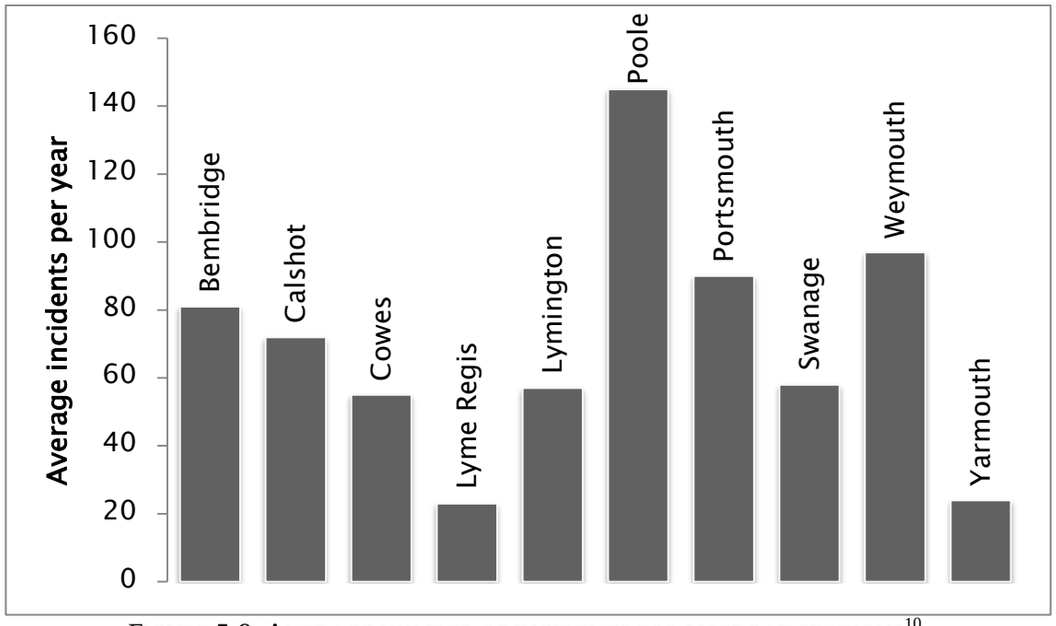


FIGURE 5-9: AVERAGE NUMBER OF INCIDENTS FOR LIFEBOAT STATIONS[10].

Based on lack of detailed data, each incident has a uniformly distributed random position uncertainty between 50 and 3,000 metres. Larger uncertainty alludes to persons immersed in water with a small target size of 0.2 by 0.2 metres (essentially a head in water), a loose `DetectionCriteria` requiring identification only, a longer UAS `loiter` time upon detection and a higher UAS flight altitude. Conversely, smaller position uncertainty defined incidents to be small boats of target size 3 by 10 metres with a tighter `DetectionCriteria` requiring recognition

[10] Based on publicly available data from www.rnli.org.uk

or even detection, less UAS `loiter` time upon detection and lower UAS flight altitude. This setup recreates reality in the sense that the position of a person in water is known less well than that of a small boat (which might even have a transponder radioing its exact position). Small boat incidents represent people that have gone overboard but are still near the boat (hence more easily found by rescuers or UAS).

However, all incidents are essentially persons immersed in ocean water. Submersion time is by far the most influential factor for survival in water (Suominen et al. 2002). A simple probability of incident survival links to water immersion time as in Figure 5-10. It assumes that ocean water temperature around the Solent remains constant at 14 °C throughout the year[11], people wear clothes and a life-jacket and are of average shape and fitness (Wissler 2003).

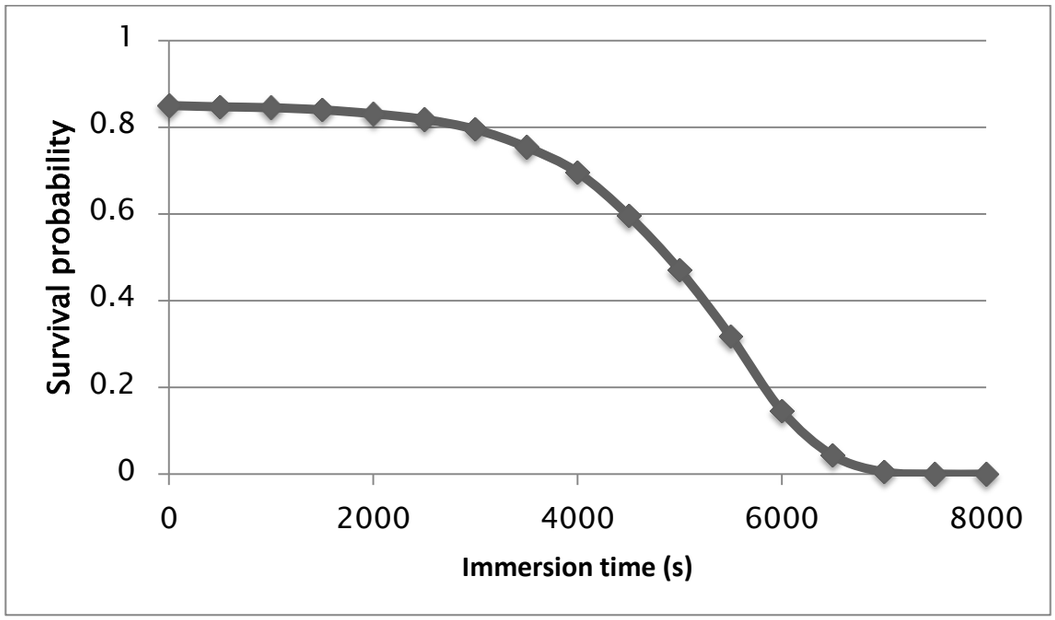


FIGURE 5-10: INCIDENT SURVIVAL PROBABILITY BASED ON WATER IMMERSION TIME. ADAPTED FROM (Wissler 2003).

5.4.1.3 Vessels

The baseline scenario features VESSEL agents representing lifeboats by setting `category=boat` and `type=boat` (Figure 3-12). In reality, the Solent stations feature different types of lifeboats and varying number of boats per station. For this case study, however, one VESSEL type applies for each station because engine performance data was available for one lifeboat type only, namely the Mersey. The Mersey lifeboat is an all-weather boat with a maximum speed of 17 knots[12]. It is powered by two Caterpillar 3208T marine diesel engines consuming

[11] This is a reasonable assumption based on the World Ocean Atlas 1994: http://iridl.ldeo.columbia.edu/SOURCES/.LEVITUS94/, accessed on 11/07/2013.

[12] See http://rnli.org/aboutus/lifeboatsandstations/lifeboats/Pages/Mersey.aspx, accessed on 10/07/2013.

between 10 and 90 kg/hr of maritime diesel, depending on cruise speed. Appendix 13 details the power consumption for varying cruise speeds. The Mersey has a fuel capacity of 821.4 kg and a search cruise speed of 7 m/s. Each lifeboat assumes one lookout searching for the incident. Due to lack of a human visual model, the camera model (Section 4.10) was adapted to emulate human vision. The horizontal and vertical field-of-view were set to 40° and 15° respectively, mirroring the human cone of visual attention. The horizontal and vertical pixels were set to 1,000 each, reflecting the total number of signal fibres going from the eye to the brain (Defense 1999). The camera tilt angle was set to 0° as the lookout will scan the horizon. Moreover, setting `camera_recognition_factor=3` increases the chance of spotting an incident threefold. This factor was chosen to match the baseline output `average_incident_waiting_time` with the observed RNLI waiting times[13].

5.4.2 Revised UAS scenario

This section describes the "revised" scenario including UAS setup within OSCAR. The lifeboat stations, VESSELS and INCIDENTS setup is identical to the "baseline" setup. The only difference in the "revised" scenario is the addition of a UAS agent. This agent is a VESSEL where `category=fixedWing` and `type=aircraft` (Appendix 3). For DECODE, BBC and 3i, `performanceModel=fixedWing_aircraft_petrol` and `fuelType=petrol` while SULSA uses `performanceModel=powerAgainstSpeed` and `fuelType=electric`. Further parameters base on the UAS in question as outlined in Section 5.2.3 and Appendix 12. The UAS agent operates from the BASE at Lyme Regis, the westernmost BASE in this case study. Thereby, the UAS is forced to cover long distances to most of the incidents, stretching its operational capabilities as much as possible. This helps design comparison as performance between two similar designs can be difficult if designs are not strained to their limits.

5.4.3 Value model

A simple value model demonstrates useful post-processing of OSCAR outputs for value-driven design. Real applications would feature more elaborate value or cost models, including currency depreciation but this is beyond the scope of this work. In general, the value of a product can be described as the revenue it generates minus the costs it accumulates. However, search-and-rescue operations do not create any incoming revenue so an alternative "benefits"

[13] The RNLI aims to "reach at least 90% of all casualties within 10 miles of lifeboat stations within 30 minutes". (RNLI, 2009).

model was devised, monetising the perceived search-and-rescue benefit of saving lives and scanning large ocean areas. Vanguard Studio[14] generated the models and statistical distributions.

5.4.3.1 Cost model

The cost model used for this case study employs a comprehensible bottom-up cost estimation approach. Usually, parametric or analogous (*i.e.* top-down) cost estimation methods are used during conceptual design due to the lack of available information (Curran et al. 2004; Fielding 1999). However, OSCAR processes more detailed information during the conceptual design phase and merits the application of a bottom-up approach. The cost model structure can be seen in Figure 5-11.

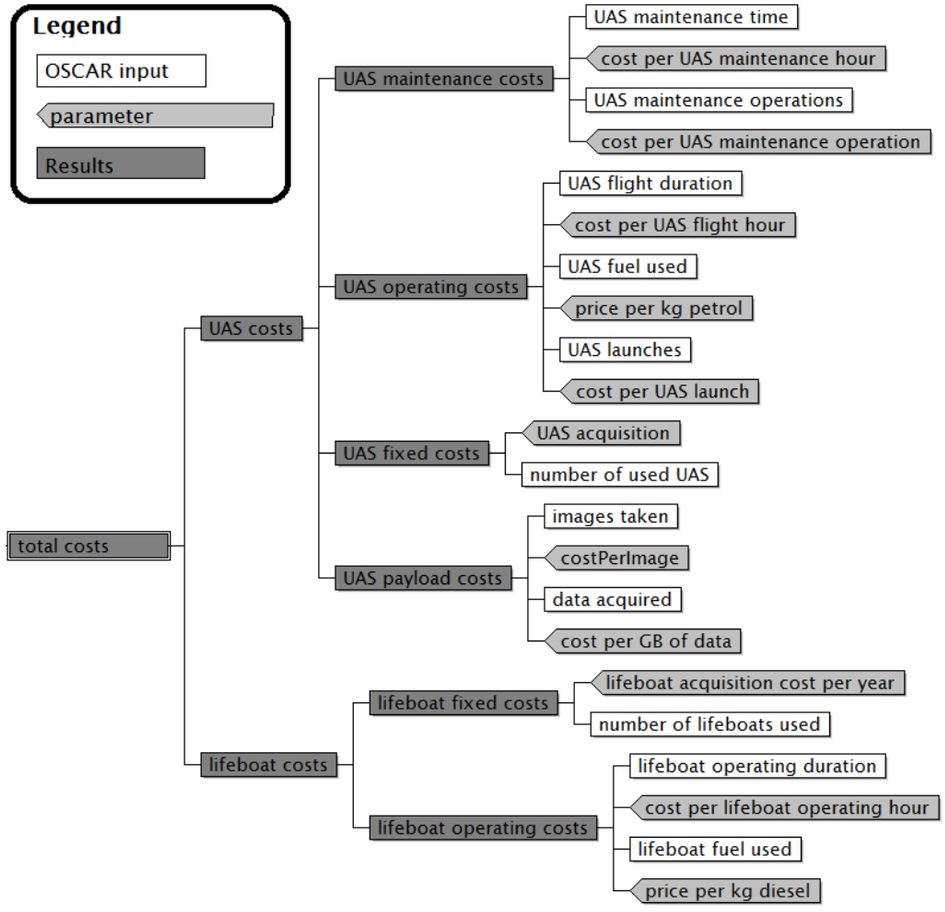


FIGURE 5-11: COST MODEL OVERVIEW

The model consists of OSCAR inputs provided by the operational simulation, cost parameters and results. The total system cost is subdivided into UAS and lifeboat costs, reflecting the case study setup. The "baseline" scenario has `UAS costs = 0`. UAS costs are further sub-divided into maintenance, operating, fixed and payload costs while lifeboat costs only consists of fixed

[14] Vanguard Studio Version 5.2.0, available at http://www.vanguardsw.com/.

and operating costs. This reflects the additional OSCAR outputs available for UAS while keeping the lifeboat outputs as simple as possible. Actual cost calculation is conducted via simple factoring of OSCAR outputs and given parameters. By way of example, the UAS maintenance costs in Figure 5-11 are calculated as in Eq. 5-1.

$$C_M = (t_M \times C_{Mh}) + (\varphi_{Mo} \times C_{Mo}) \qquad \text{Eq. 5-1}$$

Where C_M is the maintenance cost, t_M is the maintenance time, C_{Mh} is the cost per maintenance hour, φ_{Mo} is the number of maintenance operations and C_{Mo} is the cost per maintenance operation. Appendix 14 derives the cost parameters (light grey arrowed boxes in Figure 5-11). Each OSCAR output (white boxes in Figure 5-11) was included into the value model as follows: First, output values for each iteration were summarized by fitting a statistical distribution using Vanguard Studio distribution fitting. Distributions were included into the cost model such that 10,000 Monte Carlo simulation runs could be produced to obtain overall cost distributions.

5.4.3.2 Benefit model

As search-and-rescue operations do not produce economic revenue or income, the perceived benefit of saving lives and scanning large ocean areas were monetised in a benefits model. Figure 5-12 shows the benefits model used for this case study.

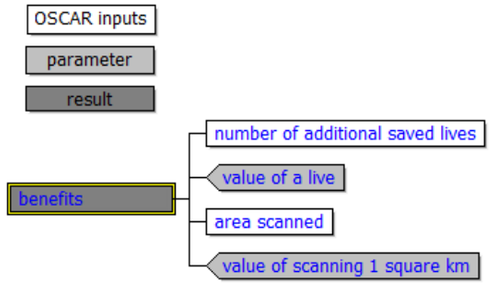


FIGURE 5-12: BENEFITS MODEL OVERVIEW

This model uses the same factoring approach as the cost model above. It quantifies the benefits of search-and-rescue in monetary terms to compare it to the costs caused by search-and-rescue operations. The major benefit of conducting search-and-rescue is saving lives. In order to save more lives, rescuers must find and rescue incidents as fast as possible as the prime driver for death at sea is time in water (Suominen et al. 2002). Therefore, the absolute number of saved lives is the main benefit. However, in this case study, we are interested in the improvement over the baseline case so the number of additional saved lives over the baseline case is used. Somewhat arbitrarily, this benefit model also includes the scanned area as a benefit. This founds upon the assumption that scanning more area can reveal other incidents that search-and-rescue opera-

tors were not aware off, increasing total benefits further. However, it also demonstrates how different OSCAR outputs can be applied for value calculation.

Several estimates for the statistical value of a live exist, varying by two orders of magnitude. The value of a statistical life of a prime-age worker in a developed country is estimated to be around $ 7M, based on a review of 100 studies on mortality and injury risk premiums (Viscusi & Aldy 2002). However, using such a high value for saving one additional live would strongly dominate the cost-benefit calculations, preventing analysis of more subtle influences. A different valuation is used based on RNLI statistics: the daily operational cost of the RNLI is £ 385,000 (RNLI 2009). On average, 22 lives are saved every day by the RNLI (ibid). As this service is financed exclusively through charitable donations, the public appears to value saving a live at sea at about £ 17,500, the value used for this case study.

The value of scanning one square kilometre is arbitrarily set to $ 1 as there are no relevant estimates for a similar application.

5.4.4 UAS landing loss plug-in

UAS landings are risky operations since a human operator or an autopilot must remotely land the UAS, often on grass or uneven ground. VESSEL loss due to landings is a relevant influence on UAS operational performance that is not captured in the generic OSCAR framework. Therefore, this Section presents an OSCAR add-in modelling the risk of VESSEL loss upon landing at BASES for UAS only.

Weibel et al. (2004) identified UAS weight as one of the most critical factors for UAS crash probability. However, a crash is more likely to occur upon landing with high speed as well. Therefore, the plug-in assumes that the probability of landing loss is related to the UAS landing kinetic energy.

There is very little scientific data of UAS landing losses for two reasons: First, manufacturers and operators do not want to publish crash data for confidentiality. Second, little data on civil UAS operations exist because their application is very new. There is not much data on landing losses for this aircraft category.

Therefore, users define a custom relation between landing loss probability and landing kinetic energy. For the case studies in this thesis, the relationship in Figure 5-13 defines the probability of landing loss based on landing kinetic energy. The setup is arbitrary and based on engineering judgement, consultation with UAS pilots and designers.

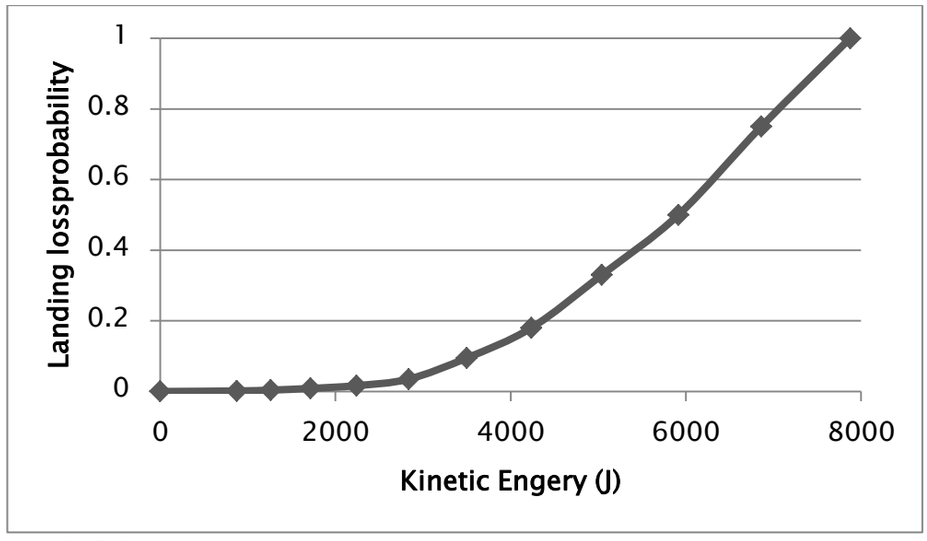


FIGURE 5-13: LANDING LOSS PROBABILITY BASED ON LANDING KINETIC ENERGY FOR UAS.

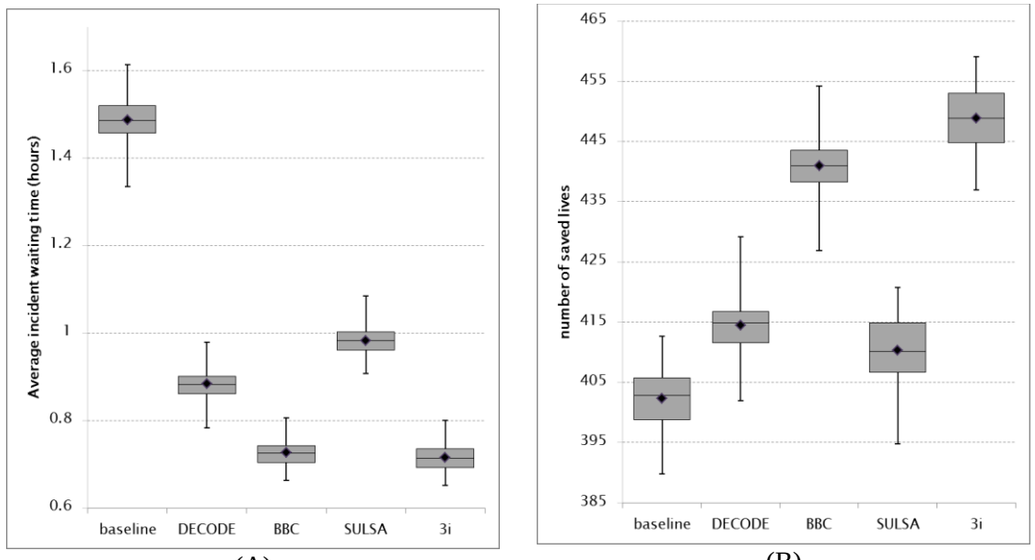
Upon UAS arrival at its BASE, the current landing speed (calculated from the aircraft performance add-in, see Appendix 9) and the current UAS weight define the landing kinetic energy. The UAS is lost with the given probability. If the UAS is lost, a replacement UAS immediately takes over and adds to the fixed UAS costs as the number of used UAS rises (Figure 5-11).

5.5 Results and Analysis

This section presents and analyses all results from the case study, starting with the raw OSCAR outputs (Section 5.5.1), followed by the costs (Section 5.5.2), benefits (Section 5.5.3) and value outputs derived from the value model (Section 5.5.4).

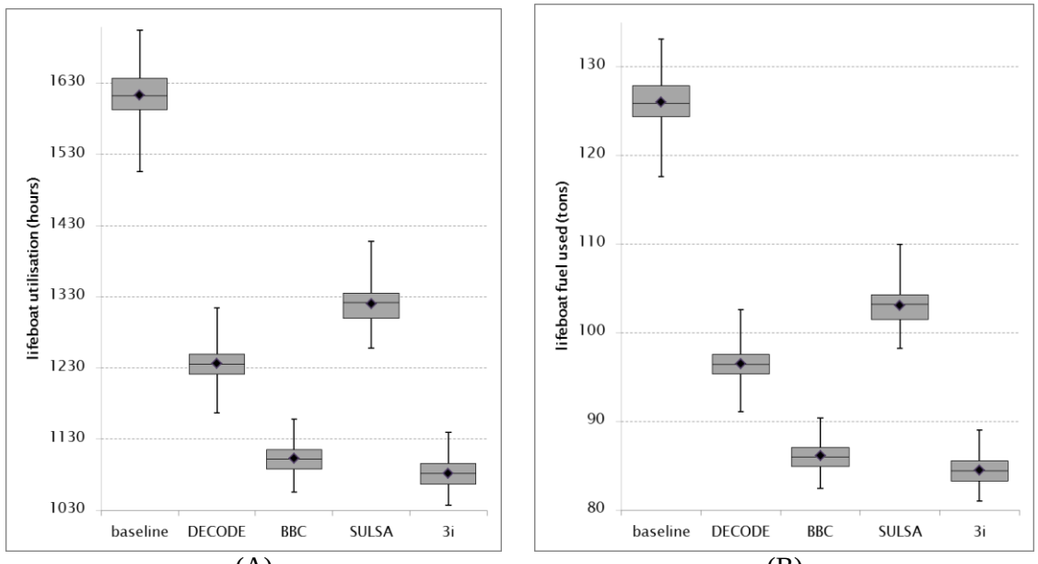
5.5.1 OSCAR outputs

For this case study, 14 outputs were recorded for 120 iterations for each of the simulation runs (one baseline case and four UAS designs), totalling 8400 outputs. The following boxplots summarise outputs aggregated over the simulation period of one year. Note that whiskers indicate the maximum/minimum of data (unless otherwise stated).



(A) (B)

FIGURE 5-14: AVERAGE INCIDENT WAITING TIME (A) AND THE NUMBER OF SAVED LIVES (B)



(A) (B)

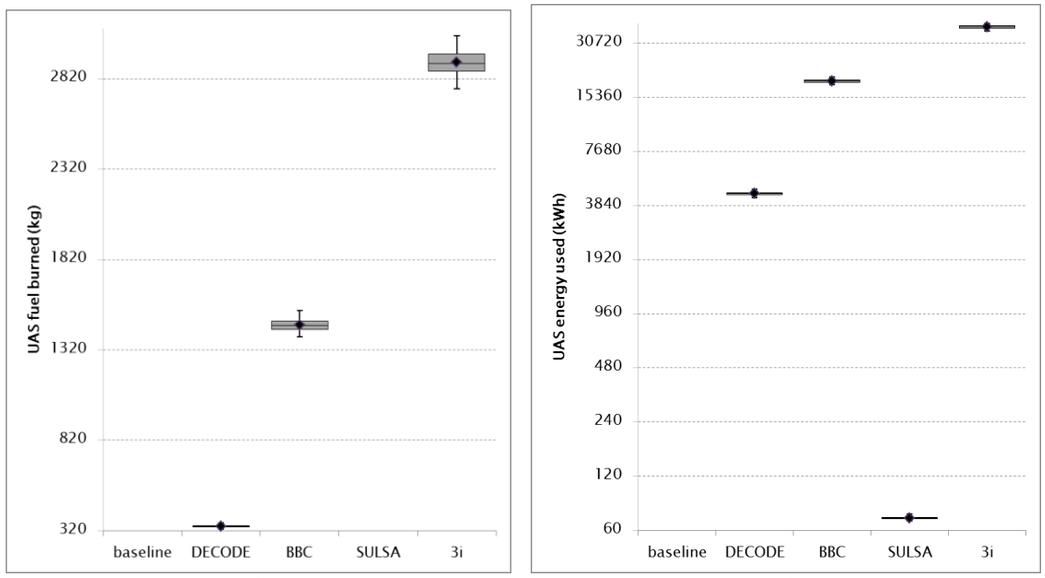
FIGURE 5-15: LIFEBOAT UTILISATION (A) AND FUEL USED BY LIFEBOATS (B)

Each UAS design reduces the average incident waiting time (Figure 5-14 (A)) by at least 30 minutes compared to the baseline case. The BBC design and the 3i design are more successful at this than the DECODE and SULSA design. The waiting time reduces because the UAS searches in parallel with lifeboats, thereby spotting incidents earlier, on average. The better performance of the BBC and 3i aircraft originates from their higher maximum and search speeds, reaching the search area faster and spotting incidents earlier, on average.

The increase in the number of saved lives in Figure 5-14 (B) is similar in trend but less strong in absolute numbers. The BBC and 3i design increase the number by about 10 % while DE-

CODE and SULSA add less than 3% to the number of saved lives. As incident survival is critically linked to time in water (Suominen et al. 2002), the faster BBC and 3i designs can rescue more people alive.

By introducing UAS to the baseline case, lifeboats utilisation decreases (Figure 5-15 (A)). The BBC and 3i design reduce the required lifeboat hours by about 33% while DECODE and SULSA reduce it by 22.5% and 19%, respectively. In absolute terms, each lifeboat is used between 18 and 27 minutes per day, on average. Fuel usage reduces accordingly (Figure 5-15 (B)): Each lifeboat burns between 23 and 35 kg of diesel per day. Lifeboat utilisation is inversely proportional to the number of saved lives (Figure 5-14), because the faster BBC and 3i designs help spotting incidents earlier, allowing lifeboats to return home quicker, thereby reducing their overall utilisation.

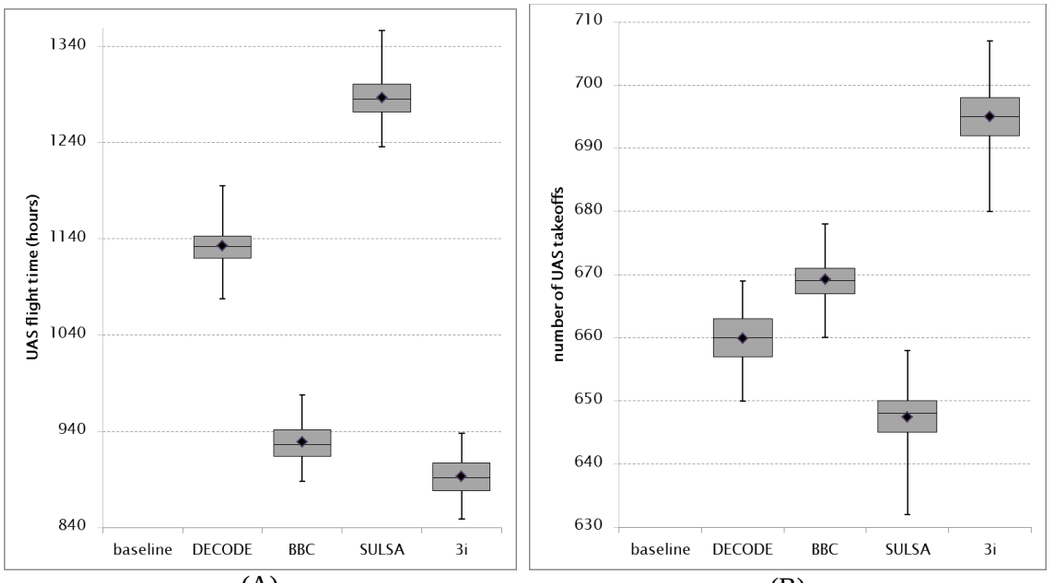


(A) (B)

LOGARITHMIC SCALE.

FIGURE 5-16: UAS FUEL BURNED (A) AND UAS ENERGY USED (B)

Figure 5-16 (A) plots the fuel burned for petrol-driven UAS (DECODE, BBC and 3i). DECODE burns about 1kg each day, BBC requires 4 kg and the twin-engine 3i burns 8 kg each day, on average. SULSA does not burn any fuel as it is propelled by an electric engine. To compare SULSA energy consumption, the energy consumed by the petrol designs was converted using the calorific value of petrol, as in Figure 5-16 (B). Still, SULSA consumes 64 times less energy than DECODE at 0.2 kWh or 720 kJ per day. In general, fuel consumption is inversely proportional to UAS flight times (Figure 5-17) because a fast design burns more fuel but also returns home earlier, on average.



(A) (B)

FIGURE 5-17: UAS FLIGHT TIME (A) AND NUMBER OF UAS LAUNCHES (B)

UAS flight times (Figure 5-17 (A)) correlate with lifeboat utilisation (Figure 5-15 (A)). The reason is that the slower DECODE and SULSA designs (flight times at about 3.3 hours each day) generally need longer to spot incidents (Figure 5-14 A). The faster BBC and 3i designs (operating for about 2.5 hours per day) spot incidents earlier, accruing less flight time. This is not mirrored by the number of take-offs shown in Figure 5-17 (B): While DECODE, BBC and SULSA launch around 1.75 times per day, 3i launches 1.9 times each day. The reason is found in Figure 5-18 showing the number of refuelling operations conducted during searches.

While BBC and SULSA virtually never need to refuel during a search, the high search speed and performance of 3i require it to interrupt a search in order to return home for refuel almost twice a month. Each refuel adds a take-off in Figure 5-17(B). However, despite more refuel operations for DECODE compared to BBC, it has less take-offs accrued. This indicates the following specific case: the slow DECODE design searched for an INCIDENT while a second INCIDENT appeared somewhere else. Due to its slow speed, it could not respond to that second INCIDENT which was rescued by lifeboats instead. As the UAS still searched for the first INCIDENT, it accrued less take-offs in total, not needing to dispatch to the second INCIDENT anymore (see discussion in Schumann et al. (2011) for more details of this specific case).

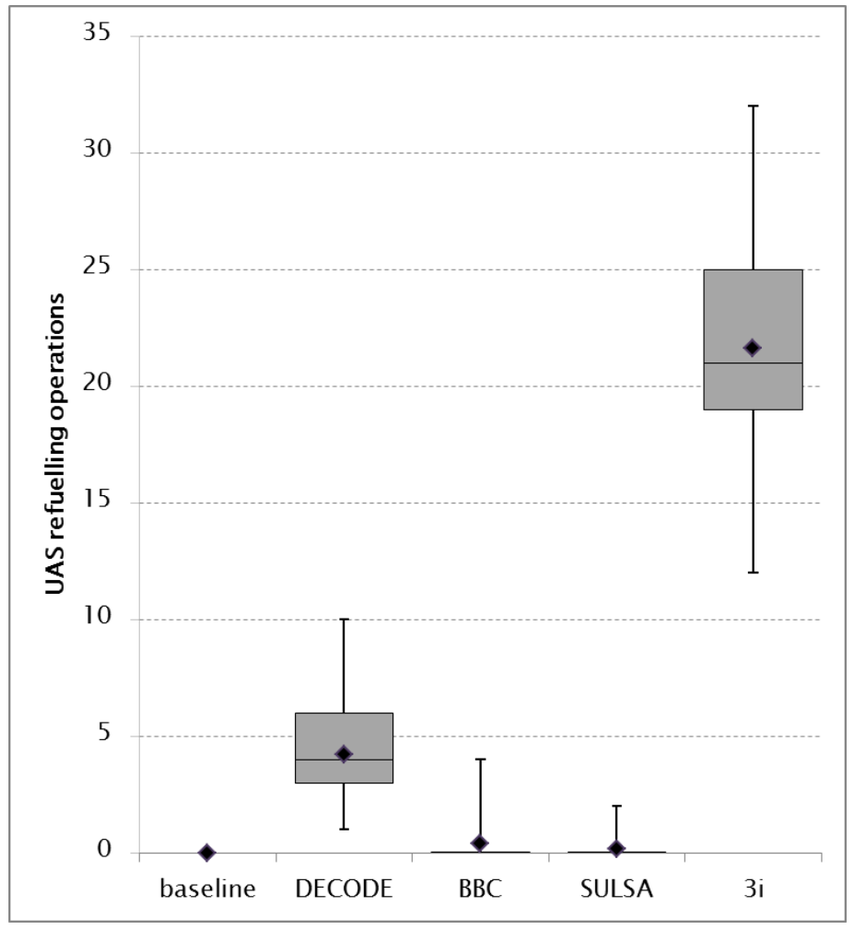


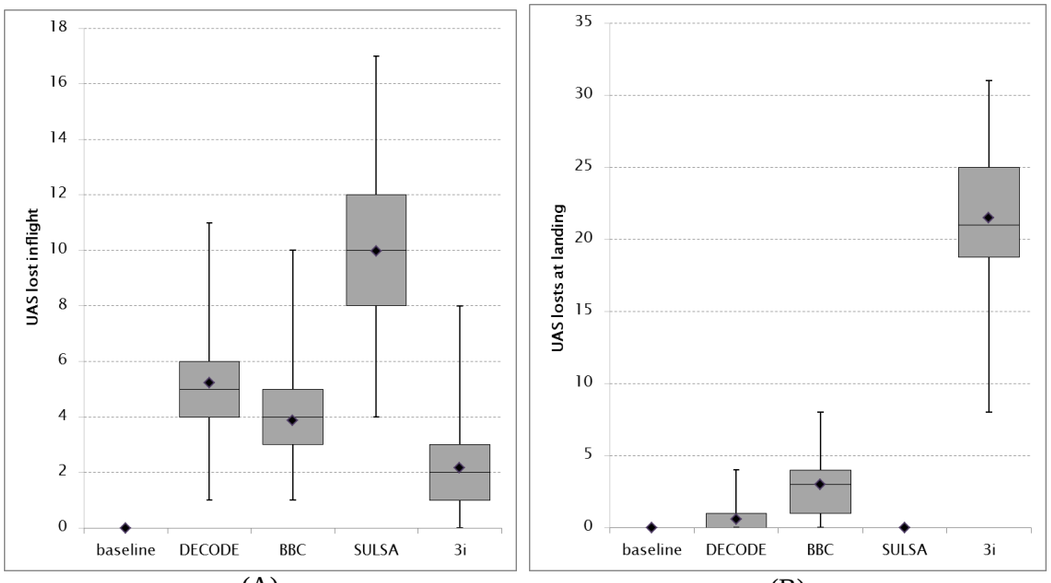
FIGURE 5-18: UAS REFUELLING COUNT

Figure 5-19 sums the UAS losses occurring inflight (A) and upon landing (B). DECODE and BBC losses are similar at about one inflight loss every 3 months and a landing loss once a year. SULSA is lost inflight almost every month and never upon landing (due to its low mass). Contrary, 3i is lost very rarely inflight but almost every 2 weeks upon landing.

Inflight losses occur when any component without redundancy stops working and causes loss of control (Section 4.12). DECODE and BBC feature identical components (compare Appendix 12) and the slightly higher DECODE inflight losses are caused by the larger flight time of DECODE. SULSA has fewer parts than the other designs and a much more reliable fuselage. However, its large number of inflight losses is also rooted in its increased flight time, causing more COMPONENT deterioration. 3i has less inflight losses because it has redundant engines, throttle servos, ignitions and propellers. Moreover, inflight losses are reduced by 3i's large number of landing losses because a new UAS is purchased after any loss, increasing the time to the next loss due to the new COMPONENTS aboard.

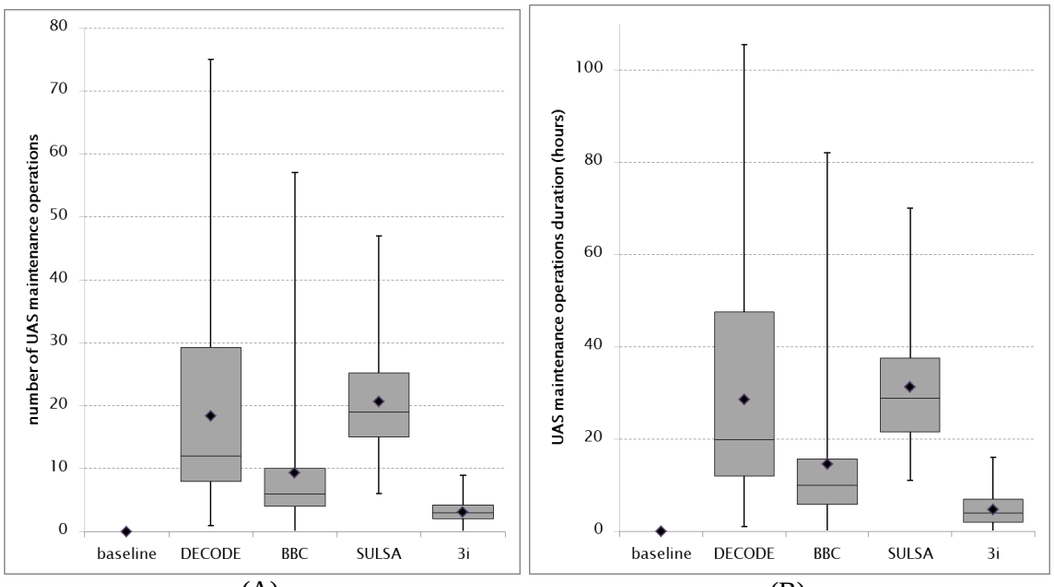
Landing losses occur based on the vessel kinetic energy upon landing (see Section 5.4.4). 3i is by far the heaviest design with the highest landing speeds and, therefore, features the highest risk of crashing upon landing. SULSA as the other extreme is very light and has a lower landing

speed, its average landing kinetic energy being about 20 times lower than 3i. Both DECODE and BBC are in between these extremes.



(A) (B)

FIGURE 5-19: UAS INFLIGHT LOSSES (A) AND UAS LANDING LOSSES (B)



(A) (B)

FIGURE 5-20: UAS MAINTENANCE OPERATIONS OUTPUTS WITH COUNT (A) AND MAINTENANCE OPERATIONS DURATION (B)

Figure 5-20 (A and B) indicate the maintenance operations details. As the maintenance duration for specific component repairs is deterministic (Appendix 4), both outputs correlate closely. They feature separately here because the cost model accounts for both separately.

The 3i aircraft has least repair requirements not because it has a sturdier structure or more reliable COMPONENTS but because it crashes more often upon landing (see Figure 5-19). A new UAS featuring new COMPONENTS is purchased after these frequent landing crashes, increasing the time to subsequent COMPONENT failures (See Schumann et al. (2012) for a more detailed analysis of this counter-intuitive result). The large spread of data for all designs roots in the relatively short simulation period of one year. During this time, too few maintenance operations are conducted to reduce spread satisfactorily. However, the OSCAR maintenance outputs are of minor importance to the value model results below, not justifying the additional workload of defining longer life cycles.

