The University Of Sheffield.

Multilevel Design for Complex Engineered Systems

Author:

Christopher J. Hambley

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Rolls-Royce Control and Systems University Technology Centre
Department of Automatic Control and Systems Engineering

December 2018

Supervisor:

Professor Visakan Kadirkamanathan

Dedicated to my wife Maya, for all her love, understanding and support.

Abstract

Multilevel Design for Complex Engineered Systems

by Christopher J. Hambley

This thesis presents a multilevel design approach that uses formalized requirements to facilitate design synthesis techniques, such as optimization, at increasing levels of fidelity. The benefits are that verification is built into the design, guaranteeing requirements satisfaction. It also focuses the design effort higher up, spending more time considering what the system needs to do, and the attributes it should have. Specific examples of design synthesis techniques are developed in the main chapters, showing how they fit into the wider multilevel framework.

Architecture optimization has been implemented at both high and low levels. High-level architectures are composed as a combination of physical means for achieving a set of functions. A multiobjective genetic algorithm is used to produce a set of Pareto-optimal solutions. Refinement to a single solution is then implemented using a customer-oriented approach. This produces solutions that the customer wants whilst reducing the need for iterative discussions with engineers.

Low-level topology is represented as a graph with nodes (system components) and edges (interconnections between components). The topology requirements are formulated as constraints on the graph and synthesis is achieved via constrained optimization. The approach is applied to a turbofan oil system case study with two objectives: increasing controllability, via the addition of controllable valves, and minimising cost. The methodology provides benefits to system designers by selecting cheaper architectures with fewer valves when the need to control oil chambers separately is small.

A simulation-based approach for performing control synthesis with signal temporal logic (STL) requirements is presented. The goal is to find control parameters that maximise the margin of satisfaction of the STL formulae. The quantitative semantics of STL are extended to a multiobjective formulation called multiSTL. In multiSTL each requirement margin is displayed on a parallel coordinates plot, which allows tradeoffs between different requirements to be analysed. This can also be used to highlight where relaxing of some requirements might yield better performance in other areas.

Acknowledgements

I would firstly like to thank my PhD supervisors Professor Visakan Kadirkamanathan and Dr Bryn Jones. Without their guidance and support over the last few years, it would have been impossible to complete this project.

Secondly, I would like to thank other students and staff from the Rolls-Royce Control and Monitoring Systems University Technology Centre. The friendly support and informal discussions have helped enormously in developing the research presented in this thesis. In particular, I would like to acknowledge: Professor Tony Dodd for his supervision in the development of the SATS tool; Dr Andy Mills, for acting as a supervisor throughout the PhD and being constantly available to bounce ideas off; and Dr Andrew Hills, who has been a guru for all things related to IT, modeling and LaTeXdocuments.

Finally, I would like to thank my wife, family and friends who have helped me to relax and enjoy the last few years, despite the pressures of postgraduate research. Without their support, this PhD would surely not have been possible.

Contents

List of Figures

List of Tables

Abbreviations

C&I	**C**ontrol and **I**nstrumentation
CBD	**C**ontract **B**ased **D**esign
EC&I	**E**lectrical **C**ontrol and **I**nstrumentation
EPS	**E**lectric **P**ower **S**ystem
FP	**F**eed **P**ump
GA	**G**enetic **A**lgorithm
GUI	**G**raphical **U**ser **I**nterface
HE	**H**eat **E**xchanger
HP	**H**igh **P**ressure
IP	**I**ntermediate **P**ressure
IRMA	**I**nteractive **R**econfigurable **M**atrix of **A**lternatives
LP	**L**ow **P**ressure
LTL	**L**inear **T**emporal **L**ogic
MFESA	**M**ethod-**F**ramework for **E**ngineering **S**ystem **A**rchitectures
MILP	**M**ixed **I**nteger **L**inear **P**rogramming
MINLP	**M**ixed **I**nteger **N**on**L**inear **P**rogramming
MOGA	**M**ulti**O**bjective **G**enetic **A**lgorithm
MPC	**M**odel **P**redictive **C**ontrol
NRE	**N**on-**R**ecurrent **E**ngineering
OC	**O**il **C**hamber
PBD	**P**latform **B**ased **D**esign
PID	**P**roportional **I**ntegral **D**erivative
PWR	**P**ressurized **W**ater **R**eactor
QFD	**Q**uality **F**unction **D**eployment

RPM	**R**evolutions **P**er **M**inute
SATS	**S**ystem **A**rchitecture **T**rade **S**tudy
SME	**S**ubject-**M**atter **E**xpert
SOA	**S**ystem **A**rchitecture **O**ptimization
SP	**S**cavenge **P**ump
STL	**S**ignal **T**emporal **L**ogic
VRV	**V**ariable **R**estrictor **V**alve
WSTL	**W**eighted **S**ignal **T**emporal **L**ogic

Symbols

Contract Theory

$C = (A, G)$	assume-guarantee contract
A	contract assumptions
G	contract guarantees
$C' = (A', G')$	refinement of contract $C = (A, G)$
$\preccurlyeq$	refines
$\models$	satisfies
$\otimes$	composition operator for contracts
$\cap$	intersection
$\cup$	union
$\subseteq$	subset

Temporal Logic

φ	logic formula
$\neg$	negation
$\wedge$	conjunction (AND)
$\vee$	disjunction (OR)
$\rightarrow$	implies
$\bigcirc$	next
$\mathcal{U}$	until
$\Box$	always
$\Diamond$	eventually
$\mu(.)$	Booleanizing function

S	set of system behaviours
$\mathcal{R}(S)$	reachable set of system behaviours
$\rho(x, \varphi, t)$	space robustness function
$\rho^{-}(x, \varphi, t)$	backwards time robustness function
$\rho^{+}(x, \varphi, t)$	forwards time robustness function
w	WSTL weighting

Graph-Based Topology Optimization

$\mathcal{N}$	graph nodes
E	adjacency matrix
$e_{i,j}$	edge value between node i and node j
C_{f_r}	flow requirement similarities matrix
$\otimes$	Kronecker product
$\bullet$	Hadamard product
f	objective
$\hat{f}$	normalized objective
w	objective weight

Oil system modeling and control synthesis

$H(t)$	heat flow at time t
$W(t)$	fluid flow at time t
$T(t)$	temperature at time t
$\omega(t)$	shaft angular velocity at time t
R	thermal resistance
c_p	specific heat constant
A	contact surface area
U	heat transfer coefficient
F_c	tube and shell correction factor
K_{p_i}	control gain for oil chamber i

Chapter 1

Introduction

This PhD project has been undertaken in the Rolls-Royce Control and Monitoring Systems University Technology Centre. It addresses some of the challenges faced by the company in designing complex engineered systems such as gas turbine engines.

1.1 Defining Complex Engineered Systems

There is some debate surrounding the definition of complex systems. A survey of complexity measures presented in [1] covers computational complexity, number of states and connectivity amongst many others. These would be considered measures of *complicatedness* by the definitions of [2] and [3]. They define complicated systems as having a large number of components working together with well-defined behaviour to accomplish a specific goal, whereas complex systems exhibit some sort of emergence or adaptability. For example, an aircraft is complicated, while a flock of geese is complex. However, in the context of *engineered* systems, [3] notes that the process of designing a complicated system is actually complex because of the many interacting design teams and stakeholders. For the purpose of this PhD project no distinction is made between the words, referring to the systems developed by Rolls-Royce and other big industrial companies as *complex engineered systems*.

1.2 Challenges for Complex Systems Engineering

There are three main aspects to designing complex systems: 1) defining the system specification; 2) designing a system which meets the specification; 3) verification and validation of the design. As time progresses, we are demanding more from our engineered systems in terms of functionality, performance and safety. To meet these increasingly complex requirements, the design solutions are also becoming more complex. For example, the number of lines of source code in fighter jets has increased from around 300,000 in 1980 to nearly 4.5 million lines in the modern joint strike fighter programme [4]. This increase in design complexity consequently makes the task of verification more difficult. Managing this complexity introduces a variety of challenges which must be addressed by systems engineering techniques.

Challenges in defining the system requirements
Firstly there is the challenge of getting good requirements from the customer. This requires defining the core functionality and then prioritising the non-essential but desirable attributes. It is vital that this step is done right because any defects in the requirements will result in defects in the design of the system. Ideally requirements should be defined in a formal language to make verification easier, but the stakeholders providing the requirements generally find it easier to specify linguistically.

Challenges in the system design process
One of the big challenges in high-level system design is making decisions about the architecture of the system. There is often a reliance on the opinions of subject matter experts, but this is not objective enough and there is a need to use design optimisation, modelling and simulation as much as possible. Unfortunately modelling and simulation can be very difficult because of the multi-level, distributed nature of complex systems design. How can subsystems developed by multiple design teams, defined at different levels of fidelity, using different design tools be compared? There is a need for a mathematical framework to understand the complex interactions between different models of the system.

Challenges for verification
Verification refers to the process of checking that a system has been designed correctly, meeting all the system requirements. As systems become more complex, manual verification of the design becomes a challenging task. Therefore there is a need for automation in the verification process. This requires *virtual integration* of models to verify not only that the subsystems themselves have been designed

correctly, but also that the system still functions as required when the individual components are put together. The ideal scenario would be to design systems with a *correct-by-construction* framework, whereby the verification is built into the design process itself and the final design is guaranteed to meet the requirements.

Challenges for adapting, upgrading and evolving systems
This is an area which is relevant to Rolls-Royce since they design high-value products with long operating lifespans. Adaptability is needed because gas turbine engines are sold to a variety of airlines and fitted on multiple aircraft, all with different requirements. Long product lifespans mean it would be beneficial to have a design that could easily be upgraded to make use of new technology such as better sensing equipment. At the same time any changes to the design have to satisfy stringent certification requirements. It is therefore desirable to have a formal mathematical framework that can handle these changes in design quickly and efficiently without a lengthy redesign and verification process. There are a huge number of engineering companies across a wide range of industries that face similar problems and seek a similar solution.

Addressing these challenges in a formal scientific framework is very important to the academic and industrial systems engineering community. It was identified as the top grand challenge priority in a brainstorming session amongst attendees of the *2015 Systems-NET Annual Research Grand Challenges for Systems Engineering Workshop* [5]. This confirms the importance of the research presented in this thesis.

1.3 Multi-Level System Design for Complex Engineered Systems

This thesis proposes a multilevel design framework to address some of the complex systems engineering challenges discussed in Section 1.2. The framework, outlined in Figure 1.1, utilises formal requirements with design synthesis techniques such as optimization. This allows engineering effort to be focused at the upper levels, where the system requirements and performance measures are defined. Spending more time at this stage of the design helps to address the first big challenge (defining requirements).

The use of design synthesis methods solves two big challeges. Firstly, it ensures good system design, with less engineering effort through use of techniques such as

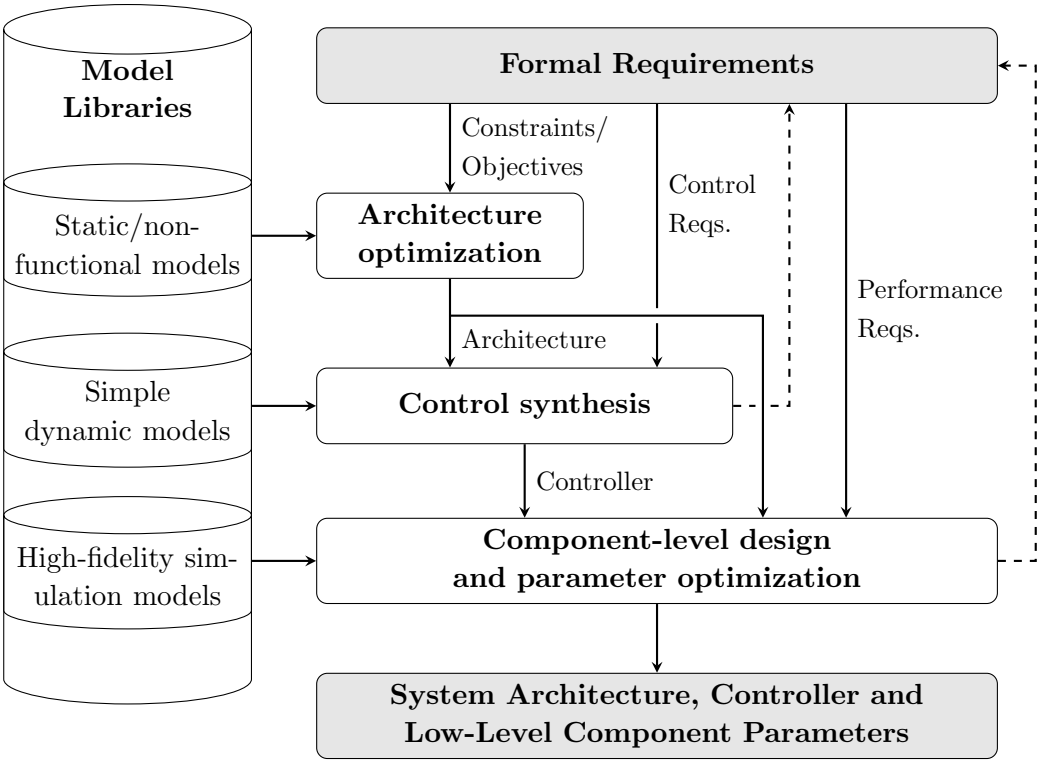


FIGURE 1.1: A multilevel system design framework with design automation using formal requirements. Cylindrical buckets represent libraries of components/models at different levels of fidelity, compiled from bottom-up abstraction. Grey rectangles represent inputs and outputs of the top-down design flow. White rectangles represent design activities at the different levels. Black arrows indicate information flows. Dashed arrows indicate a change in the requirements for upper levels when no feasible solution can be found at a lower level.

optimization. Secondly, the use of formal requirements as inputs into the synthesis allows a correct-by-construction approach whereby the design is guaranteed to satisfy the requirements, or even maximise the margin of satisfaction. This removes the need for a separate verification exercise.

The synthesis techniques rely on a library of models at appropriate levels of fidelity. Therefore, there is a bottom-up phase of populating these libraries. However, this only needs to be carried out once, allowing companies to re-use models when designing similar systems. With a sufficient model library, system design is then a top-down approach progressing rapidly from requirements to detailed design. This helps to address the challenge in adapting or upgrading systems, since it is easy to synthesise an alternative solution for a different set of requirements.

Referring to the connections between levels in Figure 1.1, the aim is for designs to be specified formally, so that they can be passed down as constraints for the

synthesis at the next level of fidelity. This allows an automated flow from requirements to low-level design.

Sometimes decisions at a higher level make it difficult or impossible to meet requirements at a lower level. Therefore feedback loops are included in the framework. For example, a decision on a particular architecture may make it impossible to meet the performance requirements at the control synthesis stage. This introduces a new constraint on the architecture design, which in turn triggers a repeat of the architecture and control synthesis stages.

1.4 Main Contributions

To the best of this author's knowledge, the original contributions of this thesis are:

1. The customer-oriented architecture refinement framework. This provides a rapid approach for reducing a large set of potential architectures to a small set of interest to the customer. It only requires a set of customer preference weightings, which can be provided at the outset, eliminating the need for lengthy iterative discussions with the customer. The framework also includes an approach to analyse resilience of architectures to changing customer preferences. This helps engineers to select solutions that are likely to remain good options, even if the customers change their preference weightings as a result of external factors, such as budget cuts. The result is a lower chance of having to rework or modify designs. The SATS tool has been developed to implement the approach in a graphical user interface, which is currently being used by the industrial sponsor of the PhD in real-world architecture design problems.

2. A graph-based topology optimization approach for system architectures. The approach is demonstrated on a turbofan oil system case study, which involves a novel heuristic algorithm for determining similarities between oil chamber flow requirements. The approach allows sensible coupling of oil chambers to shared valves, to reduce the cost of the architecture. The graph-based approach to modelling system architectures and optimizing connections between nodes also has wider applicability to any system with a set of interconnected components.

3. The multiSTL control synthesis framework. This allows the margins of satisfaction for performance requirements specified as Signal Temporal Logic (STL) formulae to be compared on a parallel coordinates plot, highlighting tradeoffs between requirements. The approach gives designers far more information about how the system is performing than if the overall margin of satisfaction is taken as the minimum of the individual requirement margins. This helps with choosing a set of control parameters that achieve an optimal system response, with respect to the priorities of the multiple, often conflicting performance requirements. To demonstrate the approach on a real-world problem, a nonlinear dynamic oil system model is developed and used to perform a multiSTL analysis.

1.5 Thesis Layout

The fully integrated multilevel design flow outlined in Section 1.3 is beyond the scope of a single PhD project. To apply this approach to a real-world complex system would require teams of engineers working for long timescales. This provides a challenge for demonstrating the novelty, relevance and importance of the framework. The thesis handles this by presenting specific instances of design at different levels of fidelity, showing how the research fits into the wider multilevel framework.

Chapter 2 discusses the relevant literature focusing on previous research into multilevel design frameworks, architecture optimization, formal requirements and control synthesis techniques.

Chapters 3 and 4 focus on the architecture synthesis level. This is further split into high-level architecture *framework* and low-level architecture *topology*, as shown in Figure 1.2.

Chapter 3 presents a two-sided approach to high-level architecture design. The first side involves synthesising a large number of Pareto-optimal architectures. This means that for each candidate architecture, there is no architecture which performs better against every decision criterion. Therefore, all the architectures are optimal in some respect, depending on the priorities of the decision criteria. The second side focuses on refining the large set of Pareto-optimal architectures to a chosen solution or subset of solutions.

The main contributions of this chapter focus on a method to rapidly refine the architecture set, based on a set of customer preference weightings. These weightings

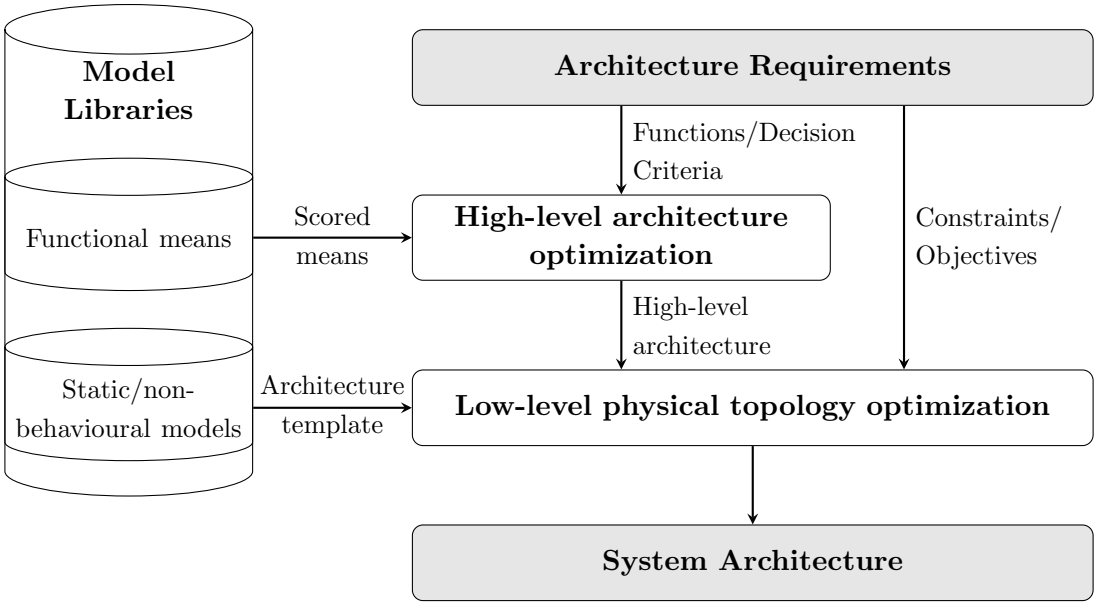


FIGURE 1.2: System architecture design at multiple levels. Cylindrical buckets represent libraries of functional means/components/models used in the design. Grey rectangles represent inputs and outputs of the top-down design flow. White rectangles represent design activities at the different platforms. Black arrows indicate information flows.

are transformed via a relational matrix into decision criteria (engineering characteristic) weightings, which are then translated into upper bounds on the decision criteria. Any solutions that do not fall under all of the upper bounds are removed from the solution set. The solutions produced via the customer-oriented refinement method are analysed for their resilience to changing customer preferences. This allows designers to select solutions that are likely to remain good options over long development periods, even in the presence of external factors such as budget cuts or management change. The approach is demonstrated on a nuclear reactor and a turbofan oil system case study.

Chapter 4 focuses on lower level architecture topology optimization. While the high-level architecture defines the key technologies that will be used, the low-level architecture topology defines the specific components and the structure of their interconnections. The approach in this chapter models the topology as a graph with nodes representing components and edges representing physical connections between them. Topology requirements are defined as constraints on the nodes and edges. Objectives are then minimised using constrained optimization.

The approach is applied to a turbofan oil system design, following on from the high-level architecture case study. The novelty of the case study is in the use of variable restrictor valves to control flows to one or more oil chambers. The two

conflicting objectives are minimising cost (by using less valves) and maximising controllability of flows to the oil chambers. Here controllability refers to the ability to maintain good system performance. When two oil chambers share similar flow requirements, good controllability can be achieved without independent valves. Using a matrix of flow requirement similarities, populated by a new heuristic algorithm, the optimization algorithm can couple similar oil chambers to a shared valve. This maintains good controllability whilst minimising cost.

Chapter 5 focuses on the control synthesis design level. The approach uses simulation -based optimization to maximise the margin of satisfaction for a set of system requirements specified in the formal signal temporal logic (STL) language. The key contribution is the development of the multiSTL framework. Rather than calculating the overall system margin as the minimum of the individual requirement margins, multiSTL displays each margin on a multidimensional parallel coordinates plot. For a set of simulation results, this allows tradeoffs between different requirements to be analysed.

The control synthesis stage uses multiSTL to analyse simulation performance for a set of different control parameters, iteratively tuning the parameters to reach a Pareto-optimal set of gains. This is then refined to a single set of control parameters by progressively prioritising the requirements, as done in the architecture refinement of Chapter 3. The approach is demonstrated on the turbofan oil system case study used in the upper design levels. A nonlinear simulation model is developed, and a set of natural language requirements are converted into STL. These are used to perform the simulation-based control synthesis with multiSTL. The resulting controller is able to satisfy all system requirements and achieve good thermal and lubrication efficiency.

Chapter 6 summarises the research presented in the thesis, highlighting the original contributions and suggested areas for further research.

1.6 Publications, Presentations and Technical Reports

Some of the research presented in this thesis is related to the following publications, presentations and technical reports.

- C. J. Hambley, 'Contract-based design for complex engineered systems', poster presentation at *ACSE Postgraduate Research Symposium*, 2015.

- C. J. Hambley, 'SATS V1.0 User Guide', Tech. Report, 2017.
- C. J. Hambley, 'Customer-oriented architecture refinement in multi-criteria synthesis of large-scale system architectures', oral presentation at *IEEE International Symposium on Systems Engineering*, 2017.
- C. J. Hambley, W. Bradley, R. Shirtcliffe, A. R. Mills, T. J. Dodd, V. Kadirkamanathan, 'Customer-oriented architecture refinement in multi-criteria synthesis of large-scale system architectures', in proceedings of *IEEE International Symposium on Systems Engineering*, 2017.
- C. J. Hambley, B. Ll. Jones, I. Griffin, A. R. Mills and V. Kadirkamanathan, 'Optimized synthesis of cost-effective, controllable oil system architectures for turbofan engines', *Systems Engineering, Special Issue on Model-Based Systems Engineering*, 2018.
- C. J. Hambley, B. Ll. Jones and V. Kadirkamanathan, 'Simulation-based control synthesis with multiple, conflicting performance requirements', in preparation.

Chapter 2

Literature Review

This chapter reviews some of the relevant research in complex systems engineering, looking at typical approaches in industry, multilevel design approaches, system architecting, formal requirements and control synthesis.

2.1 Systems Engineering in Industry

The systems engineering processes commonly used in industry are described in [6, 7]. These approaches are divided into *vertical processes* and *horizontal processes*. Vertical processes involve splitting the design up via abstraction and refinement stages. Horizontal processes relate to the decomposition of the system at the same abstraction level. The aim in this case is to develop components which are fairly independent of each other with well-defined interfaces. The system is then designed by composing a set of components connected via their interfaces [6, 7].

Model-based design is an approach that has been widely adopted by industry. Languages such as SysML [8] replace the document-centric approach of the past and provide useful features like auto code generation, which helps to reduce design errors.

As the use of model-based design becomes more widespread, *virtual integration* becomes possible. This refers to the process of integrating the models of the system and performing verification of the design requirements without the need to actually build the physical system. This is commonly done in industry through a tool-based approach. For example, [6, 7] outline a variety of software for virtual integration such as Ptolemy II, Metropolis, Modelica and SimScape. The problem

with these approaches is that they focus on providing a translation between the semantics of the different modeling languages, rather than an underlying mathematical framework.

2.2 Multilevel Design Approaches

There are a variety of vertical processes used for designing complex systems in industry. These consist of multiple levels of design at increasing levels of abstraction. For example, the layered approach of the AUTOSAR standard consists of abstraction levels from the microcontroller layer (bottom) to communication/operating system layer (middle) to the application layer (top) [9]. Another commonly used vertical process is the systems engineering V-modell® XT [10], a well-defined standard that all German military engineering companies must follow. The wider systems engineering community uses various V-shaped models generally consisting of project definition/design activities on the left side and testing/integration activities on the right side.

In the academic community, a variety of more formal multilevel approaches are proposed for systems engineering, as discussed in the following subsections.

2.2.1 Model Order Reduction

One of the challenges with a multilevel design framework is ensuring that simplified models used for upper level design activities accurately reflect the dynamics of the lower level behaviour. This can be guaranteed mathematically, under bounded approximation error, using model order reduction techniques. Model order reduction is defined mathematically in [11] as:

Given the original system,

$$\Sigma : \begin{cases} \frac{d}{dt}x = f(x,u) \\ y = g(x,u) \end{cases}, \quad \text{where} \quad u \in \mathbb{R}^m, \quad y \in \mathbb{R}^p \quad \text{and} \quad x \in \mathbb{R}^n \tag{2.1}$$

find,

$$\hat{\Sigma} : \begin{cases} \frac{d}{dt}\hat{x} = \hat{f}(\hat{x},u) \\ y = \hat{g}(\hat{x},u) \end{cases}, \quad \text{where} \quad u \in \mathbb{R}^m, \quad y \in \mathbb{R}^p, \quad \hat{x} \in \mathbb{R}^k \quad \text{and} \quad k < n \tag{2.2}$$

where Σ is the original system, x is the full state vector, u is the input vector, y is the output/measurement vector, f and g are functions of the full states and inputs, $\hat{\Sigma}$ is the reduced order system, $\hat{x}$ is the reduced order state vector and $\hat{f}$ and $\hat{g}$ are functions of the reduced order states and inputs.

The model order reduction problem is pervasive across a variety of engineering fields; see [12] for a collection of examples. Extensive discussion of model order reduction techniques is given in [11] for linear systems and [13] for nonlinear systems.

2.2.2 Model Bounding

Another multi-level approach to system design is presented in [14]. Here the design progresses sequentially as the set of potential solutions is narrowed down to a choice set, with increasing levels of fidelity in the models used to make decisions. Low fidelity models are coupled to higher fidelity models through use of bounding functions which specify the upper and lower bounds on variables. This allows information from the detailed models to be considered at the conceptual design stage, without the need for complex analysis or simulation of the high-fidelity details [14].

2.2.3 Platform-Based Design

Platform-based design (PBD) [15, 16] is a method for combining vertical abstract/refine processes and horizontal composition/decomposition approaches. The basic idea is to define a set of *platforms* corresponding to different abstraction layers in the design. Each platform has a *library* of components and a set of rules for how they can be composed. A collection of platform components with specific configuration parameter values is called a *platform instance*, which defines the system design at that platform level. The PBD approach is represented in Figure 2.1. A *mapping function* defines the relationship between platform instances (designs) at different platforms.

The example given in [15] is based on embedded systems with a high-level function/application platform, an architecture platform and a silicon implementation platform. An example of PBD applied to an aircraft electric power system is presented in [17]. Other applications of PBD are given in [4], with a JPEG encoder and distributed automotive design case studies.

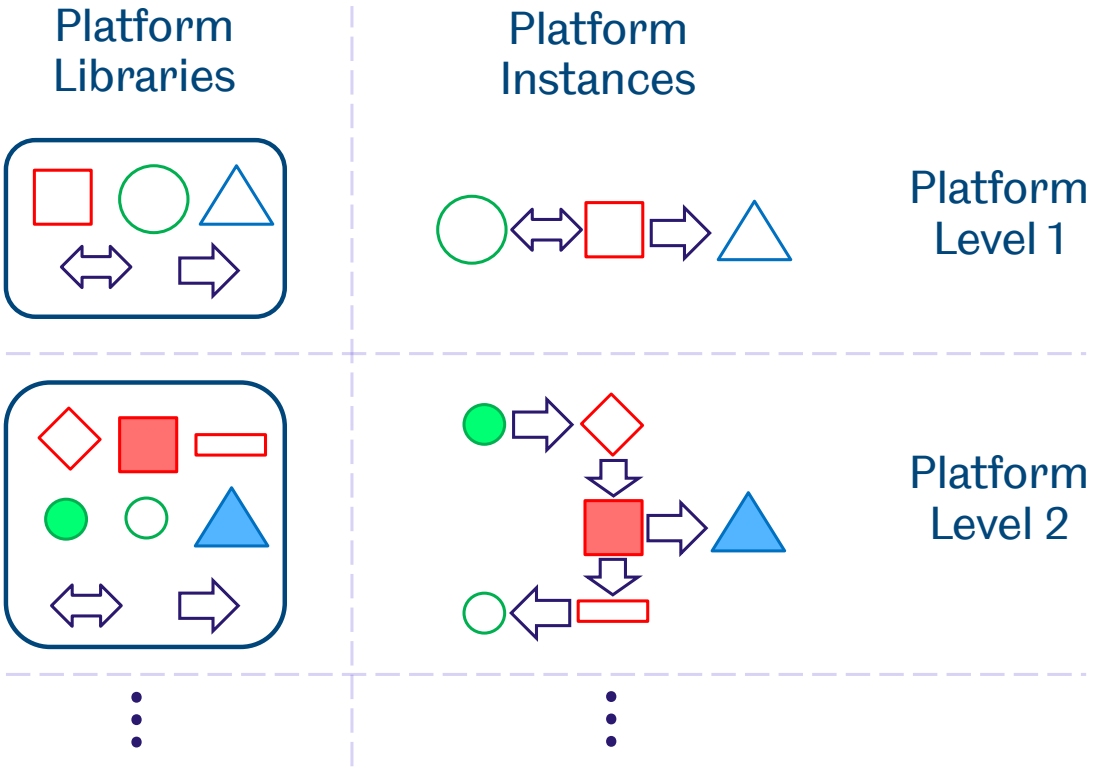


FIGURE 2.1: An illustration of PBD. Platforms are the levels of abstraction. Libraries contain the set of components with rules for connecting them. Platform instances are the system designed by composition of platform components.

One of the main challenges in PBD is determining how to move downwards through the framework. In [18] this is handled using an optimisation-based approach. At each platform the functional specification (what the system needs to do) is separated from the architecture (how to do it). The functional specification and the rules of the platform define the constraints, whilst the architecture (platform instance) is the output of the optimisation algorithm. The selected architecture then becomes the functional specification for the next platform down and a similar optimisation process is followed. The main problem with the optimisation approach is that the design space can easily become too large to efficiently explore. This is particularly true when there are both decisions on components (e.g. how many resistors) and decisions on configuration parameters (e.g. resistance) [18].

Another big challenge for PBD is establishing platform rules for connecting components described by heterogeneous models. In [4] this is handled using a tool-based approach called Metropolis, which acts as a translation between different modeling languages. Another practical approach is taken in [17] where tools such as SysML requirements diagrams, state-machine diagrams and Simulink models are used at

different platforms. The downside to these ad hoc approaches is that they lack an underlying mathematical theory such as assume-guarantee contracts.

2.2.4 Contract-based design

Contract-based design (CBD) is a formal mathematical approach to PBD. Each component within CBD is defined by an *implementation* and a *contract* [19]. A component implementation M consists of a set of variables, configuration parameters and models. A component contract $C(A, G)$ consists of *assumptions* and *guarantees* [6, 7, 20]. Assumptions (A) define the assumed behaviour of the environment or inputs from other components (uncontrollable variables). Guarantees (G) define promises on the behaviour of the component outputs (controllable variables) subject to the assumptions on the uncontrollable variables. A component implementation M satisfies its associated contract C whenever it satisfies its guarantee, subject to its assumption.

Splitting the component contract in this way makes it very clear what the component is responsible for (guarantees) and what the other components are responsible for (assumptions). This can be particularly useful when working with multiple subsystem suppliers who may not have good communication amongst each other [6]. In CBD the compatibility and virtual integration testing is based on the contracts of components, rather than their models [7]. Since there are well defined rules for compatibility, consistency, composition, refinement and conjunction of contracts this is much more simple (see Appendix A). This means that the verification process can be automated, particularly when contracts are expressed in a formal language (see Section 2.4).

CBD has been applied to case studies for water level control [21], aircraft electric power systems [19, 22] and ultra-wide band receivers for intelligent tire systems [23].

2.3 System Architecture Design

A system architecture is a definition of the system structure including the major components and the way in which they are connected in order to meet the system requirements [24]. In [25] system architecting is defined as "the embodiment of concept, the allocation of function to elements of form, and definition of relationships among the elements and with the surrounding context". The importance

of system architectures is argued in [26] as a tool for understanding the design, operation, and complex behaviours of engineered systems.

This high-level decision making process is a key part of the systems engineering discipline [27]. The way these decisions are made can vary greatly between different applications. Six main patterns of architectural decision making are shown in [28] including: combining, downselecting, assigning, partitioning, permuting and connecting. Regardless of the pattern followed for a particular application, it is important for the architectural decisions to be made using a well-defined process.

The impact of effective architecture design is highlighted in a recent survey of 46 industrial defense contractors, which shows a strong positive correlation between increasing system architecting activities and improved product development cost, schedule and scope goals [29]. Despite these clear benefits, [30] shows a marked need for an improvement in the uptake of system architecting activities within industry. The research found that even a world-leading industrial automation company with more than 50% of the global market share still has no formal architecting process. A U.S. Government Accountability Office review also found that 10 defence programs out of the 32 investigated pursued a pre-selected solution without performing any analysis of alternative architectures [31]. This motivates the need for more research into system architecting activities, as wider uptake by companies will rely on the continued development of effective tools and frameworks.

There are various types of architecting processes with different levels of formality. Improved project performance can be gained even through less formal approaches whereby the majority of systems architecture decisions are carried out using the knowledge and expertise of engineers, but in a well-defined task-flow [32]. More formal approaches attempt to automate some of these tasks and use optimization to produce optimal architectures that maximize satisfaction of some objective function and ensure system requirements or constraints are met [19, 33–39]. Use of optimization also ensures a fuller exploration of the search space and limits the effects of cognitive biases.

Guidelines for system architecting are given by INCOSE [27] covering a range of activities from obtaining customer requirements to verification of the design. Another framework for systems architecting called the Method-Framework for Engineering System Architectures (MFESA) is presented in [24]. This covers the entire systems architecture process for an industrial company, which includes activities like assigning engineering effort. The areas of interest for the research in chapters 3 and 4 focus on MFESA tasks T5 to T8 which look at generating a list of suitable

architecture candidates and selecting the best choice. A variety of approaches for carrying out these tasks are presented in Sections 2.3.1 to 2.3.6.

2.3.1 Informal Approaches

It has been established in [30] that some substantial multinational companies still do not have a formal process for designing complex system architectures. The reasons for this can be due to lack of understanding or knowledge of more formal processes such as optimization techniques. However, improvements can still be made from use of more simple techniques. The INCOSE systems engineering handbook [27] does not provide specifics about how the systems architecting tasks must be accomplished. Therefore tasks such as "evaluate alternative design solutions" could be implemented in a straightforward fashion, simply by using subject-matter experts (SMEs) to rank candidate architectures against various criteria in a Pugh matrix [40]. The same is true of many of the activities within the MFESA framework [24].

Another SME-driven architecting procedure is the 9-step method presented by [32]. This lists activities from stakeholder requirement elicitation to validation of a chosen architecture. In this approach architectural candidates are listed via brainstorming sessions and a final architecture chosen through use of SMEs. The method is demonstrated on an automotive telematics system case study.