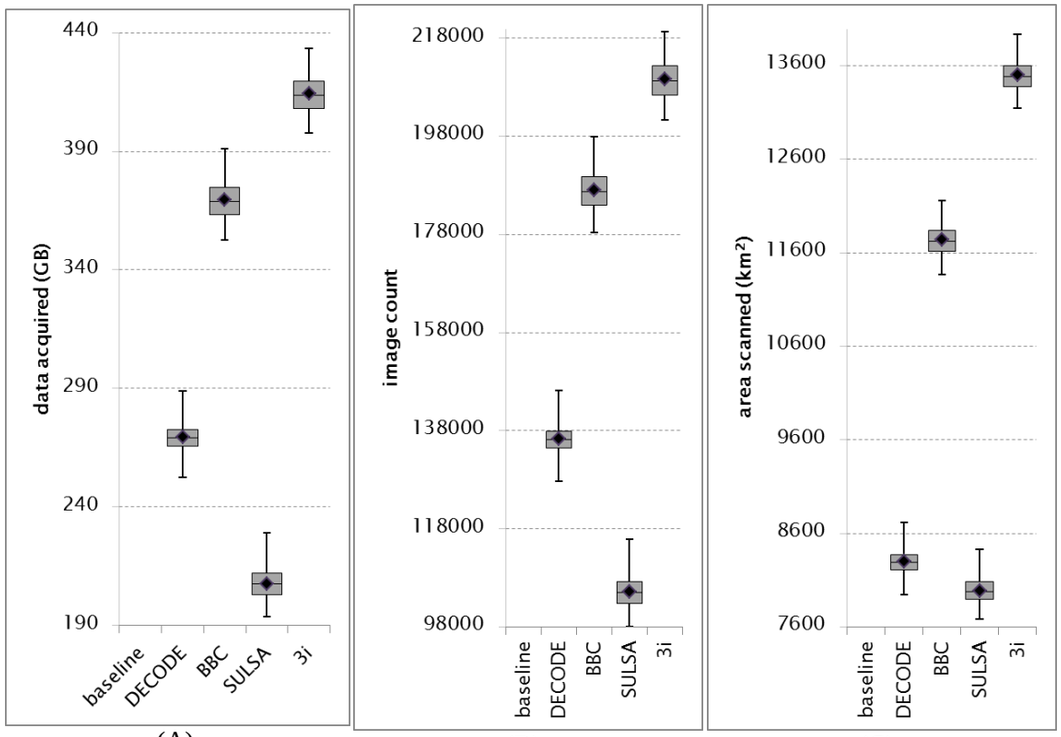


FIGURE 5-21: UAS CAMERA OUTPUTS: ACQUIRED DATA (A), IMAGE COUNT (B) AND AREA SCANNED (C)

Figure 5-21 (A-C) depicts measures collected for the UAS payload camera system. Data distribution is highly correlated between the acquired data, image count and area scanned because each depends linearly on the number of pictures taken (Section 4.10). Camera outputs are inversely proportional to UAS flight times (Figure 5-17 A) so that much data is gathered when a design spends little time in the air. The cause for this counter-intuitive correlation is found in the search-speeds for each design (see Appendix 12): BBC and 3i search at much higher speeds than DECODE and SULSA, thereby covering a larger area, taking more pictures and arriving home earlier (*i.e.* less flight time accrued).

5.5.2 Costs

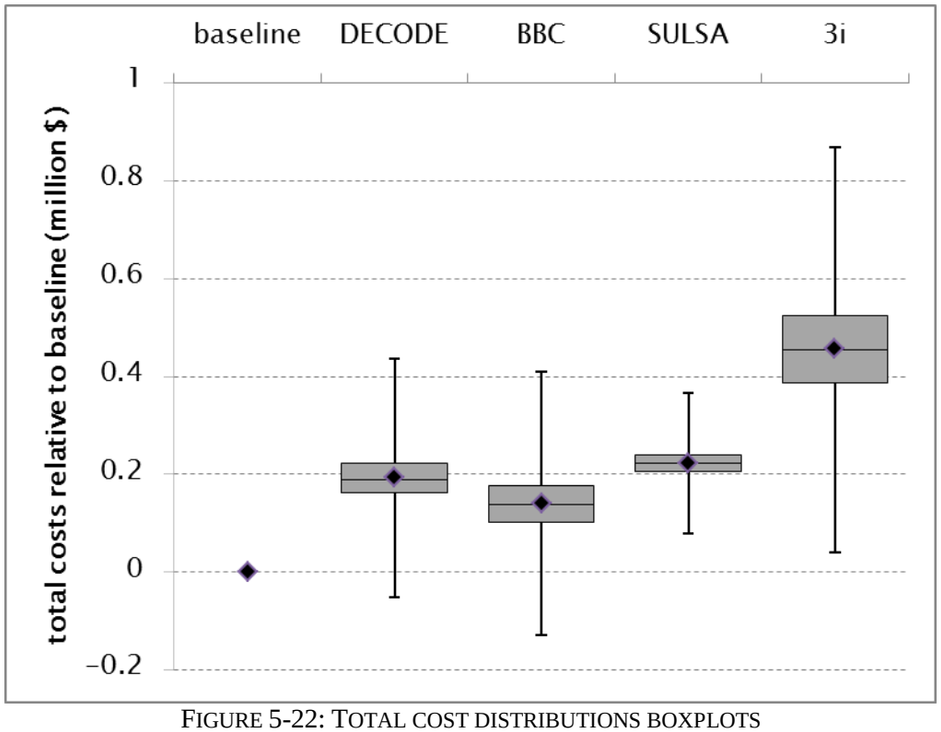


FIGURE 5-22: TOTAL COST DISTRIBUTIONS BOXPLOTS

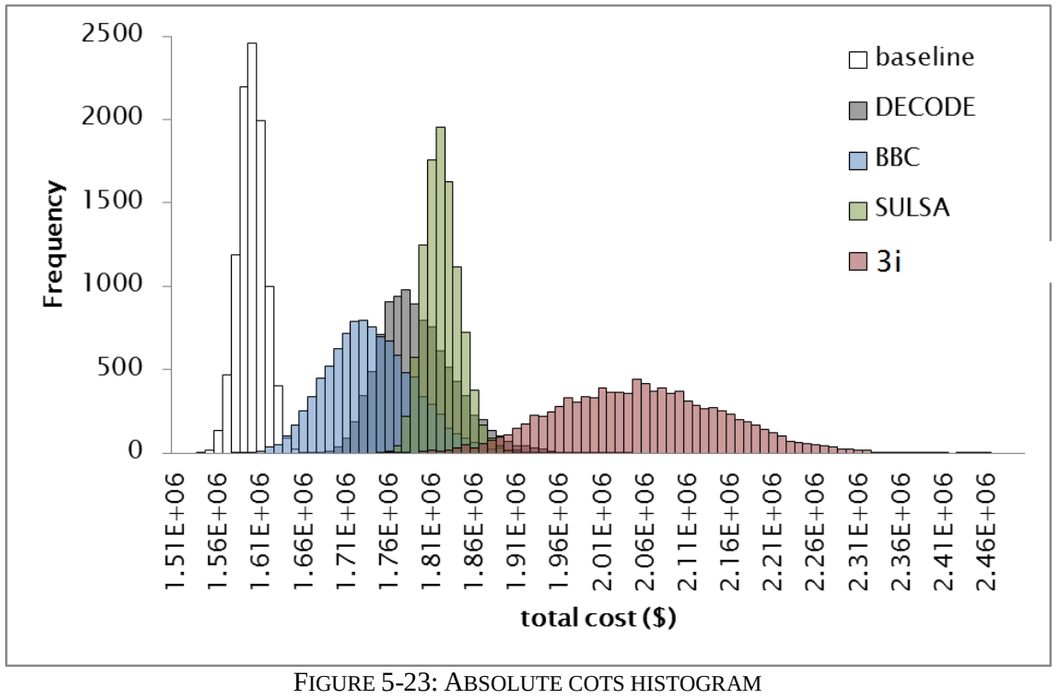


FIGURE 5-23: ABSOLUTE COTS HISTOGRAM

As shown in Figure 5-22, total system cost relative to the baseline case increases upon introducing any type of UAS. This result bases on 10,000 Monte Caro runs of the cost model (Section 5.4.3.1).

DECODE and SULSA add costs of about $ 200,000, BBC has the lowest increase at about $ 150,000 while introducing 3i is most expensive at an additional $ 460,000, on average. Figure 5-23 shows the absolute system cost histograms to assess the spread of data.

All design costs are normally distributed. The baseline and SULSA case have the smallest spread while 3i features the largest spread at $\sigma \approx \$\ 99{,}000$ and $\mu \approx \$\ 2{,}050{,}000$.

5.5.3 Benefits

Figure 5-24 depicts the benefits relative to the baseline case, based on 10,000 Monte Carlo runs of the benefits model (Section 5.4.3.2).

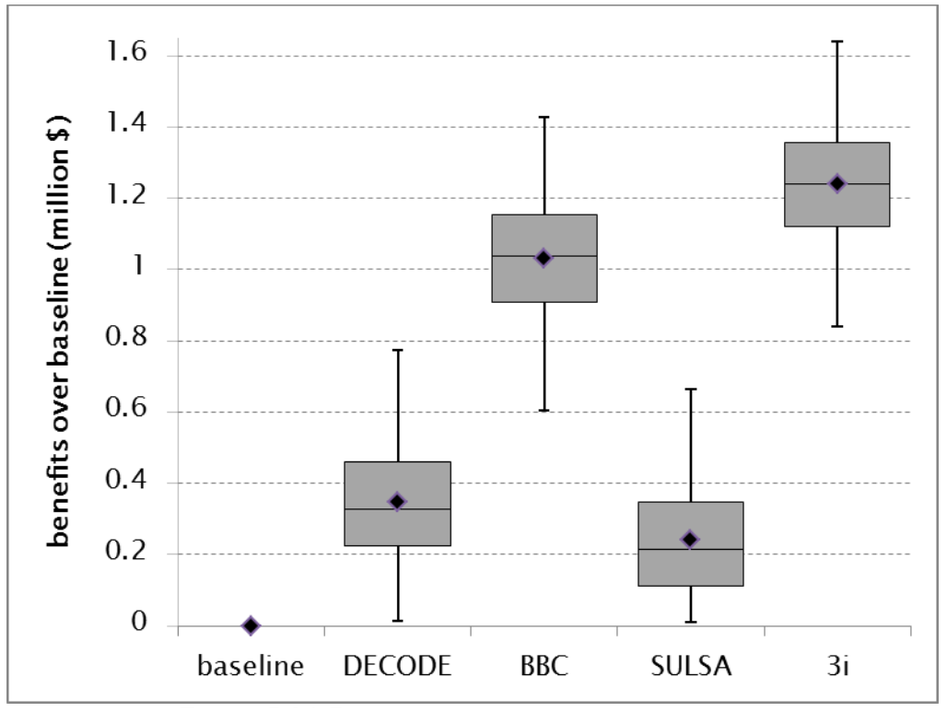


FIGURE 5-24: BENEFITS RELATIVE TO BASELINE CASE.

Introducing UAS has a beneficial effect in any case: on average, each UAS design accrues between $ 0.2M and $ 1.2M. BBC and 3i generate about $ 0.9M more than DECODE and SULSA. For each design, the major benefit contribution is the number of additional saved lives because the value of scanning one square kilometre is not large enough to compare. This is as expected because it is more desirable to save lives than to scan very large areas of the ocean. In some cases, both DECODE and SULSA can cause a benefit of nearly $ 0 because they do not save any additional lives compared to the baseline case. This is reflected in UAS flight times (Figure 5-17 A) where DECODE and SULSA feature very long flight hours while only saving marginally more lives (Figure 5-14 B). The reason is the slower dash and cruise speeds causing these designs to spot incidents later, sometimes even after the lifeboats.

5.5.4 Values

By subtracting the additional costs over the baseline (Figure 5-22) from the additional benefits over the baseline (Figure 5-24), the expected value over the baseline can be plotted as in Figure 5-25.

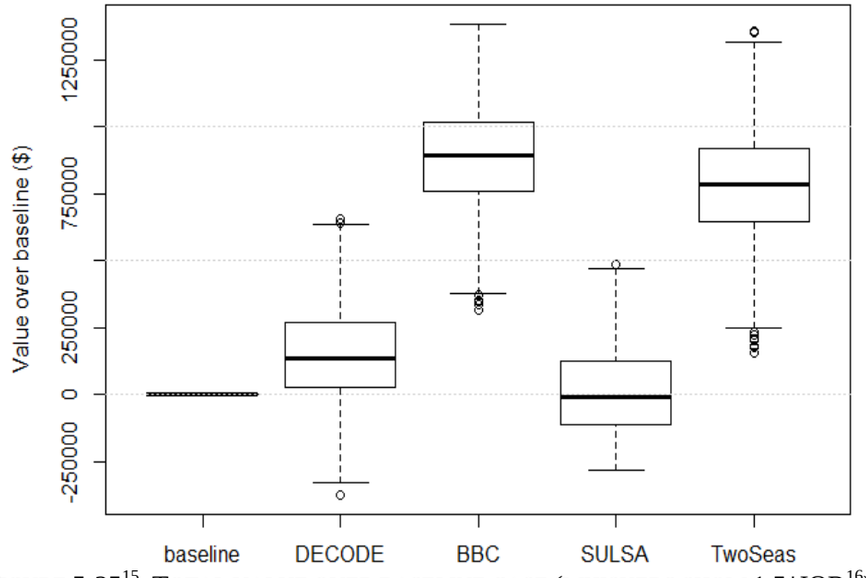


FIGURE 5-25[15]: TOTAL VALUE OVER BASELINE CASE (WHISKERS SHOW 1.5*IQR[16]).

On average, the most valuable design to use is BBC, closely followed by 3i, both creating about $ 0.8M of additional value compared to the baseline case. Both DECODE and SULSA score considerably worse with SULSA featuring an average negative value.

In this comparative case study, it is not useful to plot the absolute value of all designs and the baseline case because the underlying parameter assumptions are too vague to justify discussion.

5.6 Discussion

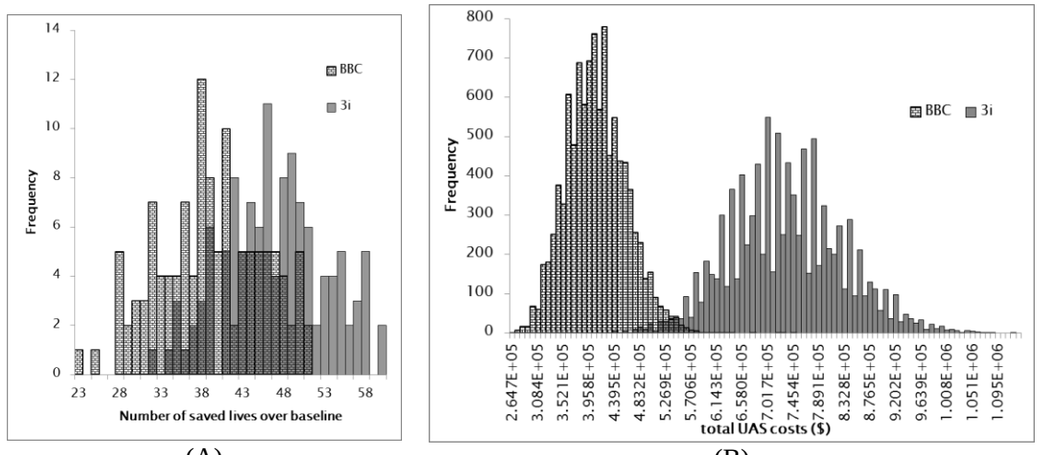
This section discusses the results obtained concerning the initial case study objective, namely to assist in decision support for value-driven design. As with all but the simplest engineering problems, there cannot be a single right answer as to which design should be used. The following sub-sections consider decision support from pure value or cost considerations as well as from a qualitative viewpoint. Additional benefits of the OSCAR approach are debated followed by a discourse into the applicability of OSCAR conceptual value-driven design.

[15] In this graph, boxplot whiskers indicate the third quartile plus 1.5 times the inter-quartile range and the first quartile minus 1.5 times the inter-quartile range.

[16] Here, IQR abbreviates Inter-Quartile Range

5.6.1 Value-based decision support

Computing a numerical value as part of the value-driven design approach allows comparing different designs based on a single, monetary number. Using the simple value model above, Figure 5-25 identifies the BBC design as the one with the highest average value, closely followed by 3i. Numerically, the decision is clear. However, the proximity of value of both designs constitutes a closer look. Figure 5-26 compares the additional saved lives over the baseline case with the UAS cost distribution for BBC and 3i.



(A) (B)

FIGURE 5-26: SAVED LIVES OVER BASELINE HISTOGRAM (A) AND UAS COSTS (B) FOR BBC AND 3I.

The latter saves about 6.8 more lives, on average (worth $ 119,000 ing $value\ of\ a\ statistical\ life = \$\,17{,}500$). However, it requires additional average costs of $ 325,000 to achieve this. The largest part of this additional cost is rooted in the fixed costs as seen in Figure 5-27.

While operating costs are almost identical for both UAS, the large number of 3i losses blow up its fixed costs (see Figure 5-19 A). In this study, fixed costs consist of the number of lost UAS only, so high fixed costs indicate a high number of UAS losses.

With this information, value-based decision support allows a more informed decision as to which design to go for. Before, value-driven design would argue to choose the BBC design because it has a higher value over the baseline. Now, analysis shows that not only has the BBC design a higher value, it also crashes less often, a characteristic highly desirable in civil UAS applications.

In fact, a 3i design is lost almost every two weeks during operations while the BBC design breaks every 8 weeks on average (see Figure 5-19 A). However, this level of reliability is far too low for practical purposes. As shown in Figure 5-28, both BBC and 3i have similar number of inflight losses but 3i has far more landing losses.

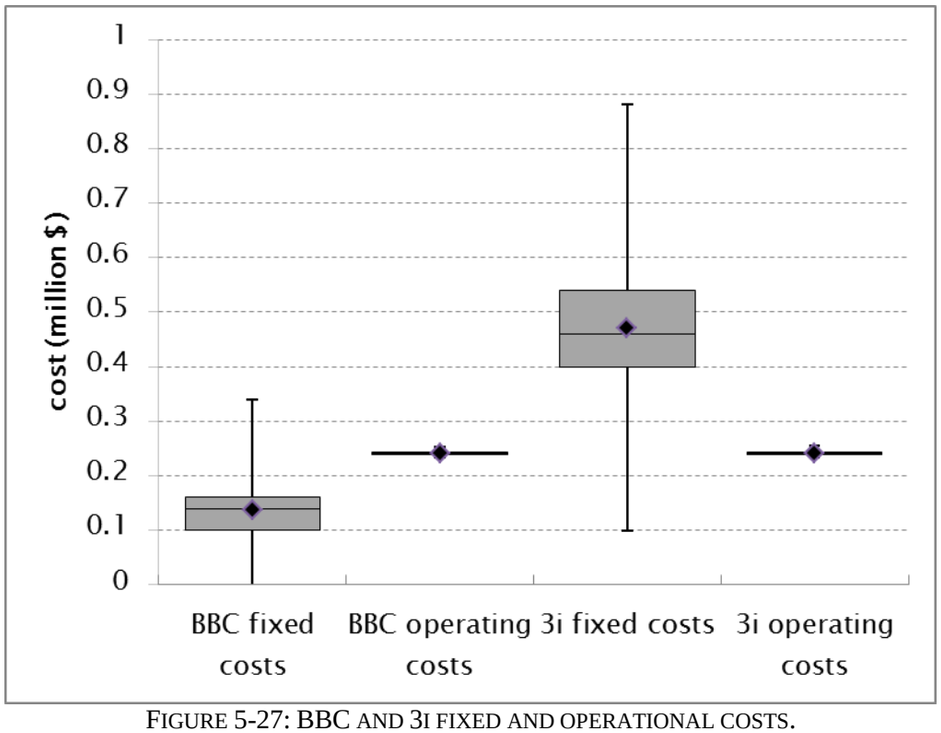


FIGURE 5-27: BBC AND 3I FIXED AND OPERATIONAL COSTS.

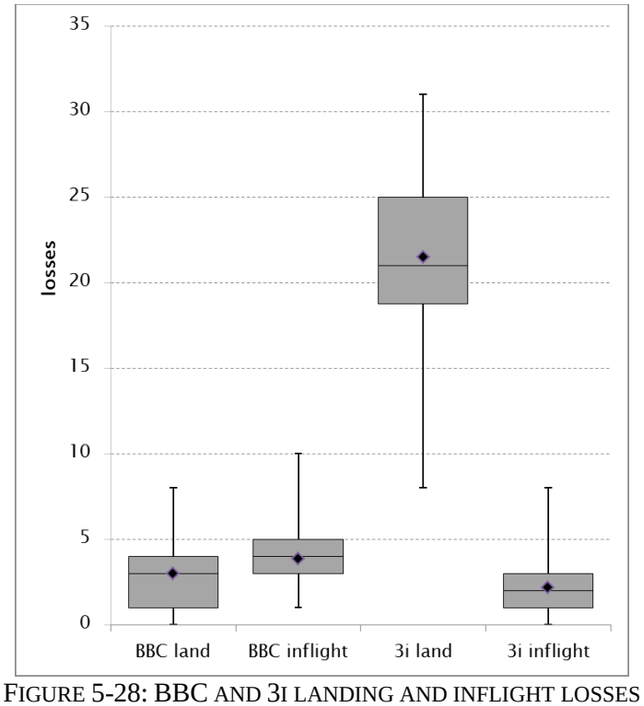


FIGURE 5-28: BBC AND 3I LANDING AND INFLIGHT LOSSES

The high number of landing losses is attributed to the high kinetic energy upon landing. As both BBC and 3i have similar landing speeds, it is the higher dry weight of the 3i design causing the additional landing losses. This higher weight originates from the prudent design approach for 3i, being tailored for maritime search-and-rescue. BBC, on the other hand, was designed for broadcasting (Section 5.2.3.2). In this case study, the additional weight of 3i, caused by twin

engines and other backup COMPONENTS as well as a sturdier structure, is only partly balanced by improved reliability: 3i has the lowest number of inflight losses but landing losses due to high weight undo these benefits. In order to reduce the inflight losses, the COMPONENT reliability model allows calculating the average losses for each component in order to identify hot spots as in Figure 5-29.

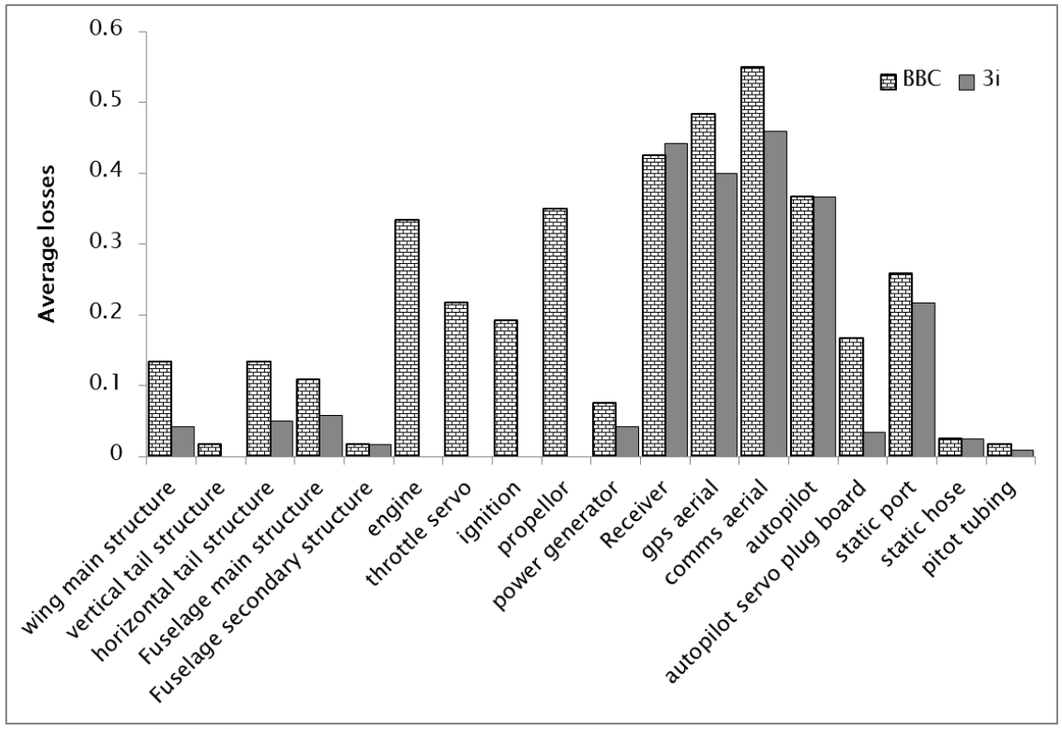


FIGURE 5-29: AVERAGE INFLIGHT LOSSES FOR BBC AND 3I, BASED ON COMPONENTS.

Both designs fail frequently due to receiver, GPS & comms aerial and autopilot issues. These COMPONENTS are not subject to loads and physical fatigue so they can be improved by buying better components. Moreover, the 3i design has no losses caused by the engine, throttle servo, ignition or propeller while BBC suffers a sizable number of losses from these COMPONENTS. Here, the benefit of redundant COMPONENTS in 3i becomes visible and should be taken into account by decision makers. The 3i aircraft will be more reliable inflight and would become a feasible alternative once landing issues are resolved. If the number of landing losses for 3i reduces, its overall cost would decrease, making it the most valuable design option.

5.6.2 Cost-based decision support

The value model in this case study is a very simple factoring of parameters. The parameters (`valueOfAStatisticalLife` and `valueOfScanning1SquareKm`) are very difficult to quantify, as search-and-rescue is an intrinsically non-commercial enterprise lacking an obvious monetary profit. Several estimates for the statistical value of a live exist, varying by two orders

of magnitude. Therefore, it is useful to neglect benefits and discuss OSCAR results based on costs only.

As seen in Figure 5-22, introducing any UAS increases costs relative to the base case. In fact, the analysis of *absolute* system costs in Figure 5-30 shows that not only do costs increase but also their spread.

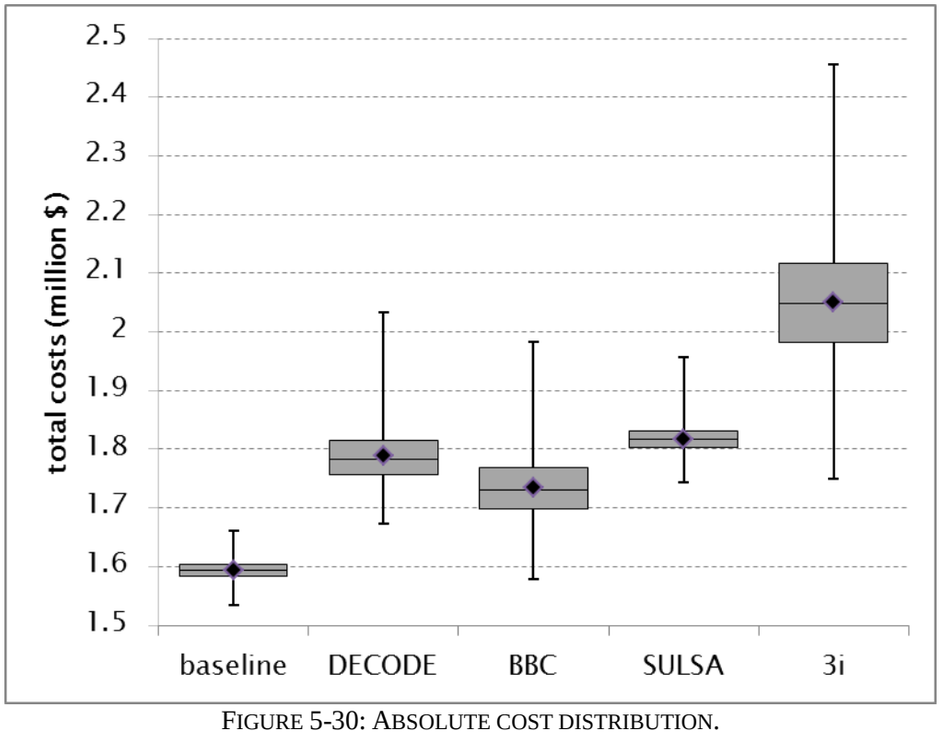
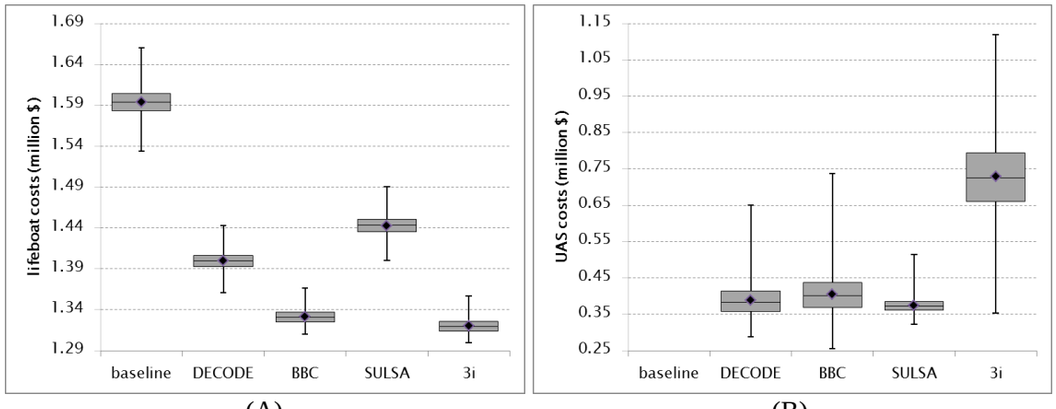


FIGURE 5-30: ABSOLUTE COST DISTRIBUTION.



(A) (B)

FIGURE 5-31: ABSOLUTE LIFEBOAT COSTS (A) AND ABSOLUTE UAS COSTS (B)

Intuitively, one might expect total cost to reduce upon introducing UAS because they are cheap to operate and reduce utilisation of expensive lifeboat equipment. In fact, lifeboat costs do decrease using UAS (see Figure 5-31 A). Cost decrease ranges between $ 150,000 and $ 270,000, attributed to less lifeboat utilisation and fuel burn as the UAS helps spotting incidents

earlier. However, the net rise in total cost originates from additional UAS costs depicted in Figure 5-31 B (ranging between £ 350,000 and $ 720,000).

The large whiskers in Figure 5-31 B are caused by outliers and do not represent highly uncertain cost estimates, as can be seen in the UAS cost histograms in Figure 5-32.

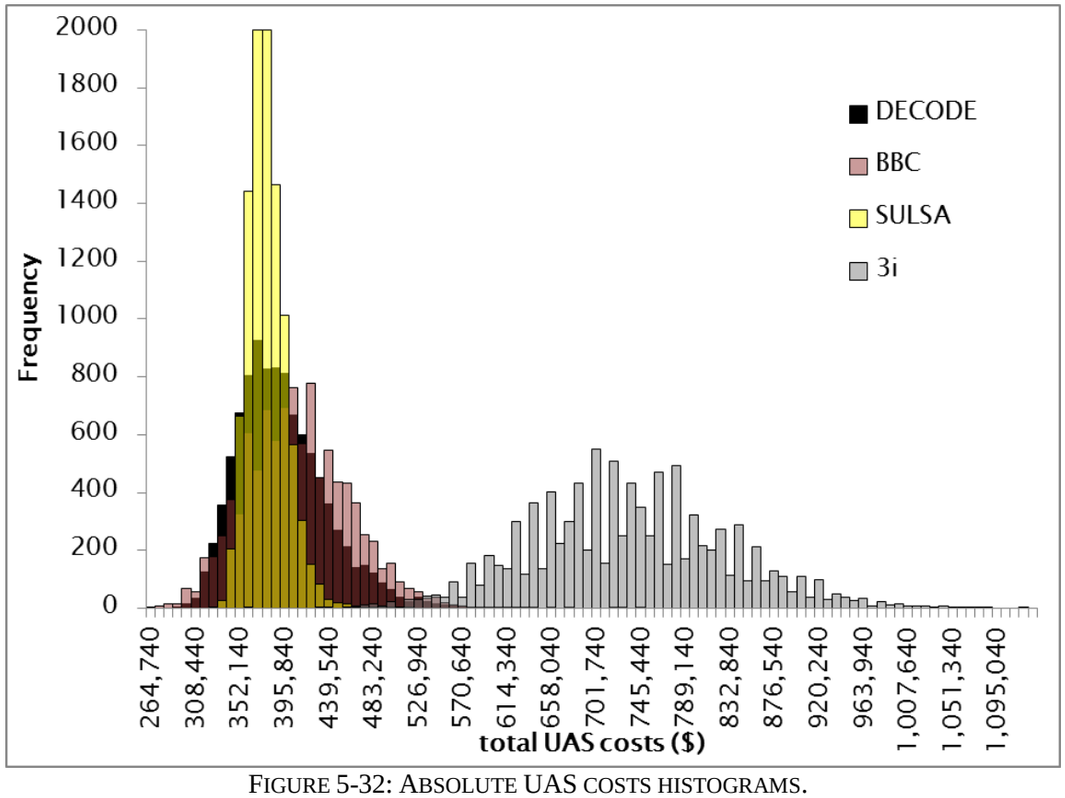


FIGURE 5-32: ABSOLUTE UAS COSTS HISTOGRAMS.

The main contributing factor to UAS costs is found through sensitivity analysis of the cost model. For BBC, `cost per UAS flight hour` (172.09 $/*hr*) and `UAS acquisition` ($20,000) are the most influential parameters (Figure 5-33 A). For 3i, it is the same but `UAS acquisition` is more influential than `cost per UAS flight hour` (Figure 5-33 B). Designers should focus upon reduction of UAS losses while the cost per UAS flight hour parameter needs to be reviewed or, if deemed correct, labour cost must be reduced through further automation.

However, sensitivity analysis of these parameters as in Figure 5-34 reveals that any cost parameter (except BBC cost per UAS flight hour) would need to reduce by 100% to reduce total costs to below the baseline case average.

Therefore, a combined parameter reduction is desirable to achieve total cost reduction through UAS implementation. Moreover, `cost per UAS launch` and `cost per UAS maintenance operation` influence on total cost is not negligible and should be reduced as much as possible. Only the `cost per image` and `cost per GB of data` parameters are insignificant and require no further analysis (GB is Gigabytes).

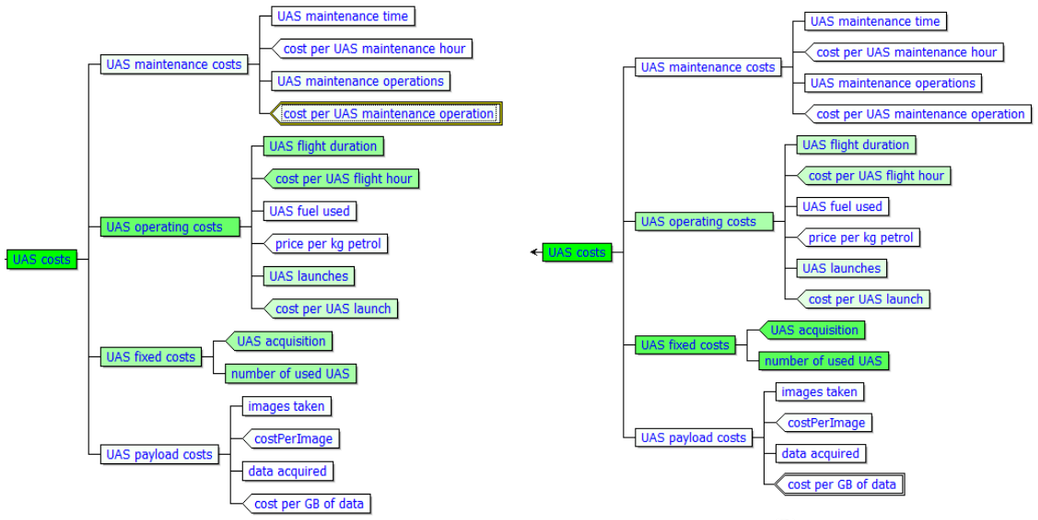


FIGURE 5-33: UAS COST SENSITIVITIES FOR BBC (A) AND 3I (B): DARKER SHADING INDICATES STRONGER INFLUENCE

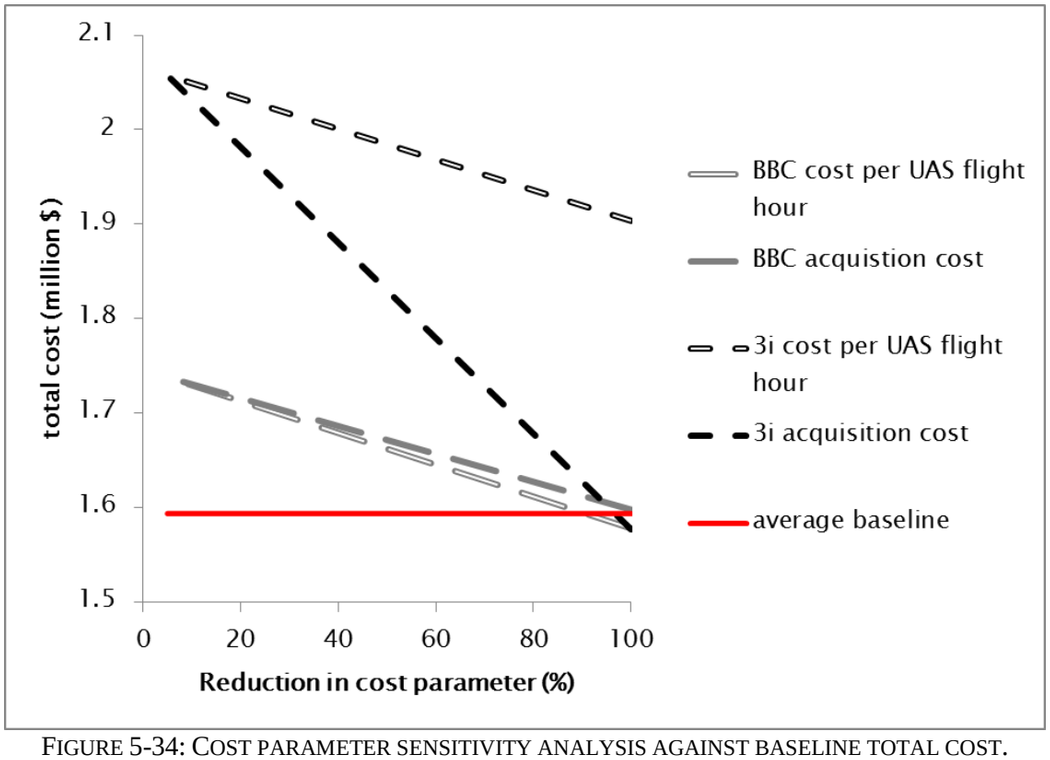


FIGURE 5-34: COST PARAMETER SENSITIVITY ANALYSIS AGAINST BASELINE TOTAL COST.

Based on cost, designers should decide for the BBC design as it has a lower additional cost impact and it has the potential to reduce total cost by optimizing several cost parameters. However, if designers can reduce the number of UAS losses for 3i (see discussion in Section 5.6.1), UAS costs would reduce dramatically and 3i could become the design of choice.

5.6.3 Qualitative decision support

Decision support discussed so far based on quantitative figures only. However, OSCAR outputs can be assessed directly, bypassing value model challenges. This section discusses some key points from OSCAR outputs as a basis for a more qualitative decision support.

Take, for example, the case where UAS acquisition is examined by a search-and-rescue authority that needs to increase its live-saving performance above all. Figure 5-14 B indicates that primarily, BBC and 3i deem a closer analysis. They allow a fourfold increase in saved lives over DECODE or SULSA. This trend is backed by the reduction in waiting time in water (Figure 5-14 A) as a twofold improvement over DECODE and SULSA. Following the analysis in Section 5.5.1, the most critical design features for saving more lives are the dash and search flight speeds. Therefore, even if none of the designs is chosen and a new design is developed for the search-and-rescue operator, this critical insight remains. Moreover, it was shown that landing crashes and inflight crashes are not only a nuisance to operators and the public, but that they are a big cost driver that must be minimized with all effort.

5.6.4 Unforeseen insights

Conducting OSCAR analysis during conceptual design reveals quantitative results and qualitative recommendations as above. However, insight into the operational environment, the product design and their interaction grows along the way. This can lead to answers that nobody knew required asking. This section reviews two sample insights for Solent search-and-rescue design that were first described in Schumann et al. (2012).

5.6.4.1 Number of launches

Figure 5-17 B depicts the number of UAS launches (*i.e.* take-offs) for all four designs. Despite a fixed number of missions, each design has a distribution of launches indicating that sometimes, they launch more often than other times. Moreover, the number varies strongly between designs so the design influences the number of launches. The reason for these variations can be explained only by considering the operations of the designs.

Consider the case of a faster design (like 3i) and a slower design (like DECODE): Sometimes, two incidents can occur nearly at the same time. The UAS is dispatched to the first incident and starts searching. A fast UAS might spot the first incident quickly, return home for a refuel and be dispatched to the second incident that has not yet been found by other lifeboats. A slow UAS takes longer to spot the first incident. Upon returning home, it finds that the second incident has already been found by other lifeboats so there is no need to dispatch anymore. The fast UAS collects more take-offs while the number of incidents stays constant.

Similarly, the fuel tank size, fuel burn characteristics and flight speed and altitude can influence the number of take-offs (see Schumann et al. (2011)). If a small fuel tank (or high fuel burn) require a design to return home during searches for refuels, the number of take-offs increases. Figure 5-18 shows that 3i requires far more refuel operations than the other designs, adding to its number of take-offs.

As the number of take-offs varies, so does the number of landings, naturally. In turn, expected landing numbers are very important for undercarriage design, as more landings require a sturdier landing gear. This can be achieved through heavier material or higher cost by employing new high-tech materials such as carbon fibre. If heavier material is used, the total design weight increases which, in turn, reduces the overall flight speeds (all other parameters being constant). However, flight speed has been identified as the most critical design parameter for a search-and-rescue UAS and any reduction results in less lives saved. Therefore, the number of take-offs (and landings) should be minimized in order to maximise saved lives.

5.6.4.2 Save more – cost more?

In this case study, payload stays constant to allow comparison of UAS design. However, consider the case where payload is varied such that a better camera system requires more weight, hence adding to the design weight. On the one hand, a better camera system would spot incidents faster as UAS would overfly incidents less often, on average. More lives could be saved. On the other hand, a better camera system is heavier and a heavier UAS requires more power (*i.e.* fuel) to be propelled through the air at constant speed. Cost increases (more fuel) when additional benefits (more lives save) are required. First order interactions like these are well known to engineers and often intuitively implemented in designs.

However, OSCAR allows analysis of the *operational effect* of a heavier payload through second order operational interactions. In this case, there is a balancing effect to burning more fuel due to a heavier camera: the UAS spots incidents *earlier* and therefore reduces its overall flight time, thereby reducing its overall fuel burn again. It is possible that this effect is stronger than the additional fuel required for the higher payload weight. In this case, saving more lives could actually decrease cost. Only a mission-modelling tool like OSCAR is able to combine first order design relations with second order operational interactions arising from the environment and procedures.

5.7 Summary

This section reviews the case study results with respect to its applicability in conceptual value-driven design.

OSCAR was developed for real conceptual design applications, keeping inputs, setup and outputs generic and variable while allowing for quick and easy data generation. A trained engineer can setup this case study within one hour, given the right data. Input data consists of geographical information, UAS parameters, incident data and implicit operational knowledge. Geographical maps can be generated within two days if the engineer works together with a search-and-rescue operator. UAS parameters are well-known conceptual design parameters that any engineer can produce in minutes from existing design tools. Implicit operational knowledge is more difficult to quantify, as each scenario requires operational alterations. The search functionality presented here is flexible enough to allow for different search patterns. However, very specific operational details (such as fleet coordination, *etc.*) cannot be incorporated easily. The computing time per UAS design is about two hours on a 2010 desktop PC with eight cores, using parallel computing. Data outputs can be queried quickly by any engineer familiar with SQL database management. Using a plotting program of choice, output data can be visualized easily.

The insights, quantifications and decision advices produced by about 30 labour-hours warrant the additional effort. A conceptual design can be chosen based quantitative and qualitative arguments not available by existing conceptual design phase mission models.

6. CASE STUDY – OPTIMISATION

This case study examines the use of the OSCAR simulation for conceptual design phase optimisation based on a real application for maritime UAS design. In reality, OSCAR would either be part of a design optimisation loop similar to the DECODE software suite (Section 5.2.2) or it would be used for manual optimisation. It is beyond the scope of this thesis to describe a complete, integrated optimisation process. Therefore, this case study will demonstrate a manual three-step optimisation indicating its potential for automated optimisation within a larger design framework (see Fu et al. (2005) for a survey of simulation optimisation techniques).

Section 6.1 explains the case study background and assumptions: they base upon the 3i research project (Section 6.2) investigating cross-border UAS application for the English Channel. Section 6.3 describes the real scenario and how UAS are intended to be used while Section 6.4 shows how this was translated into OSCAR concepts to create a viable simulation scenario. Section 6.5 analyses and discusses the results of a baseline UAS design. It is refined into a first design iteration varying the UAS design in Section 6.6. Subsequently, Section 6.7 presents a second design iteration where operational parameters vary. Section 6.8 discusses the case study findings and presents two mission-related trade-offs.

6.1 Background

The ability of UAS to provide airborne intelligence in real-time or near real-time led to the instigation of the 3i research project (Section 6.2). One of the stakeholders is the PRA (Section

6.3.1), responsible for the save conduct of operations in the harbour of Rotterdam. In this case study, assume that the 3i project finished with a working UAS design, called the "initial design" from now on. PRA is interested in using UAS to increase intelligence within their harbour area. However, they want a design optimised for their specific operational mission requirements. The 3i UAS design goal was to carry sensors for long periods, focussing on robustness and redundancy required in maritime applications. Therefore, PRA commissions a UAS manufacturer to assess the robust initial design[1] for PRA requirements and modify it if required. Thereby, PRA hopes to exploit the advantages of the existing design while improving performance specific to PRA operations.

This case study describes the manufacturer assessment and optimisation progress, assuming that it conducts a manual optimisation. Two design iterations occur, based upon the initial 3i design (named "3i-a" and "3i-b" from now on).

6.2 3i project

As part of the European Union Interreg 2Seas program[2], the 3i-project ("*Integrated Coastal Zone Management via Increased Situational Awareness through Innovations on Unmanned Aircraft Systems*") investigates improving maritime safety in the English Channel through cross-border collaboration between UK, French and Dutch industrial partners and academic institutions. This shall be achieved through implementing purpose-built UAS designed for the specific tasks of the stakeholders. These include Police authorities, harbour masters, environmental and border agencies. Therefore, tasks vary from search-and-rescue operations, ship tracking, emergency support, environmental monitoring to policing and border patrols.

The University of Southampton designed the 3i UAS airframe. The main challenge was to incorporate varying stakeholder requirements. Police forces need very fast UAS that can reach crime scenes and incidents quickly. Harbour masters require a long endurance to allow continuous harbour monitoring as well as on-board detection systems. Search-and-rescue operators ideally want a fast UAS with a long endurance. However, all stakeholders share two design requirements: First, certification and public acceptance require high operational reliability. Second, the harsh maritime environment (salty air, storms, fog, rain) ask for robust components with redundant backup systems where possible.

These considerations led to the 3i design as presented above in Section 5.2.3.4. Most prominently, it features two engines to reduce UAS losses due to engine failure. Moreover, its entire

[1] The 3i design will be publicly available since it is publicly funded.
[2] See http://www.interreg4c.eu/ for more details, accessed 03/08/2013.

structure and aerodynamic performance is designed prudently to provide a robust and reliable design. This design is used as the baseline design for PRA analysis.

6.3 Scenario

This section describes the reality of operations in the case study area covering the Port of Rotterdam (Section 6.3.1). Subsequently, possible integration of UAS is discussed based on PRA feedback and comments (Section 6.3.2). Some information in this chapter is based upon interviews with PRA employees and does not necessarily reflect the official view of the PRA.

6.3.1 The Port of Rotterdam

With about 34,000 sea-going vessels each year, the port of Rotterdam is the largest port in Europe[3]. The inshore harbour area stretches along parts of the Rhine river mouth for about 40 kilometres, from the centre of Rotterdam out to the artificially reclaimed land of Maasvlakte area (Figure 6-1). The harbour processes all kinds of goods, ranging from consumer goods and minerals to dangerous freight such as oil, gas and petrol. The harbour houses five oil refineries, 45 chemical plants, four gas power plants and one coal power plant. Therefore, any aeronautical operation above the harbour area is safety critical and must be assessed carefully.

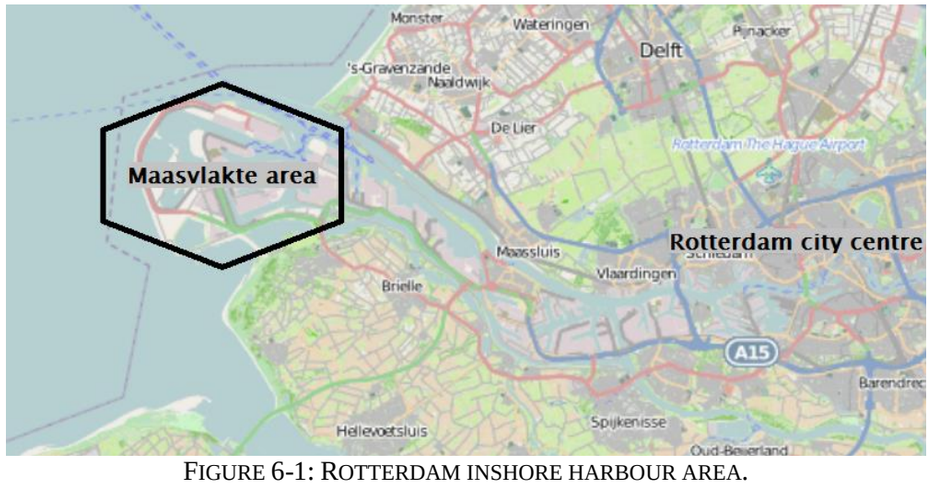


FIGURE 6-1: ROTTERDAM INSHORE HARBOUR AREA.

In addition to the inshore area, the port boundary includes offshore areas stretching 60 kilometres out at sea. These areas include several offshore anchor areas as shown in Appendix 15. Here, vessels wait for their terminal to become available if occupied.

[3] See http://www.2seas-uav.com/images/stories/downloads/Port%20of%20Rotterdam%20ppt.pptx, accessed 04/08/2013. You may need to register first on http://www.2seas-uav.com/.

The port is operated by the PRA, which is responsible for the safe conduct of harbour operations. PRA operates five patrol vessels (only one of which can go out to open sea[3]). They are used to conduct "inventories" and search-and-rescue missions. An "inventory" includes patrolling to every in-shore and offshore anchor position to investigate possible law violations such as waste dumping, smuggling or illegal anchoring. These operations are expensive, slow and ineffective (since they are conducted infrequently). Search-and-rescue missions occur irregularly. In 2012, PRA recorded 16 "significant" safety incidents leading to fatalities, severe injuries, major equipment damage or harbour closure (Smits et al. 2011).

6.3.2 UAS integration

PRA is part of the 3i research project in order to assess the usefulness of UAS for improving harbour safety through increased intelligence. The goal for PRA is "to get a real time operational picture in the remote sea areas and anchorages as well as in the port"[3]. Current equipment is not capable of delivering that vision due to harbour area size, tall structures (from a small patrol vessel, you cannot monitor the deck of an ocean liner) and human limitations. Autonomous UAS can improve the current state by supplying 24-hour bird-eye views of all harbour and offshore areas.

Beside regulatory issues (flying UAS above oil refineries and LNG tankers is not a trivial undertaking), PRA envisages using UAS for continuous patrolling and intermittent missions as required by harbour incidents. Continuous patrolling would be conducted above the inshore harbour area to obtain real-time imagery of harbour activities, possibly also at night. Intermittent missions include regular "inventories" of inshore and offshore anchorages as well as responding to any incidents occurring. The latter includes search-and-rescue operations as well as adding situational awareness to crisis staff during fires, crashes or oil spills. Moreover, troubled ships asking to find a refuge port could be investigated before harbouring.

6.4 Simulation setup

This Section describes how the UAS operations anticipated by PRA were translated into OSCAR concepts to build a realistic scenario.

During the 3i project flight tests, the UAS was assembled on a grass strip at the southern tip of the Maasvlakte reclaimed land area (see Figure 6-1). In this case study, assume that PRA will use this site as a permanent UAS base as it is far from any dangerous harbour area. Take-offs and landings can be conducted out to the sea, reducing the risk of external damage when the UAS crashes in these precarious flight phases.

The case study simulation runtime is set to one year because UAS missions would repeat every year. This is much shorter than the anticipated UAS life cycle but sufficient for the purpose of this case study. In reality, the entire life cycle should be modelled with varying mission setup and UAS performance.

The UAS must fulfil three main operational capabilities: Patrol the harbour continuously, conduct regular anchorage "inventories" and respond to any search-and-rescue emergency. The rest of this section describes each of the three missions in more detail (Sections 6.4.1-6.4.3). Subsequently, Section 6.4.4 compares mission setup inputs. Appendix 16 details the rationale behind setting the number of simulation replications to 500 for this case study.

6.4.1 Harbour patrol

A continuous harbour patrol is conducted every day of the year between 8am and 6pm[4]. The UAS patrols along a zigzag path covering most parts of the inshore harbour area (Figure 6-2). This mission has the lowest `priority` (see Table 6-2) because it is conducted much more frequently than the other missions and because it features no emergencies.

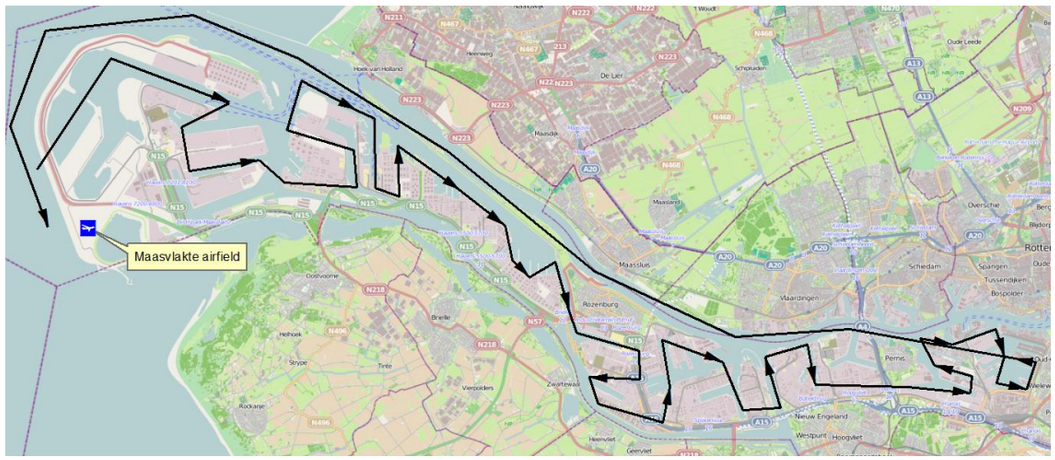


FIGURE 6-2: ROTTERDAM HARBOUR PATROL MISSION MAP. LOITER OCCURS AT THE ARROWHEADS.

The UAS launches from Maasvlakte at 8am, dashing at maximum speed to the start of the patrol path nearby. It follows the path at a constant 25 m/s and 100 m height. At 19 designated positions, the UAS loiters (at the same speed and height) for a specified duration to monitor activities in more detail. Upon completing one patrol round, it restarts the patrol. This is repeated until fuel dictates returning to base for a refuel (after which the patrol is continued) or until 6pm or until another mission with higher priority occurs.

[4] In order to cover daylight hours throughout the year at the given latitude.

6.4.2 Anchorage monitoring

Based upon the PRA "inventory" mission conducted by patrol vessels, two missions recreate "inventories", namely `AnchorageCensus` and `AnchorageEmergency`. Both share the same route, mission details and flight profiles as in Figure 6-3.

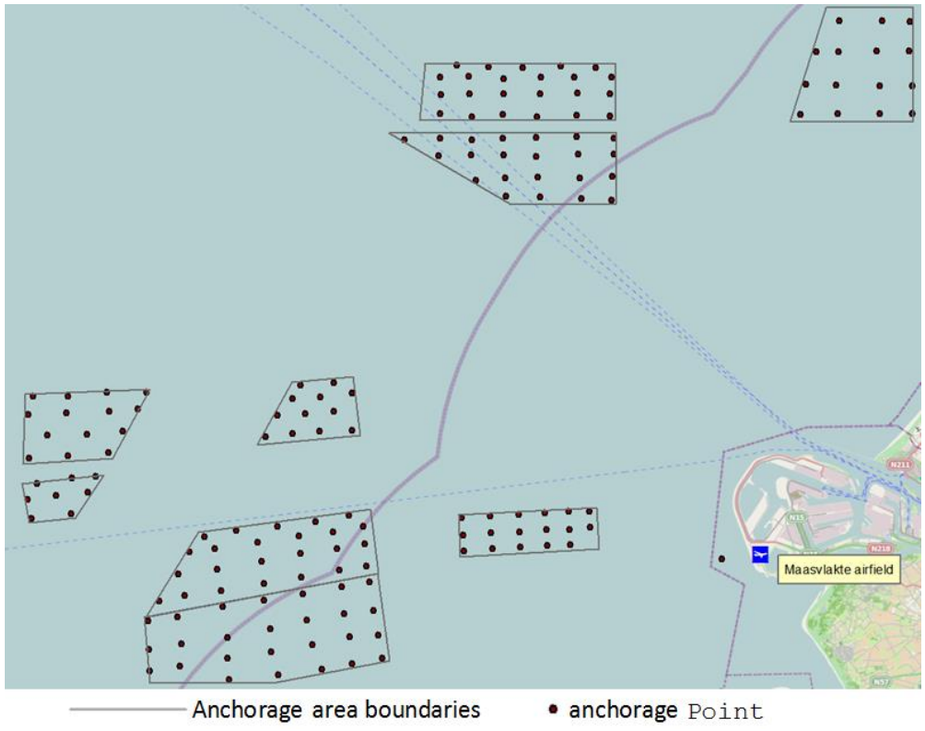


FIGURE 6-3: ANCHORAGE MISSIONS OVERVIEW MAP.

If an anchorage mission is scheduled, the UAS dashes to the initial cruise point near Maasvlakte at maximum speed and 100 m altitude. Subsequently, it visits 160 anchorage position POINTS in nine offshore anchorage areas. Each POINT refers to a ship of varying dimensions for identification. Flying and loitering occurs at maximum speed and 100 m altitude. The UAS loiters for one minute at each anchorage position to obtain adequate intelligence. The difference between `AnchorageEmergency` and `AnchorageCensus` is as follows: during `AnchorageEmergency` missions, an anchored ship has an emergency or PRA suspects a legal violation, requiring UAS investigation. Five specific ship positions with emergencies require 10 minutes of loitering, while the remaining 155 positions are visited as with `AnchorageCensus`, taking advantage of the fact the UAS is out at sea anyway. If the UAS detects fuel shortage, it interrupts for a refuel. Afterwards, it returns to the previous anchorage POINT.

Another difference between the two "inventory" missions is that `AnchorageCensus` occurs every two months while `AnchorageEmergency` is modelled every month, in line with real PRA security incidents (Smits et al. 2011). In reality, `AnchorageEmergency` would occur irregularly.

6.4.3 Search-and-Rescue

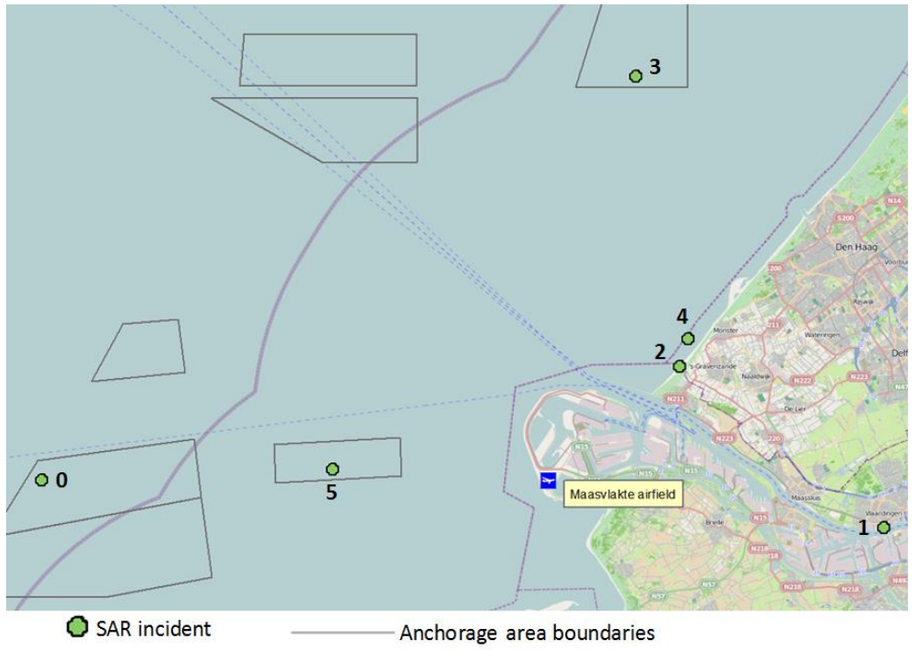


FIGURE 6-4: SEARCH-AND-RESCUE MISSION OVERVIEW MAP.

TABLE 6-1: SEARCH-AND-RESCUE MISSION DETAILS. "IPG" = INITIAL POSITION GUESS.

incident ID	**0**	**1**	**2**	**3**	**4**	**5**
Time	Jan 1st	Feb 13th	March 25th	April 17th	May 22nd	July 19th
Origin	IPG	IPG	IPG	IPG	IPG	IPG
Destination	ManOver-board	Suicide	Swimmer-mer-Missing	ManOver-board	SurferDrowning	ManOver-board
UponArrival	search	search	search	search	search	search
Type	500	500	1000	500	2000	1000
TargetHeight	0.2	0.2	0.2	0.2	0.2	0.2
TargetWidth	0.2	1.8	0.2	0.2	2	0.2
Detection-Criteria	6	3	6	6	6	6
Hover	1800	1800	1800	1800	2700	1800
Height	200	200	100	200	100	100
Speed	9999	9999	9999	9999	9999	9999

In a large harbour area, it is hard to avoid accidents and emergencies. PRA handles about 16 emergencies leading to serious injuries and death per year (Smits et al. 2011). Because conventional UAS support emergencies through searching only, a fraction of PRA emergencies feature here: PRA estimated that about six search emergencies occur each year. Figure 6-4 shows the geographical distribution of search incidents. Table 6-1 provides details on each incident.

`Destination` details the search-and-rescue incident type. Most incidents are men washed overboard and lost at sea. `Type` specifies the initial position uncertainty, indirectly indicating the time it takes to find the incident. Most targets are only small heads in the water, as specified in `TargetHeight` and `TargetWidth`. Only suicides and drowning surfers are larger because their whole body length (or board length) floating on the water dictates the characteristic width.

All search-and-rescue incidents feature the highest `priority` (Table 6-2). If the UAS is conducting any other mission while an incident requires search, the mission is cancelled and the UAS will return to BASE for refuel. Afterwards, it will dash out to the initial position guess and start searching.

6.4.4 Mission comparison

TABLE 6-2: ROTTERDAM HARBOUR MISSIONS COMPARISON.

Parameter	Harbour patrol	Anchorage census	anchorage emergency	Search-and-Rescue
`Vessel_IDs`	12	12	12	12
`Base`	Maasvlakte	Maasvlakte	Maasvlakte	Maasvlakte
`TrackFragmented`	FALSE	FALSE	FALSE	TRUE
`Destination`	Maasvlakte	Maasvlakte	Maasvlakte	Maasvlakte
`Time`	2014-01-01T08:00:00	2014-01-03T10:00:00	2014-01-07T22:00:00	2014-01-01T02:00:00
`Repetition`	36000X86400X365	0X5184000X6	0X2592000X12	0X0X0
`Priority`	0	1	2	3
`DashHeight`	100	100	100	100
`DashSpeed`	9999	9999	9999	9999
`ReturnHeight`	100	100	100	100
`ReturnSpeed`	9999	9999	9999	9999

Table 6-2 displays the relevant mission setup for all Rotterdam harbour missions. Note that only the search-and-rescue mission has `TrackFragmented=true`, indicating that its POINTS are separate incidents that should not be visited in one go (Appendix 2).

6.5 Initial design – 3i

This section presents and analyses the results obtained for the 3i UAS design (Section 6.5.1). Moreover, it provides an intermediate discussion of these initial results to justify the design changes for the first design iteration (Section 6.5.2).

6.5.1 Results and analysis

6.5.1.1 OSCAR outputs

There are 14 outputs provided by the OSCAR simulation for this case study. All figures below refer to the simulation timeframe of one year. Figure 6-5 depicts the flight performance outputs.

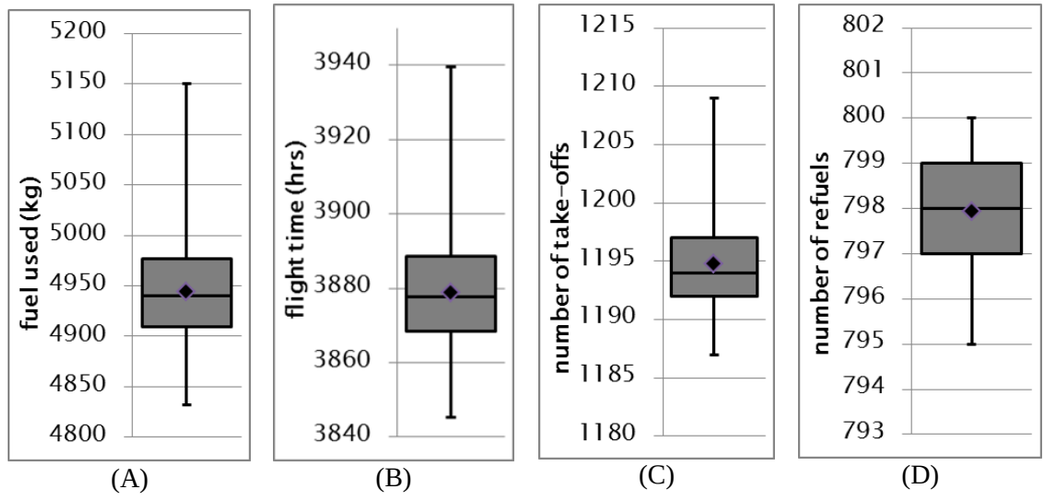
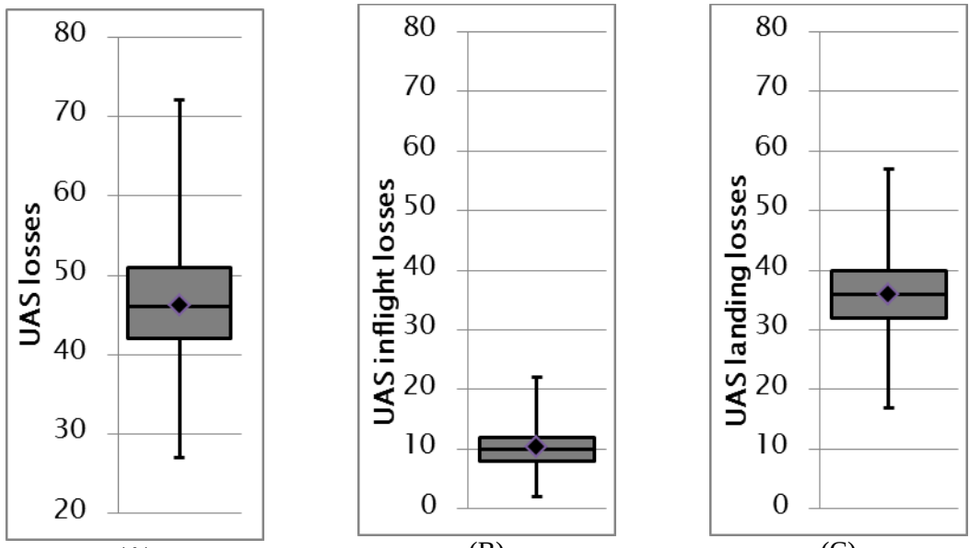


FIGURE 6-5: OSCAR FLIGHT PERFORMANCE OUTPUTS, SHOWING FUEL USED (A), FLIGHT TIME (B), NUMBER OF TAKE-OFFS (C) AND REFUELS (D).

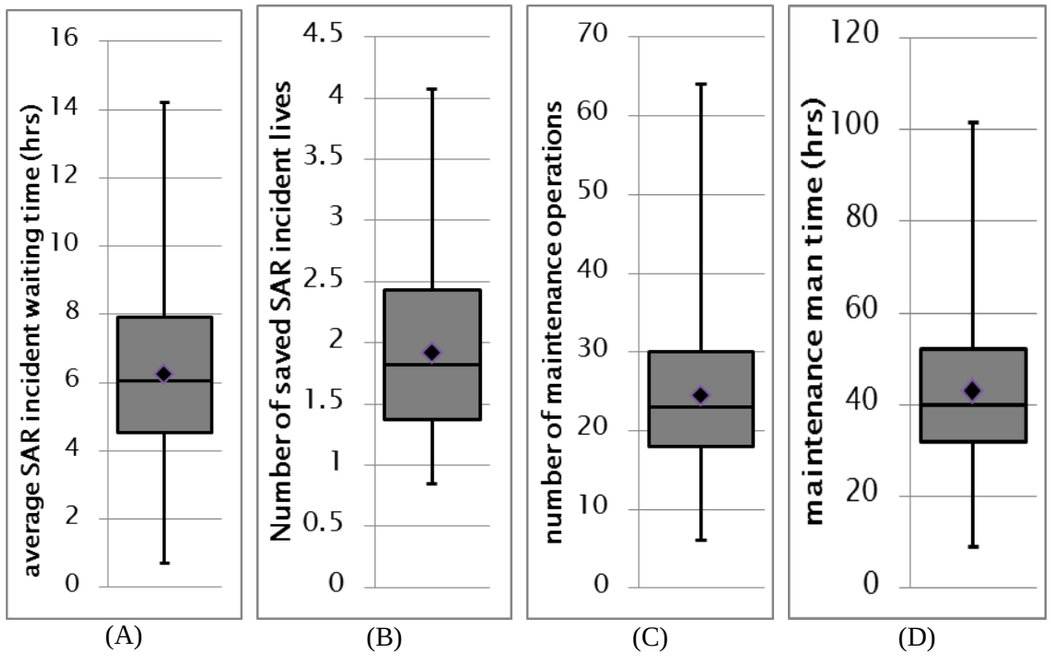
The UAS burns about 13.5 kg of petrol per day by flying for an average of 10.6 hours. The harbour patrol requires about 10 hours of daily flight duration while the remaining 0.6 hours divide into the other three missions. An average fuel burn of 1.27 kg/hr gives the 3i design an endurance of 4.5 hours. Therefore, the harbour patrol cannot be flown in one session but at least two refuels are required each day. This is reflected in the number of take-offs: the UAS launches (and lands) about 3.3 times each day out of which 2.2 times are due to refuelling.



(A) (B) (C)

FIGURE 6-6: UAS LOSSES (A) CONSIST OF INFLIGHT (B) AND LANDING (C) LOSSES.

Component fatigue and landing lead to UAS losses as shown in Figure 6-6. In total, one UAS is lost every eight days, on average. About one quarter of all losses is attributed to inflight losses caused by component fatigue. Three quarters are lost during landing, caused by the high kinetic energy due to weight and landing speed (see Section 5.4.4).



(A) (B) (C) (D)

FIGURE 6-7: SEARCH-AND-RESCUE INCIDENT METRICS (A AND B) AND MAINTENANCE METRICS (C AND D).

Search-and-rescue incident metrics (Figure 6-7 A & B) show that incidents have to wait for detection for more than six hours, on average. Less than one third of all incidents are detected alive due to the long search times involved. This rescue performance is much worse compared

to search-and-rescue operations including lifeboats, as examined in the previous case study (compare with Figure 5-14). In reality, the PRA would still need to use their fleet of patrol vessels to find incidents quicker. However, this case study is only interested in the relative performance gain by changing UAS design parameters so these baseline metrics are acceptable.

The maintenance metrics in Figure 6-7 (C & D) show that the UAS needs repair every two weeks, on average. Each maintenance operation takes about 1.7 hours.

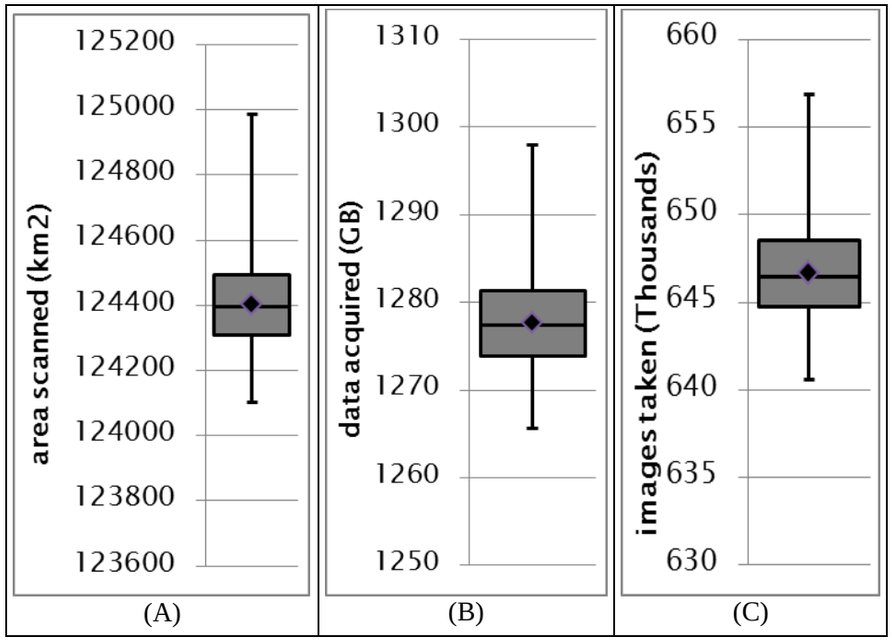


FIGURE 6-8: CAMERA PERFORMANCE OUTPUTS: SCANNED AREA (A), ACQUIRED DATA (B) AND IMAGES TAKEN (C).

Due to the long operating hours, the camera system collects large amounts of data (Figure 6-8). In total, it scans about 124,000 km^2, equivalent to an area of 340 km^2 each day (about half the size of greater London). Thereby, it collects about 3.5 GB of data each day, taking 1800 images, on average.

6.5.1.2 Costs

Using the "UAS costs" branch of the cost model described in Section 5.4.3.1, the total costs and its break-up into maintenance, operational, fixed and payload costs can be seen in Figure 6-9 (see Appendix 14 for cost parameter assumptions).

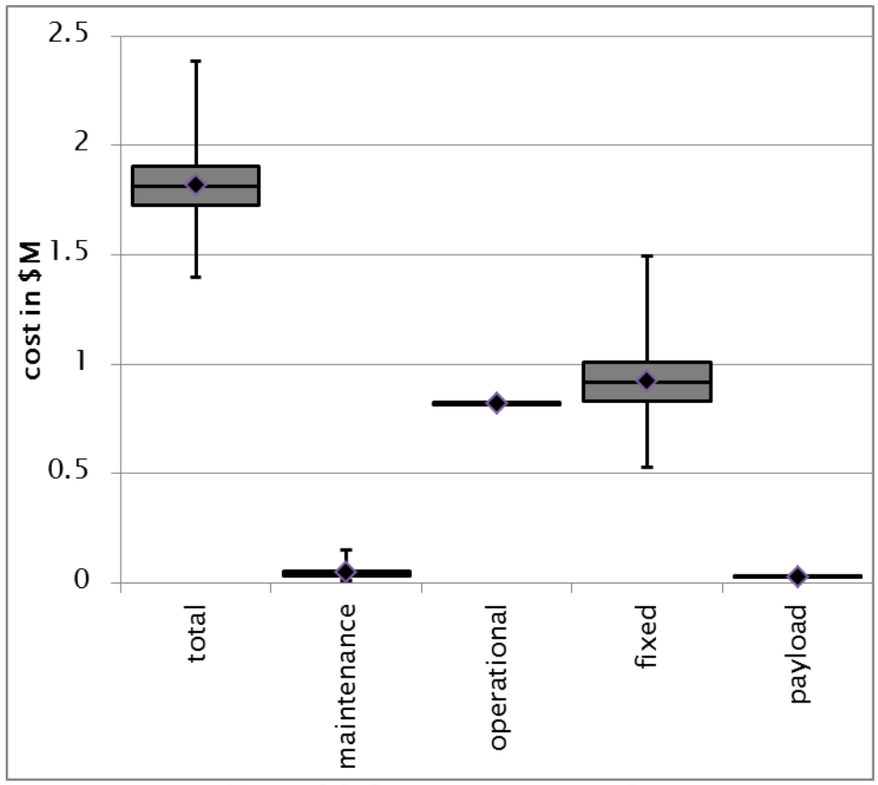


FIGURE 6-9: COST BREAKDOWN FOR 3I.

The total cost of adding the UAS to the PRA operations is expected to be 1.8 $M per year. Both maintenance and payload costs only contribute a small fraction. The majority of cost is caused by operational (0.82 $M per year) and fixed costs (0.92 $M per year). Here, fixed costs have a much higher spread, caused by the large spread of UAS losses (see Figure 6-6 A).

6.5.2 Intermediate discussion

Generally, the initial design is capable of fulfilling PRA mission requirements. However, there are two key points that merit optimisation, namely the number of launches and the number of losses.

6.5.2.1 Number of launches

The UAS flies 389 missions during the simulation period (daily harbour patrol, six anchorage censuses, twelve anchorage emergencies and six search-and-rescue operations). However, it launches three times as often (Figure 6-5 C), amassing almost refuel 800 launches. The UAS endurance is not enough to fly harbour patrol missions in one go. Every patrol is interrupted by two refuels, the second just 10 minutes before the end of the patrol. Every anchorage census (and emergency) mission requires four refuels because the UAS flies at top speed for very long distances. Last, some incidents are very hard to spot for the UAS, requiring one refuel for every

second incident (search incidents are omitted in the following discussion due to the low absolute number of launches).

Each refuel operation (stopping current SEGMENT → returning to base → refuel → take-off → dash out to current SEGMENT) takes between 25 and 60 minutes, depending on UAS geographical position while running out of fuel. Up to 1 kg of additional fuel is burnt during refuelling. Moreover, landing and taking off are the most risky part of UAS operation so reducing the number of launches is desirable. Last, more landings require a sturdier landing gear design, adding weight which, in turn, decreases the overall flight speed and increases fuel burn. Therefore, the first design iteration will feature an increased fuel tank capacity (over the 3i capacity of 5.8 kg) in order to increase aircraft endurance and reduce the number of launches. The current 3i design would require 21 kg of fuel to fly the anchorage missions without refuel (completion within 6 hours) while 16 kg of fuel would be required to finish the harbour patrol without refuel. However, a major redesign of the UAS would be required since take-off weight would double. Essentially, a completely new UAS design would emerge increasing development costs for the PRA. Therefore, a compromise increases the fuel tank size to 9 kg. Now, the current design would require 21.5 kg of fuel to fly the anchorage missions, taking 7 hours caused by two refuels. The harbour patrol would require one refuel only, burning 12.5 kg in total. Hence, the total number of refuels should reduce. Note that these estimates use the initial design but increasing the fuel tank size will change design performance as well. The effect will be analysed in Section 6.6.