A big disadvantage to SME-focused approaches is that they can sometimes lead to design fixation [41] with decisions echoing the engineers' previous experiences and neglecting novel ideas. This means that the resulting architectures are often evolutionary rather than revolutionary, potentially missing out on the benefits of new technological advancements. However, by following an SME-driven architecting procedure rather than none at all, companies are still likely to see noticeable improvements in project performance [29]. It is also worth noting that SMEs can be effective in removing unsuitable architectures from consideration early in the design, where unnecessary lower-level analysis would be costly or time-consuming [42].

2.3.2 Architecture Design Using Decomposition Matrices

Following elicitation of customer requirements, architecture design is typically carried out as a process of decomposition, splitting high-level functions into sub-functions, and physical elements into sub-systems or components [26]. Examples

of this type of decomposition are the morphological matrix [43] or matrix of technology alternatives [44]. Here the system is broken into subsystems and then components, listed alongside potential technologies for implementing them. Taking the aircraft example from [44], the subsystem "brakes" could be accomplished by 3 technologies: a) hydraulic, b) electro-hydraulic or c) electro-mechanical.

A similar approach is function means analysis [45], whereby the system is decomposed into a set of functions that must be performed, alongside a list of physical means of implementing these functions. For example, "interface with human user" could be accomplished by: a) lamps and switches; b) keyboard, mouse and monitor; or c) touchscreen.

For all of these approaches, an architecture is defined by selecting an option for each subsystem, component or function. In [46] the selection of options is implemented manually via an interactive reconfigurable matrix of alternatives (IRMA) with an ontology used to define incompatible choices. The downside to manual selection is that it limits the number of architectures that can be practically considered. An approach that uses optimization to synthesize architectures by exploring a much larger set of alternatives is therefore advocated in Chapter 3.

2.3.3 Architecture Topology Design as a Component Selection Problem

Another approach to system architecting is to have a *library* of components with their models and perform architecture design as a composition of these library elements. This naturally involves two stages as highlighted in [47, 48]:

1) Modelling phase: where the library of component models is populated either from first-principles, system identification or legacy models. It is noted that when designing complex systems there is a need to address the systems architecture problem at higher levels of abstraction, where simpler models facilitate more rapid analysis of alternatives. Therefore the modeling phase may also contain a bottom-up approach whereby high-level abstract models are derived from their high fidelity descriptions [47].

2) Component selection phase: where the architecture is constructed as a composition of components. A set of rules define the minimum and maximum number of different components of each type and the permitted interconnections. The components are then composed according to these rules until a set of formal system requirements are met.

In [47, 48] three different algorithms are used to implement the component selection phase: two "Greedy" algorithms and one simulated annealing approach. The effectiveness of the algorithms is demonstrated on a Network on a Chip (NoC) case study. The problem with the three component selection approaches here is that they only iterate until the system specification/architecture constraints are met. Therefore the algorithm may miss better architectures that satisfy the performance specification more robustly, or for cheaper cost. The reason the authors state for not following a more exhaustive optimization approach is due to the size of the search space [47, 48]. If the system design problem could be posed as a convex optimization, this large search space would not be prohibitive. However, it is known that system design is an NP-complete problem, as proved in [49]. This means that optimal systems cannot be designed with deterministic, polynomial-time procedures. Fortunately system design does not need to be carried out in real-time and architecture optimization (see Section 2.3.4) can be performed in a reasonable time-frame for smaller systems. For example, the approach presented in this thesis produces oil system architectures in less than 10 seconds on a standard desktop PC.

The architecture design problem is solved in [50] using a genetic algorithm (GA) approach. It starts with the functions that must be performed by the system and a library of components. Each function is assigned to a component from the library. If the function cannot be met by a single component alone, then increasingly large chains of components are investigated until the function is met. Using this method a population of potential candidates for the architecture is generated. The GA is then an iterative algorithm which generates a population of new architecture candidates based on the best individuals from the previous population. The method is applied to an aircraft cockpit design case study. The downside to this method is that GAs are not able to guarantee finding a globally optimal solution. Several methods for improving this are suggested including reducing the design space, considering the architecture performance in the synthesis of candidates and using constraint programming [50].

Design of system architectures as a composition of elements from a component library is also presented in [19, 33]. In these approaches optimization is used to ensure that the resulting architecture is *optimal* according to some objective function, rather than just satisfying the system requirements (constraints). This is discussed in more detail in Section 2.3.4.

2.3.4 Architecture Design as a Constrained Optimization Problem

As discussed in Section 2.3.3, it is possible to have multiple different system architecture candidates that satisfy the system requirements. Constrained optimization is a method for determining the best architecture from a set of candidates by finding the solution which maximises some desired objective such as minimising cost. The advantage of this is that it enforces a fuller exploration of the search space. In addition, the objective nature of optimization limits the cognitive biases inherent to SME-driven techniques (Section 2.3.1). The *generic* systems architecture optimization (SAO) problem is presented in [35] as comprising of three elements:

1. f - the objective function
2. R - a set of constraints
3. A - a set of architectures built in a framework N

The SAO solution finds a subset of A which minimises/maximises f whilst satisfying R. This is a generic problem and the paper does not refer to any specific optimization schemes since often these need to be tailored to suit the application.

An application of the SAO problem is presented for optimal design of an aircraft electric power system in [19, 33]. In this framework an architecture is defined as a directed graph with components represented as nodes $\{N_1, \cdots, N_n\} \in \mathcal{N}$ and interconnections between nodes N_i, N_j represented by edges $e_{i,j} \in E$ where:

$$E \coloneqq \begin{bmatrix} e_{1,1} & \cdots & e_{1,n} \\ \vdots & \ddots & \vdots \\ e_{n,1} & \cdots & e_{n,n} \end{bmatrix} \in \mathbb{B}^{n \times n}, \tag{2.3}$$

and $\mathbb{B} \coloneqq \{1, 0\}$ is the Boolean set, with $e_{i,j} = 1$ indicating a connection between components i and j and $e_{i,j} = 0$ indicating no connection. Each node has different attributes which correspond to the design objectives. Therefore inclusion/exclusion of a node from an architecture will have an effect on the overall objective function score. The set of nodes can be partitioned into subsets of components of similar types. For example $\mathcal{N}$ is partitioned as $\{\text{Tank}, FP, HE, \text{Valve}, OC, SP\}$ in Figure 2.2.

An architecture *template* is a set of nodes which are fixed. There may also be some connections between nodes which are fixed in the template as in Figure 2.2. Note

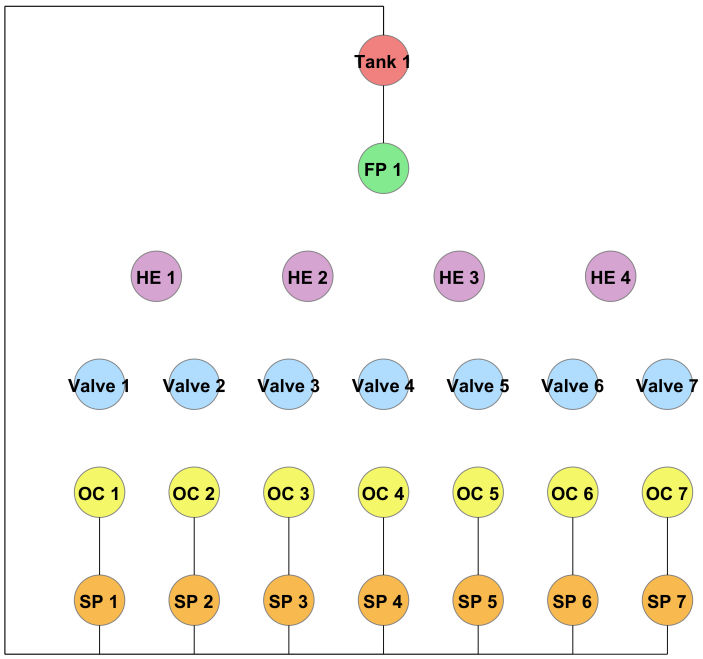


FIGURE 2.2: An architecture template for an actively controlled oil system. Connections between the tank, feed pump (FP), oil chambers (OC) and scavenge pumps (SP) are fixed. The heat exchanger (HE) and valve connections are yet to be determined. In the final architecture, any HE or valve nodes which are not connected to other nodes are not included in the design.

that the architecture template represents the maximal node configuration. There is no requirement for every node in the template to be used in the final architecture. The architecture optimization problem is then to determine the optimum set of connections between components to minimise the objective function f whilst satisfying the system requirements/constraints R. Any nodes which are not connected to other nodes in the final architecture are discarded. Note that this is approach is an example of a *connecting* architectural decision making process as described in [28].

In [19, 33] the methodology is applied to an aircraft electric power system (EPS) case study. Here the objective function is focused on minimising the cost of the architecture (number of nodes included) and complexity (number of connections amongst components). The interconnection constraints enforcing rules for how components should/should not be connected are expressed as inequalities on the edges $e_{i,j}$. In addition there are reliability constraints which are expressed as inequalities on combinations of the component reliabilities and the edges $e_{i,j}$. As the decision variable in this optimization problem is the Boolean matrix E (2.3), this is known as an integer program (IP). IPs can be solved using software such as the MATLAB toolbox YALMIP [51].

In [19, 33, 47] the architecture design is carried out at a high-level of abstraction using low-fidelity steady-state models for the components. Some research takes a different approach whereby the low-level, high-fidelity components models are simulated directly within the optimization algorithm [36, 39].

This has the advantage of being able to assess the low-level performance of candidate architectures visited by the optimization scheme. The performance of these high-fidelity simulations is the closest approximation of the real system performance and hence these methods should yield the best architectures. Unfortunately there are various downsides to these approaches. Firstly it can be impractical to simulate high-fidelity representations of more complex systems in a reasonable time-frame. In addition, when the system architecture is chosen at the start of a complex product development, these high-fidelity component representations may not have been developed. One of the key limitations for the actively controlled oil system is that the low-level performance cannot be evaluated without the controller, but the controller cannot be designed without the architecture of the system. This problem is common in any control system and hence multilevel approaches have been developed (see Section 2.2).

A recent attempt to address some of these problems through use of a two-level optimization scheme is presented in [34]. At the upper level the algorithm produces an architecture candidate using low-fidelity steady-state models. The architectural candidate is then passed to the lower level where sizing of the individual components is optimized using high-fidelity models. When no feasible component sizing can be found for a candidate architecture, a new set of constraints is added to the high-level optimization problem. For example, consider a high-level architecture that leads to a flow through a given component A which exceeds its upper bound in the low-level simulation. In this case, a new constraint can be added to neglect all architectures with upper bounds on flow which are smaller than the upper bound of A [34]. This process is repeated until an architecture is produced with a valid component sizing to meet all system requirements. Here the iterative mapping between the two levels is carried out automatically. This means the only inputs required are the system requirements and library of components with their interconnection rules. The algorithm will then run until a feasible architecture with optimum component sizings is reached.

Another multi-level optimization framework for systems-of-systems (SoS) architectures is presented in [37]. Here the framework follows a hierarchical structure with three levels resembling a tree of optimization problems. The method is applied to a noise-optimal aircraft design problem with: optimization of aircraft trajectories

at the SoS level; optimization of aircraft designs at the system level; and optimization of turbojet thrust and airfoil shape at the sub-system level. SoS architecture optimization is also considered in [38]. This approach uses multi-objective optimization (see Section 2.3.5) of all the SoS decision variables in a single Mixed Integer Non-Linear Program (MINLP), with links to dynamics and performance models to evaluate candidate solutions.

2.3.5 Multi-objective Architecture Optimization

The design of complex system architectures is usually multi-objective, with numerous, often-conflicting decision criteria. These capture the engineering characteristics of interest to the designers. For example, "robustness", "performance" or "technology maturity". When there are multiple objectives, there is often no single 'best' solution. Rather there is a set of Pareto-optimal solutions. This means that no Pareto-optimal solutions are *dominated* by other solutions which perform better against every objective [52].

A popular set of techniques for performing multi-objective optimization are evolutionary algorithms. These methods produce solutions that are a good approximation of the Pareto-front in reasonable time, despite not guaranteeing to be globally optimal [53] due to the NP-completeness of the problem [49]. Evolutionary algorithms are also able to produce solutions that would not be considered ordinarily by humans, such as the unusual organic-looking NASA evolved antenna [54]. In relation to system architectures, an unforeseen solution could be a high-scoring but unintuitive combination of options, which whilst unusual is a perfectly valid solution.

More discussion of multi-objective evolutionary algorithms in engineering design can be found in [55–58], and applications specifically for system architecture optimization in [59–61]. This is the approach used in the architecture synthesis stage of Chapter 3.

2.3.6 Determining Architectural Drivers / Decision Criteria

Whatever systems architecting approach is taken, a key task is to identify the *architectural drivers*. A 5-step method for identifying the architectural drivers by analysing and refining stakeholder requirements is presented in [62]. These

drivers are the motivating features of an architecture which correspond to either the constraints or decision criteria in more formal optimization-based design. For example, in [33] the architectural drivers are cost and complexity (decision criteria in the objective function) and reliability (one of the constraints).

2.4 Formalizing System Requirements

In many cases system requirements are comprised of both temporal and spatial constraints. For example, consider the following requirements: R1) If engine is in idle, RPM shall be between 200 and 300, else it shall be between 600 and 2000; R2) RPM should be greater than 500 within 2 seconds. R1 and R2 cannot be expressed as simple mathematical constraints, however, temporal logic languages are capable of formalizing these requirements.

2.4.1 Temporal Logic Formulae

A natural way of expressing system requirements in a formal language is temporal logic. Linear temporal logic (LTL) [63] extends traditional logic by adding the temporal operators: $\bigcirc$ (next), $\mathcal{U}$ (until), $\Box$ (always) and $\Diamond$ (eventually). For example take the LTL formula in (2.4). φ_1 is true iff whenever s_1 and s_2 are true, s_2 is not true at the next time step.

$$\varphi_1 = \Box\, s_1 \wedge s_2 \rightarrow \bigcirc \neg\, s_2 \tag{2.4}$$

The downside to LTL formulae is that they are only defined for discrete-time, finite-state systems. Real-time temporal logic [64] extends LTL to finite-state systems defined over the continuous time domain. For example, take the formula $\varphi_2 = \Diamond_{[0,20]} s_1$ which states that eventually between 0 and 20 seconds the signal s_1 becomes true. Two examples of Boolean signals which satisfy and do not satisfy φ_2 are shown in Figure 2.3.

Signal temporal logic (STL) [65] takes real-time temporal logic further to include real-valued signals. The way this is handled is through the use of *Booleanizers*. Given a signal x, a Booleanizer $\mu(x)$ is any function that converts the original real-valued signal into a Boolean signal. A simple example is a threshold function $\mu(x) = x > 0.6$ as demonstrated in Figure 2.4. After this process, the semantics of STL are the same as real-time logic, with STL formulae defined over time intervals. For example, $\varphi_3 = \Box_{[40,50]}\ x > 0.6$ would be satisfied by the real-valued

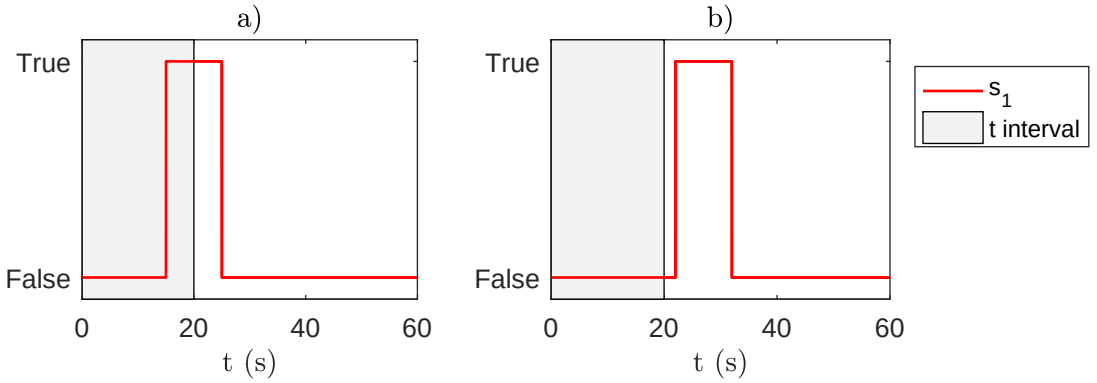


FIGURE 2.3: Boolean signals for real-time temporal logic formula $\varphi_2 = \Diamond_{[0,20]} s_1$. a) φ_2 is satisfied, b) φ_2 is not satisfied.

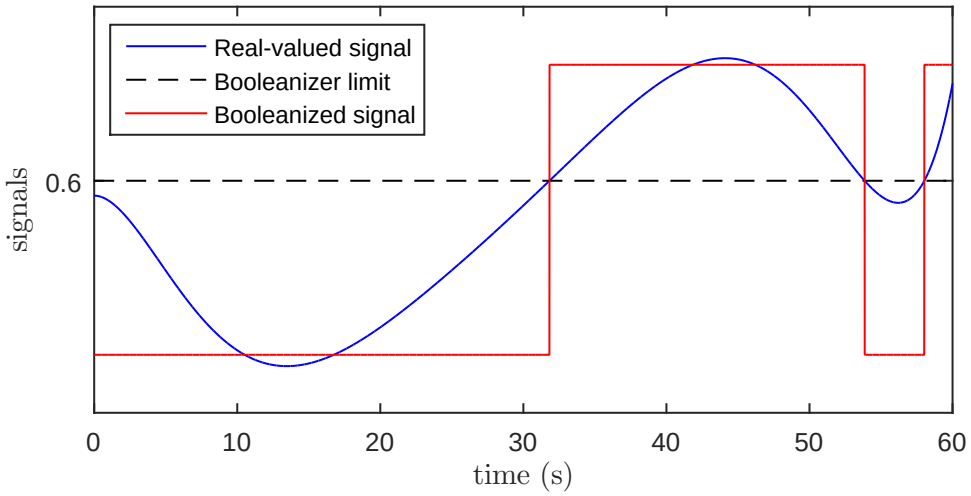


FIGURE 2.4: An example of converting from a real-valued to Boolean signal using Booleanizing function $\mu(x) = x > 0.6$

signal in Figure 2.4 because the Booleanized signal is always true between 40 and 50 seconds.

2.4.2 Verification of Temporal Logic Formulae

Analytical verification asks whether all possible behaviours of a model over infinite time satisfy a temporal logic formula i.e. $S \subseteq \varphi$. With real-valued, continuous-time signals, it is impossible to calculate the infinite-time behaviour. The use of bounded time intervals in STL means that verification can be carried out for finite-length traces; however, it is impossible to do an exhaustive search of the behaviour because real-valued signals can take an infinite number of values. One approach to solve this is to use reachable set approximation [66] to determine the entire

space of possible traces. Verification then consists of checking that the reachable set satisfies the temporal formula i.e. $\mathcal{R}(S) \subseteq \varphi$.

Another option is to use *monitoring* rather than verification. This consists of asking whether a specific trace satisfies the temporal logic formula i.e. for $s \in S$ check that $s \in \varphi$ [65]. Design of experiments is then needed to generate multiple simulation traces which cover the likely behaviour of the system. The process of monitoring for STL formulae is handled by the Breach toolbox [67, 68]. The advantages of monitoring are that it is a process that can be carried out much more quickly than analytical verification and hence can be incorporated into simulations with minimal overhead [67].

2.4.3 STL Quantitative Semantics

The problem with the use of Booleanizers for checking satisfaction of STL formulae, is that there is no indication of the *margin* of satisfaction. For example, take the simple STL formula $\varphi_4 = \Diamond_{[0,10]}\ x > 0$. Consider a signal $x_1(t)$ which briefly reaches a maximum of 0.1 at 9.9 seconds and a signal $x_2(t)$ which has a large positive value for the entire time interval. Clearly $x_1(t)$ is very close to violating the formula, while $x_2(t)$ satisfies it by a large margin. Unfortunately standard STL checking is unable to distinguish between these two cases. This led to the development of quantitative semantics for STL, presented in [69]. The idea is to define a function $\rho(x, \varphi, t)$ which is positive whenever the signal $x(t)$ satisfies the STL formula φ and negative whenever it does not. The more positive the value of $\rho(x, \varphi, t)$ the more *robust* the satisfaction of φ is. Similarly, the more negative the value of $\rho(x, \varphi, t)$, the more serious the violation of φ.

Space Robustness

There are three measures of robustness defined in [69]. The first measure, originally presented in [70], is termed *space robustness*. The space robustness at time t is defined as the distance between the magnitude of the signal and the Booleanizer limit. Consider an STL formula $\varphi_5 = \Box_{[0,10]}\ \text{speed}(t) < \text{speed}_{\text{max}}$. In this case the space robustness is defined by the function:

$$\rho(\text{speed}, \varphi_5, t) = \text{speed}_{\text{max}} - \text{speed}(t)$$

An example of two signals with large and small space robustness to formula φ_5 is presented in Figure 2.5.

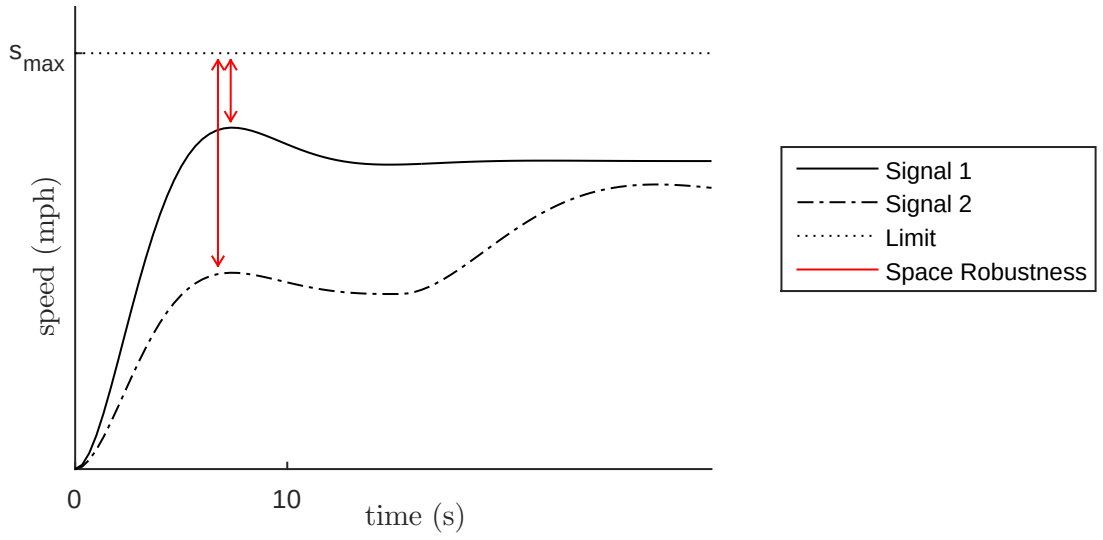


FIGURE 2.5: Two signals with different space robustness to the STL formula $\varphi_5 = \Box_{[0,10]}\ \text{speed}(t) < \text{speed}_{\text{max}}$

Time Robustness

The second measure of robustness in [69] is *time robustness*. This relates to the amount by which the signal can be shifted in time before the STL formula is violated. It is defined as a pair of functions $\rho^-(x, \varphi, t)$ and $\rho^+(x, \varphi, t)$ corresponding to shifts backwards and forwards in time respectively. Figure 2.6 shows an example of three Boolean signals with different time robustness to the formula $\Diamond_{[\tau_1,\tau_2]}x$. Clearly the signal in a) is very robust to forward shifts in time but not very robust to backwards shifts, while the opposite is true for c). The signal in b) is robust to shifts in time in both directions. In practical applications it may be more important to have robustness to shifts in a particular direction. For example, if the signal in Figure 2.6 is prone only to delays in events then clearly the signal in a) is much more robust than the one in c).

Space-Time Robustness

Space and time robustness can be combined into a single *space-time robustness* measure [69]. For a given spatial robustness c, the space-time robustness at time t can be visualised as the largest rectangle containing t of height c which fits below the space-robustness function $\rho(x, \varphi, t)$. Note that this rectangle is not unique as there are infinite choices of height c. Figure 2.7 shows that there is a trade-off between space and time robustness. With tighter space robustness c_2 the rectangle is taller but narrower. By relaxing the space robustness to c_1 the rectangle is much wider because the time-robustness is greater.

An efficient algorithm for computing robustness degrees which is linear in the size

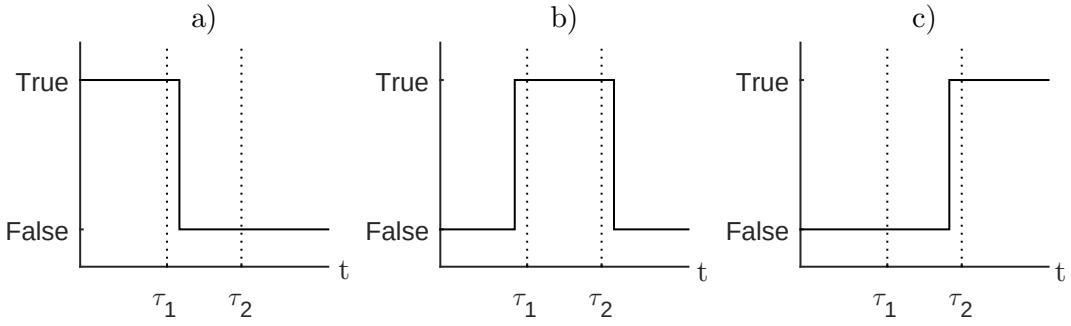


FIGURE 2.6: Boolean signals with different left and right time robustness to the STL formula $\Diamond_{[\tau_1,\tau_2]}x$: a) large forwards time robustness, small backwards time robustness; b) large forwards and backwards time robustness; c) large backwards time robustness, small forwards time robustness

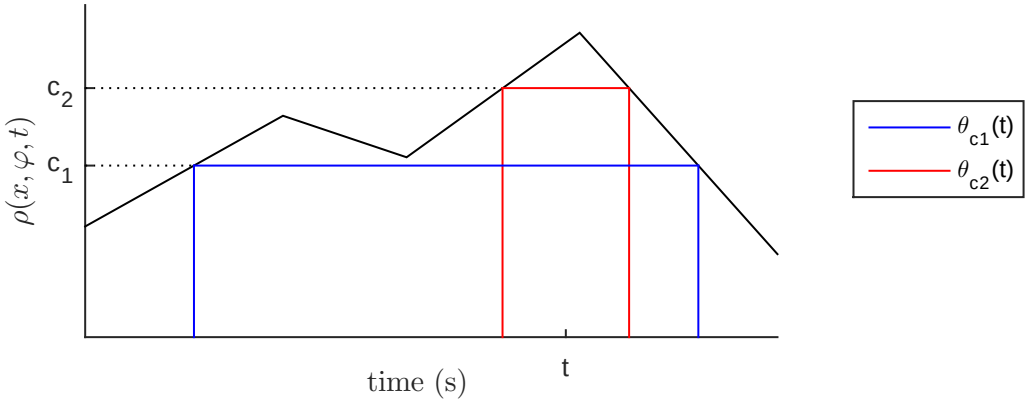


FIGURE 2.7: Space-time robustness $\theta_c(t)$ for two different space thresholds c_2 and c_2

of the signal is presented in [71]. This algorithm is implemented in the Breach toolbox [67]. One of the problems with checking STL robustness satisfaction is that the majority of algorithms use an offline approach. STL monitoring is usually based on simulations which can be computationally expensive and time consuming. To save time and resources it is desirable to terminate the simulation when a violation of the STL formula occurs. This is achieved by the *online* algorithm for robust STL monitoring presented in [72].

2.4.4 Weighted STL

One of the problems with the quantitative semantics of STL is the use of different units in compound formulae. For example, consider equation (2.5) which is taken from [73].

$$\varphi_6 = \Box_{[0,10]}(\text{speed} \leq 120 \wedge \text{RPM} \leq 4500) \tag{2.5}$$

If speed = 150 and RPM = 4530 it is clear that they both exceed their threshold by 30 units. In the standard space robustness measure, both would be violating the formula by an equal amount, but clearly the speed is violating its threshold by 25% while RPM is only violating its threshold by 0.67%. Weighted STL (WSTL) is introduced in [73] to counter this problem. Weights w_{speed} and w_{RPM} are used to solve the problem of different units. Then the space robustness calculations become:

$$\rho_{\omega_{\text{speed}}}(t) = (\text{speed}_{\text{max}} - \text{speed}(t)) \cdot w_{\text{speed}}$$
$$\rho_{\omega_{\text{RPM}}}(t) = (\text{RPM}_{\text{max}} - \text{RPM}(t)) \cdot w_{\text{RPM}}$$

The suggested choice for the weights in [73] is $w_{\text{speed}} = 1/120$ and $w_{\text{RPM}} = 1/4500$. This makes sense as a normalisation processes; however, it may be desirable for the engineers to weight a particular signal more highly. For example, if violations in speed could result in a fatal crash whilst violations in RPM result in reduced fuel efficiency, it may make sense to give a stronger weighting to the speed signal. The selection of weights remains an open research topic in WSTL.

2.5 Control Synthesis from STL Requirements

As stated in earlier sections, the advantage of using formal requirements is the ability to use them as inputs to design synthesis methods, which guarantee that they are satisfied. In [74] STL requirements are automatically encoded as a mixed-integer linear program (MILP) to be solved at each step of a model predictive control (MPC) optimization. In the absence of a cost function the quantitative semantics of the STL formula are used. This ensures not only that the controller satisfies the system requirements, but that it maximises the margin of satisfaction.

It is noted in [74] that the encoding of the MILP is computationally much more expensive than the solving. Since the encoding only needs to be done once at the start of the MPC problem, there could be potential for real-time execution of the algorithm. However, since solving an MILP is an NP-hard problem, there is no guarantee of finding a solution in polynomial time [75]. This is not a particular issue for systems with slow transients such as the smart building temperature control system used to demonstrate the approach in [74]. However, it is not an option for fast and safety-critical applications such as aerospace systems.

The STL-based MPC synthesis approach is extended in [76] to a robust control framework that can handle disturbances acting on the system. This utilises a

counterexample-guided inductive synthesis algorithm, which is computationally expensive. Other robust algorithms are explored in [77].

Some of these techniques have been implemented in the BluSTL toolbox for MATLAB [78]. The toolbox can be used to implement the encoding from STL to MILP and then solve the MPC problem whilst simulating the system.

2.6 Summary

This chapter has reviewed the literature relevant to this thesis. A general overview of complex systems engineering is given, looking at both horizontal processes of integration and vertical processes of abstraction/refinement [6, 7].

Multi-level approaches to system design can be useful for systems that are too complex to design in one high-fidelity stage. Model order reduction techniques [11, 13] can be useful for reducing the complexity of models for design exercises at higher fidelity levels. Top-down design processes such as the systems engineering V-modell® [10], the sequential model bounding approach of [14], or platform-based design [16] provide techniques for addressing the challenges of system design and verification.

One of the first design activities in a multilevel framework is system architecting. There are a variety of approaches for architecture design, ranging from informal guidelines [24, 27, 32] to multiobjective optimization [59–61]. This provides a background to the work carried out in Chapters 3 and 4.

Specifiying requirements in formal languages facilitates the use of design synthesis techniques such as optimization or control synthesis [74]. In particular, STL [65] is able to formalize a rich set of spatial and temporal requirements, with quantitative semantics defining the margin of satisfaction [69]. This margin of satisfaction can be used as an objective for optimization to maximise satisfaction of the requirements [74, 76, 77]. This provides a background to the research presented in Chapter 5.

Chapter 3

Customer-Oriented Preliminary Architecture Optimization

Referring back to the multilevel framework outlined in Figure 1.1, the first design stage is architecture optimization. This architecture design is often carried out as a two-level process, as outlined in Figure 1.2. This chapter focuses on the high-level architecture decision making which defines the *architecture framework* upon which a physical architecture topology can be designed (as in Chapter 4). The approach uses multi-objective optimization to produce a set of candidate solutions which are then refined interactively to a smaller set of interest to the customer. This chapter is partly based on research previously published in [79].

Designing complex system architectures involves analysing tradeoffs between multiple conflicting decision criteria to find a solution which best matches the preferences of the customer. This is usually done in the engineering characteristic (decision criteria) space, but the customer is generally more interested in higher-level characteristics. For example, the engineering characteristic "modularity" is not of direct interest to a customer, but it is related to their concern "through-life costs", since modular systems can be upgraded more easily. The relationships between customer and engineering concerns are many-to-many making it difficult to relate the two sets of priorities. This chapter proposes an integrated system architecture synthesis framework, which aims to maximise customer satisfaction by using their preferences directly to refine a set of candidate architectures. The main contribution of the research relates to the translation from customer preferences to decision criteria limits on a parallel coordinates plot. This automated flow facilitates rapid re-synthesis of "best" architectures following a change in customer preferences. The time saved allows customers to investigate a wider range

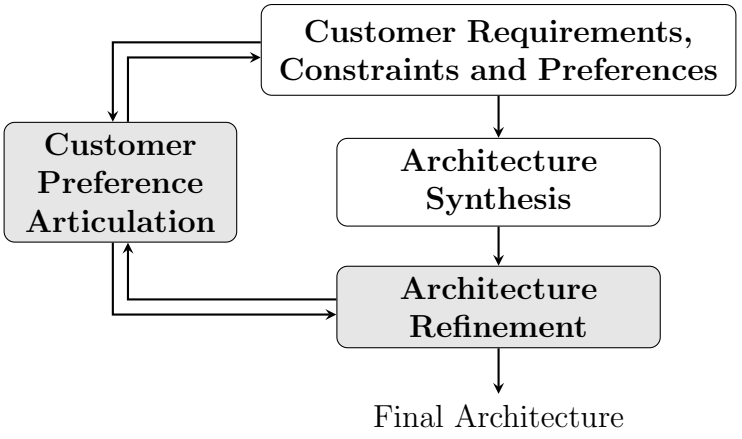


FIGURE 3.1: An overview of the architecture design process from customer concerns to chosen architecture. The contributions of this chapter are in the stages of the process highlighted in the grey boxes.

of concerns and gain a better understanding of how their priorities influence the solution set.

An approach for analysing the resilience of each solution in the set of "best" architectures to changing customer preferences is also presented. This is useful since large, complex projects with long timescales may experience changes in management, budgets or unforeseen circumstances that result in a change in customer priorities.

These ideas have also been implemented in a tool which allows system architects to perform architecture synthesis and refinement without the need to code the algorithms themselves. The research is demonstrated on two case studies: a control system for a pressurized water reactor and an oil system for turbofan engines.

3.1 Overview

This chapter presents a customer-oriented architecture design process. There are 3 main levels in the framework (see Figure 3.1) with the main developments of this chapter highlighted in the grey boxes. The first stage focuses on eliciting customer requirements and preferences. This is a complex task requiring iterative discussions with stakeholders/customers and effective communication [80]. A formal approach for determining customer preferences is presented in [81]. Multiple stakeholders are ranked according to their importance to the project. This is used to calculate an overall weighted sum of the individual customer preference weightings [81].

The second stage in Figure 3.1 is architecture synthesis which involves generating a set of candidate architectures that satisfy the customer requirements (Section 3.2). The architecture synthesis proposed in this chapter is based on multi-objective optimization, as shown previously in [35, 59–61]. There are two main advantages to this approach. Firstly, the use of optimization provides a wider search of alternatives than manually-specified architectures, whilst avoiding scalability problems associated with exhaustive searching (limited memory and processing power). Secondly, the use of multiple objectives facilitates the analysis of tradeoffs between different system characteristics which are inherent to complex systems problems.

The third stage in Figure 3.1 is architecture refinement, which involves selecting a solution or subset of solutions that best match the customer preferences (see Section 3.3). Despite optimization being used to synthesize candidate architectures in this chapter, the refinement stage is compatible with architectures derived from other methods (e.g. a manually generated set of candidates). The approach uses parallel coordinates [82] to display architecture solutions in terms of their scores for a set of decision criteria. To narrow down the solution set, users introduce limits on the upper bounds of the multiple, conflicting decision criteria to reflect their relative importance. This approach has previously been advocated in [59–61]. The refinement is achieved in [60, 61] through progressive preference articulation, whereby updating decision criteria limits triggers another optimization process focused on the new region of interest [58].