6.5.2.2 Number of losses

The UAS is lost far too often for commercial use in a port environment (Figure 6-6). About one fifth of all losses are caused by component failure happening inflight, possibly over the harbour area or above a ship. The rest is caused by inferior landing performance, based on the high speed and weight upon landing (Section 5.4.4). However, the UAS is designed to be relatively heavy (to carry heavy payload, increase endurance, carry two engines, use components that are more robust and to reduce vibrations due to weather). Moreover, it is supposed to operate fast, making it difficult to design towards slow landing speed. Moreover, the relationship between landing losses and landing kinetic energy in Section 5.4.4 is based upon engineering judgement due to lack of data for UAS. Landing losses could be reduced by reviewing the underlying assumptions in Section 5.4.4 and by designing safer landing performance (through sturdier landing gear, flight personnel training, softening the landing grass strip, *etc.*). Otherwise, landing losses can be reduced through landing less often, *i.e.* by reducing the number of refuels (see above). The following discussion will focus on reducing inflight losses.

Inflight losses are caused by failing COMPONENTS. Figure 6-10 depicts the average number of inflight losses for all COMPONENTS causing any inflight loss.

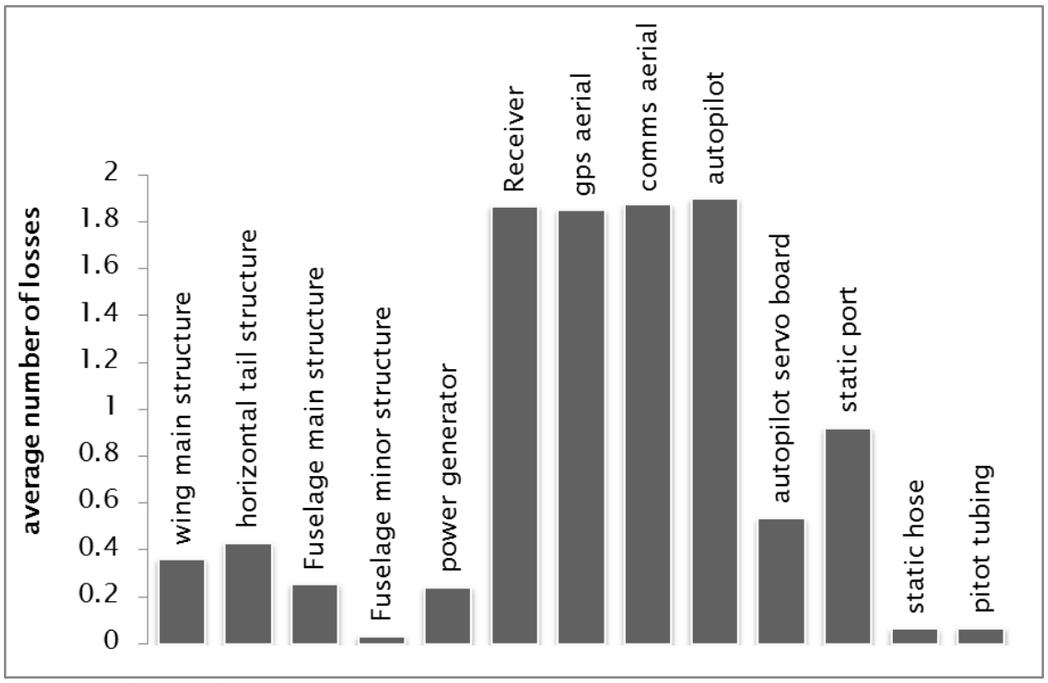


FIGURE 6-10: AVERAGE NUMBER OF INFLIGHT LOSSES FOR COMPONENTS.

Most prominently, communication equipment and the autopilot lead to over seven UAS losses each year. Sensor equipment such as the static port, static hose and pitot tubing also cause about one inflight loss per year, on average. Moreover, structural COMPONENTS like the wing and fuselage main structures and the horizontal tail structure are responsible for another loss.

Therefore, the first design iteration 3i-a will contain more robust COMPONENTS to reduce inflight failures. The communication COMPONENTS "Receiver", "gps aerial" and "comms aerial" as well as the "autopilot" and "autopilot servo board" will be duplicated on-board, effectively creating a backup system that can take over whenever a single COMPONENT fails. Additional weight is negligible and additional complexity is manageable as the technology is mature. Additional acquisition costs are not calculated in the cost model due to lack of data. Moreover, a duplicate "static port" will be installed on 3i-a, again assuming negligible weight addition and manageable complexity. The structural COMPONENTS cannot be duplicated for obvious reasons. Therefore, the `robustnessScalingFactor` of "wing main structure", "horizontal tail structure" and "fuselage main structure" will be increased from zero to one, effectively doubling the reliability of these COMPONENTS.

6.6 First design iteration – 3i-a

This section details the results from the first design iteration. As described above, the initial design 3i was altered in two ways: the fuel tank capacity was increased from 5.8 to 9 kg and a number of COMPONENTS were either duplicated or increased in reliability. In order to increase the fuel tank capacity, a UAS redesign was conducted using the conceptual design spread sheet developed for the DECODE research project (Section 4.9.2 and Ferraro et al. (2012)). Table 6-3 compares the design parameter changes between 3i and 3i-a.

TABLE 6-3: UAS DESIGN PARAMETERS COMPARISON.

Parameter	3i	3i-a
Wing area (m^2)	1.4	2
Maximum lift coefficient	1.574	1.894
Propeller diameter (m)	0.457	0.610
k1_a	0.043676	0.041786
k3_a	0.04207	0.04207
Installed power (W)	6800	6800
Propeller RPM	7000	5500
sfc_a (g/kWh)	1339	868
sfc_b (g/kWh)	-970	-649
sfc_c (g/kWh)	298	225
sfc_d (g/kWh)	0	0
maximum speed (m/s)	45	40
minimum speed (m/s)	14	14
dry weight (kg)	24.17	36
fuel weight	5.8	9
zeta_a	577	550
zeta_b	-6813	-6927
zeta_c	34459	37554
zeta_d	-87795	-102335
zeta_e	120137	149342
zeta_f	-83881	110588
zeta_g	23317	32236

Increasing the fuel tank size has a strong effect on the UAS design. Obviously, size (*i.e.* `wing area` and `propeller diameter`) and dry weight increase. As the new design is heavier, its

`maximum speed` reduces from 45 to 40 m/s. Note that the minimum speed stays constant, indicating that landing speeds generally stay constant as well. The fuel burn and drag coefficients confirm the increase in aircraft size: overall drag increases and, thereby, fuel consumption. Note that 3i-a has not been built in reality (unlike 3i, see Section 5.2.3.4).

6.6.1 Results and analysis

This section describes the operational performance achieved by 3i-a, presenting the OSCAR and cost outputs as before for 3i. However, the section starts with a mission performance comparison between 3i and 3i-a.

6.6.1.1 Mission performance comparison

To understand the OSCAR output changes better, it is useful to analyse the mission-specific performance first. Table 6-4 compares mission specific performance for 3i and 3i-a. It omits the search mission outputs, as their impact is negligible.

TABLE 6-4: MISSION PERFORMANCE COMPARISON FOR 3I AND 3I-A. ALL VALUES ARE ARITHMETIC MEANS.

Mission	Parameter	3i	3i-a
Harbour patrol	`flight time` (hrs)	10.3	10.4
	`fuel used` (kg)	12.0	12.0
	`number of refuels`	2	1
Anchorage census	`flight time` (hrs)	8.7	7.0
	`fuel used` (kg)	24.7	17.5
	`number of refuels`	4	1
Anchorage emergency	`flight time` (hrs)	9.3	8.3
	`fuel used` (kg)	27.2	20.0
	`number of refuels`	4	2

The harbour patrol `flight time` increases slightly due to the slower dash and return speeds. This is not offset by the flight time reduction due to less `number of refuels`. The `fuel used` remains constant because slower dash and return speeds offset the higher specific fuel consumption of 3i-a.

The anchorage census and emergency `flight time` reduce because the saving from less `number of refuels` does offset the increase due to slower flight speeds. This exemplifies the advantage of using geographical information during conceptual design: for harbour patrol, the combined effects of 3i-a increase the flight time, while for anchorage census and emergency the flight time reduces. Anchorage mission `fuel used` reduces significantly, owing to the fact that

less refuel operations require less additional flying. Moreover, 3i-a burns less fuel despite its higher specific fuel consumption because it flies at a slower maximum speed.

6.6.1.2 OSCAR outputs

Since the spread of OSCAR outputs is similar to the 3i design above, only (arithmetic) mean values are presented. For better comparison, the 3i mean values also appear in Table 6-5. The following analysis will examine each output in turn.

TABLE 6-5: OSCAR ARITHMETIC MEAN OUTPUTS COMPARISON BETWEEN 3I AND 3I-A. DATA BARS INDICATE RELATIVE MAGNITUDES.

Parameter	3i	3i-a
`Fuel used` (kg)	4944	4662
`Flight time` (hrs)	3879	3898
`Number of take-offs`	1195	786
`Number of refuels`	798	392
`Losses` (total)	46.3	59.9
`Losses` (inflight)	10.4	0.6
`Losses` (landing)	35.9	59.2
`Maintenance duration` (hrs)	43	37
`Maintenance operations`	24	34
`Scanned area` (km^2)	124,000	136,000
`Acquired data` (GB)	1278	1403
`Images taken`	647,000	710,000
SAR incident `waiting time` (hrs)	6.2	5.4
SAR incident `lives saved`	1.91	2.10

The total `fuel burn` has decreased by almost 300 kg. The explanation is a sum of several operational factors: First, 3i-a has higher specific fuel consumption due to its additional weight and drag. Second, dash and return SEGMENTS as well as anchorage and search-and-rescue missions use slower speeds as 3i-a's maximum speed has reduced from 45 to 40 m/s. Third, considerably fewer refuel operations are necessary during the anchorage missions, reducing the total fuel burn for these missions.

The `flight time` increases slightly by 19 hours. Again, OSCAR combines several operational factors to arrive at this result: First, all dash and return SEGMENTS as well as Anchorage and search-and-rescue missions are flown at slower speeds. This is offset by the flight time reduction caused by less refuel operations. Last, the harbour patrol duration increases slightly be-

cause slower dash and return speeds are not offset by the `flight time` reduction from fewer refuels.

The `number of take-offs` reduces by one third. This is caused by the reduction in refuel operations required for all missions. The strongest effect originates from the harbour patrol reduction as it is conducted much more frequently than the other missions are. However, the `number of refuels` still has about one refuel each day, adding to the chance of losing UAS during take-off or landing.

The (total) `number of losses` increases by almost one third. The increased reliability and the duplication of critical COMPONENTS has reduced the `number of inflight losses` to 0.6 losses per year. However, the additional weight of 3i-a combined with the same landing speed as 3i increased the kinetic energy upon landing, leading to many more landing losses, now more than one every week.

The total `maintenance duration` has decreased by 14 % while the `number of maintenance operations` has increased by more than 40 %. Figure 6-11 depicts the number of COMPONENT failures by COMPONENT type.

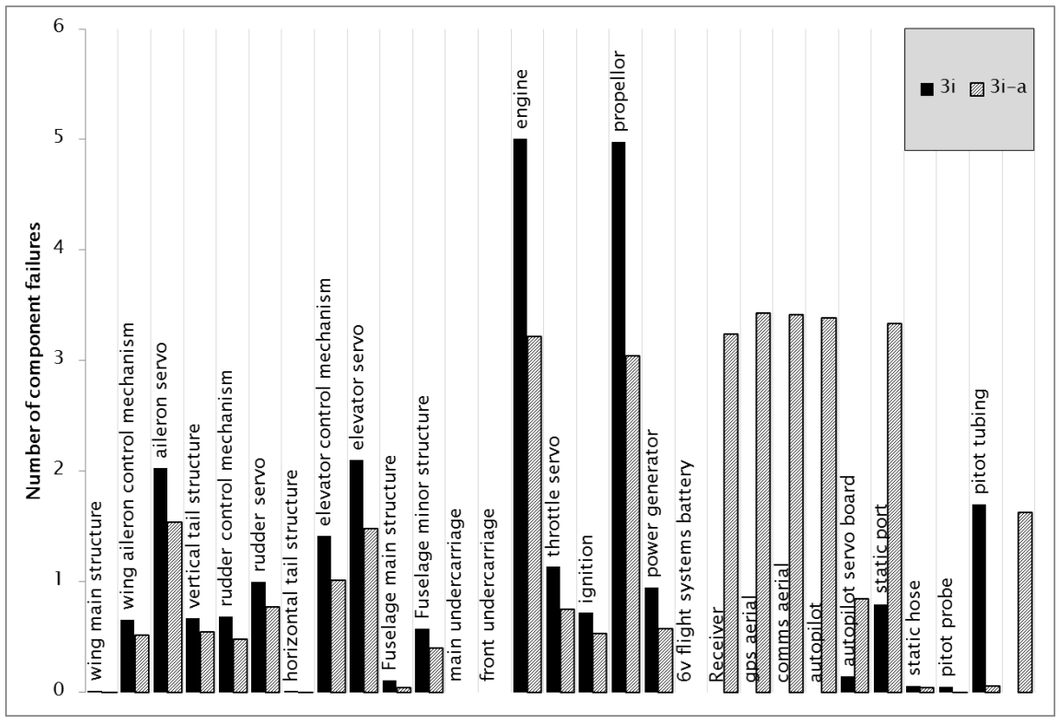


FIGURE 6-11: COMPONENT FAILURE COMPARISON BETWEEN 3I AND 3I-A. LISTS ONLY FAILURES THAT DO NOT LEAD TO AIRFRAME LOSS.

The `maintenance duration` decreases because COMPONENTS with long repair times (engine, throttle servo, ignition and power servo) cause fewer failures in the 3i-a design (check COMPONENT repair times in Appendix 17). The reason for fewer failures is operational: the higher number of landing crashes causes operators to repurchase new UAS more often. These

feature new COMPONENTS that are less likely to fail. The additional 3i-a `number of maintenance operations` is caused by those COMPONENTS that were duplicated in the new design (Receiver, GPS aerial, comms aerial, autopilot & servo plug, static port). However, repair times are shorter for those COMPONENTS, leading to an overall reduction in `maintenance duration`. Despite less COMPONENT failures for most COMPONENTS (due to more landing losses), the duplicated 3i-a COMPONENTS require maintenance operations that were not necessary with 3i. Previously, these COMPONENTS failed and led to airframe loss in any case because no redundant COMPONENTS took over. With 3i-a, redundant COMPONENTS do take over and the airframe requires maintenance upon landing, increasing the `number of maintenance operations`.

The payload performance indicators `area scanned`, `data acquired` and `images taken` all show a modest increase in the amount of collected data. First, this is caused by the additional flight time. Second, 3i-a can collect more data because it spends less time refuelling (as no images are taken during flight to and from the base).

3i-a is also performing better concerning search-and-rescue incidents despite the lower 3i-a maximum speed. The `incident waiting time` reduces by 14 % while the number of saved lives increases accordingly. This is achieved because 3i-a can search the respective area for longer without returning home for refuels. This increases the chance of spotting the incident more than the penalty paid due to the slower speed of 3i-a. Again, consolidating operational effects like these is best achieved through simulation.

6.6.1.3 Costs

Figure 6-12 presents the overall cost of using 3i-a and a cost breakdown. Compared to the 3i costs (Figure 6-9), the overall cost increases by about $ 250,000. Maintenance costs increase by about $ 17,800, operational costs decrease by about $ 44,400, fixed costs rise by almost $ 272,000 and payload costs increase by about $ 3,000.

The rise of maintenance costs originates from the increased `number of maintenance operations`, adding costs for parts, building rent, *etc.* (Appendix 14). This is not offset by the decrease in maintenance costs due to lower `maintenance duration`, leading to fewer maintenance labour-hours to be paid.

The decrease in operational costs originates from fewer `number of take-offs`, leading to less launch costs. Moreover, less fuel is used. The increase in `flight time` costs does not offset these benefits.

The strongest increase in cost is the amount of fixed costs, originating from the higher total `number of losses` of 3i-a. This is also the cause for the total rise in costs for 3i-a compared to 3i. Moreover, fixed costs feature the highest spread of data caused by the high price per lost UAS.

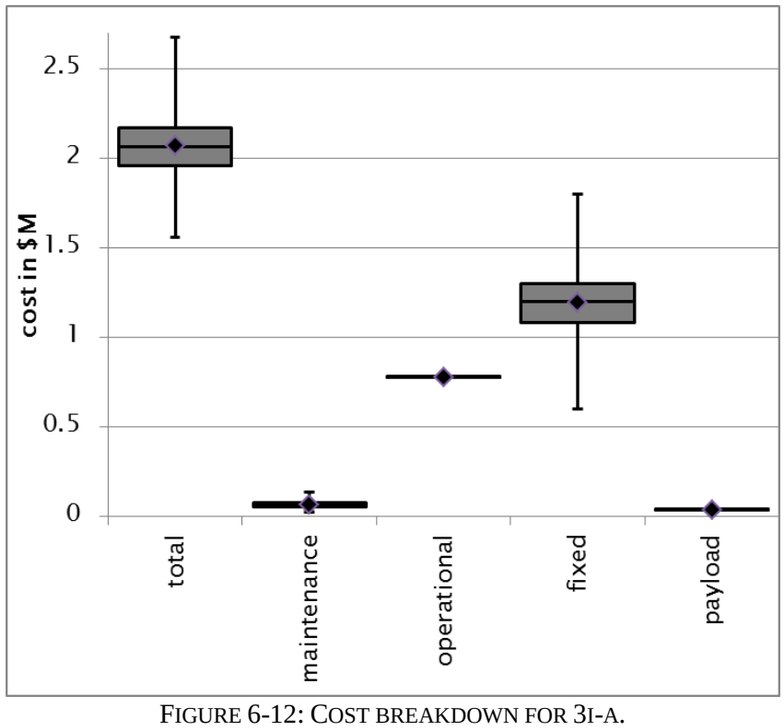


FIGURE 6-12: COST BREAKDOWN FOR 3I-A.

6.6.2 Intermediate discussion

The 3i-a design offers some operational advantages over 3i bought at higher total costs. Increasing the fuel tank capacity reduced the number of refuel operations drastically, increasing not only search-and-rescue performance. The other missions also benefit from the longer endurance as 3i-a can spend more time collecting data. The operational penalties for the redesign (slower maximum speed and higher specific fuel consumption) were offset by the added benefit of conducting missions with less refuels. Here, OSCAR demonstrated its unique benefit for conceptual design: it consolidates various (often conflicting) operational factors taking into account specific mission and airframe characteristics. For example, it is not possible with conventional conceptual design phase tools to predict if 3i-a will burn more fuel during the harbour patrol mission and less during an anchorage mission (Table 6-4). The mixture of fewer refuels, geographic mission details (distance to base when running out of fuel, *etc.*) and the airframe performance led to varying results for each mission.

Another example is the decrease in maintenance duration combined with an increase in maintenance operations. It is easy to predict that duplicating certain COMPONENTS will reduce the airframe losses. However, the operational effect of reducing losses is that COMPONENTS age more, requiring more maintenance or causing airframe losses again. Moreover, fewer missions are cancelled due to airframe loss, increasing overall performance. In this design iteration, the airframe performance was altered together with COMPONENT reliability and redundancy. This led to a heavier design featuring more losses upon landing. OSCAR consolidating this impact

against the fact that increased endurance led to fewer landings due to fewer refuels, finding that the number of landing losses still increases.

The overall performance of 3i-a improved, yet too many airframes were lost during landing, leading to unnecessary cost increases. The next design iteration 3i-b must aim to reduce landing losses, in this case by reducing the `number of take-offs` as much as possible. Conventional design could opt to increase the fuel tank capacity further until no refuels were required. Estimates require a fuel tank capacity increase of nearly 100 % to 18 kg. However, airframe size would increase further, making the UAS impractical for PRA application. The landing site would need rework to accommodate such a large design. Certification would become much harder as the possible damage of a heavier UAS increases non-linearly with weight. Moreover, cost (and thereby acquisition price) would rise manifold.

The OSCAR framework allows a value-driven design approach that takes a *holistic* view at design and operations. The PRA specified missions such that they gather as much intelligence as possible, as fast as possible and in real-time. Therefore, anchorage missions as well as all dash and return SEGMENTS were flown at maximum speed by 3i and 3i-a. Only the harbour patrol used an arbitrary speed of 30 m/s to reduce airframe losses over the harbour area.

With OSCAR, it is possible to optimise airframe design but also *operations*. Therefore, the second design iteration 3i-b will explore how changing the mission definition can reduce overall cost while keeping mission performance acceptable. For this iteration, the airframe itself will remain as in 3i-a. The number of inflight losses was acceptable due to duplicated and more reliable critical COMPONENTS. The mission performance improved over 3i so there is no reason to change the design any further in this case study. However, in order to reduce the number of landing losses, operations will be altered to reduce the number of required refuels further. For this, all mission SEGMENTS will be flown at V_{Rmax}, the maximum range speed. Depending on airframe weight, this speed varies between 19 and 21 m/s for 3i-a, essentially halving the maximum speed used during dash, return and all anchorage missions. Moreover, the very long anchorage emergency mission will feature a 30-second loiter at each non-emergency anchorage, instead of the previous 60 seconds. Last, the harbour patrol will be conducted for 9 nine hours instead of 10 each day (*i.e.* from 8am to 5pm).

These changes are expected to lead to the following results: search-and-rescue incidents will be spotted later but this might be offset by fewer refuels, possibly leading to spotting incidents earlier. Anchorage missions and the harbour patrol will take longer but manage without refuels. The total data collected will reduce as the airframe spends less time in the air. However, OSCAR quantifies the consolidated effects of less refuels, lower mission performance (less data) and lower total costs. The next Section will present and discuss the results of this second design iteration 3i-b.

6.7 Second design iteration – 3i-b

6.7.1 Results and analysis

This section presents and discusses the results from the second design iteration, called 3i-b. As before, flight performance is compared for each mission before discussing the OSCAR outputs. Subsequently, cost outputs are presented for 3i-b.

6.7.1.1 Mission performance comparison

Table 6-6 extends the mission performance indicators of Table 6-4 by adding the performance of the second design iteration 3i-b.

TABLE 6-6: MISSION PERFORMANCE INDICATORS FOR 3I, 3I-A AND 3I-B.

Mission	Parameter	3i	3i-a	3i-b
Harbour patrol	flight time (hrs)	10.3	10.4	9.2
	fuel used (kg)	12.0	12.0	8.5
	number of refuels	2	1	0
Anchorage census	flight time (hrs)	8.7	7.0	9.2
	fuel used (kg)	24.7	17.5	8.5
	number of refuels	4	1	0
Anchorage emergency	flight time (hrs)	9.3	8.3	8.2
	fuel used (kg)	27.2	20.0	7.9
	number of refuels	4	2	0

For harbour patrol missions, flying at V_{Rmax} reduces the number of refuels to zero. This a combined effect of flying at the more economical speed but also by reducing the patrol duration from 10 to 9 hours. Overall, the flight time reduces by 12 % over 3i-a. Spending less time in the air at a more fuel-efficient speed cuts fuel burn by almost 30 %.

Anchorage census and emergency missions require no more refuel operations. For the census missions, the slower flight speeds increase the flight time above the initial 3i design flight time. For the emergency missions, flight time reduces slightly because loiter times above non-emergency anchorage positions was halved. This is not offset by the flight time increase due to lower speeds. For both census and emergency anchorage missions, the fuel used reduces dramatically by 50 % and 60 %, respectively. This is caused by flying at V_{Rmax} during all flight SEGMENTS. Moreover, the reduction in refuel operations reduces the number of SEGMENTS flown at maximum speed, further reducing the fuel used.

6.7.1.2 OSCAR outputs

Table 6-7 shows all arithmetic mean OSCAR outputs for all three designs. Output distribution is similar for all designs. Therefore, refer to output distribution plots for 3i in Section 6.5 for reference. Each output will be analysed below.

TABLE 6-7: OSCAR ARITHMETIC MEAN OUTPUTS COMPARISON BETWEEN 3I, 3I-A AND 3I-B DESIGNS. DATA BARS INDICATE RELATIVE MAGNITUDES.

Parameter	3i	3i-a	3i-b
Fuel used (kg)	4944	4662	3011
Flight time (hrs)	3879	3898	3512
Number of take-offs	1195	786	395
Number of refuels	798	392	2
Losses (total)	46.3	59.9	17.0
Losses (inflight)	10.4	0.6	1.7
Losses (landing)	35.9	59.2	15.3
Maintenance duration (hrs)	43	37	153
Maintenance operations	24	34	104
Scanned area (km^2)	124,000	136,000	103000
Acquired data (GB)	1278	1403	1042
Images taken	647,000	710,000	527000
SAR incident waiting time (hrs)	6.2	5.4	7.2
SAR incident lives saved	1.91	2.10	1.66

As expected, the `fuel used` reduced drastically by about 35 %, caused by more economic fuel burn at V_{Rmax} and because the aircraft `flight time` reduced by about 300 hours. This reduction is a combination of two operational effects: First, the aircraft flies slower during all missions, thereby increasing `flight time`; second, the `number of refuels` reduced to almost none, reducing the overall `flight time`. The latter effect is stronger, reducing overall `flight time`. The `number of take-offs` reduced by 391 over 3i-a, equal to the amount of the `number of refuels` saved (subject to rounding error). This reduction is a direct consequence of flying at V_{Rmax} instead of V_{max}. The aircraft endurance has increased beyond the duration of any individual mission. The two refuels were caused by search-and-rescue incidents. Previously, search-and-rescue incidents required more refuel operations, therefore performance improved overall for 3i-b.

The number of `total losses` reduced by over 70 % to one loss every three weeks. `In-flight losses` occur very rarely at a rate of 1.7 losses per year. However, this increased almost threefold compared to 3i-a owing to two counter-acting effects: First, the lower number of `flight hours` led to fewer `inflight losses` as COMPONENTS were stressed less. Second,

the lower number of landing losses caused fewer new aircraft purchases. Hence, aircraft aged more, increasing the likelihood of inflight crashes.

The number of `landing losses` decreased by almost 75 %. This is a direct effect of fewer landings as the `number of refuels` decreased to almost zero. 3i-b is lost upon landing about once every 24 days. In reality, this number would still be far too high for reasonable operations, as is reflected in the high costs of the system (see below). Therefore, the relationship between landing KE and the likelihood of landing losses needs critical review. Real landing loss data of similar aircraft is not available but empirical data of civil aircraft could be scaled down. Alternatively, expert interviews could lead to a refined relation.

The `number of maintenance operations` and the `maintenance duration` increased by 67 and 76 %, respectively, compared to 3i-a. The aircraft requires two maintenance operations per week, each lasting 88 minutes, on average. This increase is caused by the reduced number of `total losses` leading to aircraft aging more, on average. The older an aircraft gets, the more likely it is to feature a COMPONENT failure, either leading to an `inflight loss` or requiring a maintenance operation. Therefore, the success of 3i-b in reducing fuel use and producing fewer losses is offset through more maintenance operations as in Figure 6-13.

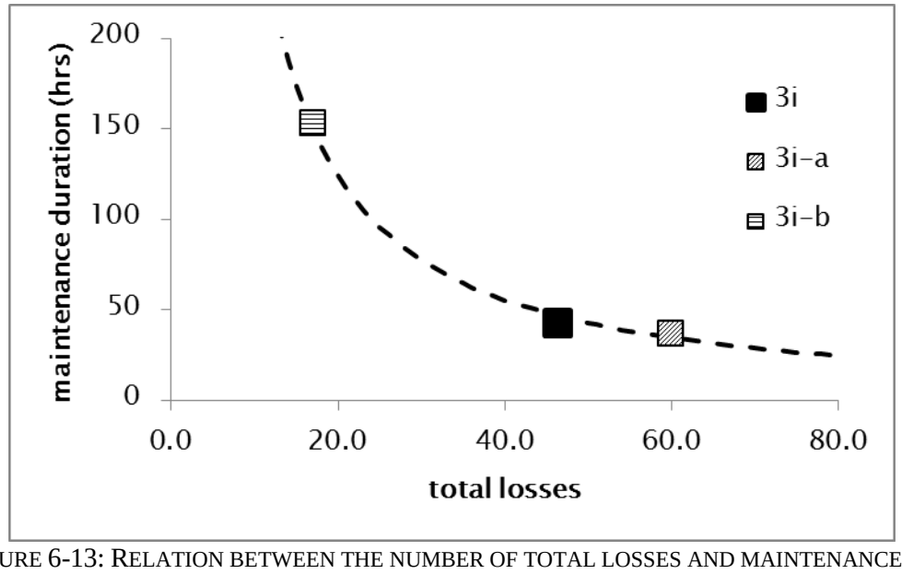


FIGURE 6-13: RELATION BETWEEN THE NUMBER OF TOTAL LOSSES AND MAINTENANCE REQUIREMENTS. TREND LINE: $y=4065x^{-1.165}$ WITH $R^2=0.95$.

If a design causes fewer losses, it requires more and more maintenance. Alternatively, designs with very low demand on maintenance feature more losses.

As seen in Table 6-7, the payload performance (`scanned area`, `acquired data` and `images taken`) decreases by about 25 % compared to 3i-a. This is a result of flying at the reduced V_{Rmax} during the harbour patrol missions. As the aircraft flies slower compared to 3i-a, it covers less distance in the same time, thereby taking fewer images. Moreover, the harbour patrol duration decreased from 10 to 9 hours, further reducing the possible payload output. Note

that the output for the anchorage and search-and-rescue missions stayed constant because the mission duration is not specified as with harbour patrol. Instead, the aircraft covers the same distance at a slower speed, taking the same amount of images, overall.

The search-and-rescue incident outputs `incident waiting time` and `lives saved` show a performance decrease for 3i-b over 3i-a. Because incidents are found 33 % later, on average, the aircraft spots 21 % less incidents alive. This was expected because flight speed is critical in finding incidents quick and alive and 3i-b flies significantly slower than 3i-a. On the other hand, 3i-b required fewer refuel operations during search-and-rescue missions, thereby spotting incidents earlier. In total, however, OSCAR found the former effect to be stronger, reducing overall search-and-rescue performance. In reality, operators should fly 3i-b at maximum speed for search operations.

6.7.1.3 Costs

Figure 6-14 depicts the total cost and its breakdown into cost components for 3i-b.

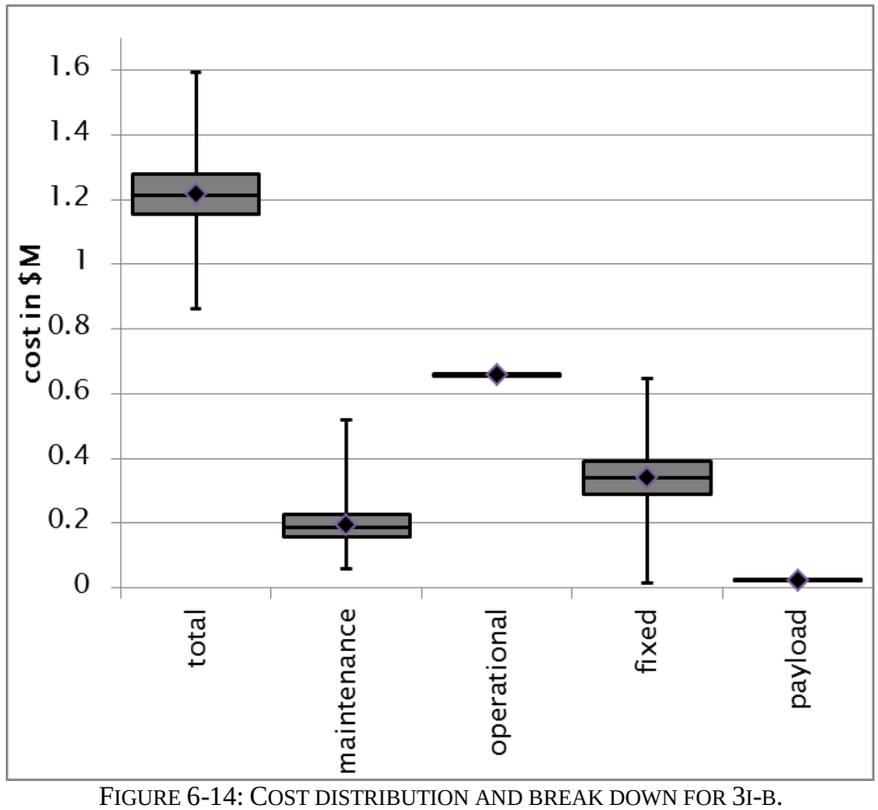


FIGURE 6-14: COST DISTRIBUTION AND BREAK DOWN FOR 3I-B.

The total cost is dominated by operational costs making up about half of the total costs. Maintenance costs total about $ 200,000 while fixed costs require about $ 340,000. Both feature relatively large whiskers but small IQR boxes. This indicates that the standard deviation is acceptable but the data has some outliers, caused by the high aircraft acquisition cost (for fixed

costs) and by the varying `number of maintenance operations` (for maintenance costs). Payload cost is trivial compared to the other cost drivers.

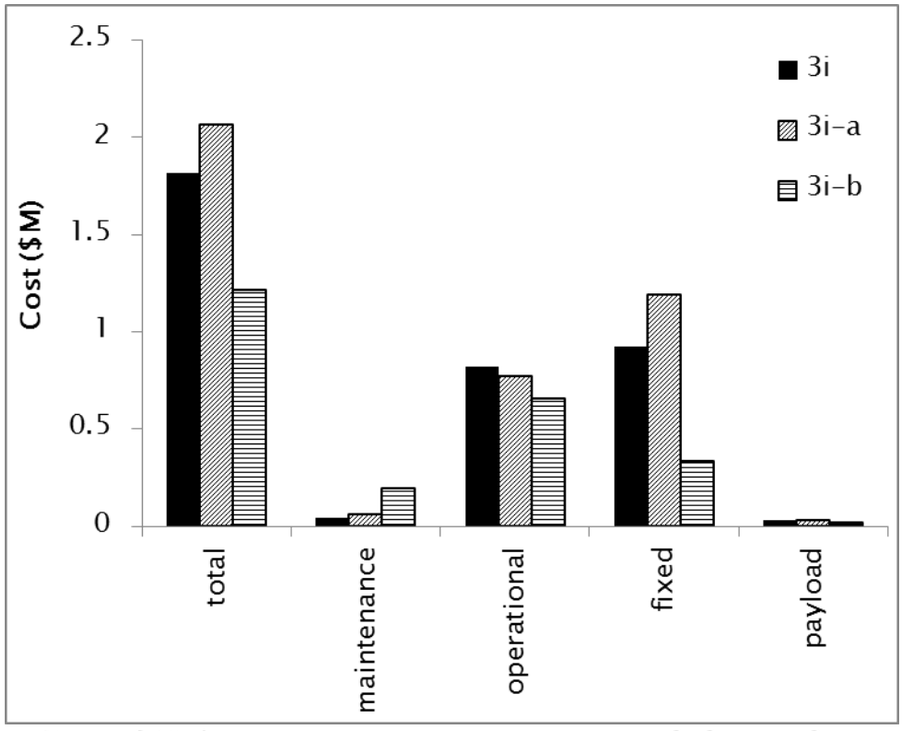


FIGURE 6-15: ARITHMETIC MEAN COST BREAKDOWN FOR 3I, 3I-A AND 3I-B.

Figure 6-15 compares the mean costs for all three designs (combining arithmetic means of Figure 6-9, Figure 6-12 and Figure 6-14). Design 3i-b is the cheapest design overall, mainly caused by a strong reduction in fixed costs. This is based upon the reduced number of `total losses`, requiring fewer aircraft purchases.

Operational costs reduce for 3i-a and 3i-b mainly because the `number of take-offs` reduces. Moreover, the `fuel used` and `flight time` reduces (except for a small increase in `flight time` for 3i-a), keeping operational costs down.

Maintenance costs increase for each design iteration because the `number of maintenance operations` increases. A slight reduction in `maintenance duration` for 3i-a over 3i does not offset the overall maintenance cost increase.

Payload costs are directly proportional to payload performance and are negligible compared to the other cost drivers. The reason is the choice of payload cost parameters (see Appendix 14) showing that payload cost does not influence total cost strongly.

6.8 Discussion

This section discusses the results of 3i-b and the overall optimisation process shown above. Moreover, it introduces two trade-off studies derived from the manual optimisation above.

Design 3i-b achieved performance improvements through changing the mission specification (and not the design itself). The `number of refuels` reduced to almost zero, reducing the number of `landing losses` and, thereby, fixed costs. Moreover, the design operated more efficiently as it wasted less time returning home for refuel and flying back out again. Combined with flying at a more fuel-economic speed, this reduced the `fuel used` drastically.

However, the performance improvements came at a price: First, more `maintenance operations` occurred, adding to costs. Second, the payload performance reduced due to reduced speed. Third, the search-and-rescue performance reduced as it took longer to find incidents.

From a cost-driven perspective, design 3i-b is still the preferred choice as it has the lowest overall cost. However, real engineering cannot optimise for one factor (lowest cost) while not penalising other performance measures. Therefore, it is useful to analyse the loss in performance versus the gain in cost using Pareto charts. In a real optimisation, many more designs would help create a smooth Pareto front. In this case study, however, the low number of iterations provide indications only. The following charts aim to present what kind of discussion and insights are possible using OSCAR during conceptual design.

6.8.1 Cost versus payload performance

As most PRA mission goals are to gather intelligence, it is interesting to investigate the relation between cost and payload performance. Figure 6-16 depicts the additional cost per extra GB of data. Designers can expect to pay an extra $ 2,341 per additional GB of intelligence data per year (or save the same amount if they are willing to sacrifice one GB of data). In reality, the relationship is non-linear, as the cost for more data rises to infinity at some point. However, the low sample number of three design iterations only allows a limited view upon this relationship.

Similar relationships can be produced for the cost per additional image taken and the cost per scanned additional square kilometre.

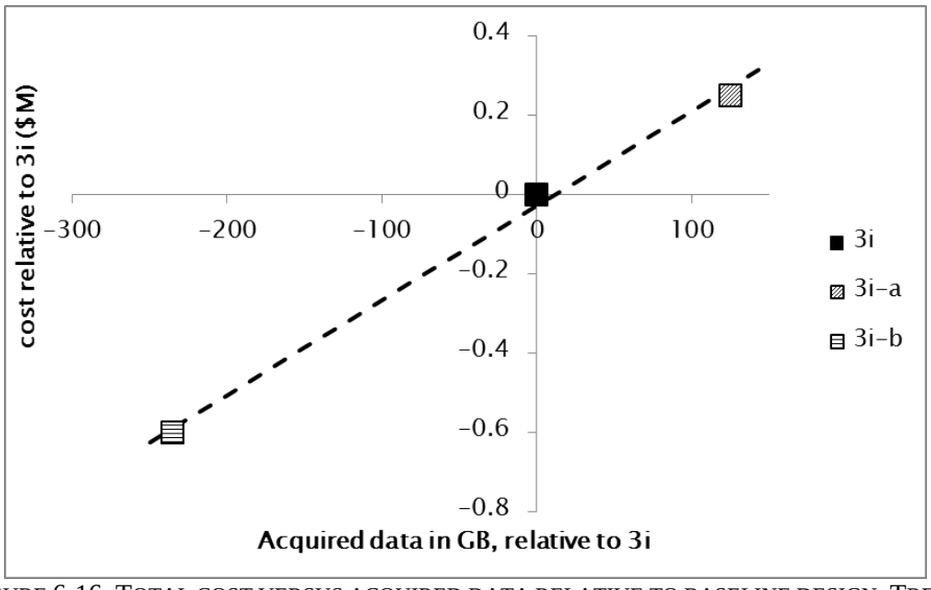


FIGURE 6-16: TOTAL COST VERSUS ACQUIRED DATA RELATIVE TO BASELINE DESIGN. TREND LINE: $Y=2384.5X-29241$ WITH $R^2=0.99$

6.8.2 Cost versus lives saved

Consider the case where PRA is primarily interested in search-and-rescue performance. Figure 6-17 depicts the relationship between total cost and the number of saved lives, both relative to the baseline design 3i.

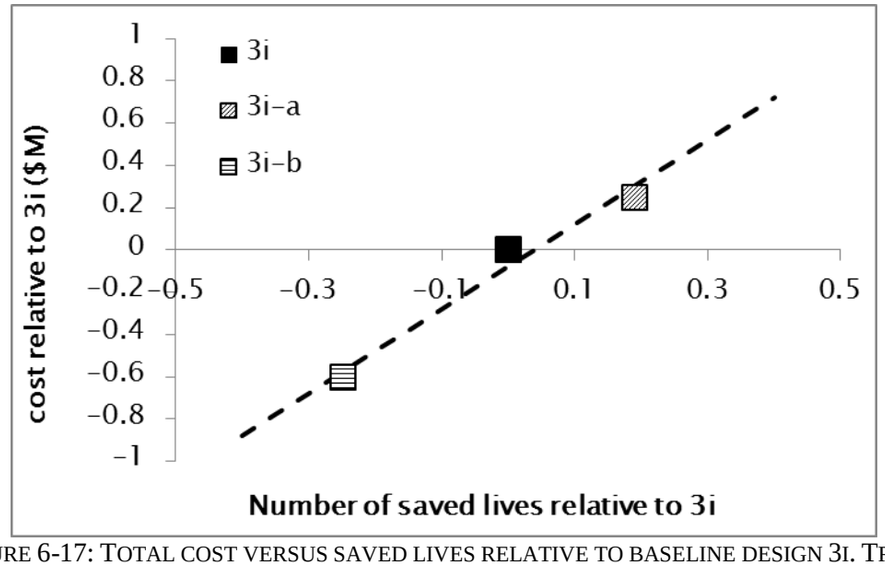


FIGURE 6-17: TOTAL COST VERSUS SAVED LIVES RELATIVE TO BASELINE DESIGN 3I. TREND LINE: $Y=2*10^6X-78974$ WITH $R^2=0.97$

This correlation allows designers and managers to assess how much more money they are willing to spend to save additional lives. In this case, saving one more live costs *additional* $M 1.9 compared to the baseline design. Interestingly, this value corresponds to the order of magnitude for the value of a statistical life (Viscusi & Aldy 2002).

6.9 Summary

This chapter presented the use of OSCAR for conceptual design optimisation. As an automated design optimisation was beyond the scope of this work, a three-step manual optimisation process was followed to indicate procedures and insights.

The first design iteration changed obvious design flaws that inhibited effective operation of the baseline design. However, by increasing the fuel capacity, duplicating or strengthening vulnerable COMPONENTS, some improvements were achieved while other metrics changed unexpectedly. For example, inflight losses almost vanished, repairs occurred less often and search-and-rescue and payload performance improved. However, the redesign increased the aircraft size and weight, increasing landing losses. Overall, more aircraft were lost and total costs increased despite opposite expectations.

The second design iteration alleviated these problems somewhat. The optimisation changed operational parameters instead of design parameters, making full use of OSCAR capabilities. Reducing the general flight speed from V_{max} to V_{Rmax} and changing the harbour patrol duration decreased the number of refuel operations to almost zero. This reduced the landing losses without the drawback of an even larger fuel tank. However, this time the improvements in landing losses and total costs were offset by increased repair demands as aircraft aged more. Moreover, payload and search-and-rescue performance suffered as they depend upon flight speeds directly.

As engineering cannot achieve an optimal design, it was shown how designers and customers could use OSCAR for trade-off studies. By linking OSCAR with a cost model, performance gains linked directly to cost increases. This helps decision makers and designers to focus upon their requirements and helps choosing the right design candidate.

7. CONCLUSION

This chapter concludes the thesis and summarises the main findings. Section 7.1 re-focuses on the context for doing the study and how it justifies the research question. Section 7.2 returns to the study objective while Section 7.3 discusses problems encountered during the OSCAR framework development. Section 7.4 does the same for the OSCAR simulation. Subsequently, Section 7.5 details the boundaries and limitations of the research, commenting on applicability in industry. Section 7.6 recommends follow-on work while Section 7.7 summaries the main conclusions.

7.1 Context

Design of complex aeronautical systems consists of trade-offs based on informed decision-making processes. In general, trade-offs are not straightforward but feature multi-dimensional multi-scale considerations. Moreover, they intersect and overlap each other. Therefore, aeronautical design is inherently complex: A small design change can result in large consequences for life-cycle operations and overall product performance. Design complexity grows at a very fast pace, exacerbating cost overruns, delivery delays and defects. There is a need for a decision-support design methodology that operates at multiple levels of abstraction throughout the entire design process. This helps to manage design complexity by providing clear design decision support. Value-driven design aims to provide this overarching signpost capability for the design of large complex systems. It supports objective design decisions and trade-offs by using a holistic life-cycle perspective that includes analysis of product missions and operations. Therefore, value-driven design requires information from a mission-modelling tool that simulates the entire life cycle of a variety of designs in a comparable, comprehensive and rational way. Value-

driven design and mission modelling should be applied whenever decisions and trade-offs occur. Some argue that the most critical decisions occur during the very early conceptual design phase. However, current conceptual design mission modelling does not apply the level of fidelity required by value-driven design analysis. Typically, current models combine parameterised building blocks in order to simulate characteristic mission profiles for one product instance alone. The process is slow, manual and neglects a lot of information and data already available during conceptual design. Current models do not include dynamic life-cycle operational changes and lack support for operational uncertainties. Moreover, they neglect spatial details, losing potentially relevant information for designers. Instead, the characteristic mission is extrapolated over a typical life cycle, yielding trivial analysis of design decision impact. Not least, mission definitions are seen as a design constraint rather than a design parameter.

Therefore, the research question asked if it is possible to create an improved life-cycle mission-modelling framework for aeronautical vessels that supports decision-making and trade-offs for value-driven design.

7.2 Objectives

This thesis identified the need for an improved conceptual design phase mission-modelling application. Accounting for the large variety of aeronautical design products, the objective was to create a generic framework that is widely applicable. Moreover, the framework should be comprehensible to allow easy integration and successful user adoption. The level of realism aims to be applicable for conceptual design phase requirements. Essentially, the goal was to be as realistic as possible taking into account the limited data, labour and computing resources during conceptual design. Last, the framework was supposed to be modular to ease adoption but also to help comprehensibility, flexibility and genericity. In the end, the framework would enable specifying entire product life-cycle operations with higher fidelity than currently possible during conceptual design. Applying the framework would provide unique design information for decision support and optimisation that can be exploited by value-driven design methods.

In order to demonstrate the capabilities of the framework practically, another objective of this thesis was to present a working prototype, *i.e.* a simulation model. It should act as an inspiration as to how to develop or expand existing tools. The simulation should incorporate and implement most of the framework features. It would include vessel and scenario features as well as automated geographical modelling capabilities. Moreover, it would be modular to allow adding plug-ins of custom fidelity.

In order to test the simulation, the thesis objective was to apply the simulation in practise. Two case studies presented real-life applications of aircraft design using the simulation. One

case study demonstrated its decision-support capabilities while the other highlighted its optimisation abilities. The goal was to describe and discuss how the tool would be used during aircraft value-driven design as well as demonstrate how the tool can benefit designers by creating new knowledge and insights.

7.3 Framework

This section concludes the *status quo* for the OSCAR framework and the issues and challenges faced during development. The OSCAR framework (Chapter 3) tries to achieve the objectives set above in order to answer the research question. It divides mission simulation into scenario-related and vessel-related information. Scenarios consist of generic building blocks (SEGMENTS, TRACKS and MISSIONS) that make up entire life cycles, if desired. All information relates to geographical modelling to exploit spatial information. VESSELS are characterised by parameters such that they can represent a large variety of moving objects like aircraft, cars, trains, humans or submarines. VESSELS consume energy at specified rates. An open plug-in approach allows specifying custom propulsion algorithms. An object-oriented approach defines COMPONENTS for VESSELS. COMPONENTS deteriorate during VESSEL operations with stochastic uncertainty. Last, VESSELS can carry inactive or active payload to compute VESSEL operational performance. Through these steps, it has been possible to provide a largely generic framework for mission simulation that enables to simulate large fleets of products in any area and detail required over long life cycles. However, several issues and challenges occurred during framework development.

The modularisation of an aeronautical life cycle into MISSIONS, TRACKS and SEGMENTS focussed on realism and genericity rather than usability. SEGMENTS enable very fine and detailed mission scenarios. Nonetheless, initial setup of life cycles requires a large amount of work. Reusing mission objects through a database alleviates these problems. However, the framework (and the simulation, for that matter) do not support pre-defined standard mission building blocks like aircraft holding patterns, airport arrival procedures or airways. In reality, these data exist in (commercial) databases. Depending on design context, such data could be implemented, reducing genericity but increasing usability.

The proposed VESSEL dynamics arising from the VESSEL parameters are very simplistic. Essentially, VESSELS are simple point masses moving through 2D space. They lack acceleration and deceleration behaviour. VESSELS can only climb or descend in discrete steps, neglecting smooth altitude changes. Moreover, directional changes occur instantaneously, lacking any turn performance. This is acceptable for VESSELS that do not perform such manoeuvres as core char-

acteristics of their life cycle, *i.e.* VESSELS that operate at level altitudes and constant speeds most of the time.

VESSEL payload definition focussed on active payload. However, many aeronautical missions carry inactive payload only. It was difficult to unify both payload types in a simple parameter set. Therefore, different parameters were suggested for both types. Moreover, the framework assumes constant payload for a VESSEL throughout its life cycle. This assumption does not hold in reality because often, each operation features different payload (*e.g.* airliner loading different passengers on each flight).

VESSEL COMPONENTS deteriorate based on a user-defined weibull function. There is no support for other distributions or non-stochastic behaviour. COMPONENTS cannot interact with each other directly. Essentially, there is no knowledge of how COMPONENTS link to others and how failure at one COMPONENT affects all other COMPONENTS. However, there is a simple model for redundant COMPONENTS of the same type that can take over work of broken COMPONENTS. Moreover, the OSCAR framework does not take into account definition of scheduled maintenance as commonly conducted with most aeronautical products.

7.4 Simulation

This section summarises the *status quo* for the OSCAR simulation (Chapter 4) and the issues and challenges faced during model development. The OSCAR simulation includes the OSCAR framework building blocks for SEGMENTS, TRACKS and MISSIONS. It applies an agent-based approach for simulating generic moving VESSELS. VESSELS interact with a geographical environment and with each other. The simulation feeds from (and to) an extensive database specified in the OSCAR framework. VESSELS apply either a generic propulsion module or custom plug-ins, as demonstrated in the case studies. They can carry electro-optical sensor payload. Payload performance depends on payload parameters, VESSEL definition and life cycle performance. Several challenges occurred during development of the OSCAR simulation.

One problem was that SEGMENT-related geographical data should be stored within Geographical Information System shapefiles. However, it was not possible to read this data within the simulation. Therefore, only spatial SEGMENT data is stored in shapefiles while SEGMENT-related data (altitude, speed, *etc.*) is stored within the external database. Care is required to combine a shapefile SEGMENT with the correct SEGMENT data in the database.

Another database problem was structural: Typically, users would store many TRACKS, MISSIONS and VESSELS in the database. Although possible, the database easily fills with hundreds or even thousands of tables with similar names. Several database files could organise data better

but would complicate post-simulation data processing because SQL queries cannot access several database files simultaneously.

In general, it was very difficult to combine geographical shapefiles and agent-based modelling within the simulation tool AnyLogic. Complex custom algorithms read shapefiles and convert them into AnyLogic "polylines" automatically. This solution is generic and unique. However, it is not optimal because agents do not interact with geographical features directly. Moreover, runtime increases due the conversion algorithms. Not least, users must create geographical shapes in the correct sequence within a Geographical Information System program to ensure VESSELS move along shapefile paths or points correctly. These drawbacks offer numerous possibilities for human error. Object-oriented geographical modelling can simplify some of these problems (Section 7.6).

The OSCAR simulation payload model focuses on electro-optical sensors while neglecting many other types of active payload (microphones, Geiger tubes, *etc.*). Moreover, inactive payload is not supported at all because the database structure does not contain the relevant parameters suggested by the OSCAR framework.

During model development, validation was a recurring problem, as with any simulation model. One can never formally prove that a model or part of it are working correctly (Sterman 2000). Therefore, traditional validation techniques were applied throughout. Unit tests and walkthroughs were conducted after each model change. Expert opinion was obtained for specific operational procedures. External modules (propulsion, geographical modelling, *etc.*) were tested using extreme case scenarios.

Moreover, it was difficult to choose ideal model fidelity levels for the various parts of the model. Some required more detail (*e.g.* payload performance) while other parts were modelled more coarse (*e.g.* climb/descent). The decision for module fidelity was based on engineering judgement, conceptual design phase requirements and development iterations. To give an example, consider the update frequency of the fuel burn calculation (Section 4.9): Initially, a very high frequency was chosen (every virtual second) because correct fuel burn measurement was considered important. However, runtimes exceeded acceptable limits and the frequency was reduced to every virtual minute. Comparison runs revealed acceptable precision loss and strong increase in computing performance.

Issues occurred not only during model development but during model application as well. While applying the OSCAR simulation as part of the DECODE project (Section 5.2.2), it was difficult to distinguish two very similar designs by their outputs. Essentially, output noise and small design change impacts merged. One of the reasons was that the virtual life cycles were very short (1-3 years) and included relatively few missions (less than 200). There are three solutions to such problems:

- Inflate the design changes arbitrarily, *i.e.* compare designs that are less similar.

- Extend the life cycles arbitrarily.
- Increase the number of replications to reduce noise.

Each option has disadvantages that must be taken into account when facing such problems. However, the OSCAR simulation delivered one insight that could not be obtained otherwise: The impact of small design changes can be neglected for operational scenarios similar to DE-CODE. This is not always the case, as was shown in the other case study in Chapter 6.

Another problem faced during model application relates to output analysis: Once designers observe different output performance for different designs, it can be difficult to explain the change. Designers usually change one design parameter such as wingspan or landing speed. However, the resulting "balanced" aircraft design varies in numerous operational parameters such as maximum speed, fuel capacity, *etc*. This can cause a complex operational "reaction" leading to step changes in outputs that were not anticipated by a (supposedly) small design parameter change (see Section 3.2.4.1 for an example). Moreover, agent interaction can cause unexpected outputs. In any case, result analysis often includes time-consuming model re-runs with human supervision in a "step-by-step" manner.

7.5 Limitations

Both the OSCAR framework and simulation are suitable for a limited area of application. Although developed with genericity in mind, OSCAR is not applicable for space and military designs. For space applications, the 2D assumption of the Geographical Information System environment does not hold. For military design, agent interactions are not sophisticated enough to model battlefields.

OSCAR can model a large variety of moving objects but only if they can control their own motion. Consider scientific balloons: They drift with prevalent winds and change altitude based on hanging weights. This could be modelled by dictating direction but there is no dynamic wind impact during model run. Moreover, vessel operations should predominantly consist of constant altitude and constant speed segments to reduce output error.

Moreover, OSCAR does not support a weather module in its current state. Therefore, operations and vessels that strongly depend on weather should be included cautiously. To some degree, any moving vessel operation depends on weather: Bicycles are predominantly used in good weather; rockets launch only below certain wind speeds; commercial airliners choose flight routes based on large weather structures; *etc*. However, most weather-dependent operations can be defined during conceptual design applying statistical averages and engineering judgement. Future work could include weather impact by adding SEGMENT characteristics such as temperature, wind or visibility.

Another OSCAR limitation for aeronautical conceptual design is the lack of take-off field length consideration. This parameter is crucial for conceptual aircraft design because it dictates landing speed and thereby many other design parameters. Currently, designers must ensure manually that their design can land at any expected airport during its life cycle.

Another limitation of OSCAR is its implicit work overhead during conceptual design. Sourcing data and creating the mission as well as vessel definitions can be time-consuming and expensive. It is beyond the scope of this thesis to quantify new cost and additional benefit that arise from using OSCAR. This thesis argues that improved decision rationales and more precise product values have the *potential* to outweigh the overhead and produce overall benefit.

Not least, this thesis argues that an explicit mission simulation triggers thinking about alternative design ideas or modes of operations that are difficult to conceive with simplified mission models. The value of such insights is hard to quantify as it depends on the specific application.

7.6 Future work

This section provides guidance and ideas for future developments based on the OSCAR framework and simulation.

The VESSEL model should include simple dynamics for altitude changes, directional changes and speed changes. Currently, VESSELS only change these characteristics incremental. However, a simple, computationally cheap and flexible model could improve conceptual design phase results without additional workload. Moreover, the existing incremental approach could remain for non-critical VESSELS of the operational scenario (like the lifeboats in Chapter 5). A very simple dynamics model could feature additional VESSEL parameters such as `turnRate` (in $^{o}/s$), `climbRate` (in m/s) and `accelerationRate` (in m/s^2).

Another improvement regarding VESSELS concerns replacement policies: What happens if a VESSEL breaks during operation? A VESSEL can break but remain fixable for future application or it can break beyond repair. Replacement policies should be definable for both cases separately. Policies could include:

- No replacement: The VESSEL is not replaced ever.
- Immediate replacement: The VESSEL is replaced at the instant of failure with an identical copy. This is the current (unrealistic) option for the OSCAR simulation. This policy is useful if unlimited resources are of interest.
- Earliest replacement: The VESSEL is replaced as soon as practically possible. BASES could feature a stock of VESSELS from which replacements are drawn (using a new BASE parameter `vesselsInStock`). If the stock is empty, the new VESSEL parameter `timeToReplace` defines how long it takes to refill the stock.

In order to provide better feedback for VESSELS carrying payload other than electro-optical sensors, the payload module requires a more generic modelling approach. Inactive payload can be included easily by adding the parameters `payloadItems` and `payloadWeight` as suggested in Section 3.3.6.1. Moreover, payload should become object-oriented: Each payload item should become a separate object such that a VESSEL can carry any number of any payload type (*e.g.* passengers *and* a camera). This would overcome the problem that current VESSELS are bound to one item of payload for their entire life cycle. Object-oriented payload could be changed for each MISSION, TRACK and even SEGMENT, if desired. If payload is assigned SEGMENT-wise, specialised operations with payload drop become possible (*e.g.* parachute flights, UAS dropping equipment or supplies, trains collecting and releasing passengers, *etc.*).

VESSEL COMPONENTS modelling can be improved in several ways. First, fixed scheduled maintenance events should be defined for individual COMPONENTS and entire VESSELS based on different conditions such as operating hours, age, operating cycles, *etc.* Moreover, a simple model of COMPONENT relations and interactions could improve the current simplistic approach of stressing remaining redundant COMPONENTS more. In this relational model, each COMPONENT could define the effect of its failure on any other COMPONENT specifically. If an engine fails, this puts additional stress on the remaining engines but also (to a lesser degree) on the rudder to keep lateral stability. Another COMPONENT modelling improvement concerns failure probability: The OSCAR simulation supports Weibull failure distributions only as it can model a wide range of data. However, the weibull distribution cannot model lognormal data often found in maintenance repair times. Users should be able to decide and apply standard statistical distributions independently.

If a COMPONENT fails, it can cause immediate VESSEL loss in the OSCAR simulation. However, if the VESSEL remains operable, the COMPONENT should define if the current VESSEL operation will be cancelled or not. Using a new COMPONENT parameter `missionCritical`, a VESSEL would return to its BASE as soon as possible for COMPONENT repair. This option would be useful for COMPONENTS that are critical to the mission: If the camera sensor fails on a mapping mission, there is no point in continuing; if a passenger becomes very sick, the mission must be cancelled; if known safety-critical COMPONENTS like the batteries on the Boeing 787 fail[1], the VESSEL should land as soon as possible.

Most conceptual design tools feature take-off and landing modules (compare Section 2.5) while this aspect is largely neglected in the OSCAR framework and simulation. One of the critical design parameters during conceptual aeronautical design is airfield runway length (except for vertical take-off and landing VESSELS). The shortest runway for the expected operations dic-

[1] After a number of battery failures on the 787, it was monitored closely during operations. See http://www.boeing.com/787-media-resource/docs/787-battery-certification.pdf, accessed 30/12/2013.

tates landing performance, *i.e.* landing speed. Therefore, BASES could feature the additional parameter `shortestFieldLength` while a simple take-off and landing module based on existing empirical relations could ensure that BASE field lengths are within limits at any time during the life cycle. However, such modules would not be applicable to VTOL (Vertical Take-off and Landing) aircraft and other VESSELS such as trains, cars or ships. Moreover, the hazardous issue of crosswind landings could be modelled by extending the current simple landing module.

Currently, OSCAR does not support weather and geographical influences such as dust. However, connecting the SEGMENT spatial data with weather databases could take into account weather effects such as average or stochastic wind speeds, temperatures, visibility or precipitation. A simple weather module would amend flight performance accordingly, possibly including effects of wind on flight speed, effects of temperature on air density, effects of visibility on operations, *etc.* Moreover, local effects such as gusts or winds could influence performance and deterioration of specific COMPONENTS (*e.g.* sand grinds engine blades).

The current geographical modelling capability does not utilise the full potential of spatial analysis. Connecting to online databases, mission definition could be simplified for commercial airliners by applying existing airways and airport departure & arrival procedures. Other operations could automatically apply routes following roads (for cars or road-monitoring aircraft), shipping routes or train tracks. Another major improvement geographical modelling would be object-oriented geographical modelling: Here, the idea is to model each spatial SEGMENT (POINT or PATH) as a specific, addressable object within AnyLogic (or any other tool). This would simplify assigning spatial details such as weather conditions. Moreover, VESSEL agents would interact with an object instead of following a passive background drawing. This would enable new functionalities for future developments: A road SEGMENT could hold a maximum number of cars; an airway could report on the number of aircraft that used it; a train track could feature overhead contact wire or not, *etc.*

7.7 Summary

This thesis introduced the OSCAR framework and simulation, a set of ideas and tools for improved conceptual design mission modelling for value-driven design. It is the first tool of its kind that takes into account mission geography specifically. Moreover, it enables designers to incorporate additional useful information into conceptual mission modelling compared to current tools. This includes simulating a whole fleet of vessels, competitors and other vessels that might interact with the design vessel. Vessels consist of components that deteriorate by operations and affect operational performance. Missions are modelled “explicitly” through defining any operation, life-cycle variation and spatial as well as fleet detail in a specific and unambigu-

ous way. This omits parameterised mission building blocks that are prevalent in current conceptual mission models. By applying an object-oriented approach, additional workload for designers is minimised. Most of the required information is available during the conceptual design phase already. The initial overhead in work (defining missions, vessels and components) is required only once. Subsequent analysis can draw from existing objects for alteration.

Technically, the OSCAR simulation is the first application of agent-based modelling specifically considering geography for conceptual aeronautical design. Although complicated, the advantages of geographical modelling for conceptual design have the potential to outweigh its technical drawbacks. Through geography, operational scenarios can cause complex interactions between vessels and their environment that cannot be predicted analytically. These can steer design decisions and improve overall value.

The OSCAR framework and simulation outputs enable detailed cost and benefit analysis producing quantitative value comparisons between different designs and design optimisation. Therefore, the initial hypothesis and research question can be answered as follows:

> *The OSCAR framework enables explicit and generic life-cycle mission modelling for aeronautical vessels supporting design decisions, trade-offs and optimisation for value-driven design.*

APPENDICES

Appendix 1: Segment parameters

This appendix details the parameters required to define a SEGMENT object.

SEGMENT parameter	Description
`Time`	An integer value (in seconds) indicating when this SEGMENT should be started relative to the start TIME of the current TRACK (Section 3.2.3.2). Hence, the initial SEGMENT of a TRACK always has `Time=0`. `UponArrival` and `Hover` can override the `Time` value. If a SEGMENT is finished earlier than the `Time` value of the subsequent SEGMENT, the vessel will loiter at the end point of the SEGMENT until the subsequent SEGMENT should be started. If a SEGMENT is finished after the `Time` value of the subsequent SEGMENT, the vessel will proceed after the `Loiter` duration of the current SEGMENT. Note that all SEGMENTS of a TRACK can have `Time=0`.
`Origin`	A string indicating a name of a POINT or the start point for a PATH. This string distinguishes POINTS and PATHS from each other beyond spatial difference. However, `Origin` is not required and has no direct operational function in OSCAR.
`Destination`	A string indicating a name of a POINT or the end point for a PATH. This string distinguishes POINTS and PATHS from each other. However, `Destination` is not required and has no direct operational function in OSCAR. Note that `Destination` and `Origin` may be identical. This is good practice for POINTS as their `Origin` and `Destination` are equal by definition.
`UponArrival`	A keyword string indicating what should be done upon arriving at this POINT or end point of this PATH. Permitted keywords are `home`, `stay`, `next` and `search`. The keyword `home` suggests that the vessel should proceed to its `Destination` airport upon arriving until it is time to proceed to the next SEGMENT of this TRACK (only applicable if not the last SEGMENT of a TRACK). This can enforce refuelling during a TRACK. The keyword `stay` requires the vessel to loiter at the current POINT or PATH end point until it is time to proceed to the next SEGMENT or the subsequent TRACK. The keyword `next` informs the vessel to proceed to the next SEGMENT disregarding the subsequent `Time`-value. This enables "sweeping" through a TRACK without interruption. The keyword `search` triggers the build-in search-and-rescue capability (Section 4.11). `UponArrival` overrides the `Time`-characteristic: if the vessel should go `home`, it will not judge if it has enough time to fly to its BASE before starting the subsequent SEGMENT. Therefore, careless data entry may cause vessels to accumulate significant delays.
`Type`	A string or integer that can be used to cause specific vessel behaviour for the current SEGMENT. Developers can define specific keywords and link those to purpose-build features. Currently, one specific behaviour is included: If `UponArrival=search` and `Type` contains an integer number, the custom search-and-rescue module will use the value as an input to search-and-rescue incident position uncertainty (Section 4.11). In any other case, this

	column is has no functionality.
TargetHeight	Based on the "target" concept defined in Section 3.3.6 and 4.10, this double value defines the target height in metres. Note that this value represents the perceived target height as seen by the vessel, not the actual physical target height. Consider the physical height of a human at around 1.8 metres: if this human swims in in water, its TargetHeight reduces to about 0.3 metres (i.e. the height of the head). If targetHeight=0, the Segment does not contain a target that requires detection (see Section 4.10.2).
TargetWidth	Similar to TargetHeight, this double value defines the perceived width of the target in metres.
DetectionCriteria	This double value can only be 0.75, 3.0 or 6.0. These key values represent the "Johnson criteria" specified by the current camera model. A higher value ensures more precise image recognition. See Section 4.10 for a full description of this value.
Loiter	An integer value indicating how many seconds the vessel should loiter upon reaching this POINT or PATH end point. Loiter overrides UponArrival and Time entries. Hence, a vessel will loiter even if it should proceed to the next SEGMENT indicated by UponArrival or if the Time of the subsequent SEGMENT has already passed. If Loiter=0, the vessel proceeds with actions defined in UponArrival.
Height	An integer value indicating in metres at what height the vessel should conduct the SEGMENT. If the SEGMENT is a POINT, the distance covered to reach the POINT and any loitering will be flown at this height (except for the first POINT of a TRACK whose Height is defined in DashHeight in the MISSION-table for the current TRACK). If the SEGMENT is a PATH, height will be used for possible manoeuvres to reach the current PATH, the total length of the PATH and any loitering. If the SEGMENT has UponArrival=search, the search and loitering will be flown at this height. A value of ``99999" indicates that the vessel should fly at the maximum possible altitude defined in the VESSEL parameter altitudeMax (see Section 3.3.2). Note that altitude changes can only occur step-wise between SEGMENTS.
Speed	An integer value indicating at what speed (in meters per second) the vessel should conduct the current SEGMENT. If the SEGMENT is a POINT, the distance covered towards the POINT and any loitering will be conducted at this speed (except for the first POINT of a TRACK whose Speed is defined in DashSpeed in the MISSION-table for the current TRACK). If the SEGMENT is a PATH, the PATH itself, any manoeuvres conducted to reach the PATH and any loitering will be conducted at this speed. If the SEGMENT has UponArrival=search, the Search and any loitering will be conducted at this speed. A value of 9999 indicates that the vessel should move at its maximum speed (defined in the VESSEL parameter speedMax, see Section 3.3.2). A value of 0 indicates that the VESSEL should move at its minimum possible speed (speedMin). If speedMin=0, the vessel will move at 0.01 m/s to avoid vessels never reaching their goal.

Appendix 2: Track parameters

This appendix details the parameters required to define a TRACK object.

TRACK parameter	Description
`Vessel_IDs`	A comma-separated list of integers defining the VESSELS to conduct this TRACK. The integers refer to the VESSEL parameter `id` (see Section 3.3.3). A TRACK can be conducted by any number of VESSELS of any type. Integers must be unique for each TRACK.
`Base`	A string defining the BASE where this TRACK will start. It refers to the `Base` parameter `StationName` (see Section 4.7).
`Track`	A string defining the list of SEGMENTS to be conducted. This loads the corresponding SEGMENT-table of this name.
`TrackFragmented`	Boolean value indicating if the SEGMENTS of this TRACK are independent (i.e. fragmented) of each other. If true, the vessel will return to the DESTINATION of this TRACK after completing each individual SEGMENT. This is useful to define TRACKS with many "sub-Tracks", for example all search-and-rescue POINTS in an area over one year. If false, SEGMENTS are conducted sequentially without return to the DESTINATION (unless a SEGMENT has `UponArrival=home`).
`Destination`	A string defining the Destination where this TRACK will end. It refers to the BASE parameter `StationName` (see Section 4.7). Care must be taken to ensure physical integrity of VESSEL operations: If a vessel ends its initial Track at `Destination=X` and is scheduled to start a subsequent TRACK from `Base=Y`, the vessel will be "beamed" from X to Y.
`Time`	Entry defining the point in time when the *TRACK* will commence, *i.e.* when the vessel will leave the *BASE*. The entry follows the ISO 8601 date format "YYYY-MM-DDThh:mm:ss" (for example: "2014-01-12T14:36:22"). All values refer to GMT (Greenwich Mean Time). If a VESSEL is not ready at that time, it will commence the *TRACK* as soon as possible afterwards.
`Repetition`	Three integer values separated by "X" define the *TRACK* repetition in the format "dXfXr" (for example "36000X86400X365"). The first integer "d" defines the patrol duration in seconds, i.e. the duration for which the *TRACK* should be patrolled without landing in between (unless refuel required). On a patrol, the VESSEL returns to the initial *SEGMENT* after reaching the final *SEGMENT* of a *TRACK*. If d=0, the *TRACK* is flown once and repeated as defined by the second and third values "f" and "r". Otherwise, the *TRACK* is patrolled for the desired duration. The second entry "f" is the repetition frequency in seconds. It defines how often the *TRACK* should be repeated. If "d"<"f", the vessel patrols for duration "d" and repeats the patrol every "f" seconds. If "d">"f", the vessel will interrupt the patrol after "f" seconds, go to its *DESTINATION* and repeat the *TRACK* afterwards, starting a patrol for "f" seconds only. If "f" is smaller than the total duration it takes

	to finish the *TRACK*, repetitions are queuing up for this VESSEL, building up delays. The third integer entry "r" defines the number of repetitions of this *TRACK*. If "f"=0 and "r">0, Repetitions are scheduled to start at the same time. The VESSEL will conduct them one by one, building up delays. If "r"=0, no patrols and no repetitions are conducted and "d" and "f" are meaningless. If "r"=2, the *TRACK* will be flown three times in total.
`Priority`	An integer indicating the relative importance of *TRACKS* within a *MISSION*. Higher values define *TRACKS* of higher priority. If a VESSEL conducts a low priority *TRACK* while being asked to start a high priority *TRACK*, it cancels the current *TRACK* and starts the higher priority *TRACK*. If a new *TRACK* with equal or lower `priority` is scheduled while a VESSEL conducts a *TRACK*, the VESSEL stores the new *TRACK* and conducts it as soon as possible afterwards.
`DashHeight`	A double value indicating the height in metres at which the dash of this *TRACK* will be conducted (i.e. the distance between the *BASE* and the first *SEGMENT*). VESSEL minimum and maximum altitudes are taken into account (*i.e.* ships do not fly, cars do not drive under water, *etc.*). If *`DashHeight=99999`*, the VESSEL's `altitudeMax` parameter defines the `DashHeight`. If *`DashHeight=0`*, airborne VESSELS conduct the *TRACK* at `altitudeMin`.
`DashSpeed`	A double value indicating the speed in metres per second at which the dash SEGMENT should be conducted. `DashSpeed` must not be equal to zero. If *`DashSpeed=-9999`*, the VESSEL will conduct the dash at its `speedMin` (see Section 3.3.2). If *`DashSpeed=9999`*, it will dash at its `speedMax`.
`ReturnHeight`	Same as `DashHeight` but for the return stage.
`ReturnSpeed`	Same as `DashSpeed` but referring to the return stage.

Appendix 3: Vessel parameters

This appendix details the parameters required to define a VESSEL.

VESSEL parameter	Description
`performanceModel`	Specify which performance model should be loaded for this VESSEL. The default model is "powerAgainstSpeed" as described in Section 4.9. Custom models can be loaded (see Section 4.8).
`fuelType`	A String specifying the fuel type used by this VESSEL. Key words include "*`petrol`*", "*`diesel`*", "*`nuclear`*", "*`coal`*" or "*`food`*". If *`performanceModel=powerAgainstSpeed`*, the calorific value of the fuel will be changed accordingly as follows: petrol → 44.4 MJ/kg; diesel → 41.1 MJ/kg; nuclear → ∞ MJ/kg; coal → 20 MJ/kg; electric → ∞ MJ/kg. Note that both keywords nuclear and electric will set the calorific value to ∞ MJ/kg. This is because the calorific value is used to convert energy used into mass of fuel used (see Section 3.3.4). However, electricity does not use up any mass while nuclear consumption of fuel elements is negligible for VESSELS.
`occupants`	Integer value indicating the number of humans required to operate a VESSEL directly, i.e. how many humans must be onboard the VESSEL to control it? For UAS, *`occupants=0`* while for civil airliners, *`occupants=2`*, usually. This value can be used to calculate operating costs but has no functionality in the OSCAR simulation.
`speedMax`	Double value larger than zero that indicates in metres per second the maximum speed that the VESSEL can move at. If a Segment requests higher speeds, the VESSEL will move at `speedMax` only, building up delays. In reality, any VESSEL type has varying maximum speeds depending on a number of parameters. However, during conceptual design one value suffices.
`speedMin`	Double value larger or equal to zero that indicates in metres per second the minimum speed that the VESSEL can move at. If a SEGMENT requests a lower speed, the VESSEL will move at `speedMin` only. Except for fixed wing aircraft, all VESSELS have *`speedMin=0`*, i.e. they can stop motion during a TRACK. For fixed wing aircraft, speedMin equates to the stall speed in landing configuration (V_{S0}), neglecting its variation across different flight regimes. A SEGMENT will never ask for `speed=0` (see Section 3.2.3.1).
`speedTypical`	Double value larger to zero that indicates in metres per second the typical speed to be used if `useTypicalSetup=true` (see below).
`altitudeMax`	Double value larger or equal to zero indicating in metres the maximum operating altitude for VESSELS. For non-airborne VESSELS, `altitudeMax=0`. For airborne VESSELS, this value neglects the variation of maximum operating altitude depending on weight, temperature, *etc*. If a SEGMENT requests

	a higher altitude, the VESSEL will move at `altitudeMax`.
`altitudeMin`	Double value indicating in metres the minimum operating altitude for VESSELS. For most VESSELS, `altitudeMin=0`, except for submarines that can move under water. Further exceptions would be mining lorries and subway trains. If a SEGMENT requests a lower altitude, the VESSEL will move at `altitudeMin`.
`altitudeTypical`	Double value indicating in metres the typical altitude to be used if `useTypicalSetup=true` (see below).
`useTypicalSetup`	A Boolean switch. If "false", the SEGMENT parameters `Height` and `Speed` shall be applicable for this VESSEL. If "true", the VESSEL will not apply the SEGMENT `Height` and `Speed` values but use its own `speedTypical` and `altitudeTypical` parameters instead for all SEGMENTS. This is useful if a SEGMENT shall be conducted by several VESSELS but one of them is controlled through user inputs on altitude and speed while the others are not. User control can be applied through specific SEGMENT `Height` and `Speed` entries while the other VESSELS have `useTypicalSetup=true`.
`weightDry`	Double value larger than zero indicating in kilograms the weight of the VESSEL without any fuel loaded (if applicable). Value includes structural and payload weight. For nuclear and electric VESSELS, this is the same as the total weight.
`weightFuel`	Double value larger or equal to zero indicating in kilograms the maximum fuel weight the VESSEL can carry. For electric and nuclear VESSELS, `weightFuel=0`. Within OSCAR, VESSELS are always fuelled up completely.

Appendix 4: Component parameters

This appendix details the parameters required to define a COMPONENT.

COMPONENT parameter	Description
weibullLifeMeasure	A string indicating by what mechanism the COMPONENT deteriorates. Value can be either "duration" or "cycles". The former will trigger the COMPONENT to deteriorate continuously while operating. This is applicable to any COMPONENT that primarily suffers fatigue from VESSEL movements, i.e. aircraft wings, car wheels, a ship's hull or a human foot. If weibullLifeMeasure=cycles, the COMPONENT deteriorates only when a TRACK concludes, i.e. every time an aircraft lands, a ship arrives in a harbour or a car parks. This is applicable to COMPONENTS primarily suffering fatigue once every VESSEL cycle, i.e. aircraft landing gear, car doors or a ship's anchor.
weibullEta	Double value indicating the shape parameter of the weibull distribution characterizing the units to failure for this COMPONENT.
weibullBeta	Double value indicating the scale parameter of the weibull distribution characterizing the units to failure for this COMPONENT. If weibullLifeMeasure=duration, unit is the time unit used in the simulation. If weibullLifeMeasure=cycles, unit is cycles.
LossProbabilityFromFailure	Double value between 0 and 1 indicating the likelihood of losing the entire VESSEL if this COMPONENT fails during service and there are no functional redundant COMPONENTS that can take over. Aircraft engines have LossProbabilityFromFailure=1 because the aircraft will crash if its only engine (or all available engines) fails inflight. A car's CD-player will have LossProbabilityFromFailure=0 because a broken CD-player cannot cause the destruction of the car itself.
unplannedMaintenanceDuration	Double value in seconds indicating how long unplanned maintenance takes on this COMPONENT. If the COMPONENT failed, it will schedule unplanned maintenance at the next suitable time.
quantityOnboard	Integer value indicating how many COMPONENTS of the same type are on-board this VESSEL. Ensure to include COMPONENTS that can take over the tasks and workload of this COMPONENT only. In aircraft wings, several aileron servos can replace a broken servo if needed. However, only servos on the left wing can replace a broken servo on the left wing. Similarly, a car usually features four identical wheels but none can take over the workload of a broken wheel, therefore each wheel should be a separate COMPONENT with quantityOnboard=1.
robustnessScalingFactor	Double value scaling the COMPONENT robustness such that scaledTimeToFailure = TimeToFailure * (1+robustnessScalingFactor). A value of zero

	indicates no change to the COMPONENT time to failure. If `robustnessScalingFactor=-0.1`, the COMPONENT will fail 10 % earlier than assumed by the proposed weibull distribution. During the conceptual design, COMPONENT failure behaviour is often unknown. Therefore, this parameter helps validating historical data or estimates against expected VESSEL performance (as conducted in Chapter 6). Moreover, this parameter enables sensitivity analysis of COMPONENT robustness.