The contribution of this chapter addresses the challenge of bridging customer and engineering concerns in architecture refinement. Architecture quality is usually assessed with respect to engineering characteristics, but customers are often more interested in a set of higher-level concerns. For example, a customer may be interested in "availability", which is the extent to which the system can operate in the presence of faults or scheduled maintenance. This is related to numerous engineering characteristics such as "maintainability" or "robustness". When there are many customer preferences, many decision criteria and many relationships between them, it can be hard to determine which architecture solutions match the customer preferences most closely. This chapter solves the problem via a new algorithm for converting customer preferences into decision criteria limits on a parallel coordinates plot. The set of "best" solutions produced via this algorithm are then analysed to show how resilient they are to changes in the customer priorities. This allows the system architect to choose a solution which is likely to remain a good choice over the entire development lifecycle.

The benefits of the approach are seen as a substantial reduction in the time and effort needed to refine a set of architectures. This allows customers to investigate a wider range of preferences, giving insight into how their priorities affect the final solution set. Note that this work specifically addresses one of the key areas suggested for improving systems architecting in [83], to "reveal tradeoffs, tensions, strengths".

In Section 3.4 the customer-oriented architecture refinement approach is demonstrated on an electrical, control and instrumentation (EC&I) system case study for a pressurized water reactor (PWR). Section 3.5 applies the approach to an oil system for a turbofan engine case study. Section 3.6 describes how these developments have been implemented in a user-friendly tool. A summary of the chapter and concluding remarks are given in Section 3.7.

3.2 Architecture Synthesis

3.2.1 Function/Means Decomposition

The high-level architecture synthesis in this chapter is based on function/means analysis as discussed in Section 2.3.2. This is a decompositional approach with a list of functions the system must perform and a set of physical means for satisfying those functions. Table 3.1 shows a basic function/means decomposition for a generic control and instrumentation (C&I) system.

TABLE 3.1: Basic function/means decomposition for a generic C&I system.

Function	**Means**			
Sense variable	Basic tech. 1	Basic tech. 2	Smart sensor	
Transmit signals	Point-to-point	Single bus	Star	Complex topology
Control actuators	Human operated (mechanical)	Pneumatic	Electrical	

A high-level architecture is defined by selecting a single means for each function. For example, the highlighted cells outlined in Table 3.2 define an architecture with smart sensors, a single bus network and electrically controlled actuators.

TABLE 3.2: A high-level architecture defined via an assignment of means for each function.

Function	**Means**			
Sense variable	Basic tech. 1	Basic tech. 2	Smart sensor	
Transmit signals	Point-to-point	Single bus	Star	Complex topology
Control actuators	Human operated (mechanical)	Pneumatic	Electrical	

3.2.2 Scoring Against the Decision Criteria

To determine how "good" a given architecture is, it needs to be evaluated against an objective. With large-scale and complex systems there are often multiple objectives or decision criteria which reflect the wide range of stakeholder concerns. With low-level design, it may be possible to determine these criteria scores via simulation, mathematical models or component datasheets giving precise values such as "cost in US$". At the high-level architecture design phase there is insufficient information to do this. Therefore the approach taken in this chapter is to use experienced engineers to provide decision criteria scores for the different solutions. However, the optimization and architecture refinement described in later sections would be compatible with any method of generating the criteria scores.

The scoring in this chapter is carried out in relation to a baseline/default solution (0), with negative scores indicating an improvement in that criterion and positive scores indicating a worse option. Negative has been chosen to mean "better" to reflect the fact that the optimization (as discussed in Section 3.2.3) attempts to minimise the objective scores. In a scoring-based approach, engineers would typically give criteria scores for a whole architecture. However, this limits the number of options that can be reasonably evaluated. In this research, the scoring is instead carried out for each of the means. This allows an overall score to be calculated for any architecture via summing of its individual means scores, as shown in Table 3.3.

TABLE 3.3: Means scored against decision criteria for a minimal C&I system example.

Function	Means	Criterion 1	···	Criterion N
Sense variable	Basic tech. 1	0	···	0
	Basic tech. 2	-1	···	4
	Smart sensors	4	···	-2
Transmit signals	Point-to-point	0	···	0
	Single bus	-2	···	2
	Star	-2	···	2
	Complex topology	-4	···	2
Control actuators	Human operated	0	···	0
	Pneumatic	-2	···	4
	Electrical	-4	···	-4
	Overall	**-2**	···	**-4**

This approach allows criteria scores to be generated for any combination of means, facilitating a much larger search of the solution space (as via multiobjective optimization in Section 3.2.3). For example, the case study outlined in Section 3.4 contains 7 functions with 30 individual means to be scored against the decision criteria. These means can be combined to give 16,200 unique architectures, which would be too time consuming to score manually.

Note that other methods of combining the scores for a combination of means could be used. For example, a weighted sum or multiplication of the individual values. This would have an impact on the solutions generated by an optimization algorithm (as presented in Section 3.2.3).

3.2.3 Multiobjective Genetic Algorithm Optimization

As mentioned in Section 3.2.2 the approach taken in this chapter is multiobjective optimization. The generic multi-objective system architecture optimization problem is defined in [35] as having 3 parameters:

1. $f = (f_1, f_2, \cdots, f_n)$ - a multi-criteria objective function.
2. R - a set of constraints defining an admissible set of parameters/variables.
3. N - an architectural framework for deriving a set of candidate solutions A.

The solution involves finding a subset of solutions $A' \subset A$ which satisfy the constraints R whilst being nondominated with respect to f (meaning that improvements for one criterion cannot be achieved without decreasing performance for another). The constraints R are the combinations of means that are incompatible. For example, "means m_i is not compatible with means m_j", expressed in the form $\neg(m_i \wedge m_j)$.

Note that, as defined, the multi-criteria objective function f can have any number of criteria n greater than or equal to 2. The architecture design problems considered in this chapter are not just multi-objective but *many-objective* (i.e. 4 to 20 decision criteria as defined in [58]).

As mentioned in Section 2.3 it has been proven that architecture design is NP-complete [49], meaning no guarantee of finding optimal architectures in polynomial time. However, evolutionary algorithms have overcome this challenge to produce solutions which are a good approximation of the pareto front, despite not guaranteeing to be globally optimal [53]. For example, see Figure 3.2 which compares the decision criteria scores for architecture candidates synthesised via exhaustive search or evolutionary algorithm for the PWR system discussed in Section 3.4. In this figure, the decision criteria scores are represented on a parallel coordinates plot [82]. This technique allows visualisation of N-D data on a 2-D plot by representing architecture solutions as a line joining the architecture scores for each decision criterion (see Section 3.3.1 for further elaboration). Note that in Figure 3.2 the evolutionary algorithm approximates the best solutions produced via the exhaustive search (similar minimum criteria values). For this reason, they are used in this chapter to synthesise the set of candidate high-level architectures.

The type of evolutionary algorithm used in this chapter is the Multiobjective Genetic Algorithm (MOGA) approach presented in [58]. This is a nondominated ranking approach which is summarised in Figure 3.3 and described in the following steps.

- **Randomly generate an initial population** - A predetermined number of initial architectures are generated by randomly selecting a means for each of the primary functions.
- **Evaluate decision criteria scores for the initial population** - For each candidate architecture the aggregated decision criteria scores are calculated as a sum of the scores of the means chosen (as outlined in Table 3.3).
- **Loop through the following steps for a predefined number of iterations**

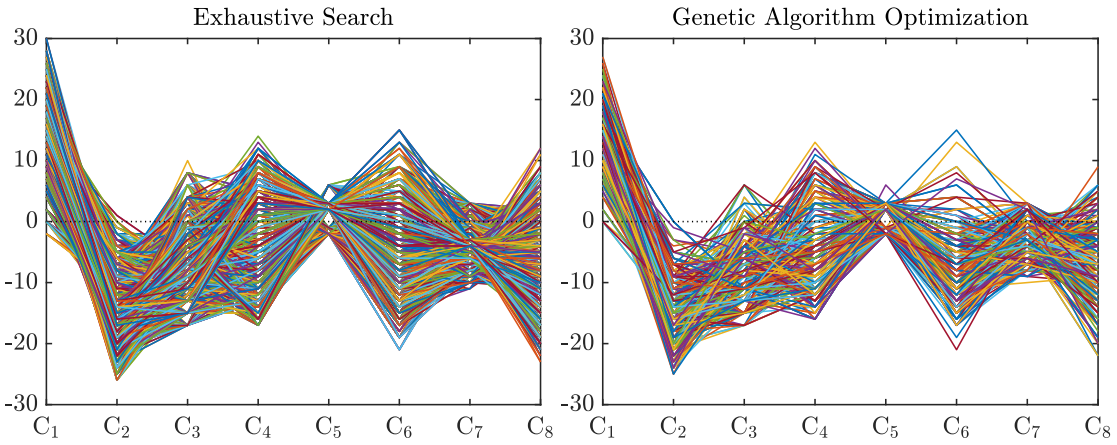


FIGURE 3.2: Candidate architecture solutions generated via exhaustive search of 16,200 potential solutions (left plot) and genetic algorithm optimization (right plot). The parallel coordinates plots represents solutions as a line joining the architecture scores for each decision criterion C_1 to C_8. It is clear that the genetic algorithm does a good job of approximating the best solutions (similar minimum criteria values).

1. **Do Pareto-based rank sorting** - This sorts the current solution set according to their Pareto-based rank as defined in [55]. The rank for each solution is calculated as the number of other solutions which dominate it (which means they score better for every decision criterion). A solution is said to be nondominated if improvements for one criterion cannot be achieved without decreasing performance for another. Therefore these solutions have a Pareto-based rank of 0, since there are 0 solutions which dominate them. A graphical outline of this principle is shown in Figure 3.4.
2. **Get parent population from sorted population using tournament selection** - In tournament selection a set number of solutions are chosen at random from the overall population (2 solutions are used in this chapter). These solutions are the tournament participants and the winner is the solution with the lowest rank. This is repeated until the parent pool size has been filled.
3. **Perform crossover to get child population** - For every pair of parent solutions, two children are made. The first child starts with the same means as the first parent, but for each function there is a chance of "crossover" occurring to inherit the means from the second parent. The opposite is true for the second child solution. These crossovers occur randomly with a probability of 0.5 in this chapter.

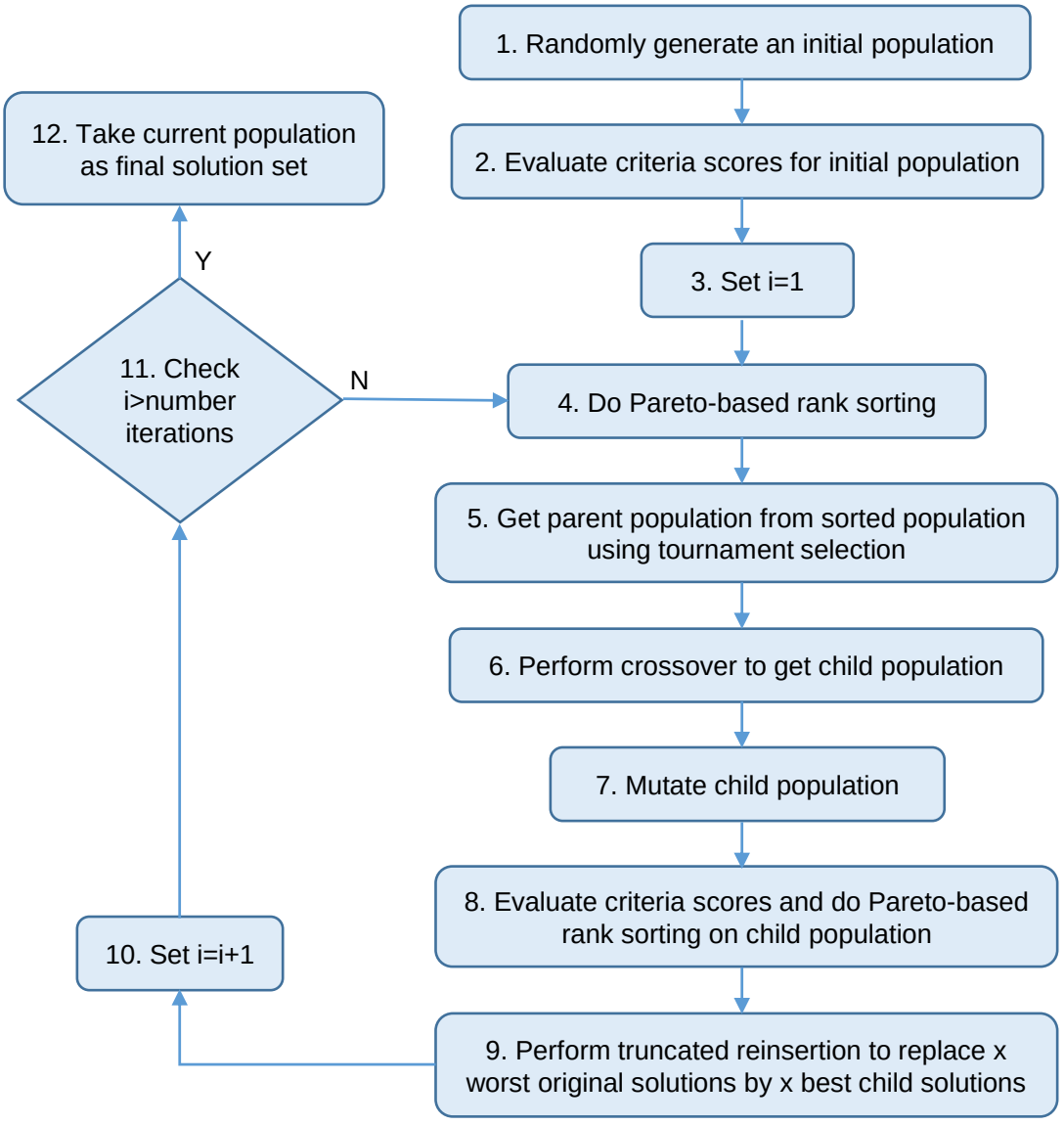


FIGURE 3.3: An overview of the architecture synthesis using multiobjective genetic algorithm (MOGA) optimization.

4. **Mutate child population** - For every child solution, the means may mutate randomly with a set probability (0.1/N where N is the number of functions). This causes some new means not present in the parent population which prevents the solution set from getting stuck at local minima.

5. **Evaluate decision criteria scores and do Pareto-based rank sorting on the child population** - This calculates the aggregate decision criteria scores of the child population and sorts them according to the Pareto-based rank.

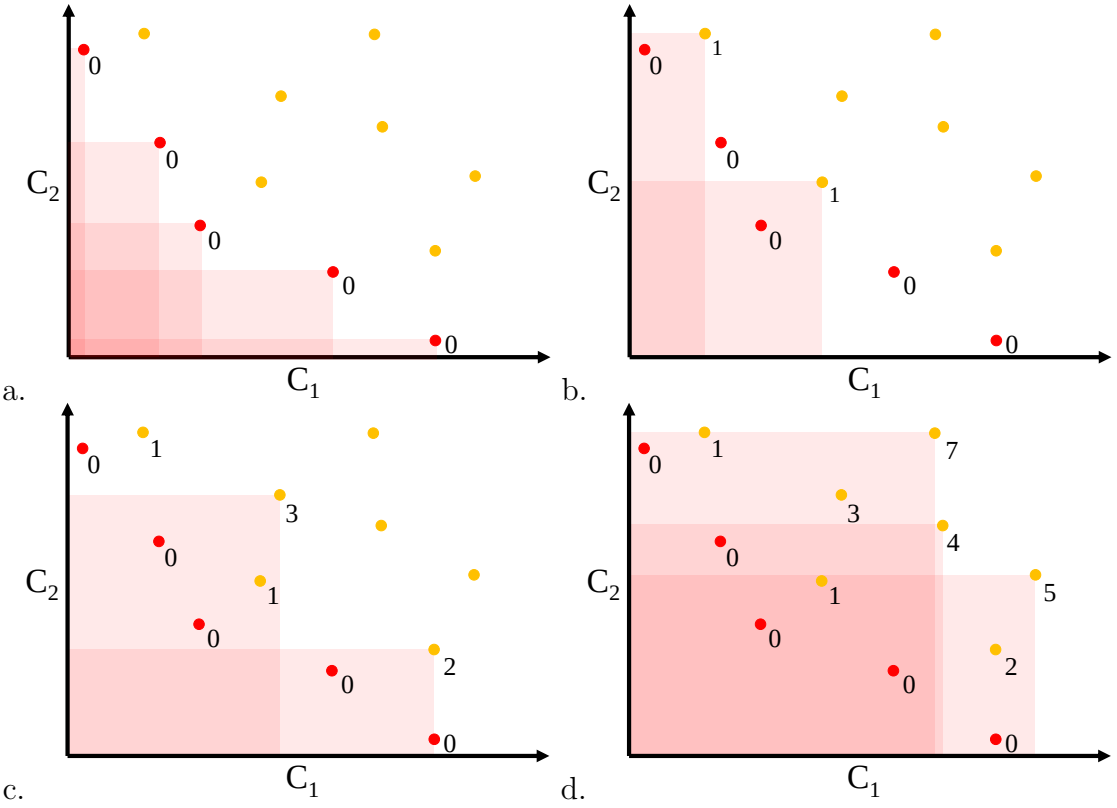


FIGURE 3.4: A graphical illustration of Pareto-based ranking [55] for two objectives. In this case, both objectives are aiming to be minimised. The rank of each solution can be visualised by drawing a square from the solution point to the origin and observing how many solutions lie within the highlighted area. Subfigure a. shows the pareto front of nondominated solutions (0 rank) highlighted by the red dots with dominated solutions shown as orange dots. Subfigure b. shows the solutions which are dominated by one other solution. Subfigures c. and d. show solutions which are dominated by more solutions.

6. **Perform truncated reinsertion to replace a percentage of the worst original solutions by the same percentage of best child solutions** - This ensures that good new solutions from the child population are added to the solution set, whilst maintaining the best original solutions. The percentage is set between 0% and 100% with a value of 50% chosen in this chapter.

- **Take the nondominated solutions from the current population as final solution set** - After the predetermined number of iterations through the algorithm, the final solution set is used as the start point for the architecture refinement discussed in Section 3.3

3.3 Architecture Refinement

The inputs to the architecture refinement stage are: 1) a set of candidate architectures, which could have been generated manually or via a formal architecture synthesis stage (Section 3.2); and 2) a set of customer preferences. The output is a subset of solutions which best match the specified preferences.

A commonly used tool for refining a set of candidate architectures is the Pugh matrix [40]. This involves ranking candidate architectures against the various decision criteria in a matrix. Weights are used to assign relative importance of the different criteria. The "best" architecture is chosen as the one with the highest overall weighted-sum of criteria scores. The downside to this approach is that it requires manual scoring of every architecture and hence the number of candidates that can be practically explored is limited.

The methodology presented in this section facilitates a much wider search by determining the overall architecture scores as a sum of the individual scores for the function means chosen (see Section 3.2 for details). This only requires scoring of the different means in order to get overall architecture scores for any valid combination of these options. Another advantage of this approach over the Pugh matrix method is the improved visualisation of tradeoffs between decision criteria as discussed in Section 3.3.1 and Section 3.4.1.

3.3.1 Reducing the Solution Set Using Parallel Coordinates

Visualising many objective scores on a single plot is difficult due to the limits of 3 dimensions. Parallel coordinates solve this problem, allowing visualisation of N-D data on a 2-D plot [82]. The use of parallel coordinates for visualising multiple engineering design objectives is introduced in [55] and applied to various multiobjective problems in [56–61]. This section uses progressive preference articulation in a multi-objective evolutionary algorithm to narrow down the region of interest on the Pareto front. When preferences are updated (via changing parallel coordinates limits) the optimization stage is repeated to find a new population of solutions in the new region of interest. An example parallel coordinates plot is shown in Figure 3.5. A solution is represented as a line linking each of its individual criterion scores with respect to a baseline solution (the zero line). To reduce the solution space, the upper bounds on the decision criteria (limits) can be tightened progressively until a solution or subset of solutions is chosen.

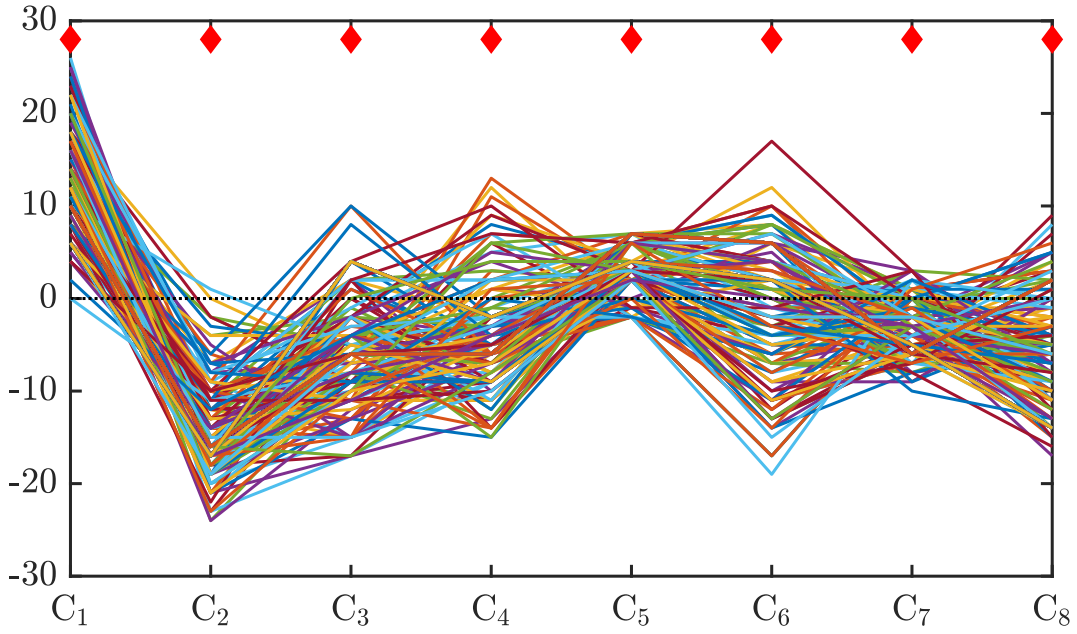


FIGURE 3.5: An example parallel coordinates plot for a system with 8 decision criteria. The decision criteria scores (y-axis) are compared against a baseline solution (the zero line), with negative values indicating an improvement over the baseline and positive values indicating a worse solution for that criterion. A solution is represented by a line linking its score for each criterion. The red diamonds represent the upper limits for each criterion, which can be tightened/relaxed to refine the solution set.

3.3.2 Customer-Oriented Architecture Refinement

Section 3.3.1 describes how progressive refinement of decision criteria limits in a parallel coordinates plot can be used to reduce the candidate solution set. However, this can be a difficult process in real world practice. The main problem is that the system architects need to select a solution based on the preferences of the customer/end-user, but these stakeholders are rarely interested directly in the decision criteria such as "modularity" or "robustness". Rather they are interested in a set of related customer preferences such as "through-life running costs" or "project delivery risk". This section presents a novel methodology for solving this problem by translating customer preferences to parallel coordinates limits.

It is highlighted in [84] that design decisions around tradeoffs in conflicting objectives are often made for non-technical reasons, and that having a structured method for performing architecture tradeoffs can be useful for identifying good solutions early in the design cycle. The approach presented in this section seeks to

facilitate this kind of insight, to narrow down on the 'best' solutions from a large Pareto optimal set.

Translating customer attributes to engineering characteristics has been achieved in Quality Function Deployment (QFD) [85] via the house of quality [86]. This essentially involves defining a matrix of many-to-many relationships between the customer and engineering characteristics, with the magnitude/sign of the matrix values indicating the strength of any positive or negative relationships. Motivated by QFD, the relationships between customer preferences and decision criteria preferences are defined here through matrix R_{pref}.

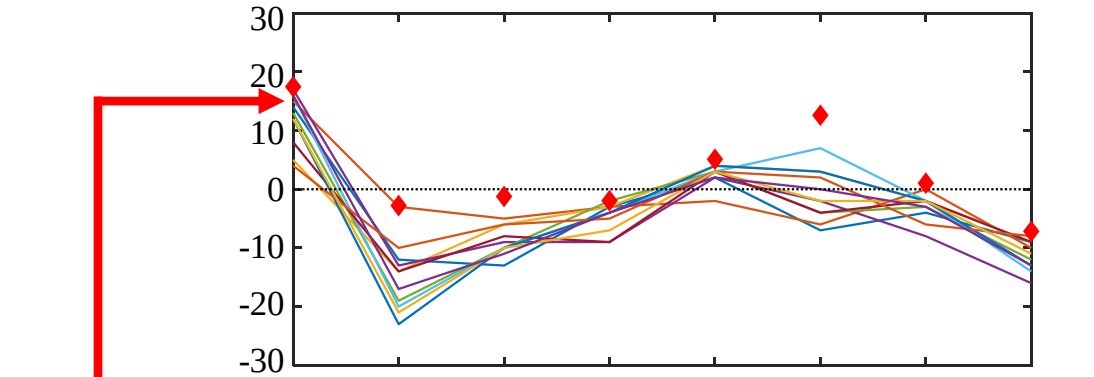
$$R_{\text{pref}} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,m} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,m} \end{bmatrix}. \tag{3.1}$$

The value $r_{i,j}$ indicates the strength of relationship between customer preference i and decision criteria preference j. In this research, a three-valued scale is used with 0 indicating no relationship, 3 indicating a weak relationship and 9 indicating a strong relationship, as used previously in [87]. However, the methodology presented is compatible with other scales and negative numbers to indicate negative relationships, as implemented in the oil system case study of Section 3.5.1. Defining R_{pref} is the most challenging task in the refinement process, since it requires a consensus to be reached between the customers and engineers about the presence and strength of any relationships. However, this only needs to be done once in the project, whereas manual preference articulation (Section 3.3.1) requires numerous iterations of customer-engineer discussions. An example of R_{pref} defined for a problem with 7 customer preferences and 8 decision criteria is shown in the orange matrix in the bottom right of Figure 3.6.

The customer preference weightings P_{weights} (blue vector in the bottom left of Figure 3.6) are specified as a percentage of the overall importance (summing to 100% in total). Once these have been defined, the decision criteria weights (green vector in Figure 3.6) can be defined using:

$$C_{\text{weights}} = P'_{\text{weights}} R_{\text{pref}}. \tag{3.2}$$

where P'_{weights} is the row vector (transposed) form of column vector P_{weights}, and R_{pref} is defined as in equation (3.1).



C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
360	270	405	465	300	255	270	510

P_1	5	0	0	0	9	9	9	3	0
P_2	30	3	0	3	0	0	0	3	9
P_3	10	3	9	3	9	0	9	0	0
P_4	5	3	0	0	0	0	9	3	9
P_5	20	0	9	3	3	9	3	3	9
P_6	25	9	0	9	9	3	0	3	0
P_7	5	0	0	0	9	0	3	3	3

FIGURE 3.6: A visualization of the conversion process from customer preferences to parallel coordinates limits. Preference weights (blue) are multiplied by the conversion matrix (orange) to give decision criteria weights (green). Darker hues of blue/green indicate more importance and darker hues of orange indicate a stronger relationship between the i^{th} preference and j^{th} decision crtierion. The decision criteria weights are converted to the parallel coordinates limits using Algorithms 1 and 2. Note that more important decision criteria (e.g. C_8, C_4, C_3) have tighter limits on the parallel coordinates plot.

The C_{weights} values show the relative importance of the decision criteria, but they do not define absolute limits or upper bounds for the criteria scores. Therefore, the weights need to be translated to parallel coordinates plot limits. Doing this requires consideration of both the relative difference between the values in C_{weights} and of the range of scores for each of the decision criteria. To explain this, see criteria C_1 and C_2 in Figure 3.5 and Figure 3.6. All of the candidate architecture scores for C_1 are greater than or equal to the baseline 0 value. An example criterion likely to show this pattern is "technology maturity" whereby very few novel architecture options would score as highly as a well-established baseline.

In contrast, almost all of the candidate architecture scores for C_2 are below the baseline 0 value. An example criterion likely to show this pattern is "performance"

in an electronic system whereby newer components would have higher processing power and lower size/weight than an older baseline solution. Considering just the values of C_{weights} (green vector in Figure 3.6) the limits on C_1 would be tighter (lower) than the limits on C_2. However, when the ranges of the different criteria scores are considered too, the C_1 limit is higher than the C_2 limit, as shown in the top of Figure 3.6. This is due to the differences highlighted in their respective criteria scores.

A method for performing the translation from C_{weights} to parallel coordinates plot limits is presented in Algorithm 1. The first part of the algorithm (lines 1 to 5) determines the pattern of the decision criteria limits based on the C_{weights} values and the maximum/minimum scores for each of the decision criteria. In particular, line 4 ensures that for each criterion the limit will be set somewhere between the maximum solution score if $C_{\text{weights}}(i) = 0$ (allowing all solutions for that criterion) and the minimum solution score if $C_{\text{weights}}(i) = \max(C_{\text{weights}})$ (allowing only the single best scoring solution for that criterion). Once, this pattern of criteria limits is set, Algorithm 2 can be used to see how many solutions fall below all of the decision criteria limits (line 6).

Initially, the limits are likely to be too constrained so that no feasible solutions are produced. Therefore the second part of Algorithm 1 (lines 7 to 16) is used to shift the limits upwards whilst maintaining the overall pattern of preferences, until a specified x number of solutions have been found. Line 12 ensures that the limits are increased relative to the range of scores for each of the criteria. Taking the example in Figure 3.5, C_5 would increase less than C_6 since there is a smaller range of scores in the different solutions.

The results of Algorithm 1 are displayed in the plot at the top of Figure 3.6. In this example $x = 10$, meaning the limits have been scaled to show the 10 best solutions according to the customer preferences. Note that the use of the two algorithms presented here produces these solutions almost instantly after a change in customer preferences, whereas a manual refinement of the decision criteria limits would take much longer.

Algorithm 1 Setting limits to find the subset of x solutions which best match the decision criteria preferences.

Input: C_{scores} - matrix of the solution set with the $(i,j)^{\text{th}}$ element defined as the score for criterion j in solution i.

Input: C_{weights} - indicating preference of the different decision criteria to the customer.

Input: x - the number of solutions required.

Output: C_x - the subset of the x best solutions according to the C_{weights} chosen.

1: **for** i = all decision criteria **do**
2: Set $C_{\max}$ and $C_{\min}$ to max/min C_{scores} values in the i^{th} column
3: Set $W_{\max}$ and $W_{\min}$ to max/min C_{weights} values
4: Set $\text{limits}(i) = (C_{\max} - C_{\min})\frac{(W_{\max} - C_{\text{weights}}(i))}{W_{\max}} + C_{\min}$
5: **end for**
6: Calculate valid solutions for the initial limits using Algorithm 2
7: Set adjustment factor $c_{\text{adj}} = 0$
8: **while** number of valid solutions $< x$ **do**
9: $c_{\text{adj}} = c_{\text{adj}} + 0.001$
10: **for** i = all decision criteria **do**
11: Set $C_{\max}$ and $C_{\min}$ to max/min C_{scores} values in the i^{th} column
12: Set $\text{limits}(i) = \text{limits}(i) + c_{\text{adj}}(C_{\max} - C_{\min})$
13: **end for**
14: Calculate valid solutions for new limits using Algorithm 2
15: **end while**
16: Export limits and the x best solutions to C_x

Algorithm 2 Finding valid solutions (satisfying all decision criteria limits) from the overall solution set

Input: C_{scores} - matrix of the solution set with the $(i,j)^{\text{th}}$ element defined as the score for criterion j in solution i.

Input: limits - upper bounds on the scores for each criterion.

Output: C_{valid} - subset of solutions satisfying all limits

1: **for** i = all solutions **do**
2: **for** j = all decision criteria **do**
3: **if** $C_{\text{scores}}(i,j) > \text{limits}(j)$ **then**
4: mark solution as invalid
5: **end if**
6: **end for**
7: **end for**
8: Export all solutions not marked as invalid to C_{valid}

The architecture refinement approach presented in this section aims to reduce the large Pareto-optimal solution set to a more manageable set of solutions for selecting an architecture from. An alternative approach is presented in [87] wherein clustering analysis is used to group the entire Pareto-optimal solution set into discrete clusters with common features. This allows tradeoffs between different clusters to be analysed using a parallel coordinates plot. Once a preferred cluster is chosen, the system architects can perform further tradeoff analysis within the cluster to refine to a single architecture. In some cases it may not be possible to group the Pareto optimal solution set into well-separated clusters. For this reason, the customer oriented architecture refinement approach is preferred in this chapter.

3.3.3 Resilience to Changing Customer Requirements

Resilience is a prominent topic in systems engineering, with an increasing amount of research being directed towards engineered resilient systems [88]. This relates to designing systems which are "effective in a wide range of operational contexts with the ability to respond to new or changing conditions through modified tactics, appropriate reconfiguration or replacement" [88].

In the case of very large-scale systems with long development cycles, this becomes relevant even before delivery of the project. For example, the development cycle from concept studies to delivery of the first UK Astute nuclear submarine was around 15 years [89]. Over these long development cycles, changes to customer priorities as the project progresses can be a particular issue. For example, "Non-Recurring Engineering Costs" or "Delivery Risk" often become more important as budgets are finalised and deadlines are approached.

This section proposes a method for analysing the resilience of different solutions to changes in stakeholder priority weightings. The analysis starts by taking the x best solutions according to the stakeholder priority weightings as outlined in Section 3.3.2. The algorithm then loops through all the solutions to check how much each priority weighting can be increased and decreased before the parallel coordinates limits have changed enough that the solution is no longer in the set of x best solutions. An outline of this process flow is shown in Figure 3.7.

This resilience data could potentially be very powerful when narrowing down the solution set with stakeholder discussions. For example, consider a particular solution which is the favoured choice with the current performance measure priorities, but which also has very small resilience values for some performance measures.

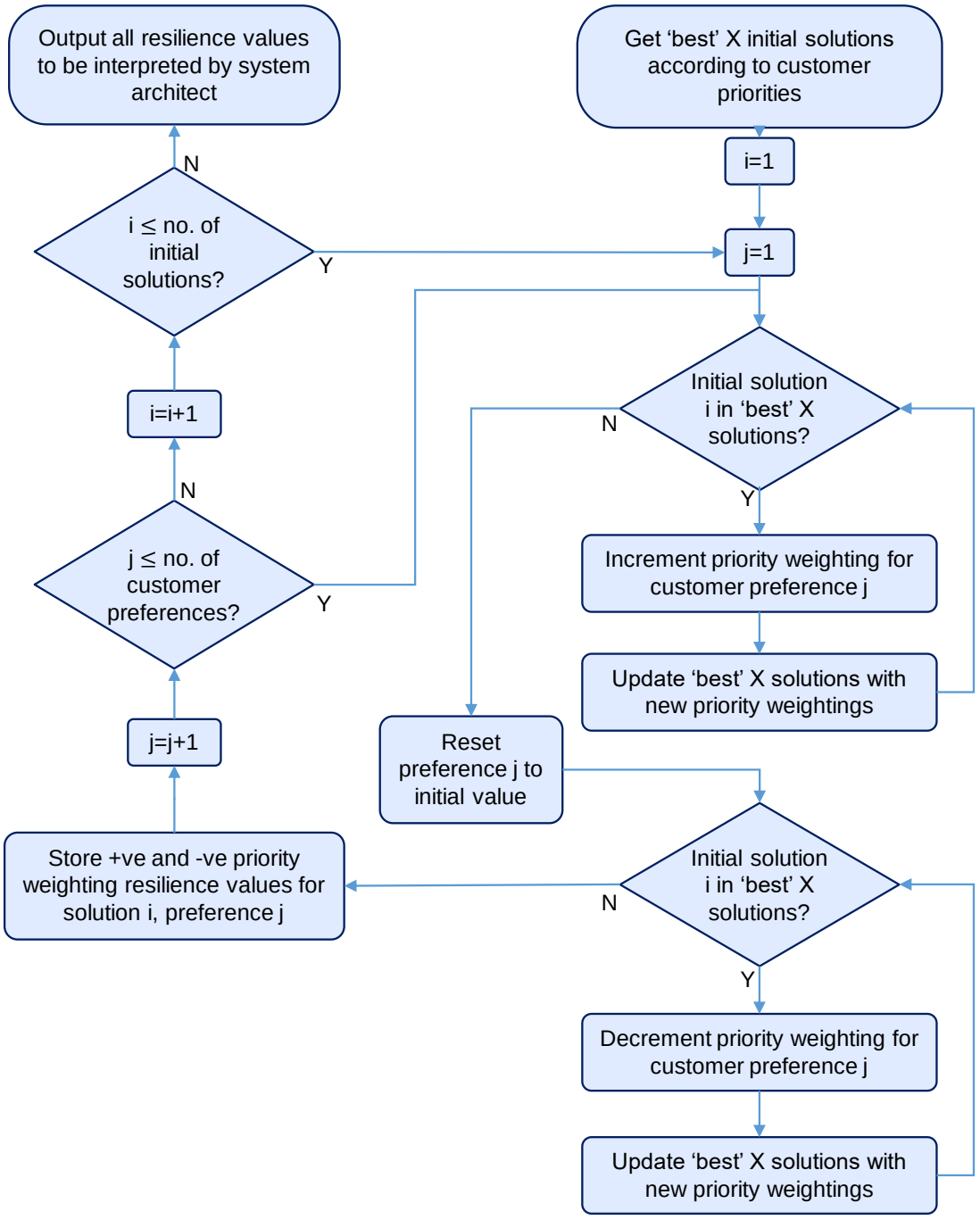


FIGURE 3.7: A process flow for the customer preference resilience analysis. The flow loops through every selected solution and every stakeholder preference to determine how much the priorities can increase or decrease before losing that solution from the 'best' x solutions.