Appendix 5: Camera footprint algorithm

This appendix details calculation of airborne VESSEL active payload camera footprints based on flight conditions. This calculation occurs every time the payload records an image. For airborne VESSELS flying level, assuming their camera is pointed towards the earth surface and targets must be found on the surface, Chen et al. (2009) define a camera footprint as in Figure A-1.

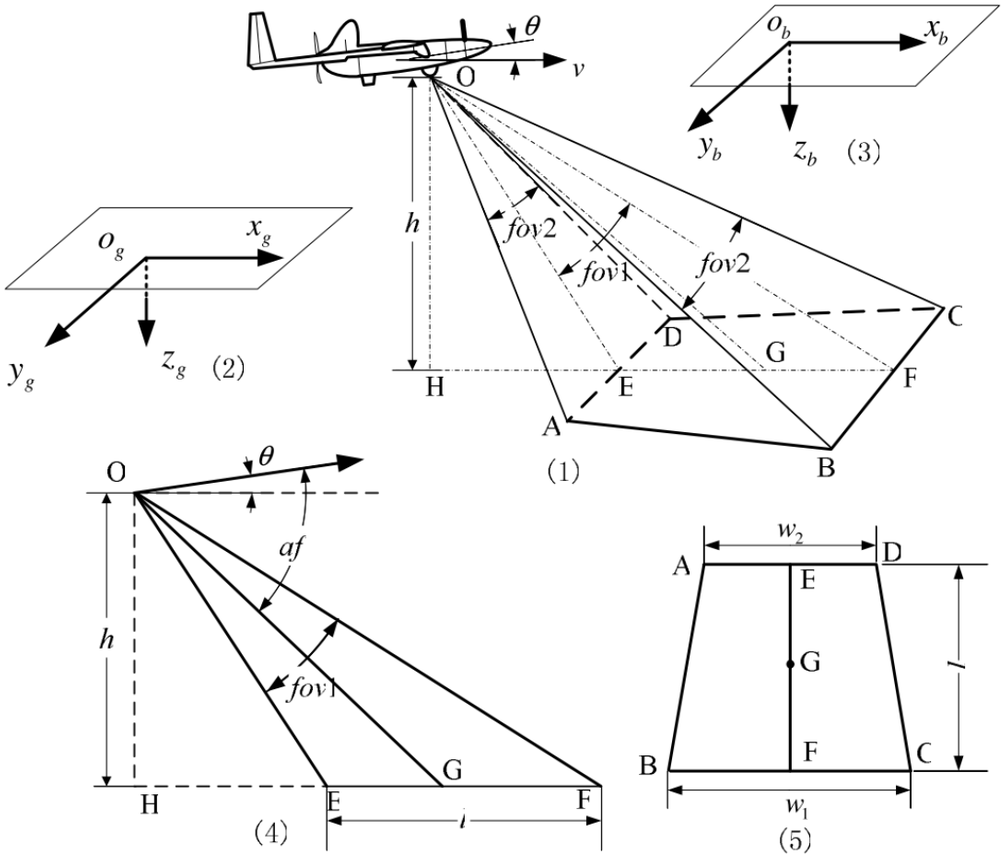


FIGURE A-1: AIRBORNE VESSEL CAMERA FOOTPRINT DEFINITION. REPRODUCED FROM CHEN ET AL. (2009).

For the OSCAR framework, the characteristic dimensions are the footprint length *"l"*, the footprint width closer to the VESSEL *"w_2"*, the footprint width farther from the VESSEL *"w_1"* and the ground distance $\overline{HE}$ from VESSEL to *"w_2"*. The VESSEL parameter `height` corresponds to *"h"*, the VESSEL parameter `sensorFOVver` corresponds to *"fov1"* and `sensorFOVhor` corresponds to *"fov2"*. Moreover, the VESSEL parameter `sensorTiltAngle` corresponds to *"af"*.

For level flight with constant pitch angle θ and altitude `height`, simple trigonometry defines the camera footprint length as

$$l = h \times \left\{ \cot\left(af - \theta - \frac{fov_1}{2}\right) - \cot\left(af - \theta + \frac{fov_1}{2}\right) \right\} \qquad \text{Eq. A-1}$$

Similarly, the camera footprint widths w_1 and w_2 are

$$w_1 = \frac{2 \times h \times \tan\left(\frac{fov_2}{2}\right)}{\sin\left(af - \theta - \frac{fov_1}{2}\right)} \quad \text{Eq. A-2}$$

$$w_2 = \frac{2 \times h \times \tan\left(\frac{fov_2}{2}\right)}{\sin\left(af - \theta + \frac{fov_1}{2}\right)} \quad \text{Eq. A-3}$$

The airborne camera footprint must be updated whenever any of the following VESSEL parameters changes: `height`, `sensorFOVver`, `sensorFOVhor` or `sensorTiltAngle`.

However, not all OSCAR VESSELS are airborne. There are ground-based and submerged VESSELS as well. Submerged VESSELS (*i.e.* submarines, *etc.*) can use the method described above if they scan the ocean surface or the water surface for targets. In this case, `height` is the difference between the VESSEL depth and the ocean or water surface. Maritime VESSELS floating on the water surface (ships…) can use the method to scan the ocean surface. Both cases assume perfectly luminous water.

For ground-based VESSELS, a different approach is used. Here, cameras are very close to the surface and the assumptions of Chen et al. (2009) do not hold anymore. Instead, cameras look horizontally towards the horizon to scan for targets on the ground. The camera footprint is not a trapezoid anymore, but a circular arc, as in Figure A-2.

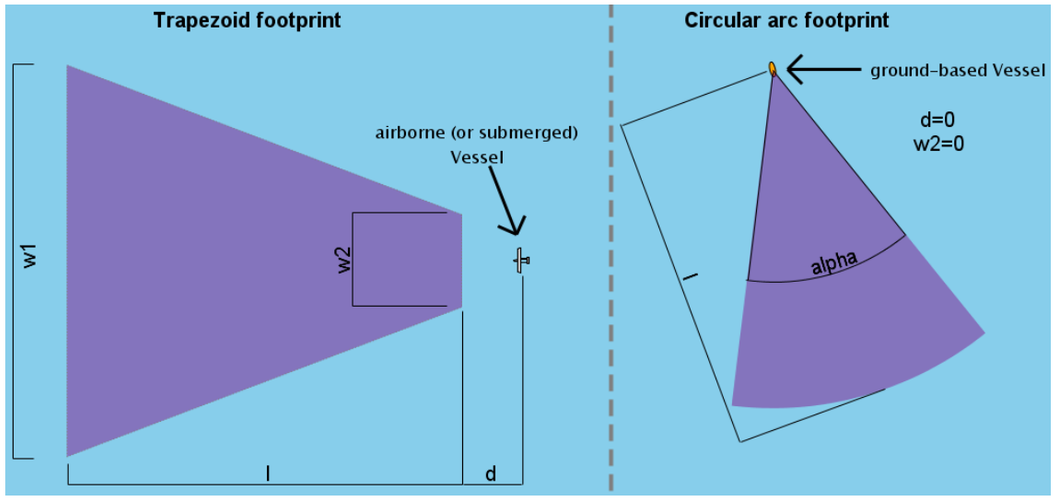


FIGURE A-2: CAMERA FOOTPRINT DEFINITION FOR AIRBORNE/SUBMERGED VESSELS AND GROUND-BASED VESSELS.

For ground-based VESSELS, assume $d = 0$ and $w = 0$ because the camera is near the ground looking towards the horizon. This assumes `sensorTiltAngle =0` for ground-based VESSELS.

Define the footprint length *"l"* as the distance to the earth horizon from the camera, neglecting refraction and assuming infinite visibility.

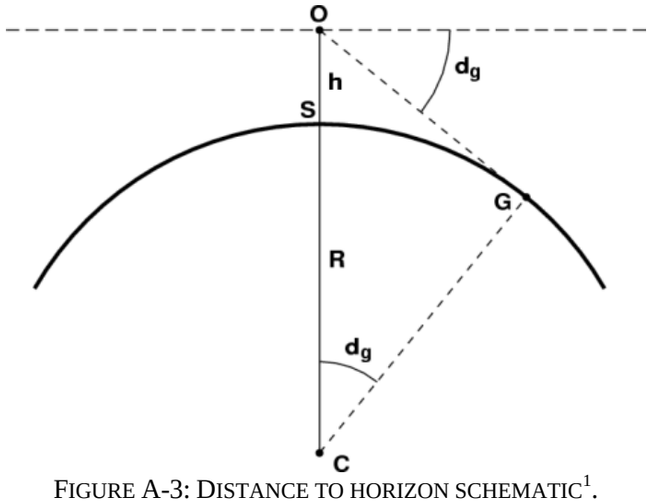


FIGURE A-3: DISTANCE TO HORIZON SCHEMATIC[1].

This distance to the horizon from an observer *O* is the same as $\overline{OG}$ in Figure A-3. From trigonometry, $\overline{OG} = \sqrt{2 \times R \times h + h^2}$ where *R* is the earth radius and *h* is the camera height above ground. However, since *R* is much larger than *h* for ground-based VESSELS, simplify the footprint length to

$$l = \overline{OG} = \sqrt{2 \times R \times h} \qquad \text{Eq. A-4}$$

Note that this discussion assumes that $\overline{OG} \approx \overline{SG}$ because *h* is small compared to these distances. Within the OSCAR simulation, all ground-based Vessels assume $h = 2.2\text{m}$, i.e. the camera sits 2.2 metres above the VESSEL `height`. Future work can make this another VESSEL parameter, named "`cameraHeight`".

The last parameter to define the circular arc footprint for ground-based VESSELS is *alpha* (compare Figure A-2). Within the OSCAR framework, *alpha* corresponds to the VESSEL parameter `sensorFOVhor`.

[1] Discussion from http://mintaka.sdsu.edu/GF/explain/atmos_refr/horizon.html, accessed 01/11/2013.

Appendix 6: Sample model walkthrough

This appendix presents a systematic tutorial on how to create and operate a simple OSCAR simulation.

1) Create geographical model:

a) Obtain a Geographical Information System program of your choice (ArcGis from www.esri.com, QGIS from www.qgis.org, *etc.*).

b) Create a new shapefile containing a number of "Point" or "Polyline" elements only. Create a new dbf-file associated with the shapefile. Add a new column "Constant". Assign all elements required to run the simulation the value "`Constant=0`" (Figure A-4).

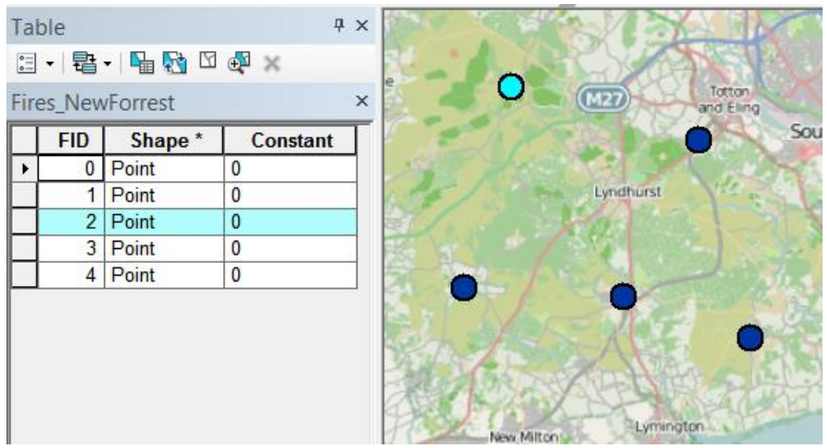


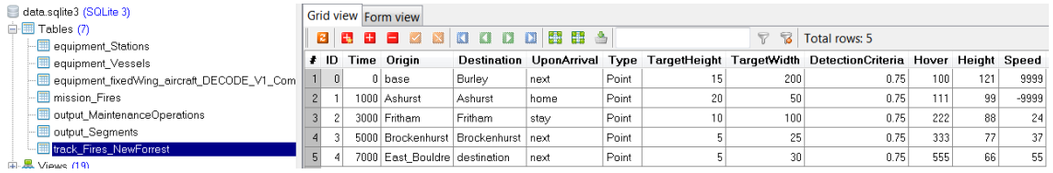
FIGURE A-4: SIMPLE GEOGRAPHICAL INFORMATION SYSTEM MAP WITH DBF COLUMN "CONSTANT".

c) Save shapefile, dbf-, ssx- and shx-file in subfolder "GIS".

2) Setup database

a) Open the file "data.sqlite3" in the model folder using any SQLite database manager (SQLiteStudio from www. http://sqlitestudio.pl).

b) Edit the input tables. Figure A-5 shows a sample TRACK table for the five POINTS in Figure A-4.



#	ID	Time	Origin	Destination	UponArrival	Type	TargetHeight	TargetWidth	DetectionCriteria	Hover	Height	Speed
1	0	0	base	Burley	next	Point	15	200	0.75	100	121	9999
2	1	1000	Ashurst	Ashurst	home	Point	20	50	0.75	111	99	-9999
3	2	3000	Fritham	Fritham	stay	Point	10	100	0.75	222	88	24
4	3	5000	Brockenhurst	Brockenhurst	next	Point	5	25	0.75	333	77	37
5	4	7000	East_Bouldre	destination	next	Point	5	30	0.75	555	66	55

FIGURE A-5: SAMPLE TRACK TABLE INPUTS.

3) Simulation setup

a) Load the file "OpSim.alp" into AnyLogic.

b) If you want to use the experiments “Simulation” or “FastRun”, open the respective active object, navigate to the “General” tab of its properties and set the parameter `missionStrings = new String[]{db_table_yourMissionName}`, following the name of the MISSION table in the database.

c) If using the “Simulation_Customers” experiment, no action is required.

d) Set remaining parameters as described in Section 4.13.

4) Run simulation

5) Analyse outputs

a) Open the file “data.sqlite3” in the model folder, navigate to the output tables.

b) Analyse and post-process data.

Appendix 7: Database

This appendix details the database developed for the OSCAR simulation. There are four different table types in the database, namely *equipment*, *mission*, *track* and *outputs*. Each has several sub-types as described below. Each database table name starts with its type followed by an underscore as in `equipment_table1`.

Equipment tables

There are three equipment table types, namely *bases*, *vessels* and *components*.

Base table

There is exactly one base table named "equipment_Bases" listing all available bases using the format below:

TABLE A-1: EQUIPMENT BASE TABLE FORMAT.

Column name	Description
`BaseID`	Integer value used as the table primary key
`BaseName`	String naming the Base
`Longitude`	Double value specifying the base longitude position in decimal degrees.
`Latitude`	Double value specifying the base latitude position in decimal degrees.

Vessel table

There is exactly one vessel table named "equipment_Vessels" listing all available VESSEL agents that can be used. The table format is as follows:

TABLE A-2: EQUIPMENT VESSEL TABLE FORMAT.

Column name	Description
id	Integer value used as the table primary key
category	String describing the VESSEL category as defined in Figure 3-12
type	String describing the VESSEL type as defined in Figure 3-12
name	Optional String for VESSEL name, *i.e.* "BMW", "Policeman", *etc*.
performanceModel	As described in Appendix 3.
fuelType	As described in Appendix 3.
occupants	As described in Appendix 3.
cameraFOVhor	As described in Section 3.3.6.2.1.
cameraFOVver	As described in Section 3.3.6.2.1.
cameraPixelHor	As described in Section 3.3.6.2.1.
cameraPixelsVer	As described in Section 3.3.6.2.1.
cameraTiltAngle	As described in Section 3.3.6.2.1.
cameraRecognitionFactor	As described in Section 3.3.6.2.1.
speedMax	As described in Appendix 3.
speedMin	As described in Appendix 3.
speedTypical	As described in Appendix 3.
altitudeMax	As described in Appendix 3.
altitudeMin	As described in Appendix 3.
altitudeTypical	As described in Appendix 3.
useTypicalSetup	As described in Appendix 3.
weightDry	As described in Appendix 3.
weightFuel	As described in Appendix 3.
speedValues	Comma-separated list of double values specifying a speed range for this VESSEL. Only required if performanceModel=powerAgainstSpeed. Can have any number of entries but total number must equal that of powerValues.
powerValues	Comma-separated list of double values specifying the energy consumption at the respective speed value in speedValues. Only required if performanceModel=powerAgainstSpeed. Can have any number of entries but total number must equal that of speedValues.

The OSCAR simulation features one additional performance model plug in (Section 4.9.2). Additional Vessel parameters run this module if `performanceModel=fixedWing_aircraft_petrol` and are described in Appendix 9.

Component table

There is an arbitrary number of COMPONENT tables depending on user requirements. Each VESSEL listed in the table `equipment_Vessels` can have its own COMPONENT table if it is supposed to deteriorate and receive maintenance (Section 3.3.5). In this case, the COMPONENT table must be named using the VESSEL'S parameters following the scheme "equipment_`category_type_name_`Components". If the user wants to assign COMPONENTS to a VESSEL of `category=fixedWing`, `type=aircraft` and `name=G-DHBX`, the COMPONENT table name were "equipment_fixedWing_aircraft_G-DHBX_Components". Upon VESSEL creation, the agent checks if a corresponding COMPONENT table exists. If so, COMPONENTS are created for this VESSEL agent. Each COMPONENT table follows the structure below:

TABLE A-3: EQUIPMENT COMPONENT TABLE FORMAT.

Column name	Description
`ComponentID`	Integer value used as the table primary key
`ComponentName`	String indicating the name of the COMPONENT.
`WeibullLifeMeasure`	As described in Appendix 4.
`WeibullEta`	As described in Appendix 4.
`WeibullBeta`	As described in Appendix 4.
`LossProbabilityFromInflightFailure`	As described in Appendix 4.
`MaintReplacementTime`	Described as "unplannedMaintenanceDuration" in Appendix 4.
`QuantityOnboard`	As described in Appendix 4.
`RobustnessScalingFactor`	As described in Appendix 4.

Mission tables

There are an arbitrary number of mission tables in the database. Each table refers to one MISSION as defined in Section 3.2.3.3. Each mission table lists an arbitrary number of TRACKS. Each mission table name starts with "mission_" and is followed by a descriptive string (*i.e.* "RotterdamPolice", "search_southernUK", *etc.*). The mission table structure follows the TRACK parameters defined in Section 3.2.3.2 and Appendix 2 as follows:

TABLE A-4: MISSION TABLE FORMAT.

Column name	Description
ID	Integer value used as the table primary key
Vessel_IDs	As described in Appendix 2.
Base	As described in Appendix 2.
Track	As described in Appendix 2.
TrackFragmented	As described in Appendix 2.
Destination	As described in Appendix 2.
Time	As described in Appendix 2.
Repetition	As described in Appendix 2.
Priority	As described in Appendix 2.
DashHeight	As described in Appendix 2.
DashSpeed	As described in Appendix 2.
ReturnHeight	As described in Appendix 2.
ReturnSpeed	As described in Appendix 2.

Track tables

There are an arbitrary number of track tables in the database. Each table refers to one TRACK as defined in Section 3.2.3.2. Each track table lists an arbitrary number of SEGMENTS. Each track table name starts with "track_" and is followed by a descriptive string. For better abridgement, use the same descriptive string used for the respective MISSION table followed by a more specific description of this TRACK. Consider a UAS designed to follow the Olympic torch relay every day. For this, the MISSION table name could be "mission_OlympicTorchRelay" listing TRACKS for each day of the relay. The TRACK tables could be "track_OlympicTorchRelay_Day1", *etc*. The track table structure follows the SEGMENT parameters defined in Section 3.2.3.1 and Appendix 1 as below:

TABLE A-5: TRACK TABLE FORMAT.

Column name	Description
ID	Integer value used as the table primary key
Time	As described in Appendix 1.
Origin	As described in Appendix 1.
Destination	As described in Appendix 1.
UponArrival	As described in Appendix 1.
Type	As described in Appendix 1.
TargetHeight	As described in Appendix 1.
TargetWidth	As described in Appendix 1.
DetectionCriteria	As described in Appendix 1.
Hover	Described as "Loiter" in Appendix 1.
Height	As described in Appendix 1.
Speed	As described in Appendix 1.

Output tables

All output data is stored into one of two output tables named "output_Segments" and "output_MaintenanceOperations". To allow flexible post-processing for users, data is stored in a raw format without processing from OSCAR. Output tables are stored within the same database as the input tables to enable matching simulation inputs and output performance. Moreover, one database file relates to one simulation run. Users can backup and store data easily by naming databases accordingly. In addition, keeping inputs and outputs within one database enables creating advanced database "views" for post-processing (see below).

Segment output table

This table named "output_Segments" stores data on every SEGMENT conducted in this simulation run. Each table entry is created through an instance of the OSCAR simulation Java class TARGETSTATISTIC corresponding to the table columns as below:

TABLE A-6: OUTPUT SEGMENTS TABLE FORMAT. SPLIT OVER SEVERAL PAGES.

Column name	Description
ID	Integer value used as the table primary key
iteration	Integer value specifying the simulation iteration (i.e. replication) that this SEGMENT was computed in.
missionName	String indicating the name of the MISSION table that this

	SEGMENT belongs to.
missionID	Integer indicating the "ID" of the TRACK in the mission table that this SEGMENT belongs to.
missionStartTimeDB	Date and time when this SEGMENT's TRACK was supposed to start as specified in the MISSION table Time column. However, the value is updated for repetitions to distinguish between repeated MISSION entries. The entry follows the ISO 8601 date format "YYYY-MM-DDThh:mm:ss" (for example: "2014-01-12T14:36:22").
missionStartTimeActual	Date and time when this SEGMENT'S TRACK actually commenced in the simulation. Use this value to find delays of TRACK starts due to tight repetition setup or previous TRACK performance. Can have keyword "cancelled due to fuel" if the VESSEL could not perform the Track due to insufficient fuel capacity.
trackName	String indicating the TRACK table name that this SEGMENT belongs to.
segmentID	Integer value indicating the "ID" of this SEGMENT as specified in its TRACK table. Can be "8888" if SEGMENT was cancelled due to insufficient fuel capacity. Can be "-1" for dash SEGMENTS. Can be "9999" for return SEGMENTS.
startTimeDB	Date and time when this Segment was supposed to start. It is the sum of missionStartTimeDB and the TRACK table entry Time for this SEGMENT. For return SEGMENTS, this is "null" since return SEGMENT start times are not defined.
startTimeActual	Date and time when this SEGMENT actually commenced within the simulation run. Can have keyword "cancelled due to fuel" if the VESSEL could not perform the TRACK due to insufficient fuel capacity.
departures	Integer value indicating how often a VESSEL departed from a BASE during this SEGMENT. For airborne VESSELS, this equates to take-offs. For dash SEGMENTS, value cannot be higher than 1. Otherwise, this value indicates how often the VESSEL had to return for refuels.
origin	Use this SEGMENT'S track table value origin. For dash SEGMENTS, use mission table Base entry.
destination	Use this SEGMENT'S track table value destination. For return SEGMENTS, use mission table destination entry.
uponArrival	Use this SEGMENT'S track table value uponArrival. Value is "landing" for return SEGMENTS.
type	Use this SEGMENT'S track table value type. Use "dash" for dash SEGMENTS and "return" for return SEGMENTS.
targetHeight	Use this SEGMENT'S track table value targetHeight. Return SEGMENTS have targetHeight=0.0 by default.
targetWidth	Use this SEGMENT'S track table value targetWidth. Return SEGMENTS have targetWidth=0.0 by default.
detectionCriteria	Use this SEGMENT'S track table value detectionCriteria.

	Return SEGMENTS have `detectionCriteria=0.0` by default.
`hover`	Use this SEGMENT'S track table value `hover`. Dash and return SEGMENTS have `hover=0` by default.
`height`	Integer value indicating the actual height in metres that this SEGMENT was conducted at. Value can differ from track table entry `height` for this SEGMENT if VESSEL could not work at specified height or if entry was keyword value like "`9999`".
`speed`	Integer value indicating the actual speed in metres per second that this SEGMENT was conducted at. Value can differ from track table entry `speed` for this SEGMENT if VESSEL could not work at specified speed or if entry was keyword value like `9999`.
`vesselID`	Integer value indicating the `id` of the VESSEL that conducted this SEGMENT. Corresponds to the `equipment_Vessels` table entry `id`.
`timeOfSpotting`	Date and time when the VESSEL spotted this SEGMENT'S target using its active payload (if any) and the payload model (Section 3.3.6.2). Value can be `spotting not required` if this SEGMENT did not require spotting (`targetHeight=0` or `targetWidth =0`). If it required spotting but the VESSEL could not spot it, value will be `null`. Value can be `cancelled due to fuel` if VESSEL had to cancel this SEGMENT due to insufficient fuel. Value can be `spotted by other vessel` if this VESSEL did not find target but a connected VESSEL looking for the same target did.
`timeOfArrival`	Date and time when the VESSEL arrived at the SEGMENT target. Value can be `null` if VESSEL aborted SEGMENT because a patrol ended in between and it never arrived. Value can be `cancelled due to fuel`.
`timeOfDeparture`	Date and time when the VESSEL departed this SEGMENT. Value can be `cancelled due to fuel`. The difference between `timeOfArrival` and `timeOfDeparture` is the VESSEL loiter duration and additional loiter due `uponArrival=stay`. Value is `null` for return Segments.
`energyUsed`	How much energy (in Joule) did the VESSEL use during this SEGMENT?
`imagesTaken`	How many images did the active payload collect during this SEGMENT, if any.
`areaScanned`	How much ground area in m^2 was scanned by the VESSEL active payload in this SEGMENT, if any? Currently, this equates to the net area scanned, neglecting overlap areas scanned multiple times. However, it is possible to record the total area including overlaps.
`dataAcquired`	Double value indicating how much data (in MB) the active payload stored by taking digital images, if any? Assumes pixel format RGBA32 where each pixel has four colour channels and each channel requires 8 bits.

segmentMeasure	String indicating a context-specific measure of choice by the client. Currently, search SEGMENTS record the incident aliveness upon spotting between 0 and 1. This can be used to find the number of total saved lives (Section 4.11). For other SEGMENTS, value is null.

Maintenance Operations output table

There is one table named "output_MaintenanceOperations" in each database. It includes raw details on all component problems causing maintenance operations (planned and unplanned) as well as VESSEL losses. Each table entry is created through an instance of the OSCAR simulation Java class MAINTENANCE corresponding to the table columns as below:

TABLE A-7: OUTPUT MAINTENANCE OPERATIONS TABLE FORMAT.

Column name	Description
id	Integer value used as the table primary key
iteration	Integer value specifying the simulation iteration (i.e. replication) where the current COMPONENT problem occurred.
vesselID	Integer value indicating the id of the VESSEL that experienced a COMPONENT problem. Corresponds to the equipment_Vessels table entry id.
ComponentID	Integer value indicating the ComponentID listed in the corresponding COMPONENT table. Value can be 9999 if VESSEL was lost upon arriving at destination (*i.e.* landing for airborne VESSELS, parking for cars, *etc.*) because no component was causing this loss.
segmentID	Integer value indicating the SEGMENT'S ID in the corresponding TRACK table where this COMPONENT problem occurred. Can be 9999 if problem occurred during return SEGMENT. Can be −1 if problem occurred during dash SEGMENT. Can be −9999 if problem occurred during landing.
timeOfProblem	Date and time when this problem occurred.
redundancy	String value indicating if the problematic COMPONENT was replaced by redundant COMPONENTS during operation (yes) or not (no). If no redundancy existed, the VESSEL may be lost. Value can be not applicable if VESSEL was lost during arrival at a BASE without specific COMPONENT problems.
crash	Boolean value indicating if the VESSEL was lost due to the current problem (yes) or not (no). If crash=yes, then by default timeOfMaintenance=not applicable and duration=0.
timeOfMaintenance	Date and time when the maintenance operation occurred that was caused by this COMPONENT'S problem. Value can be not applicable if VESSEL was lost.

duration	Integer value indicating the duration in seconds that the maintenance operation took. Corresponds to column MaintenanceReplacementTime in component table for this COMPONENT. If duration=0, either the VESSEL was lost due to the current problem or no maintenance was carried out.

Output views & analysis capabilities

A typical VESSEL conducts a large number of SEGMENTS and maintenance operations during its life cycle. If OSCAR simulates a fleet of VESSELS using several replications, the output tables become very large. Therefore, intelligent data post-processing is required to analyse outputs.

As with every database tool, SQLite creates "Views" by extracting and displaying specific information from large tables through SQLite syntax. In order to analyse outputs for the two case studies (Chapter 5 and 6), the OSCAR simulation provides 18 "Views". This section discusses one "View" in more detail to present the database analysis capability that comes with the OSCAR simulation.

For the case studies, one required output is the total fuel used by all UAS. However, the output_Segments table stores the energy used for each SEGMENT for any VESSEL. Therefore, the data must be filtered using the code below:

```
SELECT
    output_Segments. iteration, SUM ( output_Segments. en-
    ergyUsed )
AS
    energyUsedByUAS_in_Joule
FROM
    output_Segments
INNER JOIN
    equipment_Vessels ON output_Segments. vesselID = equip-
    ment_Vessels. id
WHERE
    equipment_Vessels. category = 'fixedWing' AND equip-
    ment_Vessels. type = 'aircraft' AND equipment_Vessels.
    occupants = 0
GROUP BY
    output_Segments. iteration
```

The SQL syntax sums the energyUsed column of all SEGMENTS conducted by fixedWing aircraft with zero occupants (i.e. UAS) listed in the equipment_Vessels table. Results are grouped by iteration to allow analysing statistical variation as in Figure A-6.

iteration	energyUsedByUAV_in_Joule
1	232799977751.47177
2	230857374012.5787
3	227558862967.1205
4	228545901331.4366
5	228767813720.40314
6	227249802643.26096
7	231845102189.47958

FIGURE A-6: DATABASE OUTPUT VIEW FOR UAS ENERGY USED, SORTED BY ITERATION.

This data can be copied into a spread sheets to convert it into fuel used (using the UAS fuel calorific value). See Figure 5-16 and Figure 6-5 for sample box plots of UAS fuel used.

Appendix 8: Geographical setup

This appendix details the essential structure required to import and use shapefile data within the OSCAR simulation. Moreover, it describes the actual data import process and how the data is turned into objects for AnyLogic processing.

Structure

In order to use shapefile data within the OSCAR simulation, users must adhere to a specific data structure shown in Figure A-7.

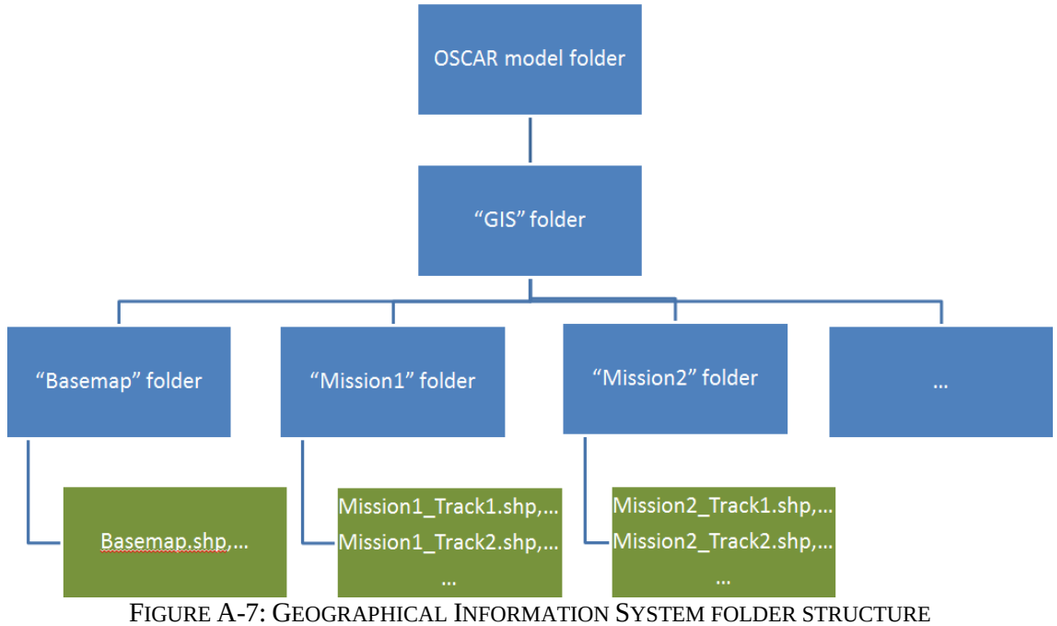


FIGURE A-7: GEOGRAPHICAL INFORMATION SYSTEM FOLDER STRUCTURE

The folder containing the actual AnyLogic OSCAR model file must contain a sub-folder named "GIS". The "GIS" folder itself contains a number of sub-folders containing the actual shapefiles. As a minimum, the "Basemap" folder contains the background world map data. Each file must be named "Basemap" plus its file-extension (.shp, .dbf, .ssx and .shx). Here, users can insert a background map of their choice. This map is for display purposes, but also allows agent interactions if required. Users can add any number of additional folders into the "GIS" folder. Folder names follow the respective MISSION table names. Similar to MISSION tables containing any number of TRACKS, folders can contain any number of shapefiles (and corresponding .dbf, .ssx and .shx files). Naming follows the same conventions as for TRACK-table names, *i.e.* the MISSION name followed by an underscore "_" followed by the TRACK name.

For the current OSCAR simulation, the Basemap data provider is the free open source geographical data provider "Natural Earth"[1]. The Basemap includes the entire planet physical landmasses including islands as polygons at a 1:10m resolution (Figure A-8).

FIGURE A-8: BASEMAP SHAPEFILE[1] USED WITHIN OSCAR SIMULATION. RESOLUTION 1:10M.

Mission folders contain one or more shapefiles (and associated .dbx, .ssx and .shx files). A shapefile can contain either Point features or Polyline features. In any case, each .dbf file must contain a specific column "Constant" assigning the value "0" to all shapes required for loading into the OSCAR simulation. Shapefiles can contain any number of features (of one type) in any geographical formation.

Import and conversion

Once users have created the shapefiles following the rules above, AnyLogic will be able to load the data and modify it to allow agent interaction with geographical features.

Upon creating the "Main" object (Figure 4-2) on start-up, the OSCAR simulation loads all shapefiles associated with the MISSION tables loaded for the current experiment (Section 4.13). The shapefiles are added to the AnyLogic map for visual display only. More importantly, the simulation associates all geographical features from the shapefiles with the SEGMENT details supplied by the database. This is the reason for the strict file formats and structural demands described earlier. The consolidated SEGMENT and shapefile data is combined using the custom OSCAR Java class "GISPOSITIONFULL". Each GISPOSITIONFULL object is a 2D point with latitude and longitude but also featuring respective SEGMENT characteristics such as `Origin`, `Destination`, `Height`, `Speed`, *etc.* (Appendix 1). Point SEGMENTS map directly to GISPOSI-

[1] Available at http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_land.zip, accessed 13/11/2013.

TIONFULL, *i.e.* each Point SEGMENT becomes a GISPOSITIONFULL object. Path SEGMENTS split into nodes and each node becomes a GISPOSITIONFULL object carrying its Path SEGMENT characteristics.

The consolidation algorithms access shapefile structures through the open source geographical tool OpenMap[2]. In fact, the AnyLogic geographical map feature imports the OpenMap toolkit to display geographical shapefiles.

After consolidating database SEGMENT and shapefile data, the OSCAR simulation can task VESSELS to move towards GISPOSITIONFULL object coordinates and use the specific GISPOSITIONFULL characteristics drawn from the database SEGMENT, i.e. `Height`, `Speed`, *etc.*

[2] https://code.google.com/p/openmap/, accessed 13/11/2013.

Appendix 9: Custom aircraft performance module

This appendix introduces a sample performance add-in developed using the steps in Section 4.9.2. It was developed by Mario Ferraro[1] to increase performance realism for fixed wing aircraft using petrol-driven propeller engines (*i.e.* the UAS in Chapters 5 and 6). Among other influences, this model takes into account the weight reduction due to fuel burn.

Inputs

Using this add-in, several additional parameters define VESSEL agents that have `category=fixedWing`, `type=aircraft` and `fuelType=petrol` as in Table A-8. These parameters feature as AnyLogic "parameter" objects as well as new columns in the database `equipment_Vessels` table (see Table A-2).

TABLE A-8: ADDITIONAL VESSEL PARAMETERS

Additional VESSEL parameters	Description
a_wing	The aircraft wing area in m^2.
cl_max	The maximum lift coefficient. Used to compute optimal aircraft landing speed based on aircraft weight upon landing.
d_prop	The propeller diameter in metres.
k1_a	Coefficient defining the drag polar. There are nine coefficients named k1_a, k1_b, k1_c, k2_a, k2_b, k2_c, k3_a, k3_b and k3_c. The polynomial is defined below.
p_inst	The installed maximum continuous engine power in Watts.
rpm	The maximum engine rpm (rounds per minute).
sfc_a	Coefficient defining the 4^{th} degree polynomial for specific fuel consumption as a function of engine power output in $g/kW.hr$. There are four coefficients named sfc_a, sfc_b, sfc_c and sfc_d. The polynomial is defined below.
zeta_a	Coefficient defining the 7^{th} degree polynomial describing the propulsive efficiency polynomial as a function of flight speed and rpm. There are seven coefficients named zeta_a, zeta_b, zeta_c, zeta_d, zeta_e, zeta_f and zeta_g. The polynomial is defined below.

Processing

[1] PhD candidate at the University of Southampton, see http://www.southampton.ac.uk/engineering/postgraduate/research_students/mf1o07.page, accessed 22/01/2014.

The module code resides in the file `performance_fixedWing_aircraft_petrol.jar` in the main model folder. The OSCAR simulation loads it upon model start as a dependency. Based on the additional VESSEL parameters described above, the module calculates fuel burn from first principles. The aircraft drag polar is defined as

$$c_D = k1 + k2 \times c_L + k3 \times (c_L)^2 \qquad \text{Eq. A-5}$$

where c_L and c_D are the lift and drag coefficients. The terms *k1*, *k2* and *k3* refer to the aircraft flight conditions take-off, cruise and landing, respectively. Each is defined as

$$k1 = k1\_a + k1\_b \times V + k1\_c \times V^2 \qquad \text{Eq. A-6}$$

where *V* is the current aircraft flight speed. The coefficients *k2* and *k3* are defined similarly. The specific fuel consumption ϕ is defined as a 4th degree polynomial based on function of engine power output as

$$\phi = sfc\_a + sfc\_b \times r + sfc\_c \times r^2 + sfc\_d \times r^3 \qquad \text{Eq. A-7}$$

where *r* is the ratio between the required engine shaft power and the maximum available power, *i.e.*

$$r = \frac{P_{req}}{P_{inst}} \qquad \text{Eq. A-8}$$

The available shaft power P_{inst} is defined as

$$P_{inst} = \frac{P_{req}}{\zeta} = \frac{D \times V}{\zeta} \qquad \text{Eq. A-9}$$

Where ζ is the propeller efficiency, *V* is the aircraft velocity and *D* is the current aircraft drag defined as

$$D = \frac{1}{2} \times \rho \times V^2 \times S \times c_D \qquad \text{Eq. A-10}$$

where ρ is the air density and S is the wing area. The current performance module is streamlined for low altitude UAS, therefore $\rho = 1.225\ \frac{kg}{m^3}$ independent of flight altitude. However, the model can be extended to include altitude-dependent air density using the standard atmosphere. The propeller efficiency ζ is defined as

$$\zeta = zeta\_a + zeta\_b \times y + zeta\_c \times y^2 + zeta\_d \times y^3 + zeta\_e \times y^4 + zeta\_f \times y^5 + zeta\_f \times y^6 \quad \text{Eq. A-11}$$

where the "zeta_" factors refer to the `zeta_` parameters described above. y is defined as

$$y = \frac{V}{\eta \times o_{prop}} \quad \text{Eq. A-12}$$

Where η s the propeller rounds per minute and o_{prop} is the propeller diameter.