This would highlight to the customer that the solution would only remain the best option if they are certain that these performance measures will not become more important later in the project cycle. With this feedback it may be decided

that a more sensible choice is a slightly worse scoring solution which is more resilient to changes in priorities over the life cycle. This is discussed more detail in the case study in Section 3.5.

3.4 Case Study - Pressurized Water Reactor EC&I System

The approach presented in Section 3.2 and Section 3.3 has been applied to an architecture selection problem for a pressurized water reactor (PWR) electrical, control and instrumentation (EC&I) system. A PWR is a type of nuclear reactor which uses a closed, pressurized loop of water that passes through the reactor core, absorbing heat. This heat is transferred to a second loop of water which is not pressurized, allowing vaporization and the production of steam. The steam generated is used to drive a turbine to produce power (see Figure 3.8) [90]. The main roles of the EC&I system in a PWR are to provide all of the sensing, signal transmission, data processing, human/machine interface and actuation needed to maintain stable temperatures/pressures and initiate protection or shutdown procedures when necessary.

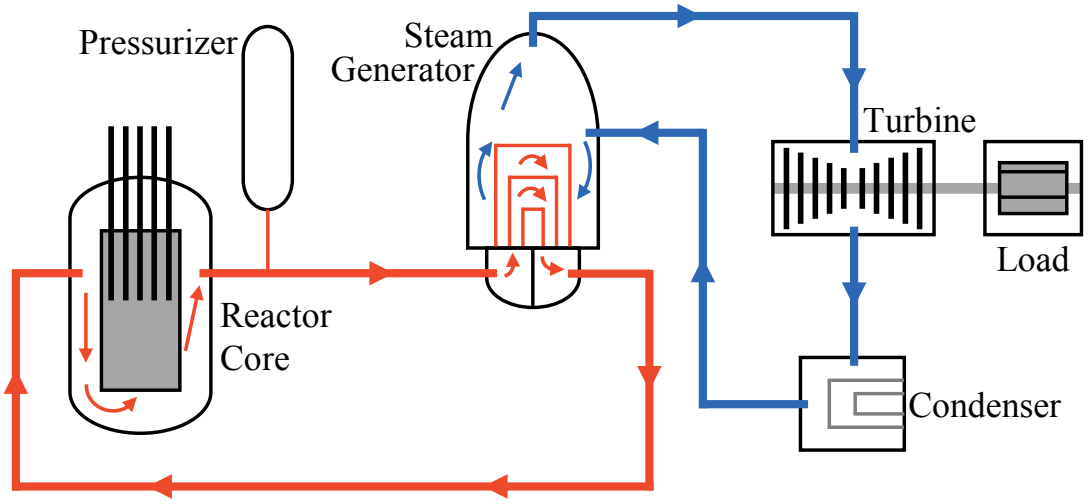


FIGURE 3.8: A diagram of a pressurized water reactor. The pressurized water loop (red) transfers heat from the reactor core to the unpressurized loop (blue) via the steam generator. The steam is used to drive a turbine, linked to a load. The steam is condensed then returned to the steam generator [90].

.

TABLE 3.4: Decision criteria for a PWR EC&I system.

No.	Decision Criterion	Description
C_1	Internal commonality	The degree to which features, attributes and environments can be shared.
C_2	Modularity	The degree to which system components can be separated and recombined.
C_3	External consistency	The degree to which physical and functional elements are governed by established standards.
C_4	Maintenance and test	The degree to which the system supports effective and efficient maintenance and testing.
C_5	Security	The degree to which the system is protected from threats.
C_6	Robustness	The degree to which the system can perform its function under stated conditions for a specified period of time.
C_7	Usability	The degree to which the human interfaces are easy to use.
C_8	Performance	The degree to which the system accomplishes its designated functions within given constraints.

TABLE 3.5: Customer preferences for a PWR EC&I system.

No.	Customer Preference	Description
P_1	Availability	The degree to which the system is able to operate at any time.
P_2	NRE Costs	The up-front costs of designing, verifying and certifying the system.
P_3	Through-life costs	The ongoing costs of operation, maintenance, upgrading and decommissioning.
P_4	Size & mass	The physical size and mass of the system.
P_5	Sustainability	The long-term availability of spare parts, knowledge and expertise needed to maintain the system.
P_6	Delivery risk	The degree of risk that the system will not be delivered on time and/or on budget.
P_7	Manning	Reduction in operation, monitor and maintenance requirements of the system.

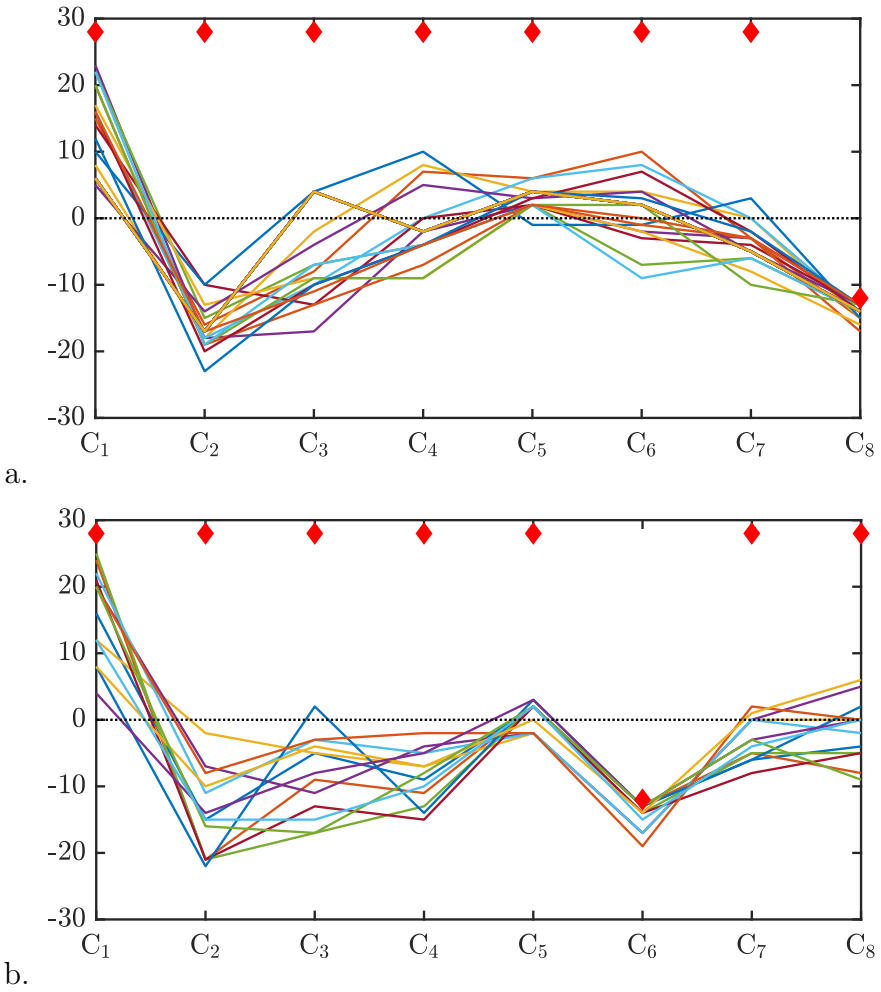


FIGURE 3.9: An analysis of tradeoffs in the PWR EC&I case study. These have been explored by setting the decision criteria limits manually. Plots a. and b. show a tradeoff between performance (C_8) and robustness (C_6). This tradeoff is represented by the fact that the best scoring solutions for either criterion, score moderately to badly for the other.

The PWR EC&I architecture is decomposed into 7 functions with 31 different options and a total of 16,200 candidate architectures. To protect commercial interests, these are not described in detail here. There are 8 decision criteria and 7 customer preferences as shown in Table 3.4 and Table 3.5 respectively. Performing multi-criteria architecture optimization produces a set of 255 nondominated solutions as shown in Figure 3.5.

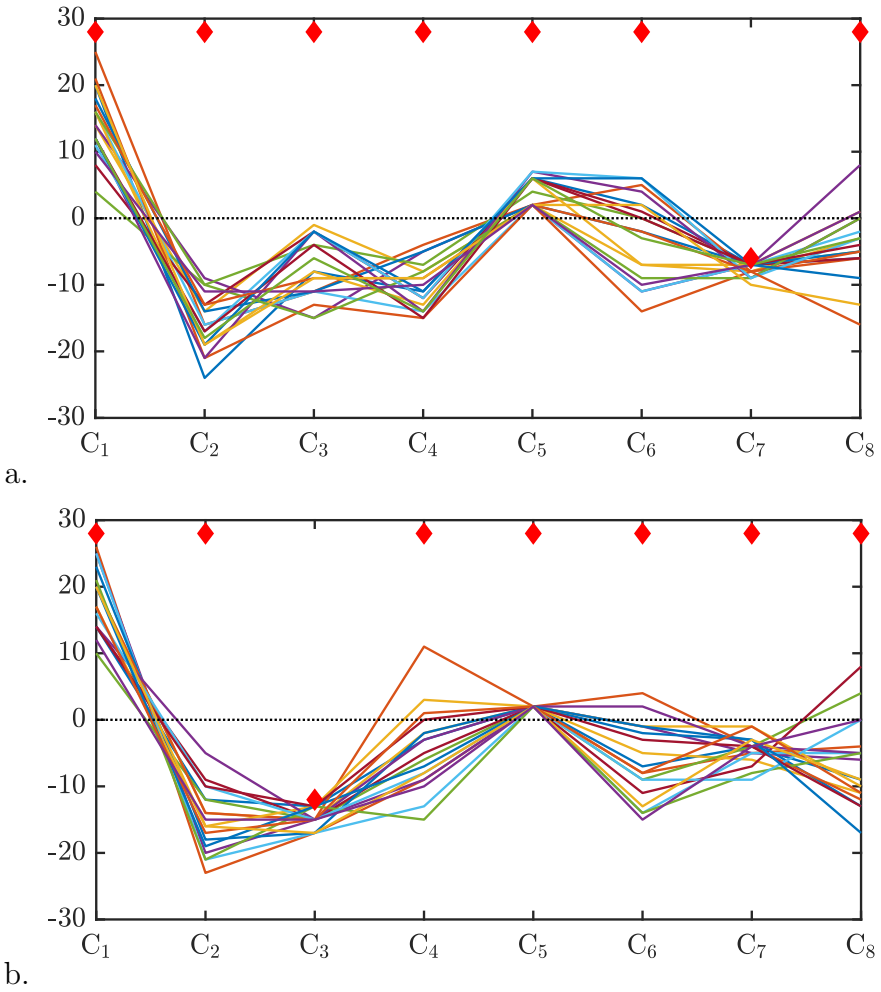


FIGURE 3.10: An analysis of complementary decision criteria in the PWR EC&I case study. These have been explored by setting the decision criteria limits manually. Plots a. and b. show two complementary criteria, usability (C_7) and maintainability (C_3). This is represented by the fact that the best scoring solutions for either criterion score well for the other.

3.4.1 Manually Investigating Tradeoffs and Complementary Decision Criteria

The set of architecture solutions in Figure 3.5 can be manually refined by altering the parallel coordinates limits. This can be useful for identifying tradeoffs between decision criteria e.g. "Performance" vs "Robustness" as shown in Figure 3.9 a. and Figure 3.9 b. It can also show complementary criteria, such as "Usability" and "Maintainability" as in Figure 3.10 a. and Figure 3.10 b. The downside to this manual approach for the PWR case study is that it is very hard to alter the limits to accurately reflect the 7 customer preference weightings, since each preference is correlated to numerous decision criteria.

3.4.2 Automating Parallel Coordinates Limits Using Customer Preferences

The results of applying Algorithm 1 to the PWR EC&I problem are shown in Figure 3.6. Note that the parallel coordinates limits (and hence the chosen solutions) change automatically with changes in customer preferences. This allows the large range of alternatives (as in Figure 3.5) to be narrowed down rapidly to a region of interest (as in Figure 3.6), unlike the time-consuming manual process described in Section 3.4.1. The engineers may want to make some small manual changes to the limits after Algorithm 1 to enforce concerns such as "security must be *at least* as good as the baseline". However, the overall time taken is still greatly reduced.

3.5 Case Study - Novel Turbofan Oil System

The approach presented in Section 3.2 and Section 3.3 has been applied to a high-level architecture design problem for a novel turbofan oil system. The key purpose of the oil system is to provide both lubrication and cooling to the turbofan engine bearings and gearboxes [91, 92]. In this problem, the aim is to investigate architectures which use novel components to allow the oil flow to be controlled independently of the shaft speed. This allows oil temperature and lubrication efficiency to be managed more effectively, increasing the life of components and oil. Chapter 4 describes the oil system in more detail, but an overview is given of the functions/means decomposition in Table 3.6. This function/means decomposition is developed from previous work on a similar case study in [87].

Note that for many of the functions there are only 1 or 2 means. Since the number of potential architectures is the product of the number of means for each function ($1 \times 8 \times 2 \times 7 \times 2 \times 2 \times 2 = 896$) this results in a relatively small number of solutions. There is therefore no need to use MOGA for this case study. Rather an exhaustive search of all possible combinations of means has been used.

For this problem the customer preferences are the same as those in P_1 to P_6 in Table 3.5. The reason P_7 is not used here is that the oil system is an automated part of the turbofan engine and does not require manning in normal operation. The set of decision criteria used in this problem is defined in Table 3.7. The relational matrix R_{pref} and the customer preference weightings P_{weights} are defined as shown in Figure 3.11. Note that R_{pref} uses a scale from -4 to 4 whereby: 0 indicates no relationship; negative or positive values indicate a negative or positive

TABLE 3.6: Functions/means decomposition for the turbofan oil system. The means of the architecture A1 chosen in Section 3.5.1 are highlighted in blue.

Function	Means
Contain oil	Partially pressured system via tanks and pipes
Feed oil during normal operation	Single mechanical fixed displacement (FD) feed pump
	Single electric FD feed pump (AC motor without controller)
	Single electric FD feed pump (DC motor with controller)
	Zonal ganged DC electric FD feed pump
	Zonal feed pump electric & mechanical mix
	Main feed + individual chamber flow control pump
	Individual DC electric FD feed pump
	Mechanical variable displacement feed pump (VDP)
Schedule oil during normal operation	Fixed orifice feed restrictors
	Variable restrictor valves (VRV)
Remove aerated oil from chamber sump	Single ganged mechanical fixed displacement (FD) scavenge pump
	Single ganged electric feed pump (AC motor without controller)
	Single ganged electric FD scavenge pump
	Zonal ganged electric FD scavenge pump
	Zonal ganged FD scavenge pump - electrical/mechanical mix
	Individual electric FD scavenge pump
	Drain chamber via gravity
Feed and scavenge separation	Combined feed and scavenge
	Separate feed and scavenge
Remove debris from oil	Mesh filter
	Electric charge across oil flow
Limit static charge build up	Earth bond every component
	Electrical bonding for rotating parts (e.g. brushes)

relationship respectively; and magnitude indicates the strength of the relationship. This is a different scale to the 0, 3, 9 scale used in [87] and the PWR case study, to demonstrate that the customer-oriented architecture refinement algorithm is compatible with different approaches to scoring or defining relationships between performance measures and decision criteria.

3.5.1 Oil System Architecture Refinement

Performing exhaustive search and removing dominated solutions produces the solution set shown in Figure 3.12 a. One thing that is immediately clear is that the majority of solutions score worse than the baseline for all decision criteria other than 'flow matching' (C_5). This makes sense, since the main purpose of investigating the novel oil system is to improve performance in this decision criterion. It is also obvious that the addition of new components will have a negative effect on performances for other decision criteria such as 'weight' (C_1) or 'technology

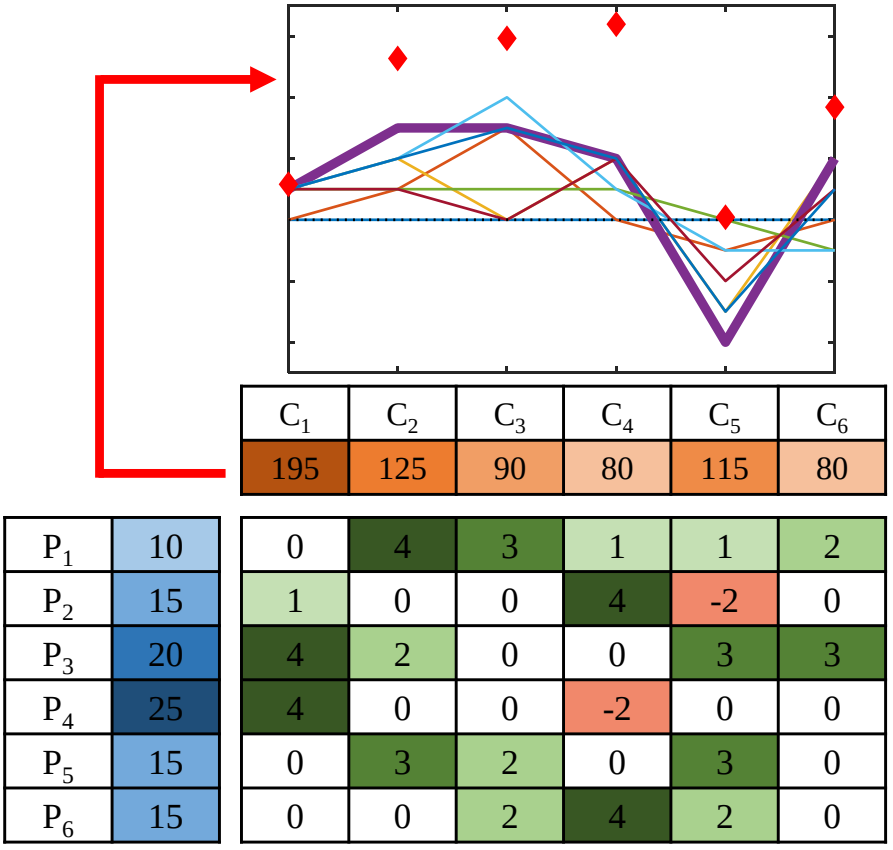


FIGURE 3.11: Customer-oriented architecture refinement for the oil system. Preference weights (blue) are multiplied by the conversion matrix (green/red) to give decision criteria weights (orange). The decision criteria weights are converted to the parallel coordinates limits using Algorithms 1 and 2. The purple solution highlighted has corresponding resilience values in column A1 of Table 3.8

.

TABLE 3.7: Decision criteria for a turbofan oil system.

No.	Decision Criterion	Description
C_1	Weight	The degree to which the weight changes relative to the baseline.
C_2	Reliability	The degree to which system components can be separated and recombined.
C_3	Safety	The degree to which physical and functional elements limit potential for hazards.
C_4	Technology maturity	The degree to which well-established technology is used in the system and manufacturing process.
C_5	Flow matching	The degree to which the system allows oil flow to be controlled independently from the shaft speed.
C_6	Maintenance	The degree to which the system supports effective and efficient maintenance.

maturity' (C_4). A particularly noticeable tradeoff highlighted in Figure 3.12 b. and Figure 3.12 c. is between 'flow matching' (C_5) and 'maintenance' (C_6).

As mentioned in previous sections, the manual preference articulation can be a time-consuming way of narrowing down the solution set. Therefore the customer-oriented architecture refinement algorithm has been implemented to rapidly reduce to the 8 'best' solutions according to the customer preferences. This is shown in Figure 3.11.

TABLE 3.8: Resilience values for the 8 'best' oil system architectures. The chosen architecture and associated resilience values are highlighted in blue.

Customer Preference	**Resilience**															
	A1		**A2**		**A3**		**A4**		**A5**		**A6**		**A7**		**A8**	
	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
Availability	11	10	90	10	90	10	90	10	90	10	9	10	90	10	9	10
NRE cost	22	15	85	8	22	15	22	15	85	8	85	4	22	13	85	4
Lifetime cost	77	20	11	17	43	20	43	20	11	20	5	20	17	20	5	20
Size & mass	75	25	75	25	75	25	75	25	75	25	75	7	75	25	75	7
Sustainability	69	15	6	15	36	15	36	15	6	15	3	15	18	15	3	15
Delivery Risk	19	15	17	10	19	15	19	15	85	10	85	5	19	15	85	5

Chapter 4 covers the low-level architecture topology optimization for the novel oil system case study. This requires a specific high-level architecture as an input, meaning the 8 'best' solutions in Figure 3.11 need further refining to a single chosen solution. Resilience analysis is a useful tool for achieving this. The resilience values for the 8 candidate solutions are shown in Table 3.8. Firstly, architectures A2, A5, A6 and A8 are ruled out because they are too sensitive to small increases in the weight of the 'sustainability' preference. Of the remaining architectures A3, A4 and A7 are highly resilient to changes in the 'availability' weighting, whilst architecture A1 is more resilient to changes in 'lifetime cost' and 'sustainability'. The long-term cost and overall lifetime of a system is one of the key concerns for aerospace customers, and therefore A1 is chosen as the solution to take forward to the next stage of the design. The means chosen for this solution are highlighted in Table 3.6.

3.6 SATS Tool Development

To help make the research in this chapter accessible to system architects who may be unfamiliar with optimization, the System Architecture Trade Study (SATS)

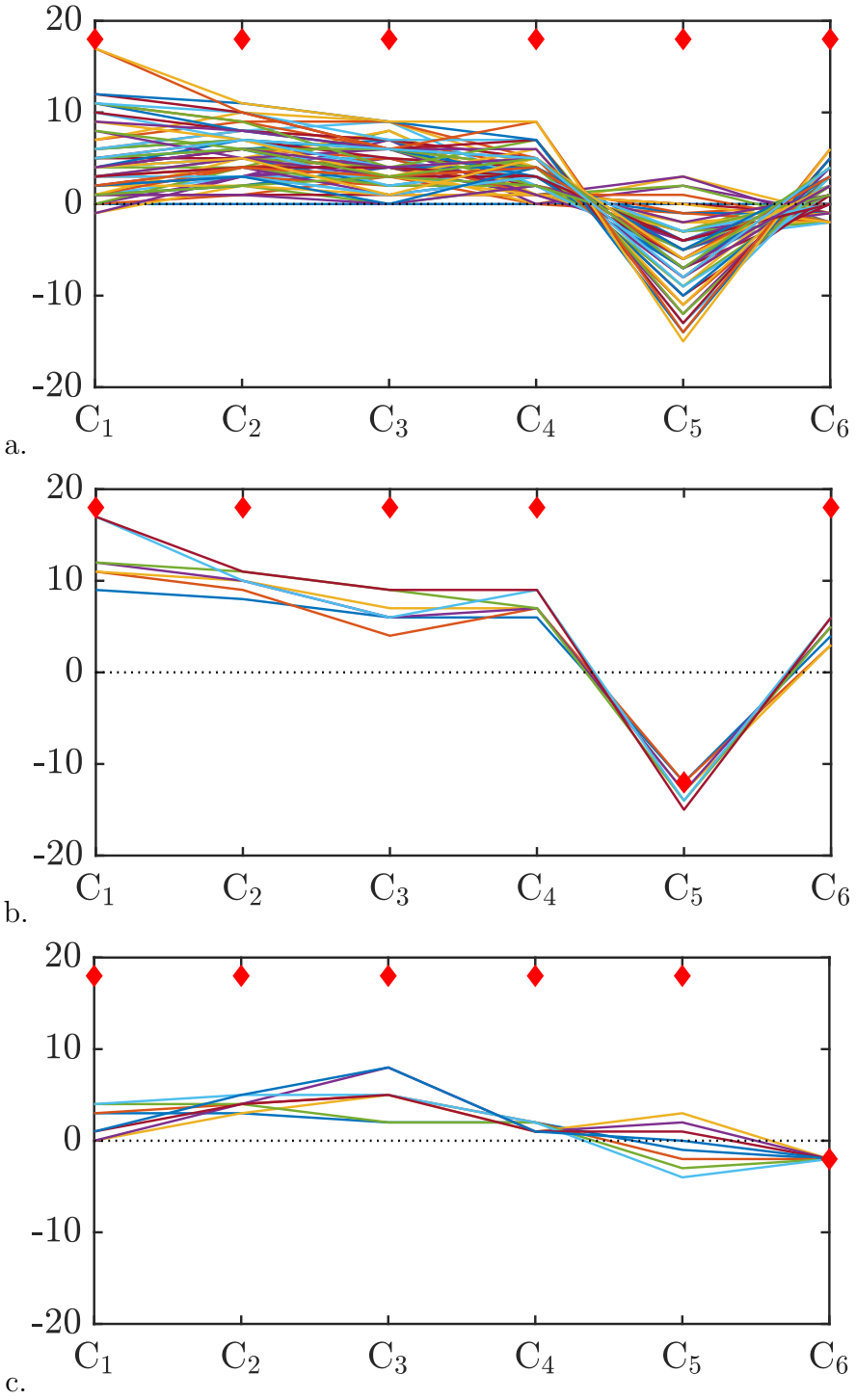


FIGURE 3.12: An analysis of tradeoffs in the oil system case study. These have been explored by setting the decision criteria limits manually. Plot a. shows the unfiltered solutions. Plots b. and c. show a tradeoff between flow matching (C_5) and maintenance (C_6). This tradeoff is represented by the fact that the best scoring solutions for either criterion, score moderately to badly for the other.

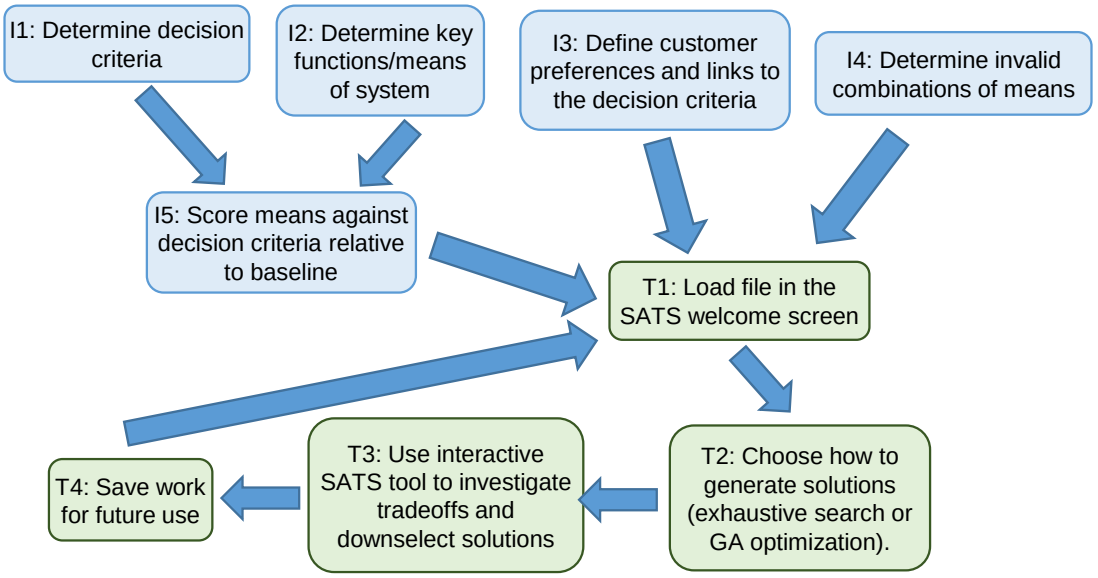


FIGURE 3.13: An overview of the SATS workflow. Blue boxes represent inputs which must be defined in the Excel input file. Green boxes represent tasks which are carried out using the tool.

tool has been developed. This is implemented in MATLAB [93] and can run either in the MATLAB command window or as a Microsoft Windows Executable (.exe) file. An overview of the SATS work flow is shown in Figure 3.13 and a view of the SATS user interface is given in Figure 3.14. The following subsections describe these features in more detail.

3.6.1 SATS Input Files

The input data required to run a SATS analysis is:

- Customer performance measures e.g. "Delivery Risk".
- Decision criteria e.g. "Modularity".
- Customer performance measure to decision criteria relational matrix.
- Primary functions e.g. "Interface with user".
- Means (e.g. "lamp and switch user interface") scored against the decision criteria.
- Incompatible combinations of means e.g. NOT "electric power supply" AND "mechanically-driven motor".

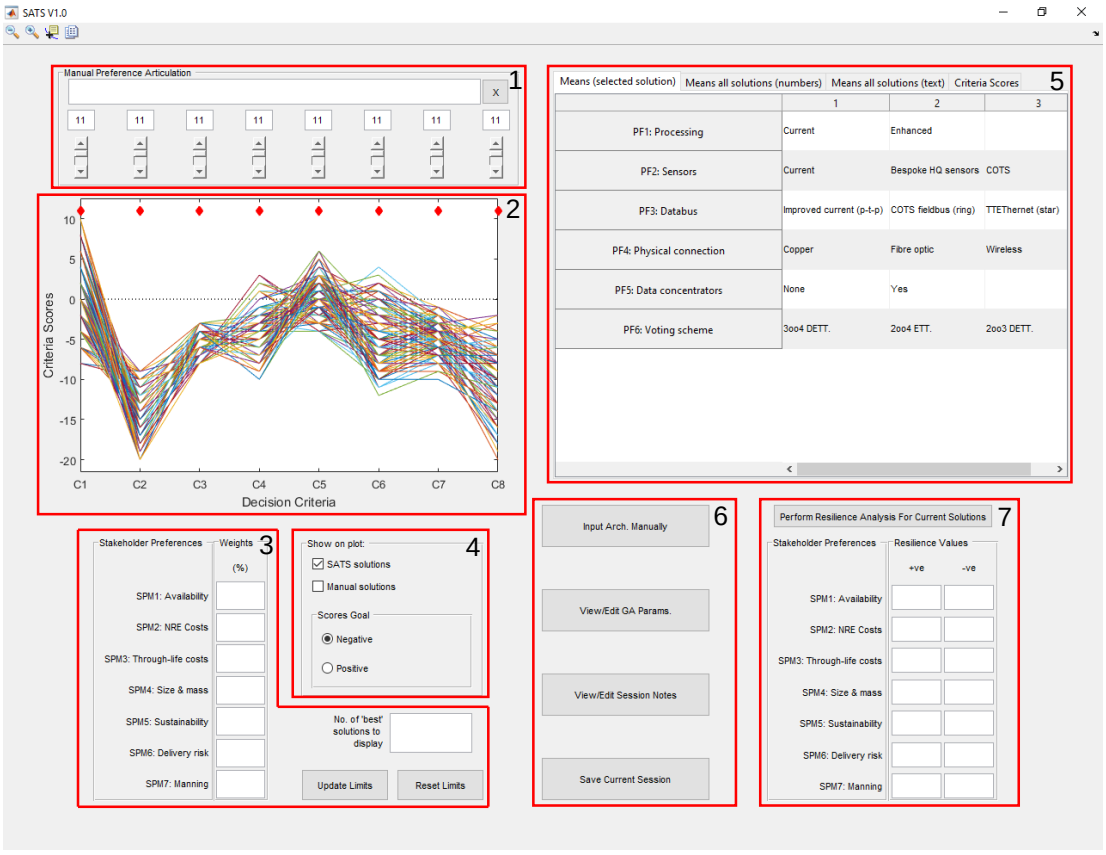


FIGURE 3.14: The SATS GUI main screen with: 1) manual preferance articulation controls; 2) parallel coordinates plot; 3) customer-oriented architecture refinement controls; 4) plot display tools; 5) solution tables group; 6) pushbutton group; 7) resilience analysis and display.

This data is defined and saved in Microsoft Excel spreadsheets before being loaded by the SATS tool. The advantage of using Excel is that it is standard software that will be installed on most PCs. This allows a wide range of engineers from different teams to contribute to the scoring of different means, without having to have the specialist SATS software installed. The system architect can then collate this information to be input into the tool and generate/refine candidate solutions.

3.6.2 Generating Solutions

SATS provides two options for generating candidate architectures. The first option is the multiobjective genetic algorithm (MOGA) as explained in Section 3.2.

The second option is to perform an exhaustive search of all potential combinations of means. Note that with this option dominated solutions (as described in Section 3.2) are removed. The advantage of the exhaustive search over the genetic

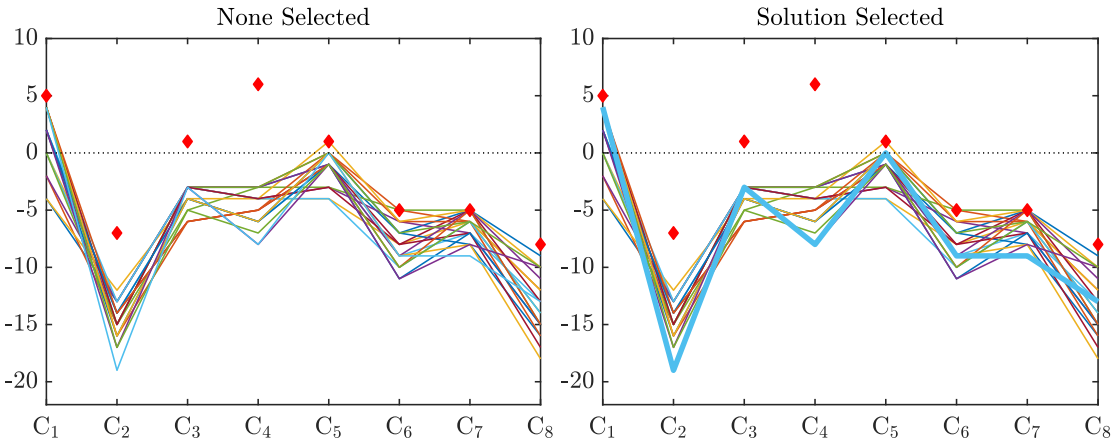


FIGURE 3.15: An example of solution selection in the PC plot. [Left] no selected solutions. [Right] one solution selected and highlighted (via clicking the line on the plot).

algorithm approach is that it is guaranteed to find *all* of the nondominated solutions. However, this is only suitable for relatively small problems since the number of combinations increases exponentially with the number of means. For relatively small problems (less than 10,000 combinations of means) this can be evaluated in reasonable time and is therefore a good option.

3.6.3 Visualising and Selecting Architecture Solutions

The SATS GUI contains two methods for visualising solutions. The first is the parallel coordinates graph which plots each solution as a line linking its decision criteria scores. This is useful for determining how the different architectures perform, but does not give any information about which means have been selected. For this reason, the table group is used to show which means have been chosen. The user can access this feature by clicking a solution, which highlights the line on the parallel coordinates plot (see Figure 3.15). Once a solution is selected, the means chosen are highlighted in the table as shown in Figure 3.16. This allows the user to select various different solutions, with a quick visualisation of what means have changed.

3.6.4 Refining Solutions

SATS offers manual preference articulation via changing of the parallel coordinates limits. For each decision criterion, the limits can be changed incrementally using the sliders or stepped via inputting a new value in the text box.

Means (selected solution) | Means all solutions (numbers) | Means all solutions (text) | Criteria Scores

	1	2	3
PF1: Processing	Current	Enhanced	
PF2: Sensors	Current	Bespoke HQ sensors	COTS
PF3: Databus	Improved current (p-t-p)	COTS fieldbus (ring)	TTEThernet (star)
PF4: Physical connection	Copper	Fibre optic	Wireless
PF5: Data concentrators	None	Yes	
PF6: Voting scheme	3oo4 DETT.	2oo4 ETT.	2oo3 DETT.

FIGURE 3.16: An example of a selected solution represented via highlighting the chosen PF options. The highlighted means are the ones chosen for the current solution. This approach allows a quick visual understanding of a given solution and a method for identifying differences between solutions at the functions/means level rather than just the criteria values level on the PC plot.

The customer-oriented architecture refinement approach described in Section 3.3.2 can also be used to update the parallel coordinates limits. Here the user inputs the customer preference weightings (as a percentage) plus the number of 'best' solutions they want to see. The limits are then scaled according to the algorithm.

There is an option to perform a resilience analysis on a set of solutions. Once these have been calculated, the respective values are displayed in the resilience section when a solution is clicked/highlighted.

3.6.5 Manually Adding Architectures to the Solution Set

Sometimes it may be desirable to manually add an architecture to the solution set. For example, if the preferred choice of the engineers is removed in the architecture synthesis stage due to being dominated by other solutions. While generally it is only desirable to select nondominated solutions, it may still be useful to see how badly this solution performs against the decision criteria. Therefore SATS has a manual input screen whereby users can select an architecture by clicking a means for each function, as shown in Figure 3.17. Manually defined solutions also override any incompatible means or parallel coordinates limits, showing any potential benefits to relaxing these constraints.

3.6.6 Editing MOGA Configuration Parameters

Since SATS is a generic tool which can be used for a variety of architecture problems, the MOGA parameters such as number of iterations, pool size and truncation

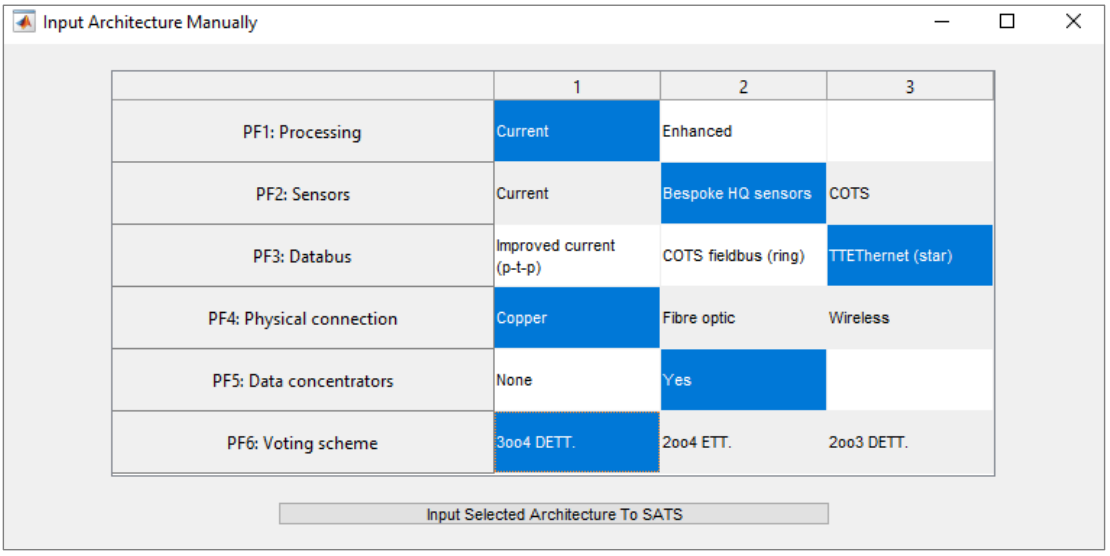


FIGURE 3.17: The manual solution input screen. For each function a means is selected and the solution added to the PC plot. This allows addition of dominated solutions which would not be generated via SATS but may be of interest to the designer (e.g. a legacy solution).

percentage may need to be changed from the defaults. SATS provides a screen to see how the algorithm has converged and rerun the optimization with different parameters if needed (see Figure 3.18).

3.7 Conclusion

This chapter has demonstrated a customer-oriented approach to designing system architectures, building upon previous work on multiobjective architecture optimization. The main benefits come from the automated step of translating customer concerns to engineering characteristic preferences, allowing a set of "best" architectures to be generated rapidly in response to a change in customer preferences. Referring back to Figure 3.1 note that there is feedback to the customer following the refinement step. They are able to directly explore tradeoffs and the effect of their preferences on the solutions generated. This additional information helps with reconsideration and refinement of the preferences until a satisfactory solution or set of solutions is generated. This approach has been demonstrated on architecture case studies for a PWR EC&I system and a turbofan oil system.

Providing this bridge between customer concerns and engineering concerns increases efficiency for both parties. In the customer preference elicitation stage, there is no need to consider engineering characteristics (which they may not fully

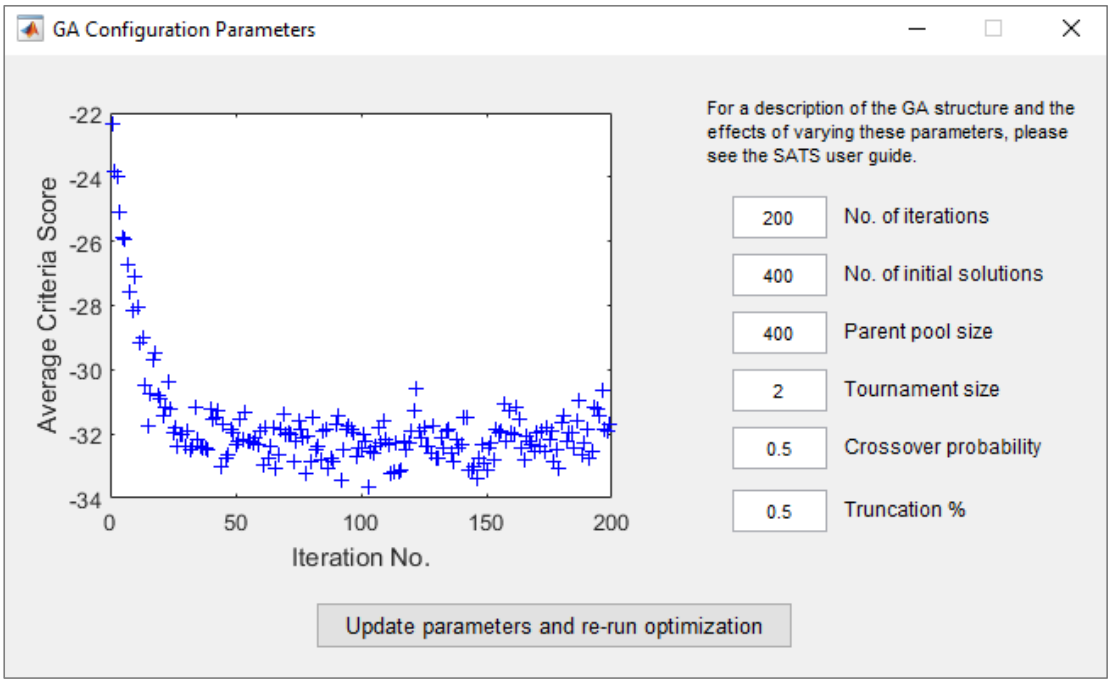


FIGURE 3.18: The MOGA parameter edit screen with options to change the number of iterations, initial solutions, parent poolsize, tournament size, crossover probability and truncation percentage.

understand) focusing instead on the areas of importance to the customer. In the architecture refinement stage, engineers have a clear definition of decision criteria priorities, reducing the need for long, iterative discussions with the customer. One thing to note though, is that this customer/architect bridge relies on having a relational matrix which has to be defined with input from both parties at the start.

In very large-scale system design, long development cycles can often result in changes to customer preferences from earlier to later stages of the project. The resilience analysis presented in this chapter helps engineers to select solutions which are likely to remain a good choice, in the presence of such changes. This is helpful for narrowing down on a final solution, as shown in the turbofan oil system case study in Section 3.5.

One of the main barriers to successful uptake of systems engineering research is the need for engineers to spend time learning new mathematical techniques, such as genetic algorithm optimization. SATS solves this problem for the techniques presented in this chapter by coding the approach into a user-friendly tool. This only requires simple spreadsheet input files to run, opening the possibilities of MOGA to all system architects.

Chapter 4

Cost-Effective, Controllable Topology Optimization

Referring back to Figure 1.1, this chapter focuses on the architecture optimization stage of the multilevel framework. Here the design is synthesised at the lower of the two architecture abstraction levels outlined in Figure 1.2. The high-level architecture decision making (as discussed in Chapter 3) defines the *architecture framework* upon which a physical architecture topology can be designed. Following this approach for the novel oil system, the architecture framework that resulted from the set of customer preferences specified in Section 3.5.1 defines an oil system architecture with variable-restrictor (actively controlled) valves. This chapter focuses on the physical topology design, and presents an optimization-based approach to determining the number and type of components to be used, and the best way in which to connect these in an oil flow network. The chapter is based on research previously published in [94].

Turbofan oil systems are used to provide lubrication and cooling in the engine. There is an increasing interest in oil system architectures which utilise electric pumps and/or valves to give optimized control of flows to individual oil chambers, leading to improved thermal management of oil and lubrication efficiency. The challenges here lie in the tradeoff between increasing controllability and minimising the addition of new components, which adds unwanted production and maintenance costs. This chapter formulates the low-level oil system architecture design as a constrained, multi-objective optimization problem. An architecture is described using a graph with nodes representing components and edges representing interconnections between components. A fixed set of nodes called the architecture template is provided as an input and the edges are optimized for a

multi-criteria objective function. A heuristic method for determining similarities between the different oil chamber flow requirements is presented. This is used in the optimization to evaluate the controllability objective based on the structure of the valve architecture. The methodology provides benefits to system designers by selecting cheaper architectures with fewer valves when the need to control oil chambers separately is small. The effect of manipulating the cost/controllability criteria weightings is investigated to show the impact on the resulting architecture.

4.1 Oil System Overview

The oil system is a vital part of a turbofan engine, providing the dual functions of lubrication and heat removal in the bearings and gearboxes. Components within an oil system architecture consist of: tanks to contain oil; pumps to move oil around the system; filters to remove debris; heat exchangers to remove heat; pipes and flow restrictors to control flow rates; oil chambers with jets directing flow to bearings or gears; deaerators and breathers to vent air to the atmosphere [91, 92]. This is shown graphically in the object process diagram [95] in Figure 4.1.

The pumps and flow restrictors that determine the amount of oil provided to the bearing chambers are typically not actively controlled in Rolls-Royce Trent [91], GE, CFM or Pratt & Whitney engines [92]. The pumps are driven by a fixed gear in the accessory gearbox, providing an output flow proportional to the speed of the high pressure shaft [91, 92]. This lack of oil flow controllability can lead to problems such as exceeding oil temperature constraints, which leads to oil degradation and higher maintenance costs. This is a particular issue during transient manoeuvres. For example, when reducing thrust the shaft speed slows more quickly than the temperature in the oil chambers due to the thermal capacitance of the metals. With a reduced oil flow, but sustained high temperature, the maximum allowable oil temperature can be exceeded [91]. Challenges such as these are likely to be even more evident in the new generation of geared turbofan engines such as the Pratt & Whitney PW1000G and the Rolls-Royce Ultrafan^{TM}. The power gearbox in these engines creates substantial new demand for lubrication and cooling. The 22MW power gearbox on the PW1000G engine generates huge amounts of heat despite being highly efficient (e.g. 1% inefficiency produces 220kW of waste heat to be absorbed by the oil system) [96]. This motivates research into novel oil system architectures. Of particular interest is the ability to utilise electrically driven pumps and variable flow restrictor valves, to provide optimal flow to the individual oil chambers at all stages of the flight cycle. This removes the need

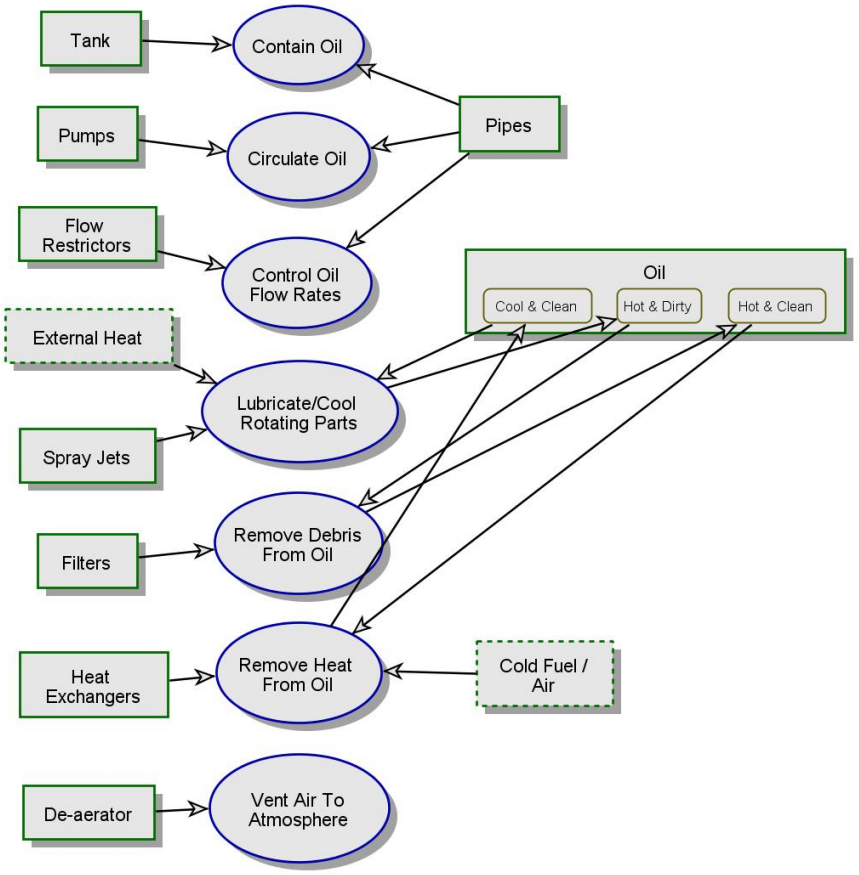


FIGURE 4.1: An object process diagram [95], showing the main objects and processes of the oil system. Blue ellipses represent processes, green rectangles represent objects and brown, rounded rectangles represent states of the oil. Arrows indicate which objects are consumed by which process, and how the processes change the state of the oil.

to constantly oversupply oil during transients and thus reduces parasitic losses on the system efficiency. In addition, better thermal management of oil means that properties such as viscosity can be more closely controlled, improving lubrication system performance and increasing component life.

Choosing the controlled oil system architecture is a multi-objective problem. It is desirable to increase the controllability of oil flows around the system, but at the same time the production cost and weight of the system has to be kept low. This presents a *tradeoff* which must be handled by the system designer in some kind of multi-criteria decision making environment. In addition to this there are safety, reliability and power consumption constraints which cannot be violated. The remainder of this chapter presents a method for handling all of these considerations in a multi-objective optimization framework.

4.2 Problem Formulation

This section formulates the turbofan oil system architecture optimization problem mathematically. The approach is based around the method presented in [33] whereby the architecture is described as a graph of nodes and edges. The resulting architecture is generated from a given template of nodes, with optimization used to determine which nodes to include and the interconnection structure between them. The novelty of this research and the main differences from the approach in [33] are:

1. **Use of a multi-criteria objective function** - this facilitates tradeoffs between the different objectives via selection of weights. The approach in this chapter covers 2 criteria (cost and controllability) but can be easily extended to include others. The need for this arises from the fact that controllability cannot be handled as a constraint as reliability is handled in [33]. This is because there is no "necessary limit" for controllability since none (direct drive) or full (individual valve for each chamber) could both be acceptable depending on the priorities of the customer.

2. **Application to a new real-world problem** - increased controllability of oil flow leads to improved lubrication efficiency meaning reduced friction and decreased fuel consumption. Better management of temperature transients also improves the life of components and oil, leading to maintenance cost savings. However, there is also a strong pressure on engine manufacturers to keep the production costs low by minimising the number of additional components. This chapter presents a new approach for handling these conflicting concerns.

Note that this approach, like [33], is an example of a *connecting* architectural decision making process [28]. However, it is also similar to a *downselecting* process due to the cost/controllability tradeoff leading to architectural solutions which are a subset of the original architecture template [28].

Section 4.2.1 defines the components and architecture template of the actively controlled oil system architecture. Section 4.2.2 discusses a heuristic approach to quantifying the similarities between different oil chamber flow requirements. Finally Section 4.2.3 to Section 4.2.4 present the constraints and objective function for the optimization.

4.2.1 An Actively Controlled Oil System

Whilst geared turbofan designs motivate the architecture optimization techniques presented in this chapter, the proposed methods are being validated against a baseline design of a conventional 3-shaft turbofan engine. The key components which make up a typical turbofan oil system architecture are outlined in [91]. These include: tanks for storing oil; mechanically driven pumps for moving oil around the system; filters for removing debris from the oil; heat exchangers for removing heat from the oil; pipes for directing oil flow around the system; flow restrictors for changing the velocity and pressure of oil flows around the system; oil chambers with jets directing flow to bearings or gears; and dearators/breathers to vent air to the atmosphere. The main differences with an actively controlled architecture are the addition of variable restrictor valves and electrically driven pumps. These modifications allow the oil flow to be controlled independently of the engine shaft speed.

Following the approach taken by [19, 33] the oil system architecture is expressed as a graph with nodes $\{N_1, \cdots, N_n\} \in \mathcal{N}$ where $\mathcal{N}$ is partitioned into subsets $\{T, FP, HE, V, OC, SP\}$ corresponding to the 6 component groups outlined in Table 4.1. The interconnection matrix E is defined as in equation (2.3). The architecture template is given in Figure 4.2. In this template the connections between the tank-pumps and oil chambers-scavenge pumps are fixed, i.e. $e_{T,FP} = 1$ and $e_{OC_i,SP_i} = 1$, $e_{SP_i,FP} = 1, \quad \forall \quad i = \{1, \cdots, 7\}$.

Assumptions

The following assumptions have been made in the formulation of the problem:

- The feed pump is an electrically-driven pump.
- Some components such as filters and the breather are essential in any architecture and therefore these are taken out of this architecture optimization for simplicity. The remaining components which are considered in this problem are given in Table 4.1.
- Oil connections to oil chambers are parallel.
- Component sizes are fixed. Architectures are composed by connecting components according to rules defined in Section 4.2.3. Some components from the template may not be used in a given architecture.

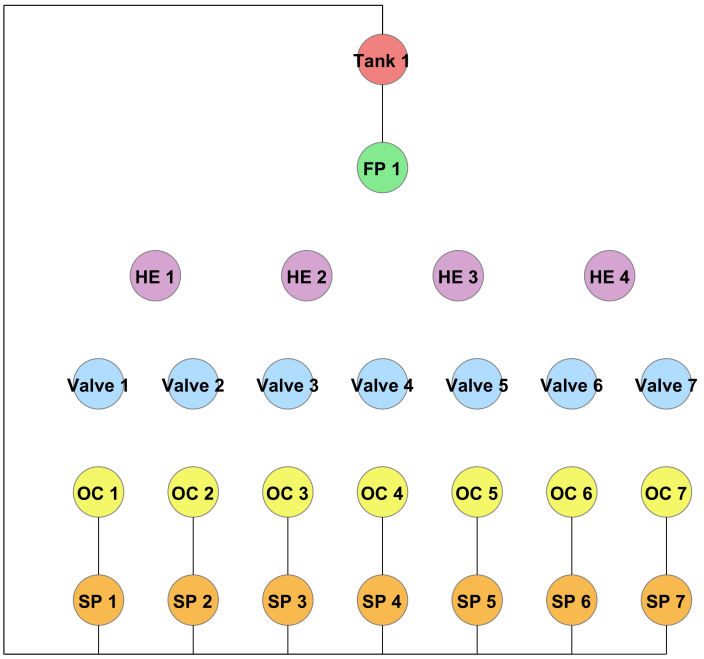


FIGURE 4.2: An architecture template for the actively controlled oil system. Connections between the tank, feed pump (FP), oil chambers (OC) and scavenge pumps (SP) are fixed. The heat exchanger (HE) and valve connections are yet to be determined by the optimization algorithm. Any HE or valve nodes which are not connected to other nodes by the optimization algorithm are not included in the final architecture.

TABLE 4.1: The component groups, functions and maximum numbers of instances.

Component	**Function**	**No.**
Tank	Contain oil	1
Feed pump	Supply oil to the oil chambers	1
Heat exchanger	Remove heat from oil	4
Valve	Control the oil flows to the individual oil chambers	7
Oil chambers	Supply oil to engine bearings or gears	7
Scavenge pumps	Remove oil from oil chamber sumps	7

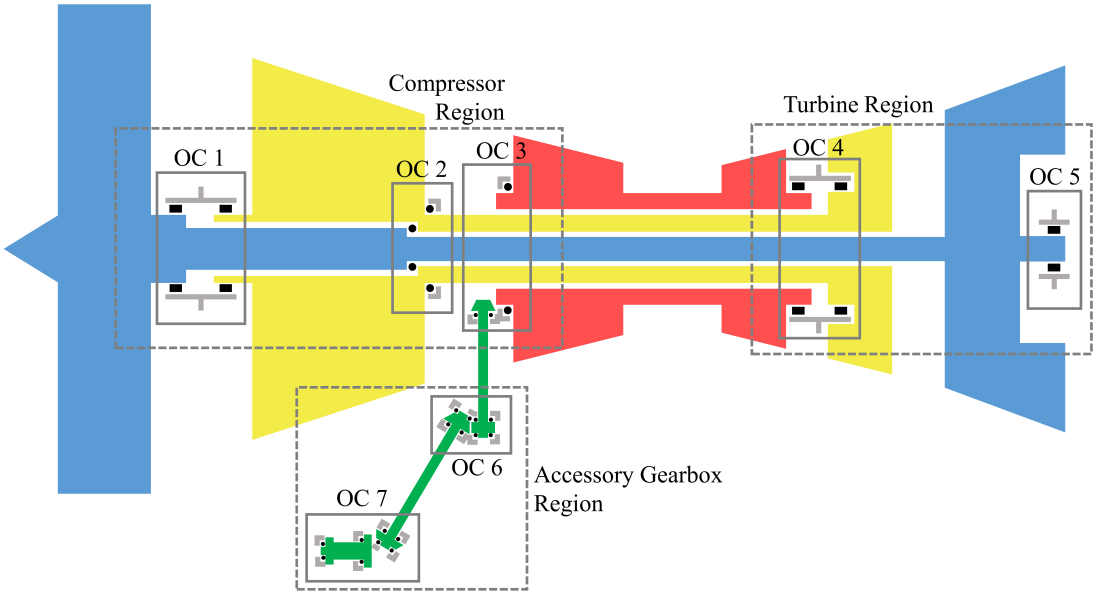


FIGURE 4.3: A schematic of a 3-shaft turbofan engine with low pressure shaft (blue), intermediate pressure shaft (yellow), high pressure shaft (red) and internal/step-aside/accessory gearboxes (green). Roller bearings (black rectangles) and ball bearings (black circles) are contained in the 7 oil chambers OC 1 to OC 7 (grey boxes). The engine regions (compressor, turbine, accessory gearbox) are highlighted by the dashed, grey boxes.

4.2.2 Quantifying Similarities Between Oil Chamber Flow Requirements

The location of the seven oil chambers is based on a typical 3-shaft turbofan engine as outlined in [91] and shown in Figure 4.3. As mentioned previously, the motivation for using an oil system architecture with valves is to better control the flow of oil to the individual oil chambers. There is also a need to keep the production costs and complexity of the architecture low. Therefore it is desirable to control multiple oil chambers with a single valve when their flow requirements are similar throughout the flight cycle.

The similarities between the oil flow requirements are contained in a matrix $C_{fr} \in S\mathbb{R}^{m\times m}$, where $S\mathbb{R}^{m\times m}$ is the set of real valued symmetric matrices of size $m \times m$ and m is the number of oil chambers.

$$C_{fr} \coloneqq \begin{bmatrix} 0 & c_{1,2} & c_{1,3} & \cdots & c_{1,m} \\ c_{2,1} & 0 & c_{2,3} & \cdots & c_{2,m} \\ c_{3,1} & c_{3,2} & 0 & \cdots & c_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & c_{m,3} & \cdots & 0 \end{bmatrix}. \tag{4.1}$$

The elements $c_{i,j}$ are a measure of the independence of oil chambers i and j with a larger magnitude indicating a greater need to control their oil flows separately. The values of $c_{i,j}$ can be assigned through a variety of methods. For example, if there was a set of optimized flow conditions for each oil chamber over the flight cycle, these could be analysed to determine the statistical correlation between chambers. These correlations would be used to populate the matrix C_{fr}.

Algorithm 3 Defining $c_{i,j}$ values.

```
1:  for all c_{i,j} do
2:    if i == j then
3:      Set c_{i,j} = 0
4:    else
5:      Set c_{i,j} = 1
6:      if i and j are in different parts of the engine (compressor/turbine/gear-
        boxes) then
7:        c_{i,j} = c_{i,j} + 1
8:      end if
9:      for each shaft (HP/IP/LP) in i and j do
10:       if shaft is unique to either chamber i or chamber j then
11:         c_{i,j} = c_{i,j} + 1
12:       end if
13:     end for
14:   end if
15: end for
```

In the absence of these optimized flow conditions, a more heuristic approach has to be taken, as outlined in Algorithm 3. Lines 2 to 5 correspond to the fact that there will be at least some difference between oil flow requirements in different chambers. Lines 6 to 8 come from the fact that oil chambers are more likely to have similar flow requirements to other chambers in the same engine region, due to coupled temperature transients, pressures and flow rates. Lines 9 to 13 correspond to the fact that any oil chamber bearings or gears which do not have a physical connection can rotate at independent speeds and hence their optimum oil flows may vary more greatly.

Using Algorithm 3 for the 3-shaft civil turbofan example in Figure 4.3, C_{fr} evaluates to:

$$C_{fr} = \begin{bmatrix} 0 & 1 & 4 & 4 & 3 & 5 & 5 \\ 1 & 0 & 4 & 4 & 3 & 5 & 5 \\ 4 & 4 & 0 & 3 & 4 & 2 & 2 \\ 4 & 4 & 3 & 0 & 4 & 3 & 3 \\ 3 & 3 & 4 & 4 & 0 & 4 & 4 \\ 5 & 5 & 2 & 3 & 4 & 0 & 1 \\ 5 & 5 & 2 & 3 & 4 & 1 & 0 \end{bmatrix}. \tag{4.2}$$

Note that Algorithm 3 is a suggested set of rules for determining similarities, but some designers may be interested in other factors. For example, oil chambers with different bearing types (ball or roller) may have less similarity between flow requirements. Additionally, a designer may want to single out a specific oil chamber (OC i) to have an independent valve based on some experience or knowledge about particular oil flow challenges in that chamber. This could be achieved by adding a large number (for example, 100) to the off-diagonal elements in the i^{th} row and column.

The multi-criteria optimization approach presented in this chapter will work regardless of the method in which C_{fr} is populated. However, since the similarities matrix is used to generate the controllability objective scores, a sensible choice of C_{fr} values will be required to produce sensible architectures.

4.2.3 Defining Architecture Constraints

Constraints are introduced to the architecture optimization to ensure system requirements are met. These requirements may define either required/forbidden interconnections or some sort of energy balance that must be satisfied.

Interconnection Constraints

There are a variety of interconnection constraints which can be expressed formally as:

$$\sum_{i=1}^{|G1|} e_{G1_i,G2_j} \diamond c \quad \forall j \in \{1, \cdots, |G2|\}, c \in \mathbb{N}, \tag{4.3}$$

where $G1, G2 \in \{T, FP, HE, V, OC, SP\}$ are component partitions and $\diamond \in \{<, \leq, >, \geq, =\}$. For example, the requirement "each oil chamber shall be connected to exactly one valve" is defined as:

$$\sum_{i=1}^{|V|} e_{V_i, OC_j} = 1 \quad \forall j \in \{1, \cdots, |OC|\}. \tag{4.4}$$

For some component groups there may be constraints on connections with upstream components, depending on the connections made downstream. These can be expressed formally as:

$$\left(\sum_{i=1}^{|G1|} e_{G1_i, G2_j} \diamond c \right) \implies \left(\sum_{k=1}^{|G3|} e_{G3_k, G2_j} \diamond c \right), \quad \forall j \in \{1, \cdots, |G2|\}. \tag{4.5}$$

where $A \implies B$ indicates A *implies* B. For example, the constraint "if a valve is connected to one or more oil chambers, it shall also be connected to exactly one heat exchanger" is given by:

$$\left(\sum_{i=1}^{|OC|} e_{OC_i, V_j} > 0 \right) \implies \left(\sum_{k=1}^{|HE|} e_{HE_k, V_j} = 1 \right), \quad \forall j \in \{1, \cdots, |V|\}. \tag{4.6}$$

Likewise "if a heat exchanger is connected to a valve, it must also be connected to the feed pump" is expressed as:

$$\left(\sum_{i=1}^{|V|} e_{V_i, HE_j} > 0 \right) \implies \left(e_{FP, HE_j} = 1 \right), \quad \forall j \in \{1, \cdots, |HE|\}. \tag{4.7}$$

These interconnection requirements are contained in the constraint set R_I.

Energy Balance Constraints

In the component library there are 4 different off-the-shelf heat exchangers each with different maximum flow rates (in arbitrary units) contained in vector flow_{HE}.

$$\text{flow}_{HE} = \begin{bmatrix} 300 & 200 & 200 & 500 \end{bmatrix}. \tag{4.8}$$

The oil chambers have maximum flow demands given by:

$$\text{flow}_{OC} = \begin{bmatrix} 30 & 40 & 50 & 100 & 80 & 20 & 20 \end{bmatrix}. \tag{4.9}$$

One or more heat exchangers can be used in the architecture, so long as they meet the downstream maximum flow demand of the oil chambers. This is termed R_B, an energy balance constraint as in [33], and is defined formally as:

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|OC|}(e_{HE_k,V_i})(e_{V_i,OC_j})(\text{flow}_{OC_j}) \leq \text{flow}_{HE_k}, \quad \forall k = \{1, \cdots, |HE|\}. \tag{4.10}$$

In this research, the interconnection and energy requirements have been manually converted from natural language to formal, programmable constraints. There is potential for a tool which allows requirements to be defined using a limited set of natural language expressions which are then automatically coded to formal requirements for the optimization problem. The challenges here revolve around getting a set of expressions which is large enough to capture any requirement that the user may wish to specify.

Safety Constraints

A key constraint for a controlled oil system is safety. If any valves become blocked leading to an interrupt in oil flow there could be serious consequences. It is assumed here that appropriate safety measures are incorporated into the physical design of the valves. For example, they could be sized to ensure that the minimum oil flow rate is always maintained, with just the upper range of flow controlled to optimize flow.

Since these safety concerns relate to the design of the valves themselves rather than the system architecture, they are not incorporated into the optimization algorithm for the oil system. In other applications, such as those whereby redundant components need to be used to achieve a certain level of reliability, safety constraints will need to be programmed into the architecture optimization as presented in [19, 33].

4.2.4 Objective Function

Architectural Drivers / Decision Criteria

Whatever systems architecting approach is taken, a key task is to identify the *architectural drivers*. A 5-step method for identifying the architectural drivers by analysing and refining stakeholder requirements is presented in [62]. These drivers are the motivating features of an architecture which correspond to either the constraints or decision criteria in more formal optimization-based design. For example, in [33] the architectural drivers are cost and complexity (decision criteria in the objective function) and reliability (one of the constraints).

In the case of the oil system architecture problem presented in Section 4.2 to Section 4.3, the architectural drivers/decision criteria are:

1. Increasing controllability of oil flow to the individual oil chambers.
2. Minimising system architecture production cost.

These are both handled in the objective function described in Section 4.2.4. Minimising cost is common in almost all applications. The meaning of *increasing controllability* is less clear since the term controllability has many definitions. Some discussion of this is given in [97] which notes that often the term controllability is used to mean state-controllability (the ability to move a system from an initial state to an arbitrary point in the state space in finite time). If we consider the exit oil temperature at each of the oil chambers as states in our system, then the ability to arbitrarily move to any point in the state space requires uniquely controllable flow to each chamber. This would require a unique valve for each oil chamber. In this chapter the term controllability relates more closely to (input-output) controllability which is linked to *performance* [97]. In the case of the oil system, good performance can be achieved when the flow to oil chambers can be controlled to manage oil temperature peaks during transients, without the need to oversupply oil during steady-state conditions. Therefore if two oil chambers share similar oil flow requirements, it may be possible to get good input-output controllability (good performance) without having full state-controllability. This would allow the production cost of the oil system to be reduced by using fewer valves.

As previously noted, there are two decision criteria in the objective function: cost and controllability.

$$f := w_{\text{cost}} f_{\text{cost}} + w_{\text{control}} f_{\text{control}}. \tag{4.11}$$

Since these are two opposing objectives, the tradeoff between them is handled through the introduction of weights w_{cost} and w_{control}. Section 4.3.2 investigates the effect of varying the weights on the resulting architecture. This section shows how the individual objective functions f_{cost} and f_{control} are constructed.

Cost

This is dependent on the production cost of the valves and heat exchangers which are used in the architecture and their interconnections:

$$f_{\text{cost}} \coloneqq \sum_{i=1}^{|V|} \delta_{V_i} \left(C_{V_{\text{base}}} + \sum_{j=1}^{|OC|} e_{V_i,OC_j} \text{flow}_{OC_j} C_{V_{\text{add.}}} \right) + \sum_{i=1}^{|HE|} \delta_{HE_i} C_{HE_i}, \tag{4.12}$$

where,

$$\delta_i \coloneqq \begin{cases} 1 & \text{if } \sum_{j=1}^{|\mathcal{N}|} e_{i,j} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

The production costs of the four potential heat exchangers (in some monetary unit) are contained in vector C_{HE}. This represents the fact that different off-the-shelf components will utilise different technologies and hence cost different amounts. There is also a rough correspondence between these costs and the flow capacities contained in (4.8).

$$C_{HE} = \begin{bmatrix} 4000 & 3000 & 3000 & 10000 \end{bmatrix}. \tag{4.14}$$

In this chapter the valves are all assumed to be equal cost ($C_{V_{\text{base}}} = 5000$) which represents the basic cost of manufacturing a valve regardless of size. It is also assumed that there is an additional cost added for each oil chamber that is connected to a valve depending on the size of the maximum flow requirements to that chamber ($\text{flow}_{OC_i} \cdot C_{V_{\text{add.}}}$) where $C_{V_{\text{add.}}} = 100$. This represents the additional material cost in larger valves with greater flow capacity. Using this cost model, the cost of the valve part of the architecture is 69,000 units for a 7 valve system, and 39,000 units for a 1 valve system. This confirms that the more valves used, the higher the cost.

Controllability

The effect of the combination of operations in equation (4.15) is to extract and sum the relevant values from the flow interconnections matrix C_{fr} based on which oil chambers are controlled via the same valves.

$$f_{\text{control}} \coloneqq \sum_{i=1}^{|V|} e_{v_i,OC} \left(\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \otimes e_{v_i,OC} \right) \bullet C_{fr}, \tag{4.15}$$

where $\otimes$ denotes the Kronecker product and $\bullet$ denotes the Hadamard product.

Consider an example with 4 oil chambers and 4 potential valves given in (4.16). In this example the first three oil chambers are controlled by one valve and the last oil chamber by another separate valve as indicated in $e_{V,OC}$.

$$e_{V,OC} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad C_{fr} = \begin{bmatrix} 0 & \mathbf{1} & \mathbf{4} & 4 \\ \mathbf{1} & 0 & \mathbf{4} & 3 \\ \mathbf{4} & \mathbf{4} & 0 & 2 \\ 4 & 3 & 2 & \mathbf{0} \end{bmatrix}. \tag{4.16}$$

Evaluating f_{control} using equation (4.15) gives a sum of the bold values in C_{fr}:

$$f_{\text{control}} = (1 + 4 + 1 + 4 + 4 + 4 + 0) = 18. \tag{4.17}$$

Note that according to this equation, a higher value indicates *worse* controllability, so the objective is to minimise f_{control}.

Normalization

As shown in Section 4.2.4, the criteria do not share the same units. Therefore they are normalized by dividing each criterion score by the maximum possible value for that criterion:

$$\hat{f}_{\text{cost}} = \frac{f_{\text{cost}}}{f_{\text{cost}_{\text{max}}}}, \quad \hat{f}_{\text{control}} = \frac{f_{\text{control}}}{f_{\text{control}_{\text{max}}}}. \tag{4.18}$$

The maximum value for the cost objective was found to be $f_{\text{cost}_{\text{max}}} = 79,000$, when using all 7 valves and the most expensive heat exchanger (HE 4). The maximum value for the controllability objective was found to be $f_{\text{control}_{\text{max}}} = 146$,

when controlling all 7 oil chambers with a single valve. These normalized criteria are then used in the overall objective function as defined in equation (4.19).