The OSCAR simulation calls any custom performance module following the process of the generic performance module (see Section 4.9.1.2), *i.e.* the fuel burn is calculated after each SEGMENT. Fuel burn is calculated by subtracting the weight at the end of a SEGMENT W_2 from the weight at the start of a SEGMENT W_1 assuming constant flight speed and altitude. Under these assumptions

$$W_2 = W_1 - \frac{c}{2}\left\{\tan\left[\tan^{-1}\left(\frac{b}{c} + \frac{2}{c}W_1\right) - \frac{R}{u}\frac{c}{2}\right] - \frac{b}{c}\right\} \quad \text{Eq. A-13}$$

where R is the segment distance in metres and

$$c = [4a - b^2]^{0.5} \quad \text{Eq. A-14}$$

$$b = \frac{k_2}{k_3}qS \quad \text{Eq. A-15}$$

$$a = \frac{k_1}{k_3}(qS)^2 \quad \text{Eq. A-16}$$

$$q = \frac{1}{2}\rho V^2 \quad \text{Eq. A-17}$$

Outputs

The OSCAR simulation outputs and stores data in the same way as for generic performance module VESSELS (Section 4.9.1.3).

Appendix 10: Experimental setup

This appendix describes the three OSCAR simulation experiments “Single run”, “Interactive single run” and “Run fast” in turn.

Single run

This experiment requires users to input the experiment parameters manually within the AnyLogic IDE. The experiment loads the specified MISSION tables and executes all MISSIONS and TRACKS once. No outputs are saved in the database. This experiment is to be used for quick model validation by the user. Users can view the animation during runtime and use AnyLogic capabilities to check on every agent state at any time. Note that the Java applet and Java application version of this experiment do not allow experiment parameter alteration.

Interactive single run

This experiment is identical to the previous “Single Run” experiment except that users can amend input data dynamically during runtime and that outputs are written to the database after the simulation run. Upon starting the experiment, an intuitive GUI guides the user through selecting and amending the correct data, as described below.

1. Initially, the user must select one or more MISSIONS that the experiment should simulate, based on the available MISSION tables in the database (Figure A-9). Optional buttons cannot be clicked yet.

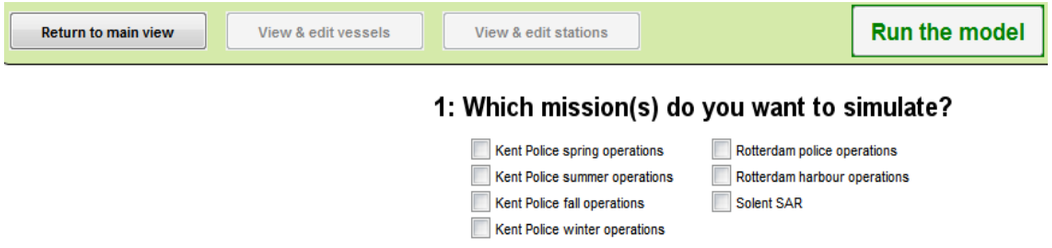


FIGURE A-9: INTERACTIVE SINGLE RUN EXPERIMENT MISSION SELECTION.

2. Upon selecting one or more MISSIONS, the GUI extends by displaying the MISSION table data in Figure A-10. Moreover, a descriptive pop-up explains the next step. Optional buttons are still non-functional.

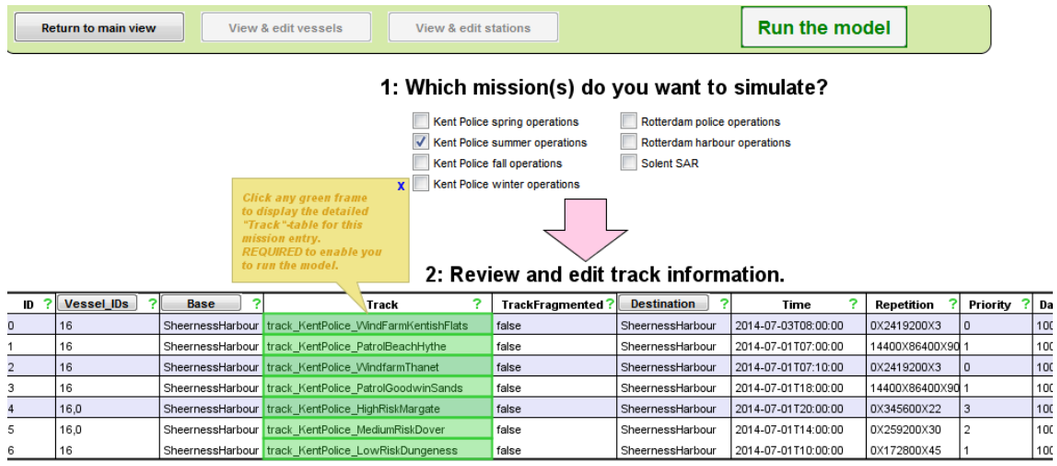


FIGURE A-10: INTERACTIVE SINGLE RUN EXPERIMENT TRACK SELECTION.

3. The user can click on any table entry and amend it. The experiment will use the amended value instead of the database value.
4. To be able to run the experiment (*i.e.* enable the buttons on top of the screen), the user is asked to click at least one green frame. This will load and display the respective database TRACK table below the current table (Figure A-11). This additional step is deemed necessary to demonstrate the full capability of the experiment setup to new users.

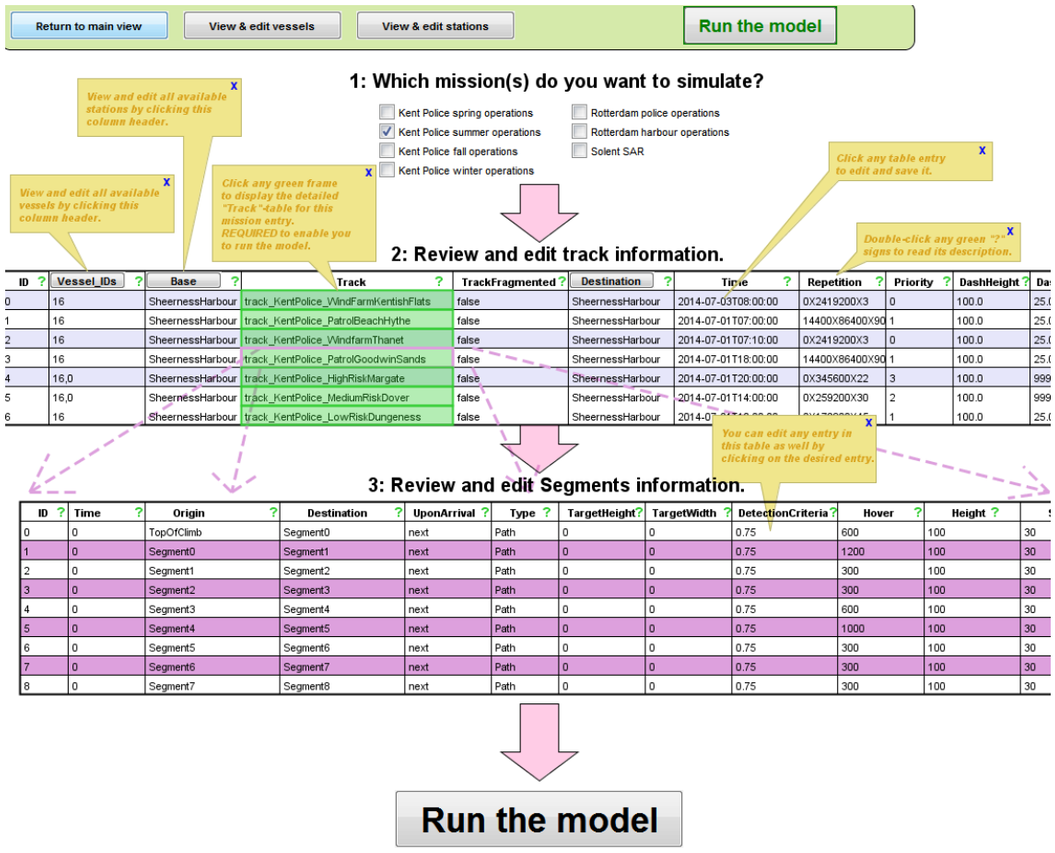


FIGURE A-11: INTERACTIVE SINGLE RUN EXPERIMENT SEGMENT SELECTION.

5. Additional pop-ups inform the user of possible actions. He can edit any visible database entry by clicking on the desired entry (Figure A-12). Moreover, the buttons at the top of the screen are now functional. There are several options.

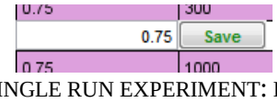


FIGURE A-12: INTERACTIVE SINGLE RUN EXPERIMENT: ENTER AND SAVE NEW VALUES.

6. The user can "view and edit VESSELS". This displays the database Vessel table as in Figure A-13. As before, any entry can be amended by clicking on it. Note that VESSEL COMPONENTS cannot be viewed and edited in the current version.

id	10	11
category	automotive	boat
type	car	boat
name	AudiA3	RNLI_Mersey_class
performanceModel	powerAgainstSpeed	powerAgainstSpeed
fuelType	petrol	petrol
occupants	3	6
cameraFOVhor	0.8	0.8
cameraFOVver	2	2
cameraPixelsHor	800	600
cameraPixelsVer	780	780
cameraTiltAngle	0.78	0.78

FIGURE A-13: INTERACTIVE SINGLE RUN EXPERIMENT VESSEL SETUP.

7. The user can "view and edit Bases". This displays the Base table as in Figure A-14. As before, any entry can be amended by clicking on it.

StationID	StationName	Longitude	Latitude
1	Bembridge	-1.06526291370392	50.6973915100098
2	Cowes	-1.29553866386414	50.7731971740723
3	Lymington	-1.52512788772583	50.7608795166016
4	Portsmouth	-1.01996278762817	50.7885704040527
5	Yarmouth	-1.50625288486481	50.7080955505371
6	LeeOnSolent	-1.19773149490356	50.8061370849609
8	Poole	-1.99048399925232	50.720142364502
9	Swanage	-1.96560299396515	50.6055870056152
10	Weymouth	-2.4572122097015 4	50.6104698181152

FIGURE A-14: INTERACTIVE SINGLE RUN EXPERIMENT BASE SETUP.

8. Once all amendments are done, the experiment can be started by clicking "Run the model".

The model executes the simulation once, applying inputs from the database except those amended by the user above. As with the "Single Run" experiment, the model animation is visible. After completion, all outputs are written to the output database.

"Run fast"

This experiment is structurally different from the previous experiments described. All experiment parameters must be defined in the AnyLogic IDE. This experiment will not display any animation to speed up model execution. Moreover, it allows running a number of replications to factor in the variations from random sampling. The number of replications must be defined in the AnyLogic IDE as well. The `randomSeed` parameter takes the value of the current replication to ensure that each replication uses different random number streams, generating variable outputs.

Once starting the experiment, it will use each available processor core to run one replication, thus speeding up total execution time. Replications are independent from each other. Every time a replication finishes, it writes all output data to the database, marking each entry with its current replication number in the column "iteration" (Figure A-15).

iteration	vesselID	componentID	segmentID	timeOfProblem
6	12	23	19	2014-12-20T13:41:30.0
6	12	21	48	2014-12-29T12:27:00.0
3	12	9999	-9999	2014-01-19T17:36:32.2
3	12	9999	-9999	2014-01-23T18:10:11.2
3	12	21	8	2014-01-29T14:58:00.0

FIGURE A-15: RUNFAST EXPERIMENT MARKS ALL OUTPUT DATA WITH COLUMN "ITERATION".

This allows post-processing to distinguish data from different replications, thus enabling calculations of statistical measures such as averages, standard deviations, *etc*.

Appendix 11: SULSA power

This appendix explains the rationale for the power-speed relation used for the electrically propelled SULSA UAS in Chapter 5. In level flight, Power *P* is defined as

$$P = D * V \qquad \text{Eq. A-18}$$

where *D* is drag and *V* is flight speed. In general, drag *D* can be defined as

$$D = \frac{1}{2} * \rho * V^2 * S * c_D \qquad \text{Eq. A-19}$$

where the density of air is assumed constant at $\rho = 1.225\,\mathrm{kg/m^3}$ (as SULSA flight altitudes are low), the wing area $S = 0.24\mathrm{m}^2$ and the coefficient of drag c_D is

$$c_D = k1_a + k3_a * c_L^2 \qquad \text{Eq. A-20}$$

where $k1_a = 0.0527$ and $k3_a = 0.0663$ are drag coefficients as in the custom performance model (Appendix 9) and the coefficient of lift c_L is

$$c_L = \frac{m*g}{\frac{1}{2}*\rho*V^2*S}. \qquad \text{Eq. A-21}$$

Here, the aircraft mass $m = 2.5\mathrm{kg}$ is assumed constant and g is the standard gravity. This yields the power-to-speed relation used, depicted in Figure 5-5.

Appendix 12: UAS inputs comparison

This appendix compares the inputs used for the different UAS designs in Chapter 5. Table A-9 summarises the OSCAR performance and camera setup inputs for the UAS designs for easy comparison. Note that the camera setup is identical to avoid output bias.

TABLE A-9: UAS DESIGNS PERFORMANCE AND CAMERA COMPARISON.

	DECODE	BBC	SULSA	2Seas
maximum speed (m/s)	24.0	37.0	19.9	45.0
search & loiter speed (m/s)	18.0	25.0	17.0	26.0
minimum speed (m/s)	10.0	13.7	10.0	14.0
dry weight (kg)	8.8	17.8	2.1	24.2
fuel weight (kg)	1.2	5.0	0.0	5.8
horizontal field-of-view (degrees)	91.7	91.7	91.7	91.7
vertical field-of-view (degrees)	91.7	91.7	91.7	91.7
horizontal pixels	680.0	680.0	680.0	680.0
vertical pixels	780.0	780.0	780.0	780.0
camera tilt angle (degrees)	68.8	68.8	68.8	68.8

Table A-10 compares the performance model inputs used to feed the petrol-engine performance model described in Section 4.9.2. SULSA, being propelled by an electric engine, used the standard performance model (Section 4.9.1) and cannot be compared to the other designs.

TABLE A-10: UAS PERFORMANCE MODEL PARAMETERS COMPARISON. NOTE: SULSA USES DIFFERENT PERFORMANCE MODEL.

	DECODE	BBC	2Seas
wing area (m^2)	1.12	1.496	1.4
maximum lift coefficient	1.5	1.908	1.57356
propeller diameter (m)	0.41564	0.508	0.4572
k1_a	0.045	0.032	0.043676

k1_b	0	0	0
k1_c	0	0	0
k2_a	0	-0.002	0
k2_b	0	0	0
k2_c	0	0	0
k3_a	0.0334	0.04	0.04207
k3_b	0	0	0
k3_c	0	0	0
installed power (W)	1490	3310	3400
propeller RPM	8000	6500	7000
sfc_a	450	459.857	1339.784
sfc_b	0	0	-970.149
sfc_c	0	0	297.7281
sfc_d	0	0	0
zeta_a	60	-0.129	577.4458
zeta_b	0	287.776	-6813.39
zeta_c	0	-1788.785	34458.75
zeta_d	0	11181.081	-87795.4
zeta_e	0	-33616.437	120137.5
zeta_f	0	46099.676	-83881
zeta_g	0	-23892.233	23317.32

Table A-11 compares all component parameters used for the UAS designs in Chapter 5. Most parameters are identical for all designs due to lack of trustworthy reliability data for UAS components. The differences are based on the fact that SULSA, being a laser-sintered aircraft with an inverted v-tail, does not feature a number of COMPONENTS such as the vertical tail structure. Accordingly, it has a unique component “aft fuselage” representing a much more endurable fuselage component. Moreover, the 3i aircraft has two engines, increasing the redundancies of the engines, throttle servos, ignitions and propellers.

TABLE A-11: UAS COMPONENT INPUTS COMPARISON.

	Weibull life				Weibull eta				Weibull beta (FH:				loss probability from				maintenance time				quantity onboard			
	DEC	BBC	SUL	2Se	DEC	BBC	SUL	2Se	DEC	BBC	SUL	2Se	DEC	BBC	SUL	2Se	DEC	BBC	SUL	2Se	DEC	BBC	SUL	2Se
wing main structure	FH	FH		FH	2	2		2	600	600		600	1	1		1	1	1		1	1	1		1
wing aileron control mechanism	FH	FH	FH	FH	2	2	2	2	600	600	600	600	0.4	0.4	0.4	0.4	1	1	1	1	2	2	2	2
aileron servo	FH		FH	FH	2	2	2	2	500	500	500	500	0.4	0.4	0.4	0.4	2	2	2	2	4	4	4	4
vertical tail structure	FH	FH		FH	2	2		2	600	600		600	0.1	0.1		0.1	1	1		1	1	1		2
rudder control mechanism	FH	FH	FH	FH	2	2	2	2	600	600	600	600	0.1	0.1	0.1	0.1	1	1	1	1	2	2	2	2
rudder servo	FH	FH	FH	FH	2	2	2	2	500	500	500	500	0.2	0.2	0.2	0.2	2	2	2	2	2	2	2	2
horizontal tail structure	FH	FH		FH	2	2		2	600	600		600	1	1		1	1	1		1	1	1		1
elevator control mechanism	FH	FH	FH	FH	2	2	2	2	600	600	600	600	1	1	1	1	1	1	1	1	4	4	4	4
elevator servo	FH	FH	FH	FH	2	2	2	2	500	500	500	500	1	1	1	1	2	2	2	2	4	4	4	4
aft fuselage			FH				2				2400				1				1				1	
Fuselage main structure	FH	FH		FH	2	2		2	600	600		600	0.7	0.7		0.7	1	1		1	1	1		1
Fuselage secondary structure	FH	FH		FH	2	2		2	500	500		500	0.05	0.05		0.05	0.5	0.5		0.5	1	1		1
main undercarriage	FC	FC		FC	2	2		2	800	800		800	0.8	0.8		0.8	1	1		1	1	1		1
front undercarriage	FC	FC		FC	2	2		2	500	500		500	0.3	0.3		0.3	1	1		1	1	1		1
engine	FH	FH	FH	FH	2	2	2	2	300	300	2400	300	1	1	1	1	3	3	3	3	1	1	1	2
throttle servo	FH	FH	FH	FH	2	2	2	2	500	500	500	500	1	1	1	1	2	2	2	2	1	1	1	2
ignition	FH	FH		FH	2	2		2	600	600		600	1	1		1	1	1		1	1	1		2
propellor	FH	FH	FH	FH	2	2	2	2	300	300	300	300	1	1	1	1	0.5	0.5	0.5	0.5	1	1	1	2
power generator	FH	FH		FH	2	2		2	400	400		400	0.3	0.3		0.3	2	2		2	1	1		1
6v flight systems battery	FC	FC	FC	FC	10	10	10	10	1000	1000	1000	1000	1	1	1	1	0.5	0.5	0.5	0.5	2	2	2	2
Receiver	FH	FH	FH	FH	1	1	1	1	1000	1000	1000	1000	1	1	1	1	0.5	0.5	0.5	0.5	1	1	1	1
gps aerial	FH	FH	FH	FH	1	1	1	1	1000	1000	1000	1000	1	1	1	1	0.5	0.5	0.5	0.5	1	1	1	1
comms aerial	FH	FH	FH	FH	1	1	1	1	1000	1000	1000	1000	1	1	1	1	0.5	0.5	0.5	0.5	1	1	1	1
autopilot	FH	FH	FH	FH	1	1	1	1	1000	1000	1000	1000	1	1	1	1	1	1	1	1	1	1	1	1
autopilot servo plug board	FH	FH	FH	FH	2	2	2	2	500	500	500	500	1	1	1	1	1	1	1	1	1	1	1	1
static port	FH	FH	FH	FH	1	1	1	1	1000	1000	1000	1000	0.5	0.5	0.5	0.5	1	1	1	1	1	1	1	1
static hose	FH	FH	FH	FH	2	2	2	2	1000	1000	1000	1000	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	1	1	1
pitot probe	FH	FH	FH	FH	2	2	2	2	5000	5000	5000	5000	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	1	1	1
pitot tubing	FH	FH	FH	FH	2	2	2	2	1000	1000	1000	1000	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	1	1	1
payload	FH	FH	FH	FH	1	1	1	1	1000	1000	1000	1000	0	0	0	0	1	1	1	1	1	1	1	1

Legend: DEC=DECODE, SUL=SULSA, 2Se=2Seas, FH=flight hours, FC=flight cycles

Changes

Appendix 13: Mersey lifeboat performance data

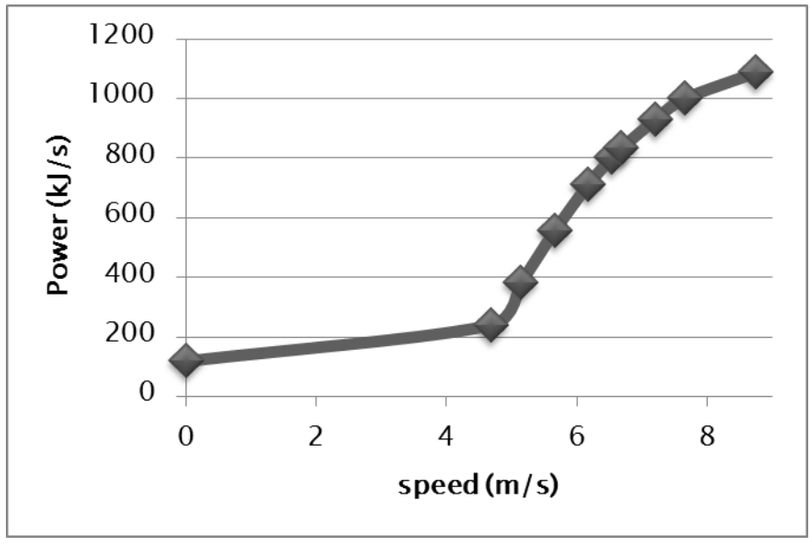
Based on publicly available fuel consumption charts[1], it is possible to derive the power consumption for various speeds for the Mersey lifeboat as follows:

Speed (knots)	Speed (m/s)	δ (gal/hr)	δ (litres/s)	δ (kg/s)	Power (J/s)
0	0	3	0.00315451	0.002775969	118533.8609
9.101	4.681958889	6.07	0.006382625	0.00561671	239833.5118
10	5.144444444	9.69	0.010189067	0.008966379	382864.3706
11	5.658888889	14.05	0.014773621	0.013000786	555133.5817
12	6.173333333	17.97	0.018895514	0.016628052	710017.8266
12.7	6.533444444	20.24	0.021282426	0.018728535	799708.448
13	6.687777778	21.09	0.022176204	0.01951506	833293.0419
14	7.202222222	23.5	0.024710327	0.021745088	928515.2435
14.894	7.662135556	25.29	0.026592518	0.023401416	999240.4471
17	8.745555556	27.5	0.02891634	0.025446379	1086560.391

Constants:	
litres/gal	3.78541178
density Diesel (kg/m^3)	880
calorific value Diesel (J/kg)	42700000

NOTE: extreme values for Speed=0 m/s and Speed=8.74 m/s are estimated.

OSCAR interpolates linearly between data points, leading to the following relationship between speed and energy consumption:



[1] See http://www.44mlb.com/fuel-consumption-charts.html, accessed on 10/07/2013.

Appendix 14 Cost model parameters rationale

This table explains the parameters used in the cost model for the decision support case study (see Section 5.4.3). All currency conversion based on www.oanda.com exchange rate from 19/07/2013 where £1 = $1.52563.

Parameter	Value	Rationale
Cost per UAS maintenance hour	$ 34.63	Average yearly income of UK airline maintenance personnel is £ 43,658[1]. Assume 40-hour workweek and 4 weeks holiday.
Cost per UAS maintenance operation	$ 1,830.76	Rent for building at £ 500; Average part cost: £ 500; Other expenditure: £ 200;
Cost per UAS flight hour	$ 172.09	Average yearly income of UK pilots is £ 72,184[1]. Assume 40-hour week and 4 weeks holidays. Assume 3 people needed to control UAS: 1 pilot, 1 ATC support, 1 image analyser)
Price per kg petrol	$ 2.88 (used for DECODE, BCC and 3i UAS)	Average price per kg petrol in Southampton on 19/07/2013.
Price per kg diesel	$ 1.40 (used for lifeboats)	Price for maritime commercial diesel in Solent area on 06/06/2013.
Price per kWh	$ 0.26 (used for SULSA UAS)	Average price based on Southampton rates for 1 kWh from British Gas on 19/07/2013.
Cost per UAS launch	$ 114.73	Use 2 people paid at pilot rate[1] for one hour (including packing up after landing).
UAS acquisition	$ 15,256 (DECODE) $ 30,512 (BBC & 3i) $ 5,340 (SULSA)	Based on simple cost model developed by DECODE research[2].
Cost per image	$ 0.0458	High quality SLR camera costs about £ 3,000 and has mean life of 100,000 images.
Cost per GB of data	$ 0.01	Extrapolated from average hard disk prices 1980-2009[3].

[1] See http://www.caa.co.uk/docs/80/airline_data/2009Annual/Table_1_14_Airline_Personnel_Cost_UK_and_Overseas_2009.pdf, accessed 19/07/2013.

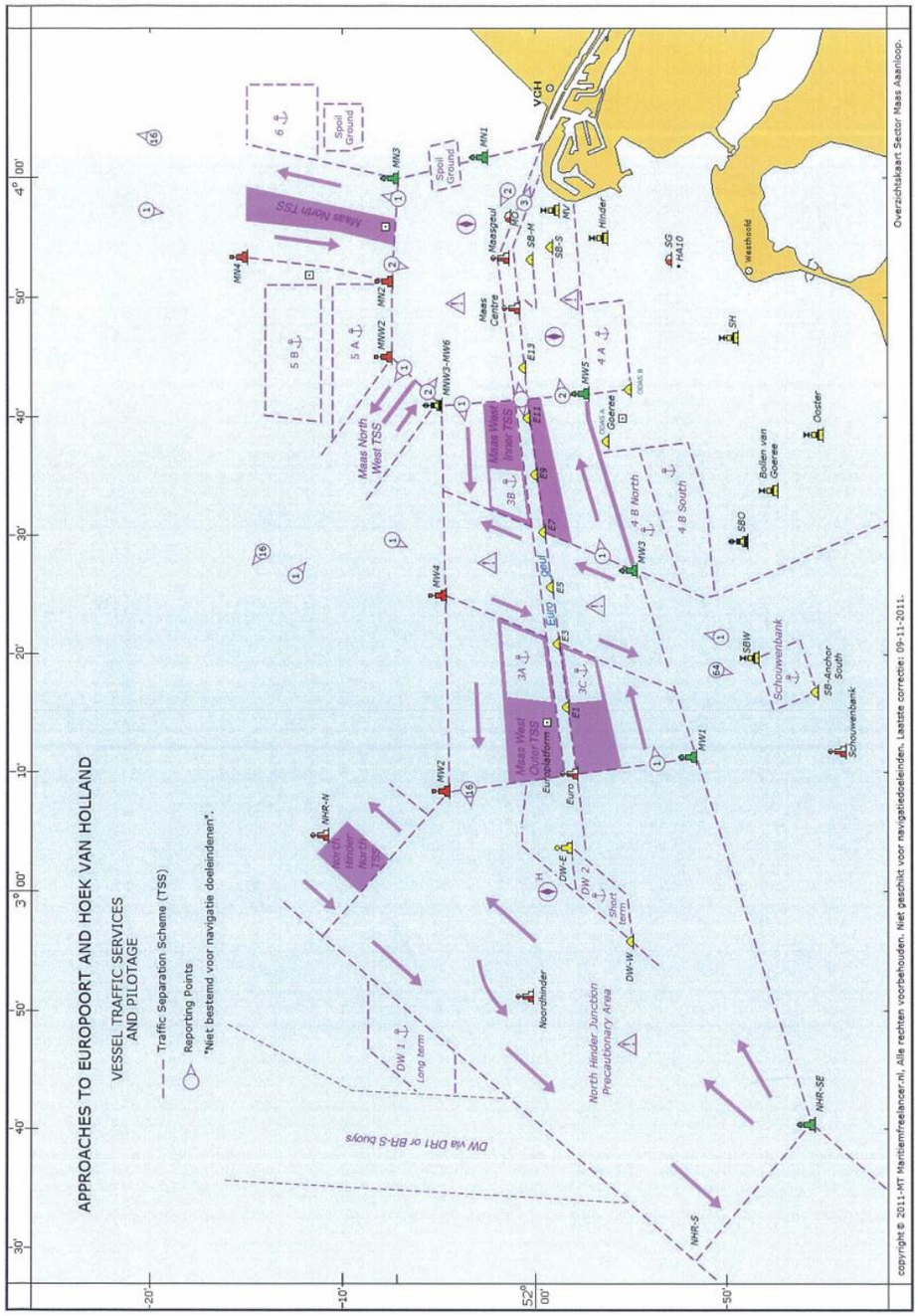
[2] See http://www.southampton.ac.uk/~decode/, accessed 22/07/2013.

Lifeboat acquisition cost per year	$ 76,281	Lifeboat similar to Mersey class costs £ 1.5M[4]. Assume lifeboat lifetime is 30 years and cost is amortised yearly.
Cost per lifeboat operating hour	$ 405.82	Daily RNLI running costs are £ 385,000 for 330 lifeboats[4]. Assume lifeboat operates for 4.4 hours daily. Includes launch costs!

[3] See http://www.mkomo.com/cost-per-gigabyte, accessed 22/07/2013.
[4] See http://rnli.org/aboutus/aboutthernli/Pages/Running-costs.aspx, accessed 19/07/2013.

Appendix 15: Port of Rotterdam offshore map

This map[1] depicts the offshore anchor position areas used in Chapter 6.



[1] Published with kind permission from M. J. Tolhuisen at www.Maritiemfreelancer.nl

Appendix 16: Replications setup

This appendix describes the rationale for defining the number of replications for the case study in Chapter 6. The same methodology was applied for the case study in Chapter 5.

The confidence interval method combined with the graphical method (Robinson 2004) were used to assess the required number of replications for each of the optimisation iterations. For this, the initial design simulation was run for 850 replications, a number estimated high enough (based on previous experience with the simulation) to achieve very good estimates of mean performance. All analysis in Section 6.5 bases upon 850 replications. To reduce runtime for the subsequent design iterations in Section 6.6 and 6.7, the minimum number of replications was investigated as follows.

For each OSCAR output, a cumulative mean $\bar{X}$ was used to describe a confidence interval as

$$\Omega = \bar{X} \pm T_{n-1,\alpha/2} * \frac{\sigma}{\sqrt{n}} \qquad \text{Eq. A-22}$$

where Ω is the confidence interval, n is the number of replications, $T_{n-1,\alpha/2}$ is the value from the Student's t-distribution with n-1 degrees of freedom and a significance level of $\alpha/2$. The common significance level $\alpha = 5\%$ is used. σ refers to the standard deviation of the OSCAR output and is defined as

$$\sigma = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1} \qquad \text{Eq. A-23}$$

where X_i is the result of replication i. Plotting the deviation from the mean (as in Figure A-16) and the confidence interval (as in Figure A-17) for each OSCAR output allowed to identify the most critical output as the `maintenance time`.

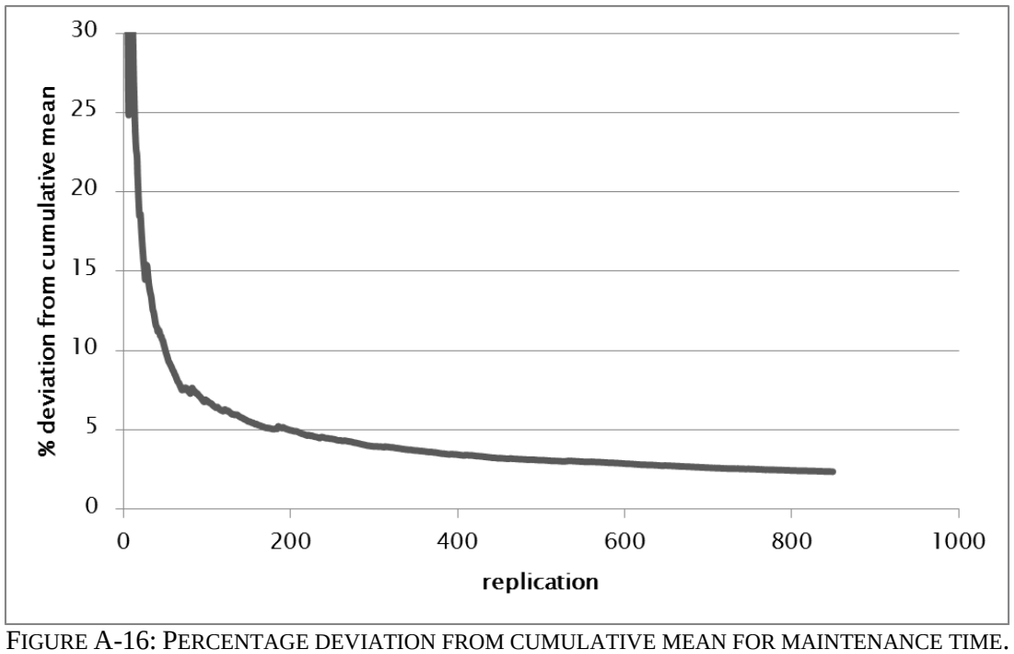


FIGURE A-16: PERCENTAGE DEVIATION FROM CUMULATIVE MEAN FOR MAINTENANCE TIME.

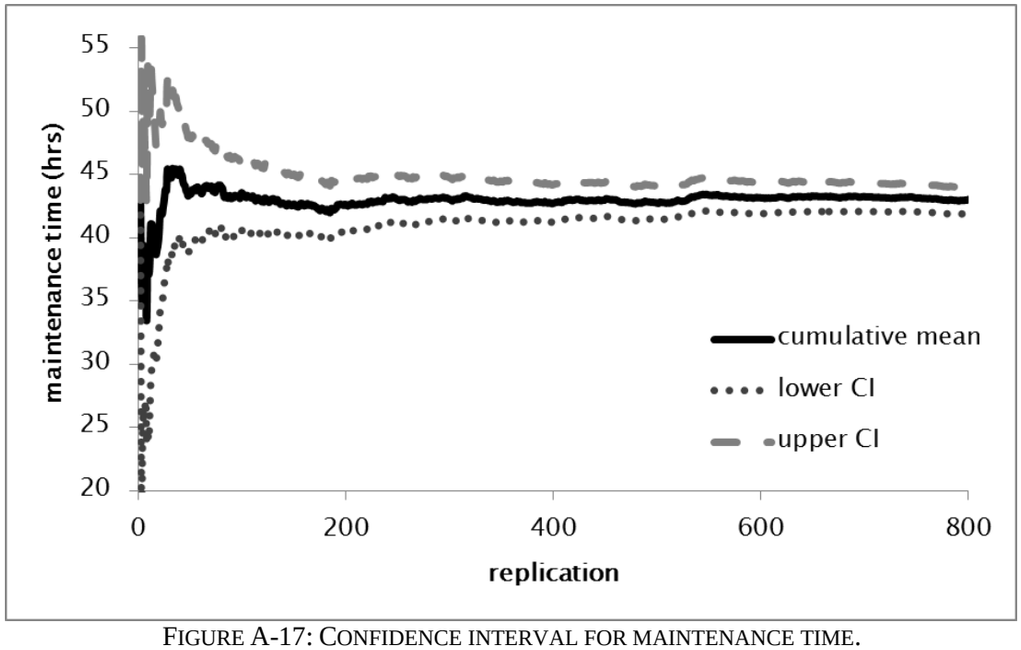


FIGURE A-17: CONFIDENCE INTERVAL FOR MAINTENANCE TIME.

In order to achieve less than 5 % deviation, 200 replications are required, as seen in Figure A-16. However, the confidence interval should not only be narrow enough but the cumulative mean should be sufficiently flat. Therefore, the graphical method was used as a sanity check. It was found that most OSCAR outputs feature a rugged cumulative mean at 200 iterations. Therefore, number of replications should be set to 400 to avoid cumulative mean errors (compare Figure A-18).

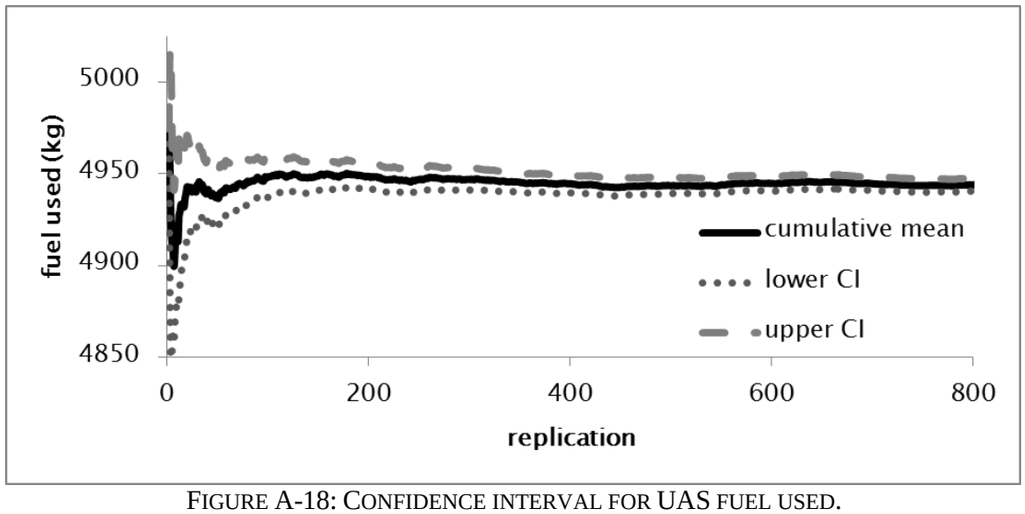


FIGURE A-18: CONFIDENCE INTERVAL FOR UAS FUEL USED.

To add a margin of safety for the design iterations, 500 replications will be used for the first and second design iteration.

In order to reduce the number of required replications, an LPτ pseudorandom sequence or Latin hypercube sampling can be employed in future work, as suggested by Keane (2012).

Appendix 17: 3i component details

This appendix shows the COMPONENTS used for the 3i and 3i-a UAS designs in Chapter 6. Changes between the designs are highlighted in yellow.