4.3 Results

After defining the constraints R and objective function f the optimization problem is expressed as:

$$\begin{aligned} \min_{E \in \mathbb{B}^{n \times n}} \quad & f \coloneqq w_{\text{cost}} \hat{f}_{\text{cost}} + w_{\text{control}} \hat{f}_{\text{control}}, \\ \text{subject to} \quad & R \coloneqq \{R_I, R_B\}. \end{aligned} \tag{4.19}$$

Since the matrix variable E only contains values in the Boolean set $\mathbb{B} \coloneqq \{1, 0\}$ this is a specific type of *integer program*. This has been solved using the MATLAB toolbox YALMIP [51] implementing a global branch-and-bound algorithm with upper solver FMINCON [93] and lower solver GUROBI [98].

4.3.1 Generated Architectures

The resulting architecture depends on the selection of weights w_{cost} and w_{control} as discussed in Section 4.3.2. Setting a strong preference for reducing cost results in a 1-valve architecture as shown in Figure 4.4. At the other extreme, when controllability is very highly weighted the optimization generates a 7-valve system as shown in Figure 4.5.

Selecting between these two extremes produces architectures with 2, 3, 4 or 5 valves (e.g. the 3-valve example in Figure 4.6). There are two things to note here. Firstly the 3-valve architecture contains two heat exchangers. Whilst the fourth heat exchanger has a flow capacity large enough to supply all of the oil chambers it is also more expensive (see (4.14)). Therefore the algorithm has chosen to use two cheaper heat exchangers (2 and 3). This was the case for the entire range of criteria weightings investigated in Section 4.3.2, apart from the architectures with a single valve. There is a requirement that "if a valve is connected to an oil chamber it shall be connected to *exactly* one heat exchanger". Therefore when there is a single valve controlling flow to all oil chambers only one heat exchanger can be used and heat exchanger 4 is the only one with sufficient capacity. A cheaper solution could be gained by allowing connection of multiple heat exchangers to a single valve in parallel. This has not been implemented because the physics of mixing multiple

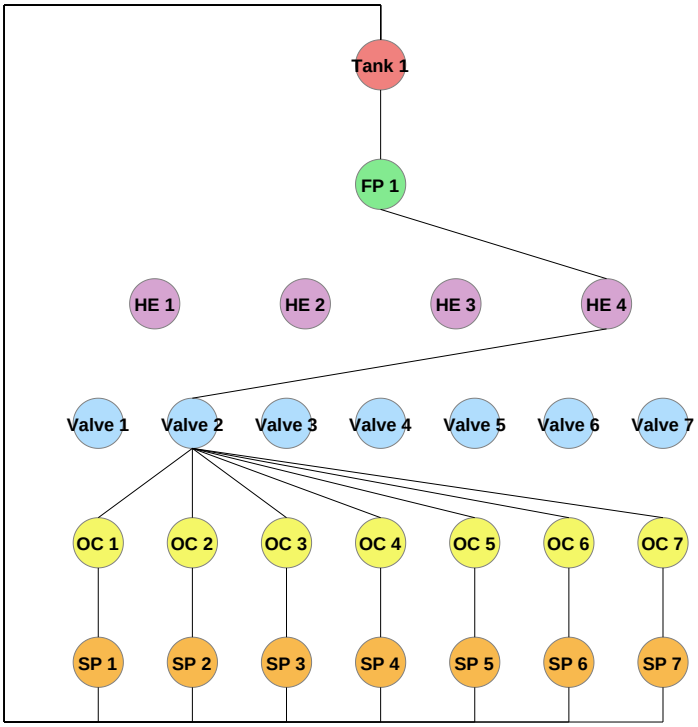


FIGURE 4.4: An example oil system architecture with 1 valve and 1 heat exchanger. The tank, fuel pump, oil chambers, scavenge pumps and their connections are fixed by the architecture template.

oil flows at the inlet to the valves would make it hard to quantify the state of the oil (e.g. temperature or viscosity) which is required for effective control. This constraint only has a small effect on the overall size of the search space, since architectures with two or more valves are not constrained to only using 1 heat exchanger (as shown in Figure 4.6).

The second point to note is that the architecture in Figure 4.6 is a sensible coupling of the oil chambers for a 3-valve system. Referring back to Figure 4.3 it is clear that the two LP/IP compressor chambers are controlled by the first valve, the two gearbox chambers and the gearbox/HP compressor chamber are controlled by the second valve and the two turbine chambers are controlled by the third valve.

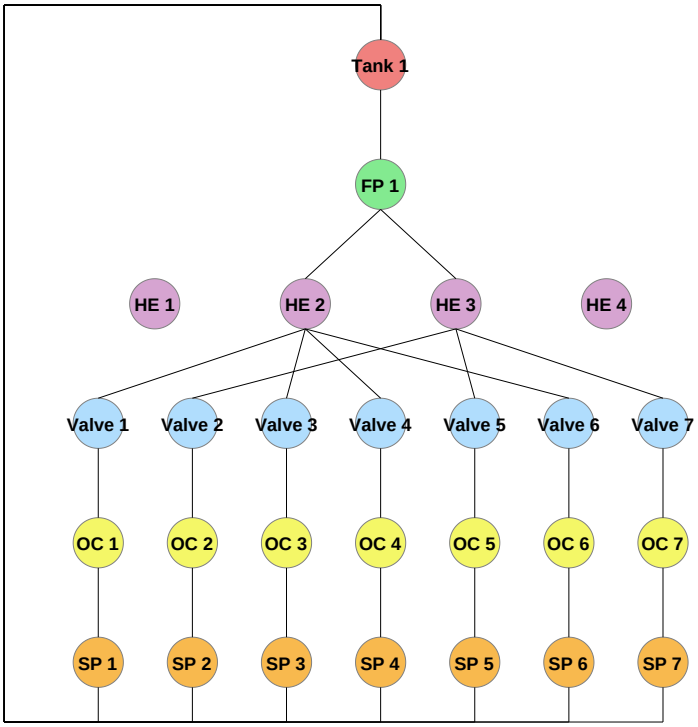


FIGURE 4.5: An example oil system architecture with 7 valves and 2 heat exchangers. The tank, fuel pump, oil chambers, scavenge pumps and their connections are fixed by the architecture template.

4.3.2 Investigating the Trade-offs Between Cost and Controllability

The solutions presented in Section 4.3.1 are examples of optimal solutions on the Pareto front, as shown in Figure 4.7. Note that there is a clear tradeoff between these criteria: improvement can only be achieved for controllability by increasing the cost and vice versa. All of the architectures shown in Figure 4.7 have some form of active control (1 valve or more), but none of the oil system architectures reviewed in the literature have controllable oil flows [91, 92]. This means they are cheaper to produce but have no controllability. Hence they would appear beyond the top-left corner of this tradeoff plot.

The solution from the Pareto front that is generated by the optimization will vary depending on the values of the weights w_{cost} and w_{control} in the objective function (4.19). Some discussion of how to choose weights is given in [81]. In particular it presents a method for determining overall criteria weights from a set

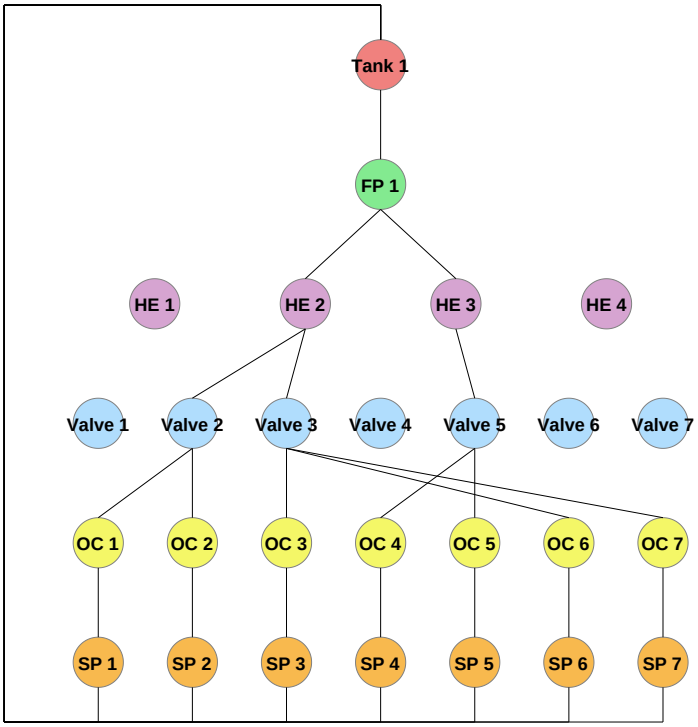


FIGURE 4.6: An example oil system architecture with 3 valves and 2 heat exchangers. The tank, fuel pump, oil chambers, scavenge pumps and their connections are fixed by the architecture template.

of weights given by multiple stakeholders. For this research, there is no access to multiple stakeholders to implement such a method. However, since the number of decision criteria is small it is possible to investigate the entire range of weight ratios $w_{\text{cost}}/w_{\text{control}}$ that produce architectures with 1 to 7 valves. This tradespace is represented in Figure 4.8. Since there are only 7 discrete possibilities for the number of valves in the architecture, the plot in Figure 4.8 shows a stepped line. One thing to note is the fact that there is a jump from a 7-valve architecture to a 5-valve architecture. The reason for this is clear when referring back to the matrix C_{f_r}.

$$C_{fr} = \begin{bmatrix} 0 & \mathbf{1} & 4 & 4 & 3 & 5 & 5 \\ \mathbf{1} & 0 & 4 & 4 & 3 & 5 & 5 \\ 4 & 4 & 0 & 3 & 4 & 2 & 2 \\ 4 & 4 & 3 & 0 & 4 & 3 & 3 \\ 3 & 3 & 4 & 4 & 0 & 4 & 4 \\ 5 & 5 & 2 & 3 & 4 & 0 & \mathbf{1} \\ 5 & 5 & 2 & 3 & 4 & \mathbf{1} & 0 \end{bmatrix}. \tag{4.20}$$

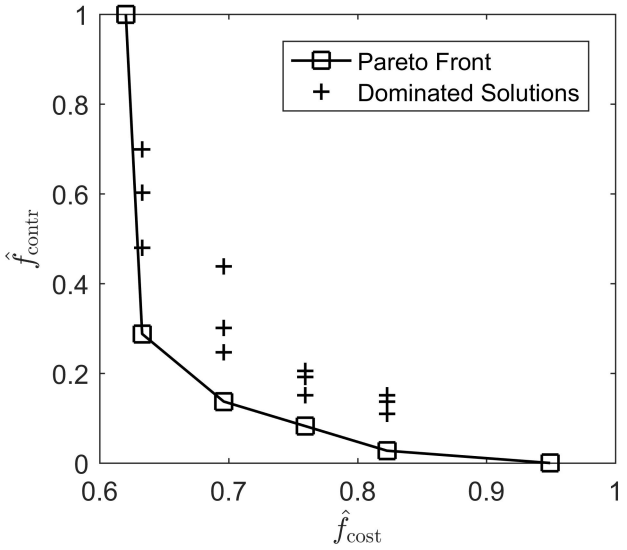


FIGURE 4.7: Architecture solutions generated via the optimization approach with the goal of minimising both criteria (squares). Moving from left to right on the x-axis, these represent solutions with 1, 2, 3, 4, 5 and 7 valves. Note that the optimization algorithm finds nondominated solutions on the Pareto front, meaning that improvement in one criterion cannot be achieved without producing a worse score for the other criterion. A few solutions have been generated randomly, without taking into account the objective function, to show the principle of dominated solutions (crosses).

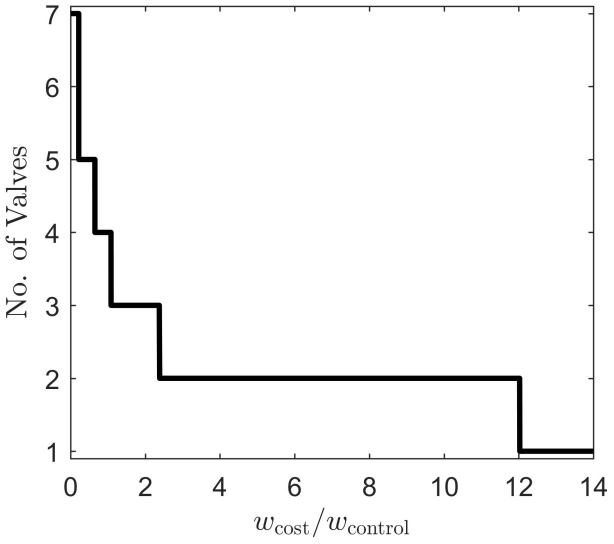


FIGURE 4.8: The effect of varying the cost to controllability weight ratio on the number of valves in the resulting architecture.

Note the 1s highlighted in bold in the top-left and bottom-right corners. The similarities between the flow requirements of OC 1 and OC 2 are identical to the similarities between the flow requirements of OC 6 and OC 7. Therefore as soon as the weight ratio $w_{\text{cost}}/w_{\text{control}}$ is great enough that it is worth controlling OC 1 and OC 2 with a single valve, the same is true for OC 6 and OC 7. This explains why the optimization never produces a 6-valve architecture.

Another observation is the fact that the stepped line shows a roughly exponential decrease. The reason for this can be explained through considering the effect of removing valves from the system on the values of the objective functions f_{cost} and f_{control}. Moving from a 7 to 6 valve architecture there is a decrease in the f_{cost} due to the removal of 1 valve. However, as two oil chambers become controlled by a single valve they *both* suffer a reduction in controllability. Similarly when moving from the 6-valve architecture to a 5-valve architecture f_{cost} continues to decrease in a linear fashion, whilst the controllability of all three oil chambers goes down. Since f_{control} decreases more rapidly than the reduction in f_{cost}, the weight ratio $w_{\text{cost}}/w_{\text{control}}$ has to increase exponentially to produce architectures with the smallest number of valves.

A sensitivity analysis has also been performed on the parameters of the cost model in equation (4.12). The net effect of increasing either $C_{V_{\text{base}}}$ or $C_{V_{\text{add}}}$ is that a smaller $w_{\text{cost}}/w_{\text{control}}$ ratio is needed to generate an architecture with the same number of valves. However, the pattern of the exponential stepped decrease shown in Figure 4.8 remains the same.

For simplicity this research has only considered the two decision criteria of cost and controllability. This allows a 2-dimensional plot to be used to visualise the tradespace. However, a more thorough optimization could consider other criteria such as weight, safety or reliability. In this case, a multi-criteria visualisation tool such as parallel coordinates [58] would be needed to investigate the effects of varying the criteria weightings.

4.4 Conclusion

This chapter has presented a multi-criteria optimization approach to the design of high-level turbofan oil system architectures. A key development is the ability to analyse the impact of using common actuators for multiple oil chambers on the controllability and cost of the system. This has been achieved through use of

a flow requirement similarities matrix which is used to identify which oil chambers should or should not be controlled together. In this research the matrix has been populated through use of a heuristic algorithm but the optimization framework would remain valid if the matrix was populated using other methods. The approach has produced sensible results and has demonstrated the ability for trade-offs to be investigated through variation of weights in the objective function. The optimization yields more suitable architectures than other computational methods investigated, such as a random coupling of oil chambers to valves. The architectures generated also match with the best architectures determined subjectively by experienced engineers. This supports the method used and provides an additional objective evidence-base upon which to make decisions.

The techniques developed have been validated on a baseline 3-shaft turbofan oil system design. This motivates the use of the approach for future geared turbofan oil system designs. Other potential case studies are alternative controlled flow networks such as smart building water-heating control or smart traffic systems. The graph-based approach to modeling system architectures and optimizing connections between nodes also has wider applicability to any system with a set of interconnected components.

The optimization-based approach presented in this chapter ensures that designs are verified and guaranteed to satisfy the formal specification. However, it is worth noting that there is a human element to formulating the problem in the choice of constraints and objective function. This means the resulting architecture will be sensitive to the problem formulation choices made by the system designers. Therefore a procedure for validation of the specification would also be required when using these techniques in practice.

Chapter 5

Simulation-Based Control Synthesis With Formalized Requirements

Referring back to the multi-level framework introduced in Chapter 1, there are multiple distinct *platforms* as shown in Figure 1.1. Below the top-level requirements, the first design platform is the architecture optimization platform. This is further split in Figure 1.2 to high-level (Chapter 3) and low-level (Chapter 4) architecture design. Following the architecture design, the next platform is the control synthesis stage, which is explored in this chapter.

The approach taken here is to use simulation-based optimization of control parameters, to maximise satisfaction of a set of formal requirements. A key contribution of this research lies in extending the principles of quantitative satisfaction of signal temporal logic (STL) formulae to a multiobjective formulation called multiSTL. When multiple STL sub-formulae are joined via conjunction to make a system-level STL formula, the quantitative semantics of STL defined in [69] state that the system-level margin of satisfaction is the minimum of the sub-formula margins. This reduces a rich set of information into a single measure which may miss some of the advantages/disadvantages of different solutions. In multiSTL each sub-formula margin is dislayed on a parallel coordinates plot, which allows tradeoffs between different sub-formulae to be analysed. This can also be used to highlight where relaxing of some requirements might yield better performance in other areas.

The case study used to illustrate these techniques is the novel turbofan oil system which is described in more detail in Chapter 4. The architecture design

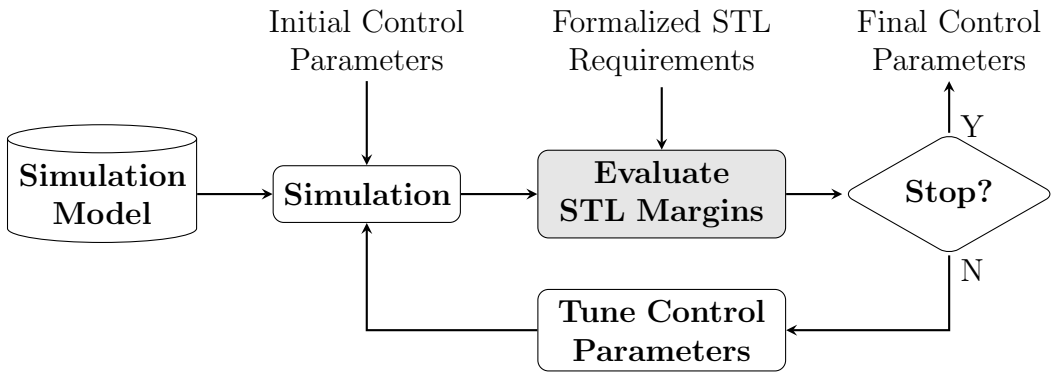


FIGURE 5.1: Simulation-based tuning of control parameters for optimal STL requirement margins. Buckets represent models, blocks represent processes, diamonds represent decisions and blank text represents inputs/outputs. The process is generic and could be applied to various different types of controllers, using different margin analysis techniques and stopping criteria. The grey block highlights where the multiSTL developments of this chapter fit into the overall synthesis.

stage only considers very simple static models, but the control synthesis requires a higher-fidelity dynamic model (as derived in Section 5.2) in order to evaluate performance. In Section 5.3 the performance requirements for the controller are converted from natural language to assume-guarantee contracts specified in STL. Section 5.4 performs the simulation-based optimization and uses the multiSTL framework to investigate tradeoffs between sub-formulae.

5.1 Simulation-Based Control Synthesis with multiSTL

As mentioned in Section 2.5, control synthesis from formal STL specifications has been achieved with model predictive control in [74, 76, 77]. The main problem with this approach is that the resulting mixed integer linear program optimization is NP-hard, which makes it unsuitable for safety-critical applications or systems which need to update control signals in short time intervals, such as aerospace systems. Therefore, for this type of system it may be desirable to synthesise more simple controllers that can be implemented in real-time, but which still perform well in simulations against a formal STL specification.

The generic structure of a simulation-based control synthesis is outlined in Figure 5.1. Note that this is not specific to any particular control law. For example,

the control parameters tuned could be PID gains, full state-feedback matrix values, lead/lag pole/zero locations etc (simple gain control is used in Section 5.4). There are also various options for the parameter tuning and stopping criterion, such as a fixed number of iterations of a genetic algorithm, or an exhaustive search of a discretised parameter space (as performed in Section 5.4).

The key part of the control synthesis (highlighted in grey in Figure 5.1) is calculating how well a given set of control parameters perform against the requirements. When requirements are specified formally in STL, this can be achieved by analysing the quantitative satisfaction as defined in [69]. This gives a measure of by how much an STL formula is satisfied or a *margin of satisfaction* (see Section 2.4.3 for more information).

Often the requirements for a control system will be a set of upper/lower bounds on states, inputs or outputs of the system joined together via conjunction (as in equation (5.18) in Section 5.3.2). For example:

$$\varphi_{\text{sys}} = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_N. \tag{5.1}$$

In this case the quantitative satisfaction for the whole system $\rho(\varphi_{\text{sys}})$ is defined in [69] as:

$$\rho(\varphi_{\text{sys}}) = \min\left(\rho(\varphi_1), \rho(\varphi_2), \cdots, \rho(\varphi_N)\right). \tag{5.2}$$

This means that the margin of satisfaction for the whole system is the minimum of the margins of satisfaction for the individual signals. The problem with using this definition is that it loses information about performance for all but the worst signal. When comparing multiple simulation results this can make it diffcult to determine which is the 'best' set of control parameters. Consider the simple example in Figure 5.2. Going by equation (5.2) the quantitative satisfaction for the two simulations in the figure is:

$$\begin{aligned} \rho(\varphi_{\text{sim}_1}) &= \min(-1, 4, 4) = -1, \\ \rho(\varphi_{\text{sim}_2}) &= \min(-1, 2, 2) = -1. \end{aligned} \tag{5.3}$$

This states that the control parameters for simulation 1 are equally as good as the control parameters for simulation 2. However, since simulation 1 performs better against the second two sub-formulae, it *dominates* the performance of simulation 2. This means it is a better designed system.

Another issue with taking the minimum margin is demonstrated in Figure 5.3. In this figure there is a tradeoff between the two simulations. Simulation 1 has greater margins of satisfaction for φ_2 and φ_3 whilst simulation 2 has a bigger margin for φ_1.

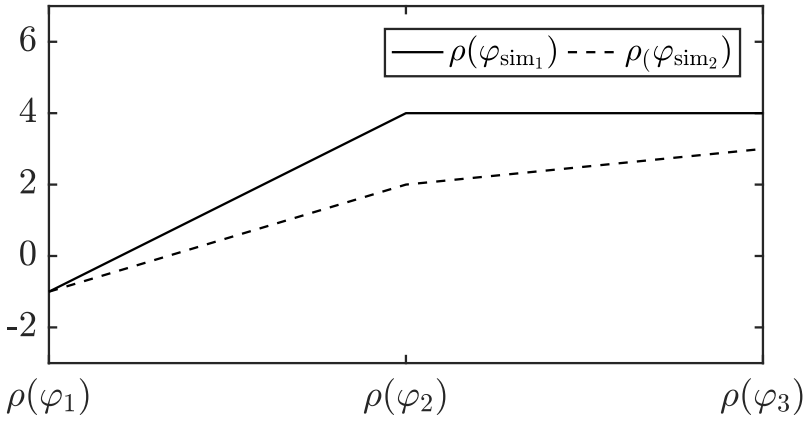


FIGURE 5.2: A simple multiSTL parallel coordinates plot for two simulations. Note that while the minimum margin is the same for both systems ($\rho(\varphi_1) = -1$), the second simulation has lower margins for both φ_2 and φ_3. Therefore the performance of simulation 1 *dominates* that of simulation 2.

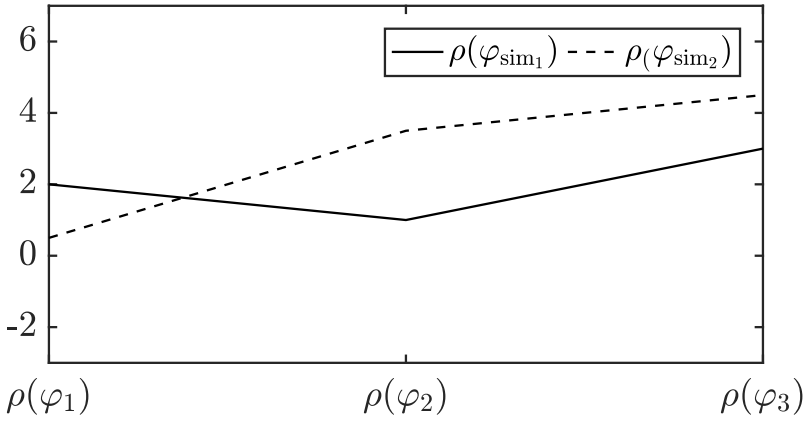


FIGURE 5.3: A simple multiSTL parallel coordinates plot for two simulations. Note that the minimum margin for simulation 2 is worse than for simulation 1 (0.5 and 2 respectively). However, simulation 2 performs better for φ_2 and φ_3. Therefore there is a tradeoff in performance between φ_1 vs φ_2 and φ_3.

Using the rules defined in equation (5.2) the overall system margins are calculated as:

$$\begin{aligned} \rho(\varphi_{\text{sim}_1}) &= \min(2, 1, 3) = 1, \\ \rho(\varphi_{\text{sim}_2}) &= \min(0.5, 3.5, 4.5) = 0.5. \end{aligned} \tag{5.4}$$

This indicates that simulation 1 is superior to simulation 2, but that is only the case if φ_1 is more important to the system stakeholders than φ_2 and φ_3.

multiSTL - Extending Quantitative Satisfaction To Multiple Dimensions

The issues highlighted in Figure 5.2 and Figure 5.3 are the same problems that led to the development of the multiobjective optimization techniques discussed in Section 3.2. The key difference in multiobjective optimization is the concept of Pareto-optimality. This refers to the fact that when there are multiple objectives, there is no single 'best' solution. Rather there is a set of solutions on the Pareto front which are all better than each other in some respect [53].

To get from the Pareto-optimal set to a single solution, a system designer is required to narrow down on a particular area of the Pareto front. In Chapter 3 this is carried out using a parallel coordinates plot, as shown in Figure 3.5. This has various advantages:

1. Parallel coordinates are a visual way of displaying large amounts of multidimensional data.
2. All individual decision criteria values are visualised - not just a combined value.
3. Tradeoffs between decision criteria are clear - both for the entire solution set and for individual solutions.
4. Refining the Pareto set to a single solution can be achieved via reducing the upper/lower bounds on decision criteria (progessive preference articulation).

For these reasons parallel coordinates are used in this research to display the individual STL sub-formula margins $(\rho(\varphi_1), \cdots, \rho(\varphi_N))$ for each simulation/set of control parameters. This approach is termed *multiSTL* and is demonstrated in the simple plots of Figure 5.2 and Figure 5.3. The approach taken to determine whether a solution is on the Pareto-front is the Pareto-based ranking as discussed in Section 3.2.3. As in the architecture optimization, simulations which are dominated by another (i.e. have lower margins for every STL sub-formula) are removed from the candidate set of control parameters. Note that while in Chapter 3 the aim was to *minimise* decision criteria scores, the aim here is to *maximise* the sub-formula margins. Therefore, when refining the candidate set of control parameters to a single solution, lower-bounds on the parallel coordinates plot are set, rather than upper-bounds.

The control synthesis approach presented in this section is applied to the novel oil system case study in Section 5.4, after defining the model (Section 5.2) and formalizing the requirements to STL (Section 5.3).

5.2 Developing an Oil System Simulation Model

A turbofan oil system consists of various different components such as tanks, pumps, valves, de-aerators, filters, heat exchangers amongst others [91, 92]. Following the more simplified architecture outlined in Chapter 4, the system modeled here contains only a tank, feed pump, heat exhcanger, variable restrictor valves (VRVs), oil chambers and scavenge pumps, as outlined in Figure 5.4. The control system architecture uses sensors to measure the oil chamber scavenge temperatures and controls the oil flows by actuating the VRVs and pumps.

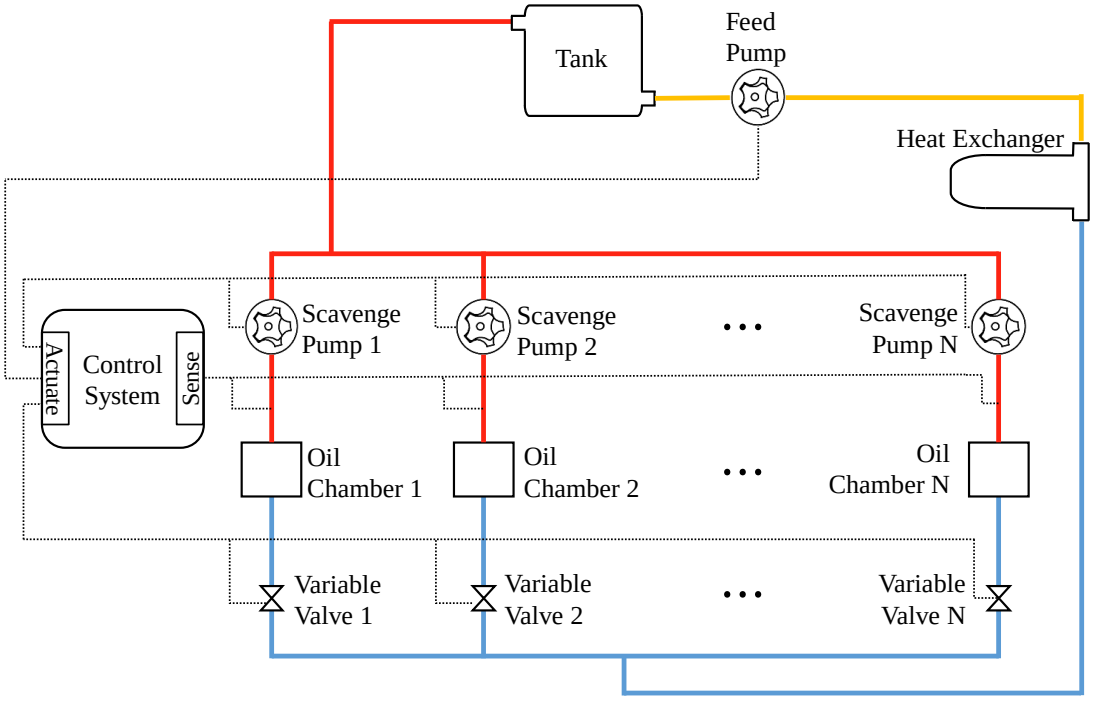


FIGURE 5.4: A schematic of an actively-controlled turbofan oil system. Solid lines indicate cold (blue), warm (orange) and hot (red) oil flows. Dashed lines indicate control system signals from sensors or to actuators.

The following sections cover the thermal modelling of the components in this system, assuming that the pumps and valves only affect the oil flow rates rather than temperatures.

5.2.1 Oil Tank Modelling

The oil tank can be modelled using the thermal energy balance equation $\text{Heat}_{\text{in}} = \text{Heat}_{\text{out}} + \text{Heat}_{\text{stored}}$. More formally this is defined as,

$$H_{\text{in}}(t) = H_{\text{out}}(t) + H_{\text{stored}}(t). \tag{5.5}$$

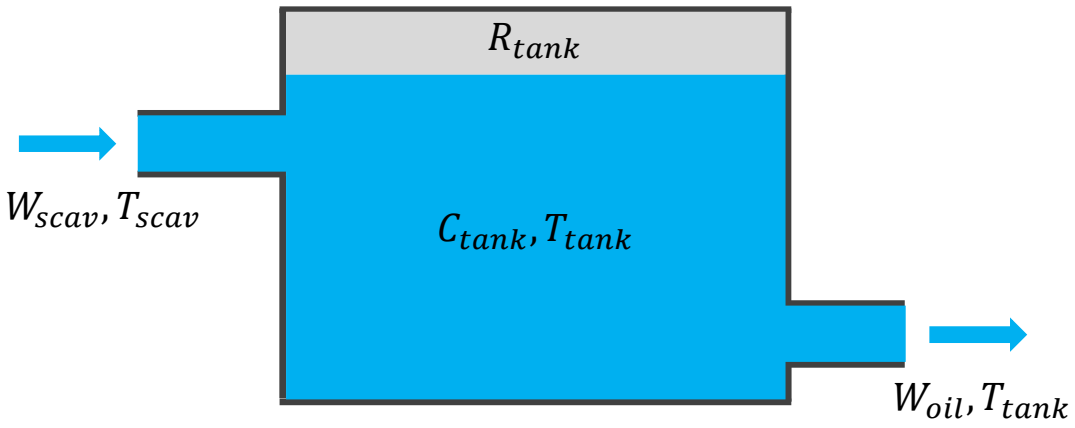


FIGURE 5.5: A simple diagram of an oil tank.

Considering a simple tank with equal inlet and outlet flow rates as outlined in Figure 5.5:

- $H_{\text{in}}(t) \in \mathbb{R}$ is provided by the hot oil coming from the combined scavenge line.
- $H_{\text{out}}(t) \in \mathbb{R}$ is the heat lost through the tank walls to the atmosphere.
- $H_{\text{stored}}(t) \in \mathbb{R}$ is the heat transferred to the oil in the tank.

This gives:

$$W_{\text{oil}}(t)\, c_{p_{\text{oil}}}\, (T_{\text{scav}}(t) - T_{\text{tank}}(t)) = \frac{T_{\text{tank}}(t)}{R_{\text{tank}}} + c_{p_{\text{oil}}}\, m_{\text{tank}}\, \frac{d\, T_{\text{tank}}(t)}{dt}, \tag{5.6}$$

which rearranges to give the first order equation for the oil tank:

$$\frac{d\, T_{\text{tank}}(t)}{dt} = \frac{W_{\text{oil}}(t)}{m_{\text{tank}}}\, (T_{\text{scav}}(t) - T_{\text{tank}}(t)) - \frac{T_{\text{tank}}(t)}{R_{\text{tank}}\, c_{p_{\text{oil}}}\, m_{\text{tank}}}. \tag{5.7}$$

where $T_{\text{tank}}(t)$ and $T_{\text{scav}}(t)$ are the temperatures of the oil in the tank and the scavenge line respectively, $W_{\text{oil}}(t)$ is the mass flow rate of the oil (assuming $W_{\text{oil}}(t) = W_{\text{scav}}(t) = W_{\text{feed}}(t)$), R_{tank} is the thermal resistance of the tank walls, C_{tank} is the thermal capacity of the tank and $c_{p_{\text{oil}}}$ is the specific heat of the scavenge oil.

5.2.2 Fuel-Oil Heat Exchanger Modelling

The fuel-oil heat exchanger is used to transfer heat from the hot oil to the cold fuel. This serves a dual purpose of keeping the oil temperature within the specified operating range, and preventing ice from forming in the fuel system [91, 92]. This heat flow $H_{\text{oil-fuel}}(t) \in \mathbb{R}$ is given by:

$$H_{\text{oil-fuel}}(t) = U_{\text{oil-fuel}}\, A_{\text{oil-fuel}}\, \Delta T_{\text{oil-fuel}}(t), \tag{5.8}$$

where $U_{\text{oil-fuel}}$ is the heat transfer coefficient and $A_{\text{oil-fuel}}$ is the oil-metal contact surface area. For counter-current heat exchangers $\Delta T_{\text{oil-fuel}}(t)$ is given by the logarithmic mean temperature difference (LMTD):

$$\Delta T_{\text{oil-fuel}}(t) = \frac{(T_{\text{tank}}(t) - T_{\text{fuel}_{\text{out}}}(t)) - (T_{\text{feed}}(t) - T_{\text{fuel}_{\text{in}}}(t))}{\log\left(\frac{T_{\text{tank}}(t) - T_{\text{fuel}_{\text{out}}}(t)}{T_{\text{feed}}(t) - T_{\text{fuel}_{\text{in}}}(t)}\right)}. \tag{5.9}$$

The most commonly used heat exchanger configuration in gas turbines is a tube-and-shell type as shown in Figure 5.6. Since this is not a true counter-current heat exchanger, a correction factor F_c has to be applied. This correction factor can be taken from lookup tables published by the Tubular Exchanger Manufactures Association (TEMA), depending on factors such as the number of tubes, passes and baffles [99–101].