	Weibull life measure		Weibull eta		Weibull beta (FH: hours, FC: cycles)		loss probability from inflight failure		maintenance time (hours)		quantity onboard		Roustness scaling factor	
	3i	3i-a	3i	3i-a	3i	3i-a	3i	3i-a	3i	3i-a	3i	3i-a	3i	3i-a
wing main structure	FH	FH	2	2	600	600	1	1	1	1	1	1	0	1
wing aileron control mechanism	FH	FH	2	2	600	600	0.4	0.4	1	1	2	2	0	0
aileron servo	FH	FH	2	2	500	500	0.4	0.4	2	2	4	4	0	0
vertical tail structure	FH	FH	2	2	600	600	0.1	0.1	1	1	2	2	0	0
rudder control mechanism	FH	FH	2	2	600	600	0.1	0.1	1	1	2	2	0	0
rudder servo	FH	FH	2	2	500	500	0.2	0.2	2	2	2	2	0	0
horizontal tail structure	FH	FH	2	2	600	600	1	1	1	1	1	1	0	1
elevator control mechanism	FH	FH	2	2	600	600	1	1	1	1	4	4	0	0
elevator servo	FH	FH	2	2	500	500	1	1	2	2	4	4	0	0
Fuselage main structure	FH	FH	2	2	600	600	0.7	0.7	1	1	1	1	0	1
Fuselage secondary structure	FH	FH	2	2	500	500	0.05	0.05	0.5	0.5	1	1	0	0
main undercarriage	FC	FC	2	2	800	800	0.8	0.8	1	1	1	1	0	0
front undercarriage	FC	FC	2	2	500	500	0.3	0.3	1	1	1	1	0	0
engine	FH	FH	2	2	300	300	1	1	3	3	2	2	0	0
throttle servo	FH	FH	2	2	500	500	1	1	2	2	2	2	0	0
ignition	FH	FH	2	2	600	600	1	1	1	1	2	2	0	0
propellor	FH	FH	2	2	300	300	1	1	0.5	0.5	2	2	0	0
power generator	FH	FH	2	2	400	400	0.3	0.3	2	2	1	1	0	0
6v flight systems battery	FC	FC	10	10	1000	1000	1	1	0.5	0.5	2	2	0	0
Receiver	FH	FH	1	1	1000	1000	1	1	0.5	0.5	1	2	0	0
gps aerial	FH	FH	1	1	1000	1000	1	1	0.5	0.5	1	2	0	0
comms aerial	FH	FH	1	1	1000	1000	1	1	0.5	0.5	1	2	0	0
autopilot	FH	FH	1	1	1000	1000	1	1	1	1	1	2	0	0
autopilot servo plug board	FH	FH	2	2	500	500	1	1	1	1	1	2	0	0
static port	FH	FH	1	1	1000	1000	0.5	0.5	1	1	1	2	0	0
static hose	FH	FH	2	2	1000	1000	0.5	0.5	0.5	0.5	1	1	0	0
pitot probe	FH	FH	2	2	5000	5000	0.5	0.5	0.5	0.5	1	1	0	0
pitot tubing	FH	FH	2	2	1000	1000	0.5	0.5	0.5	0.5	1	1	0	0
payload	FH	FH	1	1	1000	1000	0	0	1	1	1	1	0	0

Legend: FH=flight hours, FC=flight cycles Change

REFERENCES

Abbas-Bayoumi, A. & Becker, K., 2011. An Industrial View on Numerical Simulation for Aircraft Aerodynamic Design. *Journal of Mathematics in Inudstry*, 1, pp.1–14.

Adelantado, M., 2004. Rapid Prototyping of Airport Advanced Operational Systems and Procedures through Distributed Simulation. *Simulation*, 80.

Alonso, J.J., LeGresley, P. & Pereyra, V., 2009. Aircraft Design Optimization. *Mathematics and Computers in Simulation*, 79, pp.1948–1958.

Al-Salka, M.A., 2001. *Computer-Aided Design for Life-Cycle.* In: Concurrent Engineering In Product Design And Development. I. Moustapha, ed. New Delhi, India: New Age International, pp. 229–249.

Amirreze, K. et al., 2013. A New Systematic Approach in UAV Design Analysis Based on SDSM Method. *Journal of Aeronautics and Aerospace Engineering*, S1 (001).

Andersson, M. & Olsson, G., 1998. A Simulation-based Decision Support Approach for Operational Capacity Planning in a Customer Order-driven Assembly Line. In D. J. Medeiros et al., eds. *Proceedings of the 1998 Winter Simulation Conference*. pp. 935–941.

Anemaat, W.A.J. et al., 2013. Software Tool Development to Improve the Airplane Preliminary Design Process. In C. B. et al., ed. *20th ISPE International Conference on Concurrent Engineering*. IOS Press, pp. 12–18.

Ashok, S.V., 2013. *An Integrated Product – Process Development (IPPD) Based Approach for Rotorcraft Drive System Sizing, Synthesis and Design Optimization*. Atlanta, USA: Georgia Institute of Technology. PhD thesis.

Austin, R., 2010. *Unmanned Aircraft Systems: UAVS Design, Development and Deployment* I. Moir, A. Seabridge, & R. Langton, eds., John Wiley and Sons, Ltd.

Backlund, G., 2000. *The Effects of Modelling Requirements in Early Phases of Buyer-Supplier Relation*. Linkoeping, Sweden: Linkoeping University. PhD thesis.

Bandte, O., 2000. *A Probabilistic Multi-Criteria Decision Making Technique for Conceptual and Preliminary Aerospace Systems Design*. Georgia, USA: Georgia Institute of Technology. PhD thesis. Available at: http://www.reli.ari.ac.ir/english%20papers/-10.1.1.83.5914.pdf.

Banks, J. & Gibson, R.R., 2009. The ABCs of Simulation Practice. In *Software Solutions*. Software Solutions. INFORMS, 1, pp. 16–21.

Batty, M. et al., 2012. *Agent-Based Models of Geographical Systems* A. J. Heppenstall et al., eds., New York, USA: Springer. Available at: http://www.springer.com/social+sciences/-population+studies/book/978-90-481-8926-7.

Bell, P.C. & O'Keefe, R.M., 1987. Visual Interactive Simulation—History, Recent Developments, and Major Issues. *Simulation*, 49, pp.109–116.

Bergh, J. Van den et al., 2013. Aircraft Maintenance Operations: State of the Art. *status: published*. Available at: https://lirias.kuleuven.be/bitstream/123456789/426843/1/-13HRP09.pdf.

Bertoni, M. et al., 2013. Using 3D CAD Models for Value Visualisation: An Approach with SIEMENS NX HD3D Visual Reporting. *Computer-Aided Design & Applications*, 10. Available at: http://www.bth.se/fou/forskinfo.nsf/all/3ed9a6e34ae7d77ec1257b63002e9b-ca/file/CADandA_Bertoni_reviewed.pdf.

Birkin, M. & Wu, B., 2012. Agent-Based Models of Geographical Systems. In A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty, eds. New York, USA: Springer, pp. 51–68.

Boehm, B.W., 1988. A Models Model of Software Development and Enhancement. *Computer*, 21, pp.61–72.

Bond, A.H. & Ricci, R.J., 1992. Cooperation in Aircraft Design. *Research in Engineering Design*, 4, pp.115–130. Available at: ftp://ftp.eng.umd.edu/afs/glue.umd.edu/home/glue/-a/h/aharrin1/pub/Airplane%20Design/red92.pdf.

Bosse, T. et al., 2013. An Integrated Multi-Agent Model for Modelling Hazards within Air Traffic Management. In *Proceedings of the 2013 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. IEEE, pp. 179–186.

Box, G.E.P. & Draper, N.R., 1987. *Empirical Model-Building and Response Surfaces*, Hoboken, NJ, USA: Wiley & Sons, Ltd.

Brooks, C., 2001. *Survival in Cold Water*, 40 Mount Hope Avenue, Dartmouth, Nova Scotia, B2Y4K9 Canada: Survival Systems Limited.

Brown, D. et al., 2005. Spatial Process and Data Models: Toward Integration of Agent-Based Models and GIS. *Journal of Geographical Systems*, 7, pp.25–47.

Bussmann, S., Jennings, N. & Wooldridge, M., 2004. *Multiagent System for Manufacturing Control: A Design Methodology*, New York, USA: Springer.

Cassidy, P.F., Gatzke, T.D. & Vaporean, C.N., 2008. Integrating Synthesis and Simulation for Conceptual Design. In *Proceedings of the 46th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, NV, USA: AIAA.

Castagne, S., Curran, R. & Collopy, P., 2009. Implementation of Value-Driven Optimisation for the Design of Aircraft Fuselage Panels. *International Journal of Production Economics*, 117, pp.381–388.

Chan, L.K. & Wu, M.L., 2002. Quality Function Deployment: A Literature Review. *European Journal of Operational Research*, 143, pp.463–497.

Chapman, D., 1987. Planning for Conjunctive Goals. *Artificial Intelligence*, 32, pp.333–377.

Charette, R.N., 2009. This Car Runs on Code. IEEE Spectrum, New York, USA. Available at: http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code.

Chen, H. et al., 2009. Research on Search Probability and Camera Footprint of Region Coverage for UAVs. In *IEEE International Conference on Control and Automation*. Christchurch, New Zealand: IEEE, pp. 1920–1924.

Chen, Y. et al., 2012. Design and Simulation of Multi-Agent System for Marine Engineering Platform Automation. In *Proceedings of the Third International Conference on Mechanic Automation and Control Engineering (MACE)*. Washington, D.C., USA: IEEE, pp. 1526–1529.

Cheung, J. et al., 2010. Application of Value-Driven Design to Commercial Aero-Engine Systems. In *10th AIAA Aviation Technology, Integration and Operations Conference*. AIAA2010-9058. Fort Worth, Texas: AIAA.

Cheung, J.M.W., Scanlan, J.P. & Wiseall, S.S., 2009. An Aerospace Component Cost Modelling Study for Value-driven Design. In *CIRP IPS2 Conference*. University of Cranfield.

Cioppa, T.M., Lucas, T.W. & Sanchez, S.M., 2004. Military Applications of Agent-Based Simulations. In R. G. Ingalls et al., eds. *Proceedings of the 2004 Winter Simulation Conference*. Washington, D.C., USA: INFORMS, pp. 171–180.

Collins, P. ed., 2012. *Buy or Rent?*, The Economist (January 21st, 2012). Available at: http://www.economist.com/node/21543195.

Collopy, P.D., 2007. *Adverse Impact of Extensive Attribute Requirements on the Design of Complex Systems*, AIAA2007-7820. Urbana, Illinois: AIAA. Available at: http://vddi.org/reqts.pdf.

Collopy, P.D., 2001. *Economic-based distributed Optimal Design*, AIAA2001-4675. AIAA.

Collopy, P.D., 2008. Value of the Probability of Success. In *AIAA SPACE 2008 Conference and Exposition*. San Diego, California: AIAA.

Collopy, P.D. & Hollingsworth, P., 2009. Value-Driven Design. In *9th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*. AIAA2009-7099. Hilton Head, South Carolina: AIAA.

Collopy, P.D. & Hollingsworth, P., 2011. Value-Driven Design. *Journal of Aircraft*, 48 (3), pp.749–759.

Collopy, P.D. & Poleacovschi, C., 2012. Validating Value-Driven Design. In R. Curran et al., eds. *Proceedings of the Third International Air Transport and Operations Symposium*. Delft, The Netherlands: IOS Press, pp. 3–13.

Corke, T.C., 2003. *Design of Aircraft*, Upper Saddle River, NJ, USA: Pearson Education, Inc.

Cox, T.H. et al., 2004. *Civil UAV Capability Assessment*, NASA. Available at http://www.uavm.com/images/NASA_UAV_Capabilities_Assessment-2004.pdf.

Crooks, A.T., 2008. Constructing and Implementing an Agent-Based Model of Residential Segregation through Vector GIS. In *UCL Working Paper Series*. UCL Working Paper Series. London, UK: University College London. Available at: http://eprints.ucl.ac.uk/-15185/1/15185.pdf.

Cullinane, K. & Khanna, M., 1998. Economies of Scale in Large Container Ships. *Journal of Transportation Economics and Policy*, 33 (2), pp.185–208.

Curran, R., 2010. *Value-Driven Design and Operational Value.* Encyclopedia of Aerospace Engineering. In R. Blockley & W. Shyy, eds. Hoboken, NJ, USA: John Wiley and Sons, Ltd., pp. 1–11.

Curran, R. et al., 2005. Integrating Aircraft Cost Modelling into Conceptual Design. *Concurrent Engineering*, 13 (4), pp.321–330.

Curran, R. et al., 2012. Value Operations Methodology (VOM) Applied to Medium-Range Passenger Airliner Design. *Journal of Aerospace Operations*, 1, pp.3–27. Available at: http://iospress.metapress.com/content/q32t3068857j6276/fulltext.pdf.

Curran, R., Raghunathan, S. & Price, M., 2004. Review of Aerospace Engineering Cost Modelling: the Genetic Causal Approach. *Progress in Aerospace Sciences*, 40 (8), pp.487–534. Available at: http://www.sciencedirect.com/science/article/pii/-S0376042104000594.

Cusumano, M. et al., 2003. Software Development Worldwide: the State of the Practice. *IEEE Software*, 20 (6), pp.28–34. Available at: http://ebiz.mit.edu/research/papers/-178_Cusumano_Intl_%20Comp.pdf.

Dalamagkidis, K., Valavanis, K.P. & Piegl, L.A., 2011. *On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations*, New York, USA: Springer.

Davidsson, P., 2002. Agent-Based Social Simulation: A Computer Science View. *Journal of Artificial Societies and Social Simulation*, 5 (1).

Hagel, C., 1999. *Design Criteria for Military Systems, MIL-STD-1472F*, 1400 Defense Pentagon,Washington DC, USA: US Department of Defense. Available at: http://www.everyspec.com/MIL-STD/MIL-STD-1400-1499/MIL-STD-1472F_208/.

Delaurentis, D.A., Mavris, D.N. & Schrage, D.P., 1996. System Synthesis in Preliminary Aircraft Design Using Statistical Methods. In *Proceedings of the 20th International Council of the Aeronautical Sciences (ICAS) Congress*. Sorrento, Italy: Georgia Institute of Technology. Available at: http://hdl.handle.net/1853/6282.

Deshpande, S. et al., 2013. ADML: Aircraft Design Markup Language for Multidisciplinary Aircraft Design and Analysis. Blacksburg, Va, USA: Virginia Polytechnic Institute &

State Unversity. Available at: http://vtechworks.lib.vt.edu/bitstream/handle/10919/-24826/admlJAIS13.pdf?sequence=1.

Doll, T.J. et al., 1998. Robust, Sensor-Independent Target Detection and Recognition Based on Computational Models of Human Vision. *Optical Engineering*, 37, pp.2006–2021.

Duquette, M., 2009. Effects-Level Models for UAV Simulation. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference*. Chicago, Illinois: AIAA, pp. 1155–1166.

Egan, J., 2011. *The Unmanned Initiative: A Strategic Appraisal of Coast Guard Unmanned Aerial Systems*, Norfolk, VA, USA: National Defence University - School of Joint Advanced Warfighting. Available at: http://www.dtic.mil/dtic/tr/fulltext/u2/a545609.pdf.

Engler, W.O., 2013. *A Methodology for Creating Expert-based Quantitative Models for Early Phase Design*. Georgia Institute of Technology. PhD thesis. Available at: https://-smartech.gatech.edu/bitstream/handle/1853/47670/engler_william_o_201305_phd.pdf?sequence=1.

ESRI, 1998. *ESRI Shapefile Technical Description*, Redlands, Ca, USA: Environmental Systems Research Institute, Inc. Available at: http://www.esri.com/library/whitepapers/-pdfs/shapefile.pdf.

Ferraro, M. et al., 2012. Toward Value-Driven Design of a Small, Low-Cost UAV. In *53rd AIAA/ASME/ASCE/AHS/AS Structures, Structural Dynamics and Materials Conference*. AIAA2012-1723. Honolulu, Hawaii: AIAA.

Fielding, J.P., 1999. *Introduction to Aircraft Design*, Cambridge, UK: Cambridge University Press.

Forsberg, K. & Mooz, H., 1999. System Engineering for Faster, Cheaper, Better. In *Proceedings of the Ninth Annual International Symposium on Systems Engineering*. Brighton, UK: INCOSE.

Frangopol, D.M. & Maute, K., 2003. Life-Cycle Reliability-Based Optimization of Civil and Aerospace Structures. *Computers and Structures*, 81, pp.397–410.

Fu, M.C., Glover, F.W. & April, J., 2005. Simulation Optimization: A Review, New Developments, and Applications. In M. E. Kuhl et al., eds. *Proceedings of the 2005 Winter Simulation Conference*. Availabe at: http://leeds-faculty.colorado.edu/glover/-WSC2005.pdf.

Fulton, E.A., Smith, A.D.M. & Johnson, C.R., 2003. Effect of Complexity on Marine Ecosystem Models. *Marine Ecology Progress Series*, 253, pp.1–6.

Garcia, A., 2010. High Speed, Energy Consumption and Emissions. Technical Report. International Union of Railroads, 16 Rue Jean Rey, Paris, France.

Gao, C., Johnson, E. & Smith, B., 2009. Integrated Airline Fleet and Crew Robust Planning. *Transportation Science*, 43, pp.2–16.

Glas, M., 2013. *Ontology-Based Model Integration for the Conceptual Design of Aircraft*. Munich, Germany: Technische Universitaet Muenchen. PhD thesis.

Glickman-Weiss, E., Hearon, C. & Nelson, A., 1997. Does Shivering Thermogenesis Enhance the Individual's Ability to Maintain Rectal Temperature During Immersion in Cold Water. *Wilderness & Environmental Medicine*, 8 (1), pp.3–7.

Goerzen, C., Kong, Z. & Mettler, B., 2010. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57 (1-4), pp.65–100. Available at: http://dx.doi.org/10.1007/s10846-009-9383-1.

Golden, F. & Tipton, M., 1987. Human Thermal Responses During Leg-only Exercise in Cold Water. *The Journal of Physiology*, 391 (1), p.399.

Gorissen, D., Quaranta, E., Ferraro, M., Schumann, B., Schaik, J. van, Keane, A., et al., 2014. Value-Based Decision Environment: Vision and Application. *Journal of Aircraft (Accepted for publication)*: AIAA.

Gundlach, J., 2012. *Designing Unmanned Aircraft Systems: A Comprehensive Approach*, AIAA Education Series.

Hauser, J. & Clausing, D., 1988. The House of Quality. In *Harvard Business Review*. pp. 63–73.

Heilala, J. & Maantila, M., 2010. Developing Simulation-Based Decision Support Systems for Customer-Driven Manufacturing Operation Planning. In B. Johansson et al., eds. *Proceedings of the 2010 Winter Simulation Conference*. IEEE, pp. 3363–3375.

Herpel, T. et al., 2008. Multi-sensor Data Fusion in Automotive Applications. In *Proceedings of the 3rd International Conference on Sensing Technology (ICST 2008)*. Tainan, Taiwan: IEEE. Available at: http://www7.informatik.uni-erlangen.de/herpel/publications/pdf/-herpel2008sensorfusion.pdf.

Herrick, K., 2000. Development of the Unmanned Aerial Vehicle Market: Forecasts and Trends. *Air & Space Europe*, 2 (2), pp.25–27.

Hurrion, R.D., 1976. *The Design, Use and Required Facilities of an Interactive Visual Computer Simulation Language to Explore Product Planning Problems*. London, UK: University of London. PhD thesis.

Hybertson, D.W., 2010. *Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems*, Boca Raton, USA: CRC Press.

IAMSAR, 2007. *International Aeronautical and Maritime Search and Rescue Manual Volume III* 6th ed., 4 Albert Embankment, London, SE1 7SR, UK: International Maritime Organisation.

Jameson, A., 1999. Re-Engineering the Design Process Through Computation. *Journal of Aircraft*, 36 (1), pp.36–50.

Jennings, N.R., 2000. On Agent-Based Software Engineering. *Artificial Intelligence*, 117, pp.277–296.

Jennings, N.R., Sycara, K. & Wooldridge, M., 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1 (1), pp.7–38.

Jinks, S., 2012. *Integrating Supply Chain Simulation, Component Geometry, and Unit Cost Estimation*. University of Southampton. Available at: http://eprints.soton.ac.uk/348807/. PhD thesis.

Jones, R.A., Jimmieson, N.L. & Griffiths, A., 2005. The Impact of Organizational Culture and Reshaping Capabilities on Change Implementation Success: The Mediating Role of Readiness for Change. *Journal of Management Studies*, 42 (2), pp.361–386. Available at: http://dx.doi.org/10.1111/j.1467-6486.2005.00500.x.

Karban, R. et al., 2008. Exploring Model-Based Systems Engineering for Large Telescopes - Getting Started with Descriptive Models. *SPIE Proceedings*, 7017.

Keane, A.J., 2012. Cokriging for Robust Design Optimization. *AIAA Journal*, 50 (11), pp.2351–2364.

Keane, A.J. & Nair, P.B., 2005. *Computational Approach for Aerospace Design*, Chichester, UK: John Wiley and Sons, Ltd.

Keatinge, W., 1961. The Effect of Work and Clothing on the Maintenance of the Body Temperature in Water. *Experimental Physiology*, 46 (1), pp.69–82.

Keeter, H.C., 2008. *Coast Guard Partners with Government, Industy in Unmanned Aircraft System Evaluation* H. C. Keeter, ed., US Coast Guard Acquisition Directorate. Available at: www.uscg.mil/acquisition.

Keller, S. & Collopy, P., 2013. Value Modeling for a Space Launch System. *Procedia Computer Science*, 16, pp.1152–1160. Available at: http://www.sciencedirect.com/-science/article/pii/S1877050913001221#.

Kirby, M.R., 2001. *A Methodology for Technology Identification, Evaluation and Selection in Conceptual and Preliminary Aircraft Design*. Georgia Institute of Technology. PhD thesis.

Kroo, I. et al., 1994. *Multidisciplinary Optimization Methods for Aircraft Preliminary Design*, AIAA1994-4325. AIAA. Available at: http://aero.stanford.edu/reports/mdo94.html.

Krus, P., 2011. Whole Aircraft Simulation for System Design and Optimisation in Preliminary Design. In: *Proceedings of the 2011 International Conference of the European Aerospace Societies (CEAS)*. AIAA.

Krus, P. & Jouannet, C., 2010. Whole Mission Simulation for Aircraft Preliminary Design. In: *Proceedings of the 48th AIAA Aerospace Sciences Meeting*. AIAA.

Law, A.M. & Kelton, W.D., 1997. *Simulation Modeling and Analysis*, Columbus, OH, USA with: McGraw-Hill Higher Eduaction.

Leachtenauer, J.C., 2003. Resolution Requirements and the Johnson Criteria revisited. In G. C. Holst, ed. *Proc. SPIE 5076, Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XIV*. Orlando, Fl, USA: International Society for Optics and Photonics.

Leachtenauer, J.C. & Driggers, R.G., 2001. *Surveillance and Reconnaissance Imaging Systems: Modeling and Performance Prediction*, Norwood, MA, USA: Artech House.

Leonard, J., 2001. *Systems Engineering Fundamentals: Supplementary Text*, Fort Belvoir, Va, USA: Defense Acquisition University Press.

Mabus, R., 2008. NATOPS Flight Manual Navy Model SH-60B Helicopter. Technical Report A1-H60BB-NFM-000. US Navy, Patuxent River, Md, USA.

Macal, C.M. & North, M.J., 2010. Tutorial on Agent-Based Modelling and Simulation. *Journal of Simulation*, 4, pp.151–162.

Maslow, A.H., 2002. The Psychology of Science: A Reconnaissance. Richmond, Ca, USA: Maurice Bassett Publishing.

Mavris, D., Bandte, O. & DeLaurentis, D.A., 1999. Robust Design Simulation: a Probabilistic Approach to Multidisciplinary Design. *Journal of Aircraft*, 36 (1), pp.298–307.

Mavris, D. & Kirby, M.R., 1999. Technology Identification, Evaluation, and Selection for Commercial Transport Aircraft. In *Proceedings of the 58th Annual Conference of Society of Allied Weight Engineers, Inc.* San Jose, USA: SAWE, Inc.

Mavris, D.N. et al., 1998. A Stochastic Approach to Multi-Disciplinary Aircraft Analysis and Design. In *Proceedings of the 36th Aerospace Sciences Meeting & Exhibit*. Reno, NV, USA: AIAA98-0912. AIAA.

MCA, 2008. *Search and Rescue Framework for the United Kingdom of Great Britain and Northern Ireland*, Southampton, UK: Maritime Coastguard Agency. Available at: http://www.dft.gov.uk/mca/uksar.pdf.

McConnell, S., 2004. *Code Complete - A Practical Handbook of Software Construction* 2nd edition., Redmond, WA, USA: Mircrosoft Press.

McCullers, L.A., 1995. *Flight Optimisation System User's Guide*, Langley, Va, USA: NASA Langley Research Center.

McLean, C. & Leong, S., 2001. The Expanding Role of Simulation in Future Manufacturing. In B. A. Peters et al., eds. *Proceedings of the 2001 Winter Simulation Conference*. Arlington, VA, USA: INFORMS. Available at: http://informs-sim.org/wsc01papers/-203.PDF.

Mullan, C. et al., 2012. Surplus Value Sensitivity and Subsystem Analysis. In R. Curran et al., eds. *Proceedings of the 2012 Air Transport and Operations Symposium (ATOS)*. Amsterdam, the Netherlands: IOS Press, pp. 162–175.

Murman, E.M., Walton, M. & Rebentisch, E., 2000. Challenge in the Better, Faster, Cheaper Era of Aeronautical Design, Engineering and Manufacturing. *Aeronautical Journal*, 104 (1040), pp.481–489.

Murphy, L. & Collopy, P., 2012. A Work-Centered Perspective on Research Needs for Systems Engineering with Models. In C. H. Dagli, ed. *New Challenges in Systems Engineering and Architecting: Conference on Systems Engineering Research*. Procedia Computer Science. St. Louis, MO, USA: Elsevier, pp. 305–310.

Nagel, B. et al., 2012. Communication in Aircraft Design: Can We Establish a Common Language? In *28th International Congress of the Aeronautical Sciences*. Brisbane, Australia. Available at: http://software.dlr.de/p/cpacs/home/.

Nalepka, J.P. & Duquette, M.M., 2003. *A Multi-Purpose Simulation Environment for UAV Research*, Wright-Patterson Air Force Base, OH, USA: AIAA-2003-5685. AIAA.

Niedringhaus, W.P., 2004. The Jet:Wise Model of National Air Space System Evolution. *Simulation*, 80 (1), pp.45-58.

Nilubol, O., 2005. *Development of a Combat Aircraft Operational and Cost-Effectiveness Design Methodology*. Cranfield College of Aeronautics. Available at: https://dspace.lib.cranfield.ac.uk/handle/1826/3380.

Nunez, M. & Guenov, M.D., 2013. Design-Exploration Framework for Handling Changes Affecting Conceptual Design. *Journal of Aircraft*, 50 (1), pp.114–129.

Nurminen, K. & Karonen, O., 2003. What Makes Expert Systems Survive over 10 Years: Empirical Evaluation of Several Engineering Applications. *Expert Systems with Applications*, 24 (2), pp.199–211. Available at: http://lib.tkk.fi/Diss/2003/-isbn9512265745/article8.pdf.

Otto, K.N. & Wood, K.L., 2001. *Product Design: Techniques in Reverse Engineering and New Product Development*, Upper Saddle River, NJ, USA: Prentice Hall, Inc.

Paine, J., 2000. Spreadsheet Structure Discovery with Logic Programming. In *Proceedings of the European Spreadsheet Risks Interest Group (EuSpRIG)*. pp. 16–17. Available at: http://arxiv.org/ftp/arxiv/papers/0802/0802.3940.pdf.

Panko, R.P., 2008. Spreadsheet Errors: What We Know. What We Think We Can Do. In *Proceedings of the Spreadsheet Risk Symposium*. Greenwhich, UK: European Spreadsheet Risks Interest Group (EuSpRIG). Available at: http://arxiv.org/ftp/arxiv/-papers/0802/0802.3457.pdf.

Panko, R.R., 2000. Two Corpuses of Spreadsheet Errors. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*. IEEE, pp. 1–8.

Park, J.H. & Seo, K.-K., 2004. Incorporating Life-Cycle Cost into Early Product Development. *Journal of Engineering Manufacture*, 218 (9), pp.1059–1066.

Parunak, H. v. D., Savit, R. & Riolo, R.L., 1998. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and User's Guide. In *Proceedings of Multi-Agent Systems and Agent-Based Simulation (MABS)*. Springer, pp. 10–25.

Peters, J.F., 1995. The Transition of Functional Organizations to Integrated Product Teams on the Space Station Program. In *Proceedings of the Fifth Annual International Symposium INCOSE*. St. Louis, USA, pp. 767–773.

Pidd, M., 1998. *Computer Simulation in Management Science* 4th edn., Chichester, UK: Wiley & Sons, Ltd.

Pidd, M., 1992. Guidelines for the Design of Data Driven Generic Simulators for Specific Domains. *Simulation*, 59 (4), pp.237–243.

PMI, 2013. *A Guide to the Project Management Body of Knowledge* 5th ed., Newton Square, PA, USA: Project Management Institute.

Price, M. et al., 2012. A Novel Method to Enable Trade-offs Across the Whole Product Life of an Aircraft Using Value Driven Design. *Journal of Aerospace Operations*, 1 (4), pp.359–375.

Price, M., Raghunathan, S. & Curran, R., 2006. An Integrated Systems Engineering Approach to Aircraft Design. *Progress in Aerospace Sciences*, 42 (4), pp.331–376.

Prilla, M. et al., 2013. Collaborative Usage and Development of Models: State of the Art, Challenges and Opportunities. *International Journal of e-Collaboration*, 9 (4), pp.1–16.

Quinn, C., Soban, D. & Price, M., 2012. Value Driven Design in the Presence of Fuzzy Requirements. In R. Curran et al., eds. *Proceedings of the 2012 Air Transport and Operations Symposium (ATOS)*. Amsterdam, the Netherlands: IOS Press, pp. 317–327.

Raj, P., 1998. *Aircraft Design in the 21st Century: Implications for Design Methods*, Reston, VA, USA: AIAA.

Rand, W. et al., 2005. Toward a Graphical ABM Toolkit with GIS Integration. In *Proceedings of Agent2005*. Chicago, Il, USA.

Ranky, P.G., 1994. *Concurrent/Simultaneous Engineering: a Practical and Consistent Approach Centred Around Powerful Creative & Innovative Manufacturing and Product Design Methods, Tools and Technologies*, CIMware.

Raymer, D., 2006. *Aircraft Design: A Conceptual Approach* 5th Edition. J. A. Schetz, ed., Reston, Va, USA: AIAA Educational Series.

RNLI, 2009. *RNLI Annual Operational Statistics Report*, West Quay Road, Poole, Dorset, BH15 1HZ, UK: Royal National Lifeboat Institution.

RNLI, 2008. *RNLI Annual report and accounts 2008*, West Quay Road, Poole, Dorset, BH15 1HZ, UK: Royal National Lifeboat Institution.

Robinson, S., 2004. *Simulation: The Practice of Model Development and Use*, Chichester, UK: John Wiley and Sons, Ltd.

Rohrer, M.W., 2000. Seeing Is Believing: the Importance of Visualization in Manufacturing Simulation. In J. A. Jones et al., eds. *Proceedings of the 32nd Winter Simulation Conference*. Washington, D.C., USA: INFORMS, pp. 1211–1216.

Romli, F.I., 2013. Functional Analysis for Conceptual Aircraft Design. *Journal of Advanced Management Science*, 1 (4), pp.349–353.

Royo, P., Barrado, C. & Pastor, E., 2013. ISIS+: A Software-in-the-Loop Unmanned Aircraft System Simulator for Nonsegregated Airspace. *Journal of Aerospace Information Systems*, 10 (11), pp.530–544.

Rubinstein, R.Y. & Kroese, D.P., 2011. *Simulation and the Monte-Carlo Method* 2nd ed., Chichester, UK: John.

Sadraey, M.H., 2012. *Aircraft Design: A Systems Engineering Approach*, Chichester, UK: John Wiley & Sons.

SAE, 1998. *Perceptions and Limitations Inhibiting the Application of Probabilistic Methods*, SAE G-11 Probabilistic Methods Committee. Available at: http://standards.sae.org/-air5086/.

Saravi, M., Newnes, L. & Mileham, T., 2013. Optimising Performance and Cost at the Early Design Stages. *International Journal of Engineering and Technology Innovation*, 3 (3), pp.214–228.

Sargent, R.G., 2007. Verification and Validation of Simulation Models. In *Proceedings of the 2007 Winter Simulation Conference*. Washington, D.C., USA: INFORMS, pp. 124–137.

Sawyer, J.T. & Brann, D.M., 2009. How to Test your Models more Effectively: Applying Agile and Automated Techniques to Simulation Testing. In M.D.Rossetti et al., eds. *Proceedings of the 2009 Winter Simulation Conference*. Austin, TX, USA: INFORMS, pp. 968–978.

Scanlan, J. & Rao, A., 2006. DATUM Project: Cost Estimating Environment for Support of Aerospace Design Decision Making. *Journal of Aircraft*, 43 (4), pp.1022–1028.

Scanlan, J.P., 2007. A Metric-Based Approach to Concept Design. *Journal of Engineering Design*, accepted for publication. Available at: http://www.southampton.ac.uk/jps7/-Lecture%20notes/Student%20copy%20value%20based%20design.pdf.

Scanlan, J.P., 2004. DATUM (Design Analysis Tool for Unit cost Modelling): a Tool for Unit Cost Estimation of Gas Turbine Design within Rolls-Royce. *Cost Engineer*, 42, pp.8–10.

Schumann, B., Scanlan, J. & Fangohr, H., 2012. Complex Agent Interactions in Operational Simulations for Aerospace Design. In C. Laroque et al., eds. *Proceedings of the 2012 Winter Simulation Conference*. Berlin, Germany: IEEE, pp. 2986–2997.

Schumann, B., Scanlan, J. & Takeda, K., 2011. Evaluating Design Decisions in Real-Time Using Operations Modelling. In R. Curran & S. C. Santema, eds. *Air Transport and Operations Symposium 2011 (ATOS)*. Delft, the Netherlands: Delft University of Technology.

Smits, H., Menssen, T. & Smits, P., 2011. *Port of Rotterdam Annual Report*, Rotterdam, the Netherlands: Port of Rotterdam Authority. Available at: http://www.-portofrotterdam.com/en/Port-authority/finance/annual-report/Documents/-annual_report.pdf.

Smulders, F., Zwan, F.M. van der & Curran, R., 2012. A Value Operations-Based Methodology for Airport Concept Development for the Year 2050. In R. Curran et al., eds. *Proceedings of the 2012 Air Transport and Operations Symposium (ATOS)*. Amsterdam, the Netherlands: IOS Press, pp. 119–129.

Soban, D.S., 2001. *A Methodology for the Probability Assessment of System Effectiveness as Applied to Aircraft Survivability and Susceptibility*. Atlanta, Ga, USA: Georgia Institute of Technology. PhD thesis.

Soban, D.S., Hollingsworth, P. & Price, M.E., 2011. Defining a Research Agenda in Value-Driven Design: Questions that Need to be Asked. *Journal of Aerospace Operations*, 1 (4), pp.329–342.

Soban, D.S. & Mavris, D.N., 2000a. Formulation of a Methodology for the Probabilistics Assessment of System Effectiveness. In *Proceedings of the AIAA Missile Science Conference*. Montoney, Ca, USA: AIAA.

Soban, D.S. & Mavris, D.N., 2000b. Methodology for Assessing Survivability Tradeoffs in the Preliminary Design Process. In *Proceedings of the 2000 World Aviation Conference*. San Diego, Ca, USA: SAE.

Steinkeller, S., 2011. *Aircraft Vehicle Systems Modeling and Simulation under Uncertainty*. Stockholm, Sweden: Soerdertoern University. PhD thesis. Available at: http://sh.diva-portal.org/smash/get/diva2:415979/FULLTEXT01.

Sterman, J.D., 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Columbus, OH, USA: McGraw-Hill.

Stevenson, W.J., 2002. *Operations Management* 7th edition., Boston, USA: McGraw-Hill/Irwin.

Streit, A., Pham, B. & Brown, R., 2008. A Spreadhseet Approach to Facilitate Visualisation of Uncertainty in Information. *IEEE Transactions on Visualisation and Computer Graphics*, 14 (1), pp.61–72.

Sturm, P.J. & Hausberger, S., 2005. Emissions and Fuel Consumption from Heavy Duty Vehicles. Technical University Graz. Graz, Austria. Available from: http://www.-transport-research.info/Upload/Documents/200906/20090619_171904_26922_II_-COST346_WGA_FinalReport.pdf.

Suominen, P. et al., 2002. Impact of Age, Submersion Time and Water Temperature on Outcome in Near-Drowning. *Resuscitation*, 52 (3), pp.247–254.

Taguchi, G., 1986. *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Am. Supplier Inst.

Tam, W.F., 2004. Improvement Opportunities for Aerospace Design Process. *Space*, pp.28–30. Available at: https://info.aiaa.org/tac/ADSG/DETC/Web%20Pages/Paper-TamSpace-2004_26027.pdf.

Thokala, P., 2009. *Life Cycle Cost Modelling as an Aircraft Design Decision Support Tool*. University of Southampton. PhD thesis. Available at: http://eprints.soton.ac.uk/72021/.

Tikuisis, P. & Keefe, A.A., 2005. *Stochastic and Life Raft Boarding Predictions in the Cold Exposure Survival Model (CESM v3.0)*, Toronto, Ca: Defence Research and Development Canada.

Tipton, M. et al., 1999. Immersion Deaths and Deterioration in Swimming Performance in Cold Water. *The Lancet*, 354 (9179), pp.626–629.

Torenbeek, E., 2013. *Advanced Aircraft Design: Conceptual Design, Technology and Optimization of Subsonic Civil Airplanes*, Chichester, UK: John Wiley & Sons.

Torenbeek, E., 1982. *Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion and Performance*, Delft, the Netherlands: Delft University Press.

Twiss, B.C., 1992. *Forecasting for Technologists and Engineers: A Practical Guide for Better Decisions*, London, UK: Peter Peregrinus Ltd.

Viscusi, W.K. & Aldy, J.E., 2002. The Value of a Statistical Life: A Critical Review of Market Estimates Throughout the World. In *Harvard Law School John M. Olin Center for Law, Economics and Business Discussion Paper Series*. Harvard Law School, p. 128.

Weibel, R.E. & R. John Hansman, J., 2004. Safety Considerations for Operation of Different Classes of UAVs in the NAS. In *Proceedings of the 4th Aviation, Technology, Integration and Operations (ATIO) Forum*. Chicago, Il, USA: AIAA.

Weisbuch, G., 1991. *Complex Systems Dynamics: An Introduction to Automata Networks*, Redwood City, CA, USA: Addison-Wesley.

Wheeler, T.M. & Brooks, M.D., 2007. *Experiences in Applying Architecture-Centric Model-Based Systems Engineering to Large-Scale, Distributed, Real-Time Systems*, Bedford, Ma, USA: Mitre Corporation.

Wieland, F. & Satapathy, G., 2010. Aviation Analysis Using a Distributed Agent-based Infrastructure. In *Proceedings of the 2010 AIAA Infotech@Aerospace Conference*. Atlanta, Ga, USA: AIAA.

Will, P.M., 1991. Simulation and Modeling in Early Concept Design: An Industrial Perspective. *Research in Engineering Design*, 3 (1), pp.1–13.

Wissler, E.H., 2003. Probability of Survival During Accidental Immersion in Cold Water. *Aviation, Space and Environmental Medicine*, 74 (1), pp.47–55.

Wooldridge, M. & Jennings, N.R., 1995. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10 (2), pp.115–152.

Yilmaz, L. & Hunt, C.A., 2011. Advanced Concepts and Generative Simulation Formalisms for Creative Discovery Systems Engineering. In A. Tolk & L. C. Jain, eds. *Intelligence-based Systems Engineering*. Intelligent Systems Reference Library. Berlin, Germany: Springer, pp. 233–258.

Yu, T.-T., 2008. *The Development of a Hybrid Simulation Modelling Approach Based on Agents and Discrete-Event Modelling*. University of Southampton. PhD thesis.

Zhu, Y. & Sala-Diakanda, S., 2007. Integration of Underwater Sonar Simulation with a Geographical Information System. In S. G. Henderson et al., eds. *Proceedings of the 2007 Winter Simulation Conference*. Washington D.C., USA: Winter Simulation Conference, pp. 1378–1386.