The outlet temperature of the oil is then given by:

$$T_{\text{feed}}(t) = T_{\text{tank}}(t) - \frac{U_{\text{oil-fuel}}\, A_{\text{oil-fuel}}}{W_{\text{oil}}(t) c_{p_{\text{oil}}}}\, F_c\, \frac{(T_{\text{tank}}(t) - T_{\text{fuel}_{\text{out}}}(t)) - (T_{\text{feed}}(t) - T_{\text{fuel}_{\text{in}}}(t))}{\log\left(\frac{T_{\text{tank}}(t) - T_{\text{fuel}_{\text{out}}}(t)}{T_{\text{feed}}(t) - T_{\text{fuel}_{\text{in}}}(t)}\right)}. \tag{5.10}$$

Similarly, the fuel outlet temperature is given by:

$$T_{\text{fuel}_{\text{out}}}(t) = T_{\text{fuel}_{\text{in}}}(t) + \frac{U_{\text{oil-fuel}}\, A_{\text{oil-fuel}}}{W_{\text{fuel}}(t) c_{p_{\text{fuel}}}}\, F_c\, \frac{(T_{\text{tank}}(t) - T_{\text{fuel}_{\text{out}}}(t)) - (T_{\text{feed}}(t) - T_{\text{fuel}_{\text{in}}}(t))}{\log\left(\frac{T_{\text{tank}}(t) - T_{\text{fuel}_{\text{out}}}(t)}{T_{\text{feed}}(t) - T_{\text{fuel}_{\text{in}}}(t)}\right)}. \tag{5.11}$$

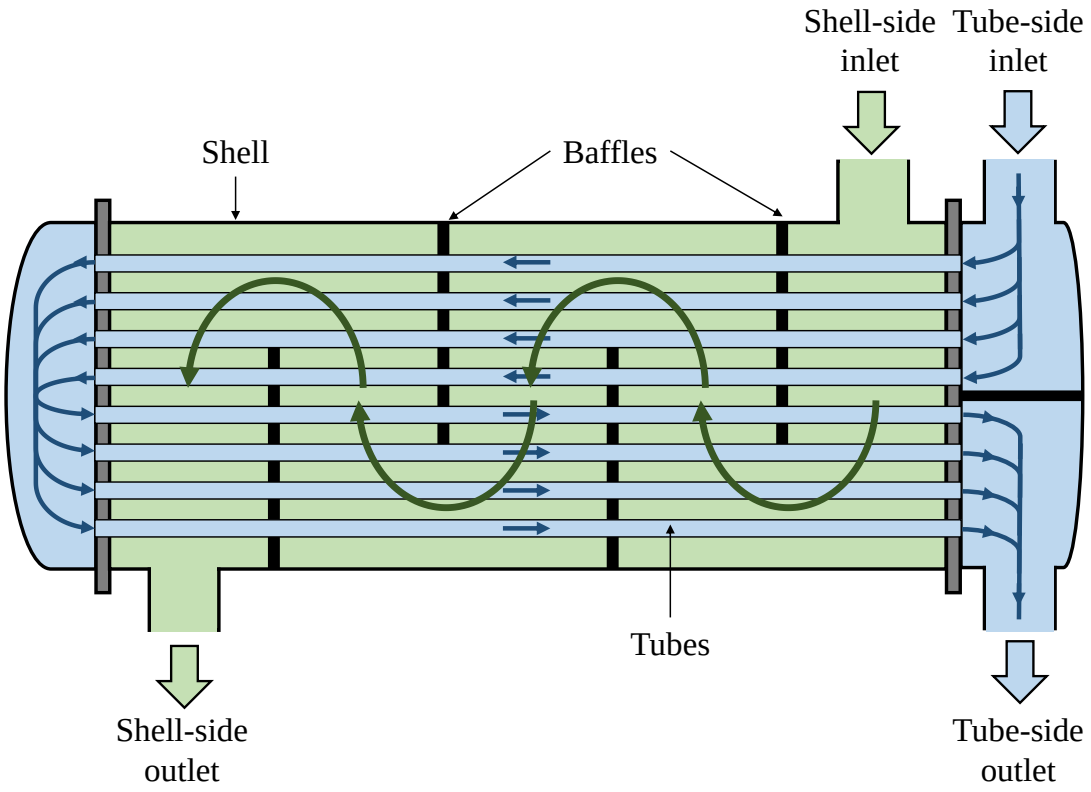


FIGURE 5.6: An example of a tube-and-shell heat exchanger. The tube flow is passed through a group of tubes running through the shell. Heat transfer is achieved by directing the shell flow around the tube group via baffles.

5.2.3 Heat To Oil Modelling

The purpose of the engine oil is to lubricate the bearings or gears in the oil chambers and to remove the heat caused by friction in these chambers. This can be considered as a heat exchanger with heat flow similar to that given in equation (5.8):

$$H_{\text{metal}-\text{oil}}(t) = U_{\text{metal}-\text{oil}}\, A_{\text{metal}-\text{oil}}\, \Delta T(t), \tag{5.12}$$

where $U_{\text{metal}-\text{oil}}$ is the heat transfer coefficient, $A_{\text{metal}-\text{oil}}$ is the oil-metal contact surface area and $\Delta T(t) = (T_{\text{metal}}(t) - T_{\text{feed}}(t))$. This heat flow can be used to calculate the resulting scavenge oil temperature based on the feed oil temperature and feed oil flow:

$$T_{\text{scav}}(t) = T_{\text{feed}}(t) + \frac{H_{\text{metal}-\text{oil}}(t)}{W_{\text{oil}}(t)\, c_{p_{\text{oil}}}} = T_{\text{feed}}(t) + \frac{U_{\text{metal}-\text{oil}}\, A_{\text{metal}-\text{oil}}\, (T_{\text{metal}}(t) - T_{\text{feed}}(t))}{W_{\text{oil}}(t)\, c_{p_{\text{oil}}}}. \tag{5.13}$$

5.2.4 Oil Chamber Metal Modelling

Starting again from the energy balance equation:

$$\text{Heat}_{\text{stored}} = \text{Heat}_{\text{in}} - \text{Heat}_{\text{out}} = H_{\text{g}}(t) - (H_{\text{metal-oil}}(t) + H_{\text{atmosphere}}(t)), \quad (5.14)$$

where $H_{\text{g}}(t)$ is the heat generated via the ambient temperature and friction forces, $H_{\text{metal-oil}}(t)$ is the heat given to the oil as specified in equation (5.12) and $H_{\text{atmosphere}}(t)$ will be assumed to be negligible for this case study. $H_g(t)$ is defined as in [102, 103] by:

$$H_g(t) = k1_\omega\,(T_\omega - T_{\text{metal}}(t))\,\omega(t)^{0.4} = k1_\omega\,((T_{\text{ambient}}(t) + k2_\omega\,\omega(t)) - T_{\text{metal}}(t))\,\omega(t)^{0.4}, \quad (5.15)$$

where $k_{\omega_1}, k_{\omega_2}$ are heat transfer coefficients and $\omega(t)$ is the angular velocity of the bearings/shafts. Combining this with equations (5.12) and (5.14) gives:

$$\begin{aligned}\frac{d\,T_{\text{metal}}(t)}{d\,t} = &\frac{k1_\omega\,((T_{\text{ambient}}(t) + k2_\omega\,\omega(t)) - T_{\text{metal}}(t))\,\omega(t)^{0.4}}{c_{p_{\text{metal}}}\,m_{\text{metal}}} \\ &- \frac{U_{\text{metal-oil}}\,A_{\text{metal-oil}}}{c_{p_{\text{metal}}}\,m_{\text{metal}}}\,(T_{\text{metal}}(t) - T_{\text{feed}}(t)).\end{aligned} \quad (5.16)$$

5.2.5 Combining the Individual Scavenge Feeds

In a gas turbine oil system there are several separate feed lines (downstream from the heat exchanger) going to the different bearings or gearboxes. For simplicity it is assumed that the valves, splitters and restrictors which guide the main feed flow down these separate lines do not have a significant effect on the oil temperature. Therefore the feed temperatures for each chamber $T_{\text{feed}_1}(t), \cdots, T_{\text{feed}_N}(t)$ are the same as the exit temperature of the heat exchanger $T_{\text{feed}}(t)$. However, since each individual chamber will have a different temperature and oil flow rate, the individual scavenge temperatures $T_{\text{scav}_1}(t), \cdots, T_{\text{scav}_N}(t)$ will be different. The downstream combined scavenge temperature can be calculated via:

$$T_{\text{scav}}(t) = \frac{\sum_{i=1}^{N} T_{\text{scav}_i}(t)\,W_{\text{scav}_i}(t)}{W_{\text{scav}}(t)} = \frac{\sum_{i=1}^{N} T_{\text{scav}_i}(t)\,W_{\text{scav}_i}(t)}{W_{\text{oil}}(t)}. \quad (5.17)$$

Here it is assumed that $W_{\text{scav}}(t)$ is equal to the sum of the individual scavenge flow rates, and that this is the same as the combined feed flow $W_{\text{oil}}(t)$.

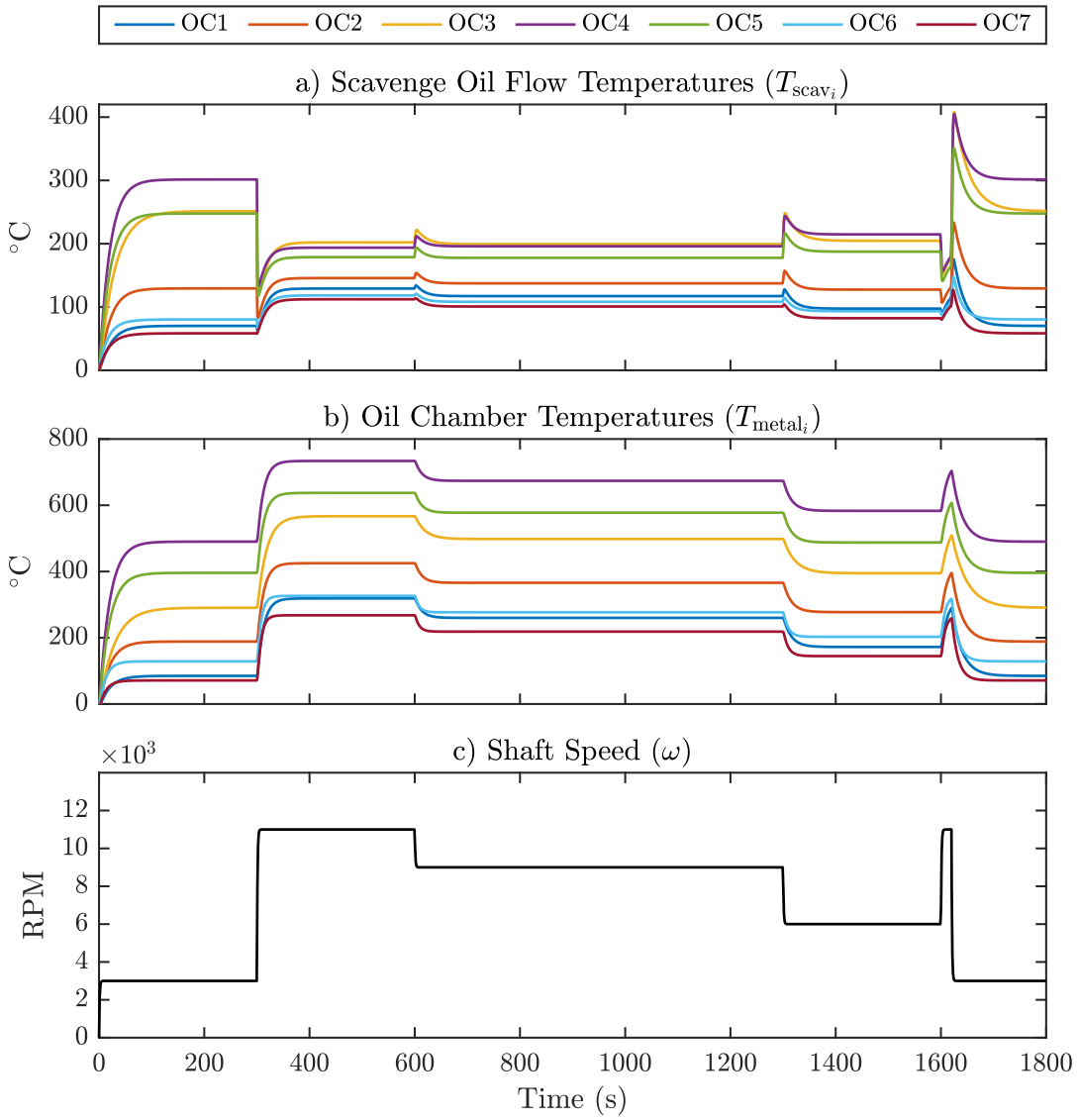


FIGURE 5.7: Oil system simulation results without active control. The flight envelope includes the 6 phases of typical flight: Ground Idle (0s-300s), Maximum Takeoff Thrust (300s-600s), Maximum Continuous Thrust (600s-1300s), Approach Idle (1300s-1600s), Thrust Reverse (1600s-1620s) and Ground Idle (1620s-1800s) [104].

5.2.6 Simulating the Nonlinear Simulation Model

The model derived in Section 5.2.1 to Section 5.2.5 has been implemented in MATLAB and Simulink [93] (see Appendix B for parameter values and block diagrams). Figure 5.7 shows a simulation of the model with oil flow linked to the high-pressure shaft speed via the accessory gearbox (i.e. no active control) for a standard passenger aeroplane flight envelope.

Note that the shaft speed spool up times are typically much faster than the oil

chamber temperature transients. For example, European Aviation Safety Agency CS-E 745 states that an increase from 15% to 95% rated Take-off thrust must take less than 5 seconds [105], whereas the oil chamber transients last tens to hundreds of seconds due to the specific heat capacity of the metal [106]. This difference can be seen in the the middle and bottom plots of Figure 5.7.

The variation between transient times causes spikes in the oil scavenge temperatures as shown in the top plot of Figure 5.7. For example, when stepping down from the 'Thrust Reverse' to 'Ground Idle' phases around 1600 seconds. The rapid decrease in shaft speed results in a rapidly reduced oil flow. Since there is less oil to absorb the heat from the slowly cooling oil chamber, the scavenge temperature rapidly increases to over 400 °C in some cases. This is well over the fire point of typical turbofan oils (e.g. 285°C for Mobil Jet Oil II [107]).

Aside from the spikes, it is also clear that lower shaft speeds result in higher steady-state temperatures. For example, in Figure 5.7 at 'Ground Idle' the highest scavenge temperatures are also close to the fire point. To solve these issues without active control, the feed pump size has to be increased. Unfortunately this results in oil being oversupplied at other thrust settings which has a negative impact on lubrication efficiency and hence on fuel consumption. These problems motivate the need for the novel oil system with active control, to manage oil temperature transients more efficiently and avoid temperature spikes.

The authors currently have no measured real-world engine data to validate these simulation results. However, qualitatively the simulations are performing as expected, matching the behaviour described by oil systems engineers.

5.3 Formalizing Requirements Using A/G Contracts

The simulation model for the novel oil system has been developed in Section 5.2, but before the controller can be designed a set of performance requirements are needed. There are many different viewpoints of interest to the system-level designer, but for simplicity this section focuses just on the thermal and flow-rate properties of the oil.

5.3.1 Informal, Textual Requirements

R1 to R4, specified in natural language, define the requirements that the novel oil system must satisfy.

Thermal Requirements
R1: Oil temperature shall not exceed 220°C at the scavenging side at any time.
R2: All oil flows from the oil system to the oil chambers shall have a temperature in the range 10°C to 120°C for all operating conditions above idle.

Flow Requirements
R3: All individual oil flows from the oil system to the oil chambers shall have a maximum flow rate of 0.4 kg s^{-1} at any time.
R4: The minimum oil feed flow to the chambers shall be 0.07 kg s^{-1} at any time.

These requirements relate to both a thermal *viewpoint* and a fluid flow *viewpoint* and are satisfied by the following subsystems:

- Heat exchanger - responsible for removing heat from the oil (up to the temperature specified in R1) to achieve R2.
- Oil chambers/valves - responsible for varying the flow to the oil chambers to satisfy R1 given R2.
- Pumping/storage - responsible for ensuring the flow rates in R3 and R4.

5.3.2 Formalized STL Assume/Guarantee Contract

In Appendix C the informal, textual requirements R1 to R4 are formalized to A/G contracts [20] specified in Signal Temporal Logic [65] for the different subsystems. These subsystem contracts are then combined via the rules of contract composition (see Appendix A) to check compatibility. This is performed to demonstrate how component-level A/G contracts can be composed to give a system-level contract, which would be useful if different components or subsystems were to be produced by different teams or supplier companies.

For brevity just the final system-level contract will be given here.

$$C_{\text{sys}} = \begin{cases} A_{\text{sys}} = \square\,(-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \wedge \square\,(0.1\,\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\,\text{kgs}^{-1}). \\ G_{\text{sys}} = \square\,(\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge \square\,(T_{\text{scav}_i} \leq 220°\text{C}) \\ \qquad \wedge \square\,(0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C}) \wedge \square\,(0.07\,\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\,\text{kgs}^{-1}). \end{cases} \tag{5.18}$$

Note that the assumptions do not correspond to any of the requirements R1 to R5. This is because they relate to environmental inputs from outside the scope of the novel oil system ($T_{\text{fuel}_{\text{in}}}$ and W_{fuel}). The fuel tank temperature can drop as low as $-35°$C when flying at max altitude (outside air temp $\approx -60°$C) and reach as high as 55°C when starting on the ground on a hot day [91]. Typical fuel flow values for a turbofan engine are around 0.5 kg s^{-1} [108] therefore upper/lower limits are set as 0.8 kg s^{-1} and 0.1 kg s^{-1}.

The guarantee corresponding to $T_{\text{fuel}_{\text{out}}}$ is also not contained in the original requirements. The upper/lower limits are set so that the fuel is always above 0°C (to avoid ice forming in the fuel filter) and below 165°C (to avoid degradation) [91].

With a simulation model and a formal A/G contract, the next stage of the design is to perform control synthesis to maximise the quantitative satisfaction [69] of the STL formulae in G_{sys}.

5.4 Oil System Control Synthesis with multiSTL

As highlighted in Figure 5.7, it is impossible to satisfy G_{sys} without using a very large sized feed pump. This results in an oversupply of oil during higher shaft speeds, which affects lubrication efficiency and increases fuel consumption. The problem can easily resolved, however, by introducing a simple gain-based feedback control of the form:

$$W_{\text{feed}_i}(k) = K_{p_i} T_{\text{metal}_i}(k-1), \tag{5.19}$$

where $W_{\text{feed}_i}(k)$ is the control input (oil chamber feed flows) at the current time step; $T_{\text{metal}_i}(k-1)$ is the oil chamber metal temperature at the previous time step; and K_{p_i} is the control gain.

This means that the hotter the metal in the oil chambers, the greater the oil flow. The advantage of scheduling oil in this way is that it is able to track the slower

transients of the oil chamber temperatures. However, measuring temperature of solids with rotating parts is technically challenging. Therefore, an alternative is to use:

$$W_{\text{feed}_i}(k) = K_{p_i} T_{\text{scav}_i}(k-1), \tag{5.20}$$

where $T_{\text{scav}_i}(k-1)$ is the scavenge flow temperature at the previous time step.

It is much easier to measure T_{scav_i} than T_{metal_i}, with a set of sensors measuring the scavenge oil temperatures downstream from the chambers. Note that since the scavenge temperature is highly coupled to the oil chamber temperature, this would still allow the slow transients to be tracked effectively, removing the spikes in oil temperature as shown in the top plot of Figure 5.8. In addition, this allows more oil flow to be supplied when the oil chambers are hotter. This occurs at higher operating points when faster rotational speeds create a larger demand for lube oil. Therefore the control law in (5.20) is able to ensure good thermal and lubrication efficiency.

Note that the choice of K_{p_i} values used in the simulation of Figure 5.8 are good for the feed flow rates as these are well within the upper/lower bounds set in the requirements. However, the scavenge temperatures are too high for oil chambers 3, 4 and 5. This represents a tradeoff, since increasing the oil flow to these three chambers will reduce the scavenge temperature, but push the feed flow rates closer to their upper bound. By performing multiple simulations with different control parameters, multiSTL can be used to investigate multiple tradeoffs such as this (see Section 5.4.4).

5.4.1 Discretising the Control Parameter Space

The control parameters tuned in this case study are the K_{p_i} gains from equation (5.20). From trial simulations, sensible choices for each K_{p_i} are in the interval $[0.0005, 0.0008]$. Larger values lead to saturation of the oil flows, and smaller values lead to large overshoots in temperature.

To reduce the possible search space, the set of potential control parameters is discretised in this range as shown in equation (5.21). The number of potential gain values for the 7 oil chambers is $\{3, 3, 3, 3, 4, 3, 3\}$ respectively. Note that this

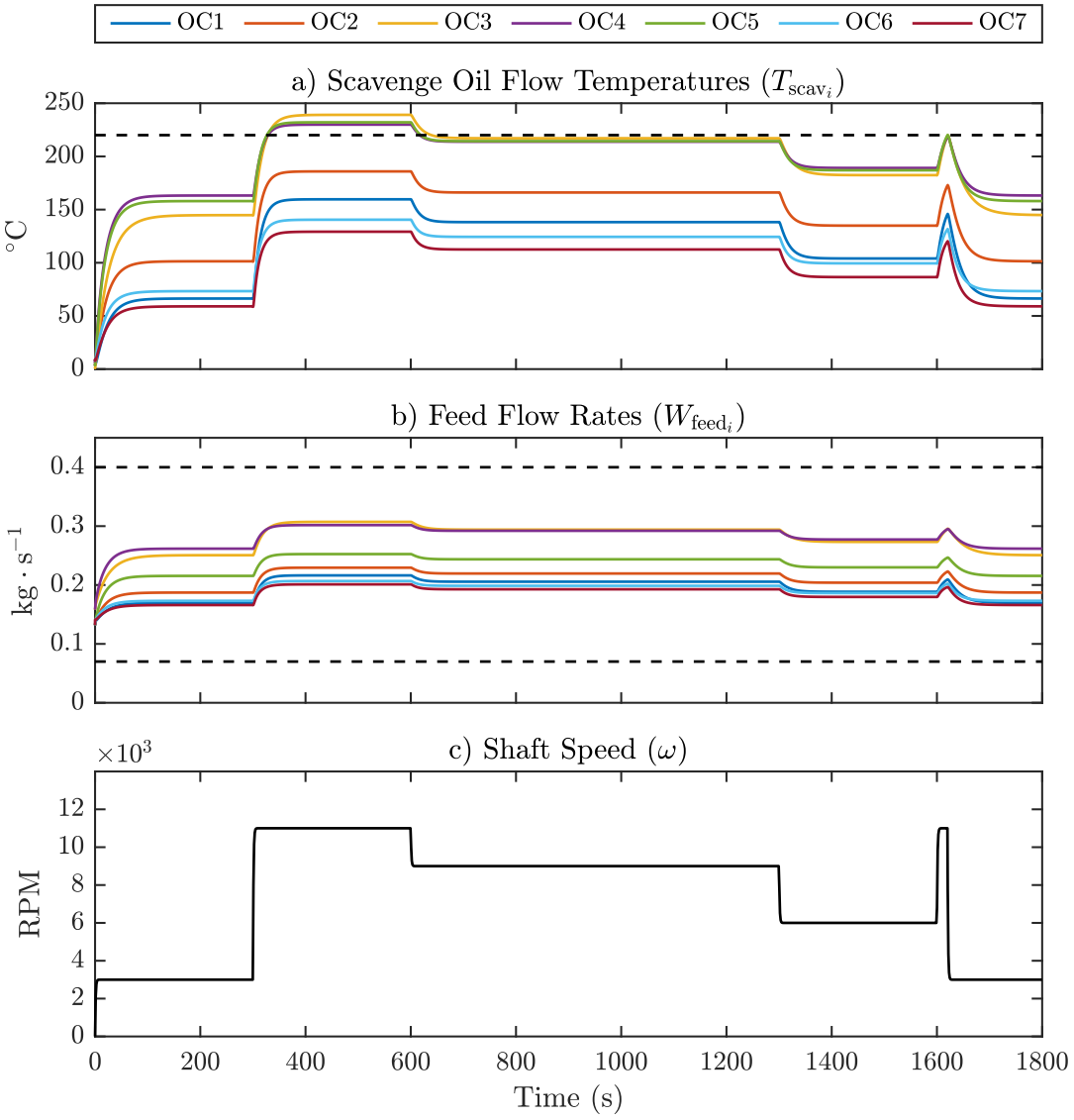


FIGURE 5.8: Oil system simulation results with feedback control. The flight envelope includes the 6 phases of typical flight: Ground Idle (0s-300s), Maximum Takeoff Thrust (300s-600s), Maximum Continuous Thrust (600s-1300s), Approach Idle (1300s-1600s), Thrust Reverse (1600s-1620s) and Ground Idle (1620s-1800s) [104]. The control gains have not been tuned and therefore some scavenge temperatures are too high (e.g. OC3, OC4, OC5).

provides $3 \times 3 \times 3 \times 3 \times 4 \times 3 \times 3 = 2916$ possible combinations.

$$\begin{aligned}
K_{p_1} &= \{0.0005,\, 0.0006,\, 0.0007\},\\
K_{p_2} &= \{0.0005,\, 0.0006,\, 0.0007\},\\
K_{p_3} &= \{0.0006,\, 0.0007,\, 0.0008\},\\
K_{p_4} &= \{0.0006,\, 0.0007,\, 0.0008\},\\
K_{p_5} &= \{0.0005,\, 0.0006,\, 0.0007,\, 0.0008\},\\
K_{p_6} &= \{0.0005,\, 0.0006,\, 0.0007\},\\
K_{p_7} &= \{0.0005,\, 0.0006,\, 0.0007\}.
\end{aligned} \tag{5.21}$$

5.4.2 Weighting the Sub-formulae

For simplicity in plotting/visualising the different formulae, G_{sys} from equation (5.18) can be written as:

$$G_{\text{sys}} = \varphi_{1,i} \wedge \varphi_{2,i} \wedge \varphi_3 \wedge \varphi_4, \tag{5.22}$$

where,

$$\begin{aligned}
\varphi_{1,i} &= \square\,(0.07\,\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\,\text{kgs}^{-1}),\\
\varphi_{2,i} &= \square\,(T_{\text{scav}_i} \leq 220^{\circ}\text{C}),\\
\varphi_3 &= \square\,(\neg\text{Idle} \implies 10^{\circ}\text{C} \leq T_{\text{feed}} \leq 120^{\circ}\text{C}),\\
\varphi_4 &= \square\,(0^{\circ}\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165^{\circ}\text{C}).
\end{aligned} \tag{5.23}$$

The margin of satisfaction for each sub-formula can then be calculated as:

$$\begin{aligned}
\rho_w(\varphi_{1,i}) &= \min\left(W_{\text{feed}_i}(t) - 0.07,\, 0.4 - W_{\text{feed}_i}(t)\right) \cdot w_{\text{flow}} && \forall\,\{t \in [0, 1800]\},\\
\rho_w(\varphi_{2,i}) &= \min\left(220 - T_{\text{scav}_i}(t)\right) \cdot w_{\text{temp}} && \forall\,\{t \in [0, 1800]\},\\
\rho_w(\varphi_3) &= \min\left(T_{\text{feed}}(t) - 10,\, 120 - T_{\text{feed}}(t)\right) \cdot w_{\text{temp}} && \forall\,\{t \mid \omega(t) > 3000\,\text{RPM}\},\\
\rho_w(\varphi_4) &= \min\left(T_{\text{fuel}_{\text{out}}} - 0,\, 165 - T_{\text{fuel}_{\text{out}}}\right) \cdot w_{\text{temp}} && \forall\,\{t \in [0, 1800]\}.
\end{aligned} \tag{5.24}$$

Note that these are weighted formulae multiplied by either $w_{\text{flow}} = 1/0.4$ or $w_{\text{temp}} = 1/220$. The choice of weights is designed to scale the margins so that they can be visualised on the same parallel coordinates plot. This makes it more difficult to determine the margin of satisfaction in °C or $\text{kg} \cdot \text{s}^{-1}$, but preserves the qualitative satisfaction with $\rho_w(\varphi) \geq 0$ indicating the requirement is satisfied and $\rho_w(\varphi) < 0$ highlighting a violation. More information on weighted STL formulae can be found in [73].

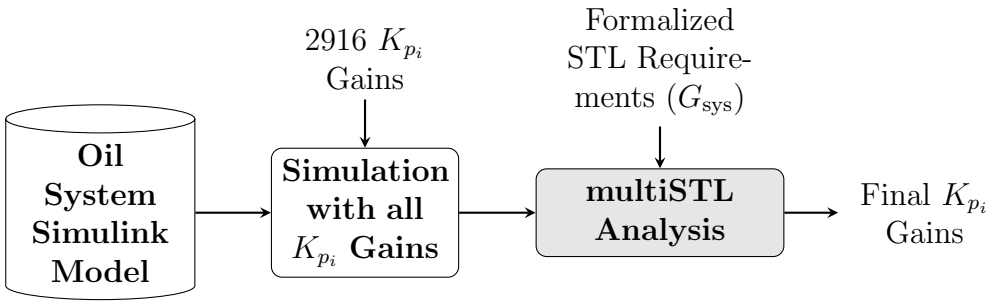


FIGURE 5.9: Oil system control synthesis using an exhaustive search of the discretised parameter space. Buckets represent models, blocks represent processes and blank text represents inputs/outputs. The grey block highlights where the multiSTL developments of this chapter fit into the overall synthesis.

5.4.3 Exhaustive Search of the Discretised Parameter Space

Each simulation takes approximately 5 seconds to run. This translates to $2916 \times 5 = 14580$ seconds (approximately 4 hours) to simulate all possible combinations of K_{p_i} values. Since this is not too unreasonable, exhaustive search is used as the specific method of tuning the control parameters. Therefore the generic flow outlined in Figure 5.1 can be updated to that shown in Figure 5.9.

After performing all the simulations and storing the weighted margins as calculated in equation (5.24), the next step is to investigate tradeoffs and narrow down on a final parameter set using multiSTL.

Note that exhaustive search is not feasible for larger problem sizes. Therefore multi-criteria optimization approaches, such as those discussed in Section 3.2, could be used for such problems.

5.4.4 Analysing Tradeoffs with multiSTL

The multiSTL parallel coordinates plot is shown in Figure 5.10. In this plot, each solution is represented as a line linking the $\rho_w(\varphi_i)$ values for that simulation. Tradeoffs in the data set occur when there is a crossover of lines between the margins for two sub-formula. For example, consider the zoomed-in parallel coordinates plot for $\varphi_{1,4}$ and $\varphi_{2,4}$ shown in Figure 5.11 a). Increasing the value of K_{p_4} results in a decrease in $\rho_w(\varphi_{1,4})$, the margin for oil flow rate (see Figure 5.11 b)) but an increase in $\rho_w(\varphi_{2,4})$, the margin for scavenge temperature (see Figure 5.11 c)).

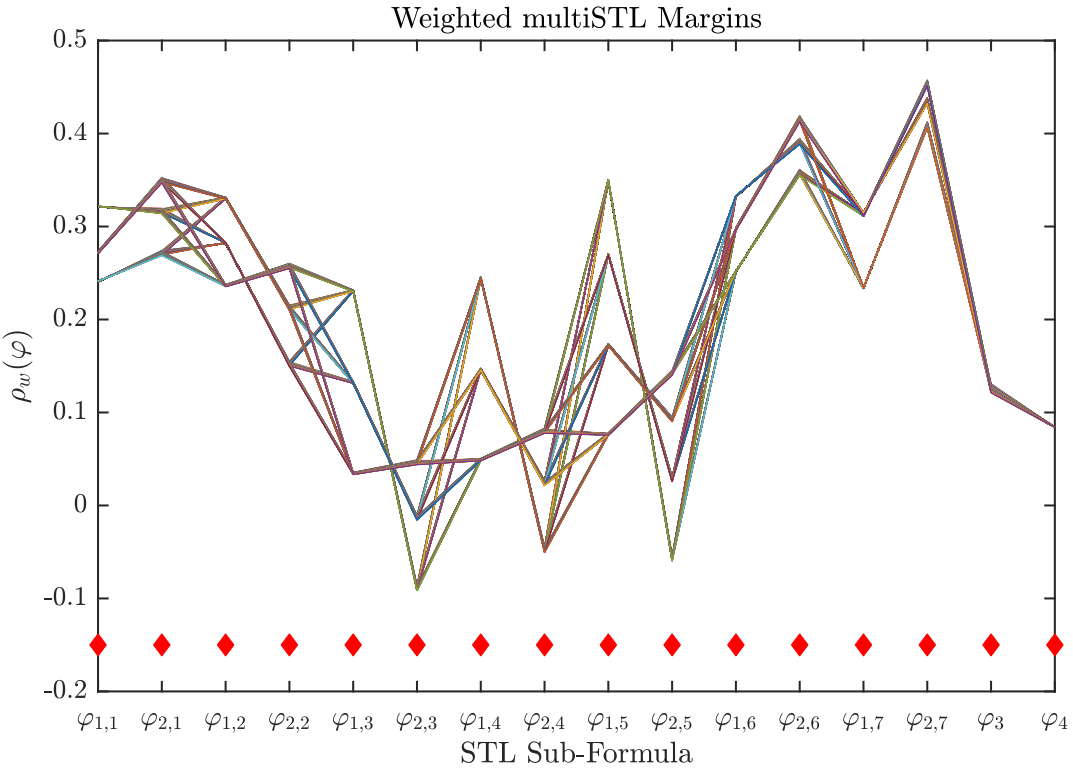


FIGURE 5.10: Weighted multiSTL analysis for the oil system controller. Each of the 2916 K_{p_i} combinations is represented as a line linking the margins for each sub-formula for that simulation. A final solution can be narrowed down by altering the lower bounds for each margin (red diamonds).

Refining to a Single Parameter Set Using multiSTL

The procedure for altering the parallel coordinates limits for this case study is as follows:

1. All $\rho_w(\varphi_i)$ values must be ≥ 0, as shown in Figure 5.12 a). This is to ensure that all the requirements have been met. This leaves only one parameter choice for oil chamber 3 ($K_{p_3} = 0.0008$).

2. All other temperature margins must be $\geq 10°$C. This corresponds to $\rho_w(\varphi_{i,2}) \geq 10/220 = 0.045$, as shown in Figure 5.12 b). This ensures large enough margins to avoid getting close to unsafe fire points of oil or fuel.

3. From the remaining solutions, flow rate margins should be prioritised to ensure better lubrication efficiency and hence better fuel efficiency, as shown in Figure 5.12 c).

This is the suggested choice of refinement used for this case study, but there could be other good choices depending on the priorities of the system stakeholders. The

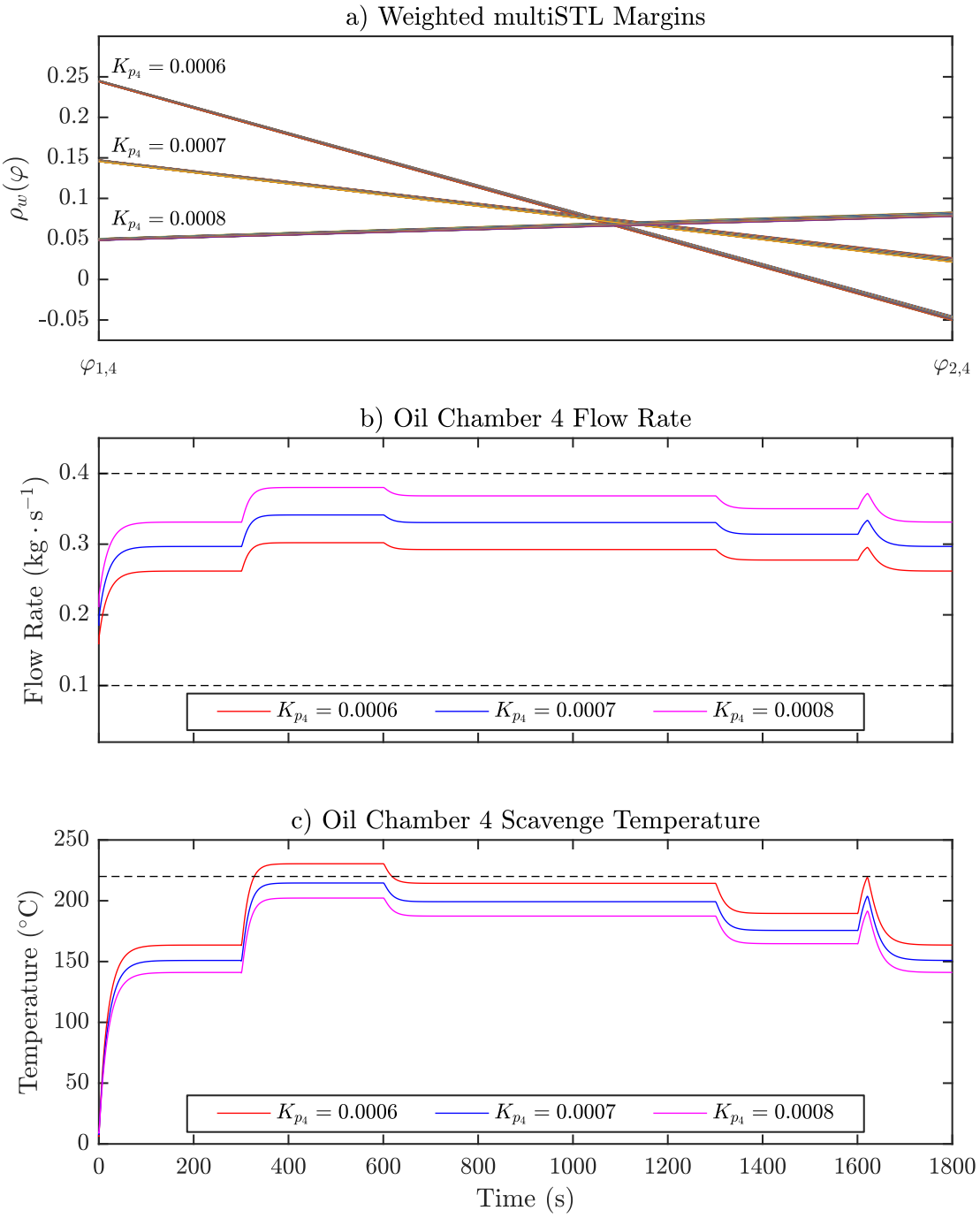


FIGURE 5.11: Analysing the tradeoff between oil flow rate and scavenge temperature in oil chamber 4. a) shows a zoomed-in multiSTL parallel coordinates plot for the two relevant sub-formulae. b) shows the oil flow rate for different K_{p_4} values in comparison to the upper/lower bounds of $\varphi_{1,4}$. c) shows the scavenge temperature for different K_{p_4} values in comparison to the upper bound of $\varphi_{2,4}$.

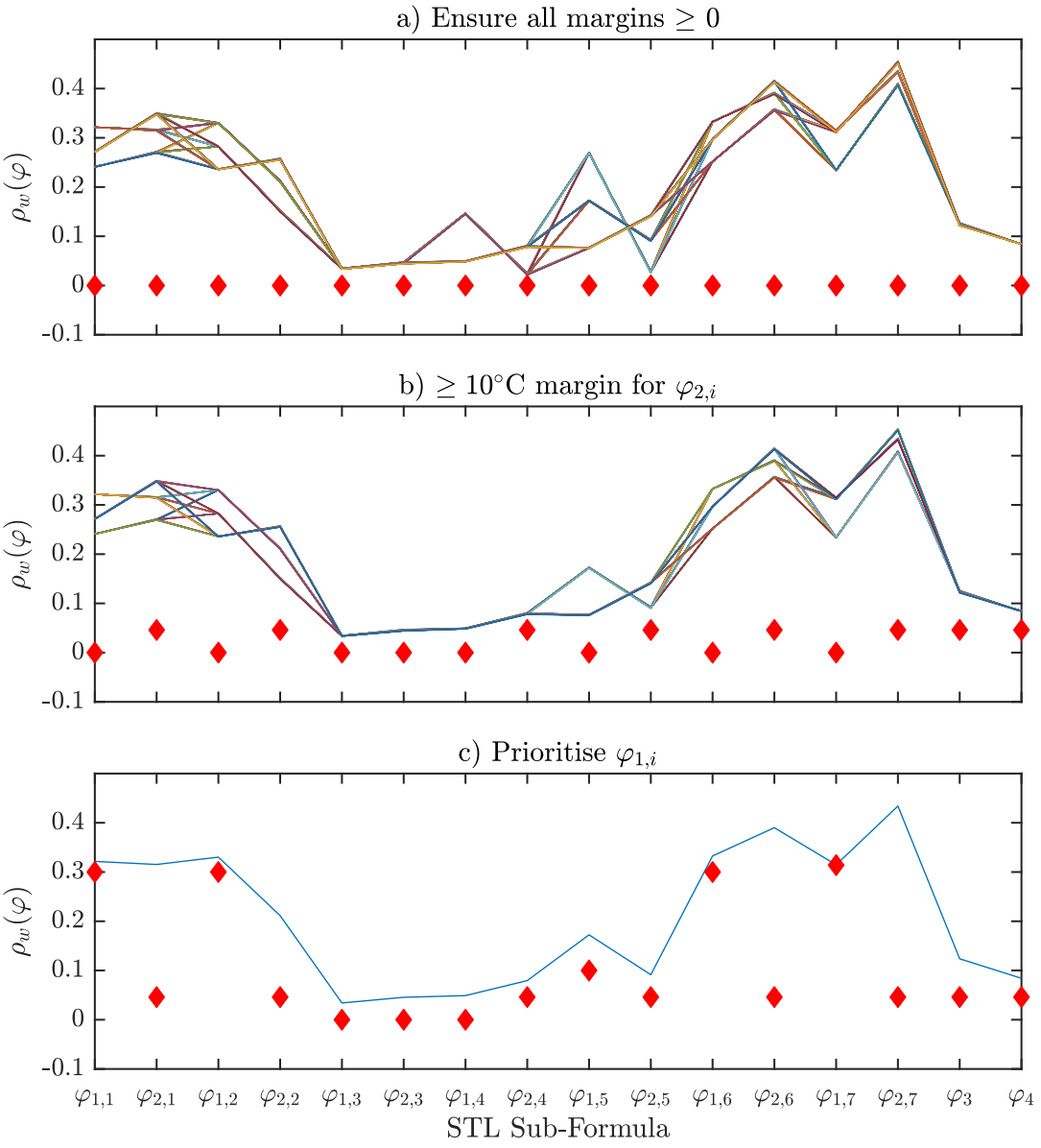


FIGURE 5.12: Three stages of refinement using a multiSTL parallel coordinates plot. a) ensuring all requirements are met ($\rho_w(\varphi_i) \geq 0$). b) temperature margins ≥ 10°C. c) Flow rates prioritised for the remaining solutions.

single parameter set resulting from this refinement procedure is:

$$\begin{aligned} K_{p_1} &= 0.0006, \\ K_{p_2} &= 0.0006, \\ K_{p_3} &= 0.0008, \\ K_{p_4} &= 0.0008, \\ K_{p_5} &= 0.0007, \\ K_{p_6} &= 0.0006, \\ K_{p_7} &= 0.0006. \end{aligned} \tag{5.25}$$

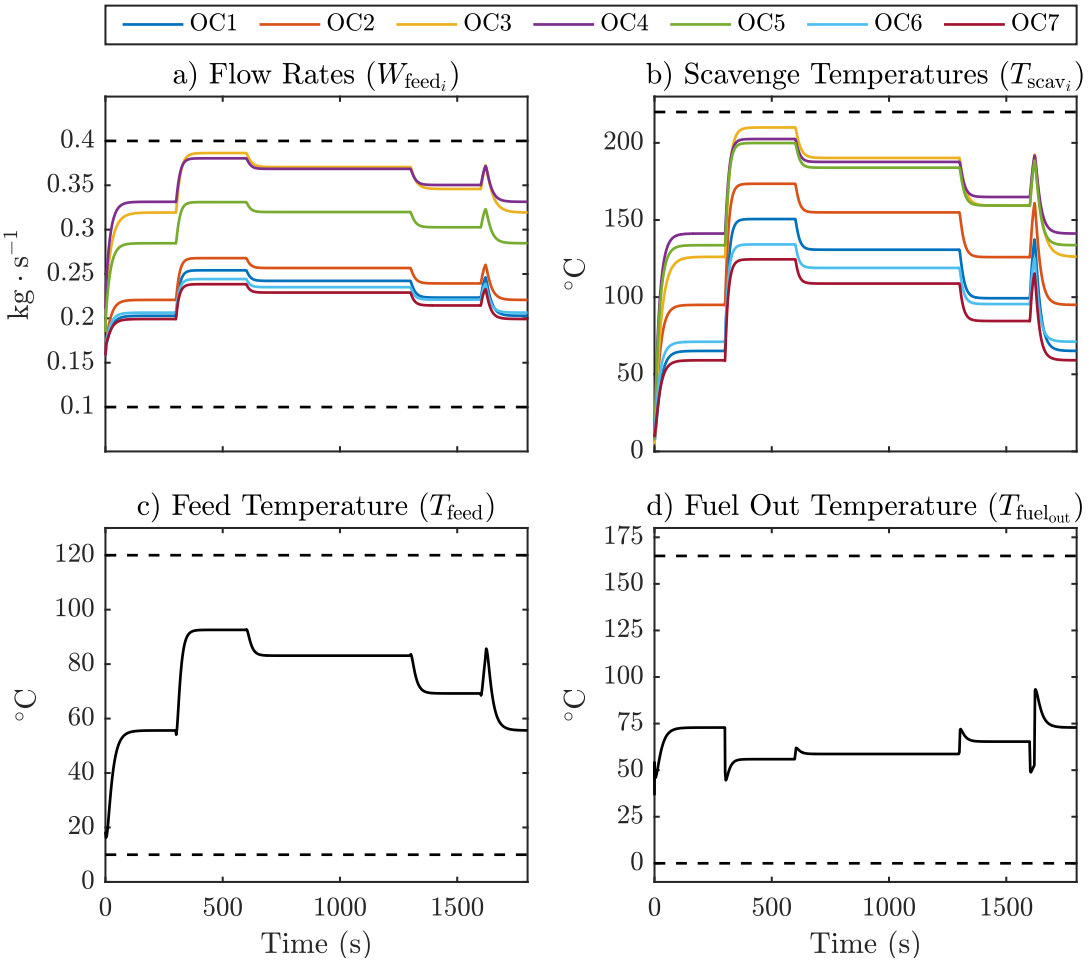


FIGURE 5.13: Weighted multiSTL analysis for the oil system controller. Each of the 2916 K_{p_i} combinations is represented as a line linking the margins for each sub-formula for that simulation. A final solution can be narrowed down by altering the lower bounds for each margin (red diamonds).

which produces the signals shown in Figure 5.13. Note that all of the signals fall well inside the upper/lower bounds at all times. The flow rates rise/fall with the scavenge temperatures, which are coupled to the oil chamber temperatures. This ensures effective thermal and lubrication system performance, with an increase in flow rates when the demand for cooling and lubrication is higher.

5.5 Conclusion

This chapter has introduced the multiSTL framework for synthesising control parameters using simulation. The advantage of using this approach over calculating

an overall system margin using the min operator is that it allows tradeoffs between different requirements to be analysed. For example, optimising oil flow rates results in non-optimal oil temperatures and vice versa.

multiSTL can also be used to show where it may be beneficial to relax certain requirements if they are too tight. For example, by allowing a greater oil flow rate to a certain oil chamber, the scavenge temperature margins can be increased. Being able to show this on a multiSTL parallel coordinates plot provides system designers with an evidence base to help persuade stakeholders that this would be a sensible change.

The developments of this chapter follow on from previous architecture level design techniques discussed in Chapter 3 and Chapter 4. To demonstrate the multiSTL approach, the novel oil system case study from previous chapters has been used. The oil system model has been derived from first principles and implemented in Simulink. This has been simulated with a large set of potential control parameters to synthesise a controller which optimises satisfaction of formal STL requirements. The refinement from the large candidate set of parameters to final chosen set has been carried out by iteratively altering the limits on a multiSTL parallel coordinates plot.

Chapter 6

Conclusions and Future Work

This final chapter summarises the work presented in Chapters 3-5, highlights the main contributions of the thesis and suggests potential areas of further research.

6.1 Summary

This thesis has presented a multilevel framework for system design using formalized requirements, as outlined in Figure 1.1. The approach is both bottom-up (in the population of libraries of components from high-fidelity to simple static models) and top-down (in the progression from requirements to detailed design).

A key aspect of the framework is the use of formalized requirements to allow design automation techniques such as optimization, and mapping between different design levels. This focuses most of the design effort at the higher levels, where tradeoffs between customer preferences, engineering characteristics and performance requirements can be analysed. Whilst this thesis has required considerable modeling effort at each design level, the goal for an industrial implementation of this framework would be to perform modeling only once in the population of a library of models. These models could then be re-used in similar design exercises with the majority of the engineering effort focused on the top level requirements.

Fully realizing this multilevel flow for a real-world complex system would require a whole team of engineers working over considerable timescales. This is out of the scope of a single PhD researcher. However, this thesis has shown multiple specific instances of design automation and shown how these exercises fit into the wider vision of the multilevel design framework.

Chapters 3 and 4 focus on the architecture optimization level, which is further split into the high-level architecture framework and low-level topology optimization (as in Figure 1.2). The high-level architecture framework optimization occurs at the concept design stage and consists of two sides: architecture synthesis and refinement. The synthesis stage begins by determining a set of potential means for implementing each system function. Each physical means is scored against a set of engineering characteristics such as 'modularity' or 'robustness'. A high-level architecture framework is defined by selecting a single means for each function. The overall architecture scores for the engineering characteristics are calculated as a sum of the scores for the means chosen. Architectures are then synthesised using a multiobjective genetic algorithm which produces a Pareto-optimal set of candidate solutions.

The architecture refinement stage uses an interactive parallel coordinates plot which displays each Pareto-optimal architecture as a line linking its decision criteria scores. Refining this set of solutions to a smaller set is achieved via progressively changing the upper bounds on the decision criteria scores. A customer-oriented refinement algorithm has been presented which allows the parallel coordinates limits to be altered rapidly, to best reflect the set of customer preference weightings. This has been extended with resilience analysis, which shows how much the weightings can change before a given solution disappears from the optimal set.

The high-level architecture synthesis and refinement techniques have been implemented in the user-friendly SATS tool, which allows system architects to use this approach with only simple text input files. The approach has been demonstrated on two case studies: a pressurized water reactor EC&I system and a turbofan oil system.

Chapter 4 takes the turbofan oil system architecture framework defined in the previous case study and performs the lower level architecture topology optimization. The oil system architecture is modeled as a graph with nodes representing components such as pumps, valves, oil chambers and heat exchangers. Edges between nodes indicate a physical pipe connection between the components. The edges are represented as an adjacency matrix with a 1 indicating an edge between two nodes and a 0 indicating no connection. The architecture requirements are formalized as constraints on the adjacency matrix.

The architecture design is carried out as a constrained optimization, with two conflicting objectives: cost and controllability. The decision variables are the values in the adjacency matrix (connections between nodes). The main tradeoff is between the number of valves to use, since less valves will reduce cost but more

will increase controllability. Cost is determined as a function of the nodes used in the architecture, and the number of connections. To determine controllability for a given architecture, an oil flow requirement similarities matrix, C_{f_r} is populated using a heuristic algorithm. This ensures that when the number of valves used is less than 1 per chamber, there is a sensible coupling of valves to oil chambers with similar flow requirements.

Following on from the architecture optimization levels, Chapter 5 focuses on the control synthesis level. The approach uses simulation-based control synthesis to tune control parameters, optimizing satisfaction of a formal requirement set. The formal language used is signal temporal logic (STL) because of the rich set of spatial and temporal requirements that can be expressed in this form.

Previous research has presented quantitative semantics for STL which define the overall system margin of satisfaction as the minimum of the margins of the individual requirement margins. This removes information about the system performance for all but the worst performing requirement. This makes it difficult to compare performance between simulations with different control parameters. For this reason multiSTL is proposed as a multi-dimensional approach similar to that of multiobjective optimization. Rather than try to find a single 'best' set of control parameters, a Pareto-optimal set of control parameters is found. Refinement to a single solution is then carried out via progressively prioritising different requirement margins.

A key advantage of multiSTL is better visualization of performance and tradeoffs between requirements, using the parallel coordinates plot. This information can be fed back to requirements engineers to initiate beneficial requirements changes. For example, relaxing Requirement 1 by 25% will result in much larger margins of satisfaction for Requirements 2 and 3.

The multiSTL control synthesis approach has been applied to the oil system case study, following on from the previous architecture design levels. A nonlinear simulation model has been developed, along with formal STL requirements derived from natural language expressions. The simple gain-based feedback control is tuned via an exhaustive search of a discretised control parameter space. This produces a large number of Pareto-optimal parameter sets, which have been refined to a single parameter set using the multiSTL approach. The final controller is able to satisfy all the formal requirements with reasonable margins, and maintains better thermal and lubrication performance than the baseline fixed-gear oil flow design. This in turn results in reduced maintenance costs and fuel consumption.

6.2 Main contributions

The main contributions of this thesis are highlighted in Section 1.4. They are repeated here as a reminder of the most important aspects of the thesis.

1. The customer-oriented architecture refinement framework. This provides a rapid approach for reducing a large set of potential architectures to a small set of interest to the customer. It only requires a set of customer preference weightings, which can be provided at the outset, eliminating the need for lengthy iterative discussions with the customer. The framework also includes an approach to analyse resilience of architectures to changing customer preferences. This helps engineers to select solutions that are likely to remain good options, even if the customers change their preference weightings as a result of external factors, such as budget cuts. The result is a lower chance of having to rework or modify designs. The SATS tool has been developed to implement the approach in a graphical user interface, which is currently being used by the industrial sponsor of the PhD in real-world architecture design problems.

2. A graph-based topology optimization approach for system architectures. The approach is demonstrated on a turbofan oil system case study, which involves a novel heuristic algorithm for determining similarities between oil chamber flow requirements. The approach allows sensible coupling of oil chambers to shared valves, to reduce the cost of the architecture. The graph-based approach to modeling system architectures and optimizing connections between nodes also has wider applicability to any system with a set of interconnected components.

3. The multiSTL control synthesis framework. This allows the margins of satisfaction for performance requirements specified as Signal Temporal Logic (STL) formulae to be compared on a parallel coordinates plot, highlighting tradeoffs between requirements. The approach gives designers far more information about how the system is performing than if the overall margin of satisfaction is taken as the minimum of the individual requirement margins. This helps with choosing a set of control parameters that achieve an optimal system response, with respect to the priorities of the multiple, often conflicting performance requirements. To demonstrate the approach on a real-world problem, a nonlinear dynamic oil system model is developed and used to perform a multiSTL analysis.

6.3 Future work

Based on the work presented in this thesis, suggested directions for further research are:

1. Investigating the sensitivity of the solutions produced via the customer-oriented architecture refinement to the scores given to the functional means, and to the relational matrix between customer preferences and decision criteria. This could help to highlight to system architects any scores or values which have a large impact on the solution set, allowing them to reconsider whether the score or value has been assigned appropriately.

2. Inverting the customer-oriented architecture refinement process discussed in Chapter 3. This would involve selecting a solution manually and then calculating the set of customer preference weightings required to select this solution. This could be useful in circumstances where engineers prefer a given solution over the solutions produced via SATS. Comparing the actual customer preferences with those required to select the chosen solution could highlight big differences, and give an understanding of why the engineer's preferred solution does not match the customers preferred solution.

3. Replacing the heuristic for populating the oil chamber similarities matrix C_{f_r} in Chapter 4. With extensive high-fidelity simulation of a turbofan system, a more scientific method for correlating the oil chamber flow requirements could be established. This could take into account information such as time constants of the oil chamber metals, rotational velocities, ambient temperatures or physical locations.

4. Genetic algorithm-based control synthesis with multiSTL. The discretisation of the control parameters in the oil system case study resulted in a search space that could be evaluated exhaustively. For larger control synthesis problems this may not be an option, requiring the use of alternative optimization techniques such as multiobjective genetic algorithms (as used in Chapter 3).

5. Feedback between multiple levels. The original multilevel vision for the PhD (Figure 1.1) includes automated feedback loops from lower design levels to higher levels. This could be implemented for example between the architecture topology level (Chapter 4) and the control synthesis level (Chapter 5), as shown in Figure 6.1. In this example an inability to meet the requirements at the control synthesis level results in a feedback to the architecture level, by adding new graph constraints. This triggers a reoptimization and

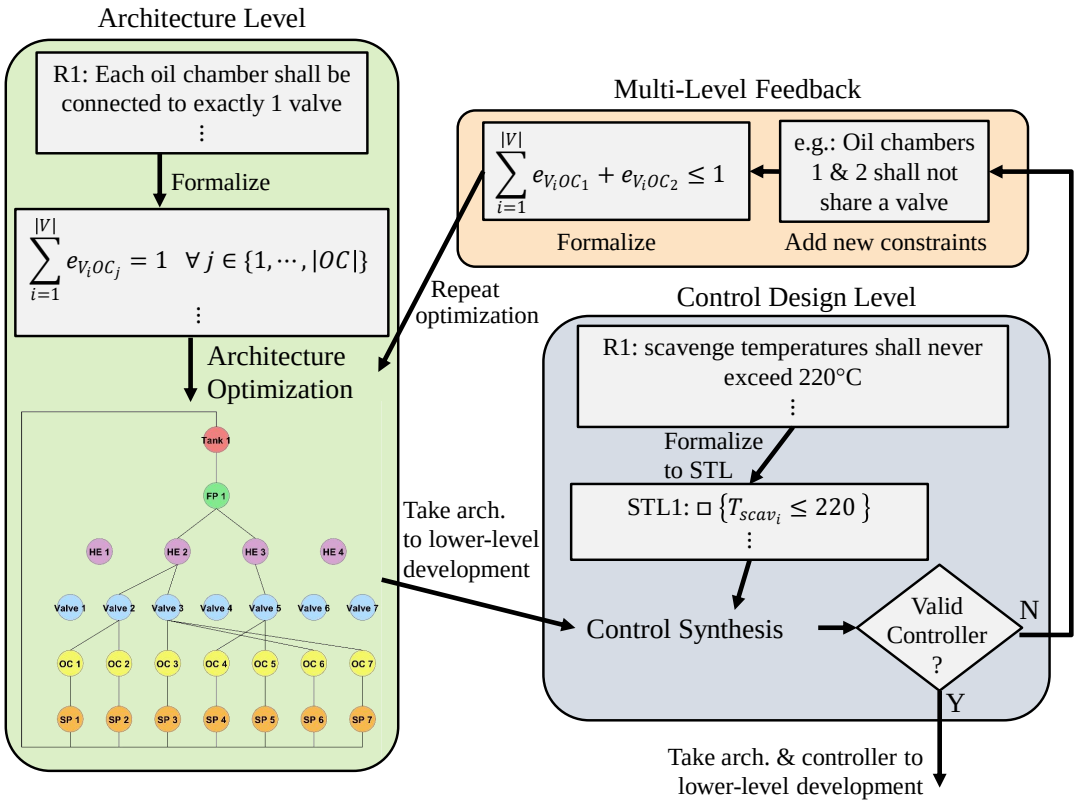


FIGURE 6.1: An example of a multilevel feedback loop. An inability to meet the requirements at the control synthesis level results in a feedback to the architecture level, by adding new graph constraints. This triggers a reoptimization and a second control synthesis exercise.

a second control synthesis exercise. The process repeats until a satisfactory architecture/controller combination are found.

6. This PhD has mainly addressed a multilevel design flow for a single subsystem. This could be linked to a parallel design flow for coupled systems/subsystems. For example, the fuel system which interfaces with the oil system. In the oil system A/G contract there are assumptions relating to the fuel flow and heat exchanger inlet fuel temperature, and a guarantee relating to the heat exchanger outlet fuel temperature. This could be checked for compatability with the A/G contract developed for the fuel system, to show that the two designs will work well together. Co-simulation of the two systems could help to confirm that both designs satisfy their individual contracts and the combined system level contract.

7. Extending the multilevel framework to lower levels. For example, the control inputs W_{feed_i} in Chapter 5 need to be realized at a lower level of fidelity by varying feed pump rate and valve restrictor orifice size. This would involve

the development of higher fidelity models and component level A/G contracts, which could be composed to check compatibility and refinement of the upper level contract.

8. Application to new case studies. The techniques in Chapter 3 have been applied to a nuclear pressurized water reactor EC&I system and turbofan oil system case studies. Further validation of the techniques could be carried out on any system that can be decomposed using a function/means analysis. The flow network topology optimization in Chapter 4 could be applied to more complex turbofan architectures such as geared turbofans, or other fluid flow networks such as smart heating systems. The multiSTL approach of Chapter 5 could be used in any control synthesis case studies where there is a simulation model and multiple performance requirements.

Appendix A

Theory of Assume-Guarantee Contracts

The theory underpinning assume-guarantee contracts is presented in detail in [6, 7, 20]. The aspects that are most relevant for CBD are discussed more briefly here.

Saturated Contracts

A contract $C = (A, G)$ is said to be in *saturated* form if $\neg A \subseteq G$ i.e. $G \vee A = True$ [6], where $\neg$ and $\vee$ are the logical negation and disjunction operators respectively. If this is not the case, then the saturated form of C can be defined by $C' = (A, G')$, where $G' = G \vee \neg A$. The rules below for composition, refinement and conjunction are defined for saturated contracts.

Compatibility and Consistency

A single contract $C = (A, G)$ is said to be *consistent* if $G \neq \emptyset$ and the guarantees do not constrain any of the uncontrolled variables. It is *compatible* if $A \neq \emptyset$ and the assumptions do not constrain any of the controlled variables [6, 20]. In the case of multiple contracts, they are said to be compatible/consistent if their composition (see below) is compatible and consistent [20]. More informally, a set of contracts are said to be compatible if the assumptions of each component are contained in the guarantees of the other components and the environment.

Composition

Composition of two contracts $C_1 = (A_1, G_1)$ and $C_2 = (A_2, G_2)$ in saturated form is defined by (A.1) [6, 7, 20].

$$C_1 \otimes C_2 = (A_{12}, G_{12})$$

where,

$$\begin{aligned} A_{12} &= (A_1 \cap A_2) \cup \neg(G_1 \cap G_2) \\ G_{12} &= G_1 \cap G_2 \end{aligned} \tag{A.1}$$

Composition is needed in CBD to determine an overall system contract, since system designs are built by composing components from the platform library.

Refinement

A contract $C' = (A', G')$ is said to *refine* a contract $C = (A, G)$, written as $C' \preccurlyeq C$, if:

$$A \subseteq A' \quad \text{and} \quad G' \subseteq G \tag{A.2}$$

This effectively amounts to tightening the guarantees and loosening the assumptions. This means that C' can be viewed as a stronger form of contract C [6, 7, 20]. Refinement is a useful relationship when checking if a composition of components satisfies the top-down vertical contract specification, i.e. $(C_1 \otimes C_2 \otimes \cdots \otimes C_n) \preccurlyeq C_s$. Tools have been developed to help with automated verification of temporal contract refinement [109]. When using LTL or STL contracts (see Section 2.4) this checking problem can become quite expensive. In [110] an algorithm is proposed which breaks the system-wide refinement problem into a series of successive refinement checks. In the example given in the paper, the algorithm reduces the computation time from 639 to 123 seconds.

Conjunction

Conjunction of two contracts $C_1 = (A_1, G_1)$ and $C_2 = (A_2, G_2)$ in saturated form is via:

$$C_1 \wedge C_2 = (A_1 \cup A_2, G_1 \cap G_2) \tag{A.3}$$

This is needed in CBD when there are several *viewpoints* of the system [20]. For example, requirements may relate to a functional viewpoint, a performance viewpoint and a reliability viewpoint. Each component will have contracts relating to the multiple viewpoints. To get the overall contract for the component these need to be joined together via conjunction.

Appendix B

Turbofan Oil System Simulation Model

The oil system is modeled by the following equations. Note that to save space (t) has been removed from the time-varying signals. See the equations presented in Section 5.2 for a clearer distinction between constants and signals.

$$
\begin{bmatrix}
\frac{dT_{\text{metal}_1}}{dt} \\
\vdots \\
\frac{dT_{\text{metal}_N}}{dt} \\
\frac{dT_{\text{tank}}}{dt} \\
0 \\
0 \\
\vdots \\
0 \\
0 \\
0
\end{bmatrix}
=
\begin{bmatrix}
\frac{k1_{\omega_1}\left((T_{\text{ambient}_1}+k2_{\omega_1}\,\omega)-T_{\text{metal}_1}\right)\omega^{0.4}}{c_{p_{\text{metal}_1}}\,m_{\text{metal}_1}} - \frac{U_{\text{metal-oil}_1}\,A_{\text{metal-oil}_1}}{c_{p_{\text{metal}_1}}\,m_{\text{metal}_1}}\left(T_{\text{metal}_1}-T_{\text{feed}}\right) \\
\vdots \\
\frac{k1_{\omega_N}\left((T_{\text{ambient}_N}+k2_{\omega_N}\,\omega)-T_{\text{metal}_N}\right)\omega^{0.4}}{c_{p_{\text{metal}_N}}\,m_{\text{metal}_N}} - \frac{U_{\text{metal-oil}_N}\,A_{\text{metal-oil}_N}}{c_{p_{\text{metal}_N}}\,m_{\text{metal}_N}}\left(T_{\text{metal}_N}-T_{\text{feed}}\right) \\
\frac{W_{\text{oil}}}{m_{\text{tank}}}\left(T_{\text{scav}}-T_{\text{tank}}\right) - \frac{T_{\text{tank}}}{R_{\text{tank}}\,c_{p_{\text{oil}}}\,m_{\text{tank}}} \\
-T_{\text{feed}} + T_{\text{tank}} - \frac{U_{\text{oil-fuel}}\,A_{\text{oil-fuel}}}{W_{\text{oil}}c_{p_{\text{oil}}}}\,F_c\,\frac{(T_{\text{tank}}-T_{\text{fuel}_{\text{out}}})-(T_{\text{feed}}-T_{\text{fuel}_{\text{in}}})}{\text{Log}\left(\frac{T_{\text{tank}}-T_{\text{fuel}_{\text{out}}}}{T_{\text{feed}}-T_{\text{fuel}_{\text{in}}}}\right)} \\
-T_{\text{scav}_1} + T_{\text{feed}} + \frac{U_{\text{metal-oil}_1}\,A_{\text{metal-oil}_1}}{W_{\text{feed}_1}\,c_{p_{\text{oil}_1}}}\left(T_{\text{metal}_1}-T_{\text{feed}}\right) \\
\vdots \\
-T_{\text{scav}_N} + T_{\text{feed}} + \frac{U_{\text{metal-oil}_N}\,A_{\text{metal-oil}_N}}{W_{\text{feed}_N}\,c_{p_{\text{oil}_N}}}\left(T_{\text{metal}_N}-T_{\text{feed}}\right) \\
-T_{\text{scav}} + \frac{\sum_{i=1}^{N}\,T_{\text{scav}_i}\,W_{\text{scav}_i}}{W_{\text{oil}}} \\
-T_{\text{fuel}_{\text{out}}} + T_{\text{fuel}_{\text{in}}} + \frac{U_{\text{oil-fuel}}\,A_{\text{oil-fuel}}}{W_{\text{fuel}}c_{p_{\text{fuel}}}}\,F_c\,\frac{(T_{\text{tank}}-T_{\text{fuel}_{\text{out}}})-(T_{\text{feed}}-T_{\text{fuel}_{\text{in}}})}{\text{Log}\left(\frac{T_{\text{tank}}-T_{\text{fuel}_{\text{out}}}}{T_{\text{feed}}-T_{\text{fuel}_{\text{in}}}}\right)}
\end{bmatrix}
$$

These equations are implemented in Simulink [93] as shown in Figure B.1 to Figure B.8.

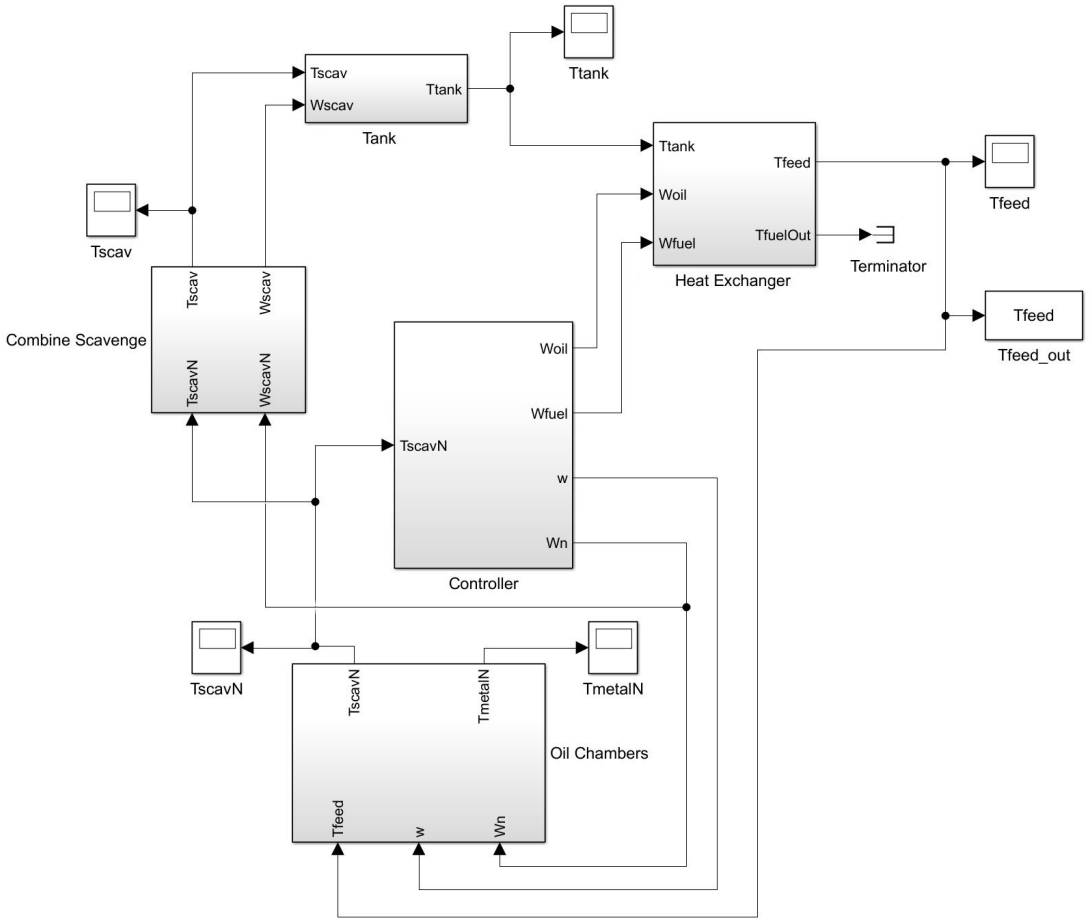


FIGURE B.1: Simulink top-level block diagram.

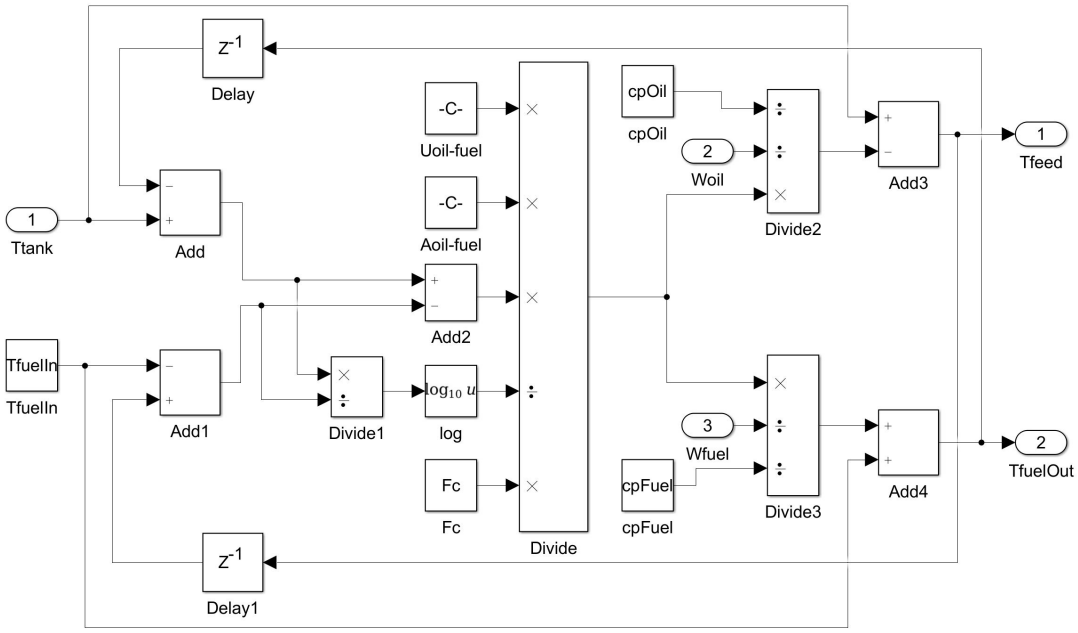


FIGURE B.2: Simulink heat exchanger block diagram.

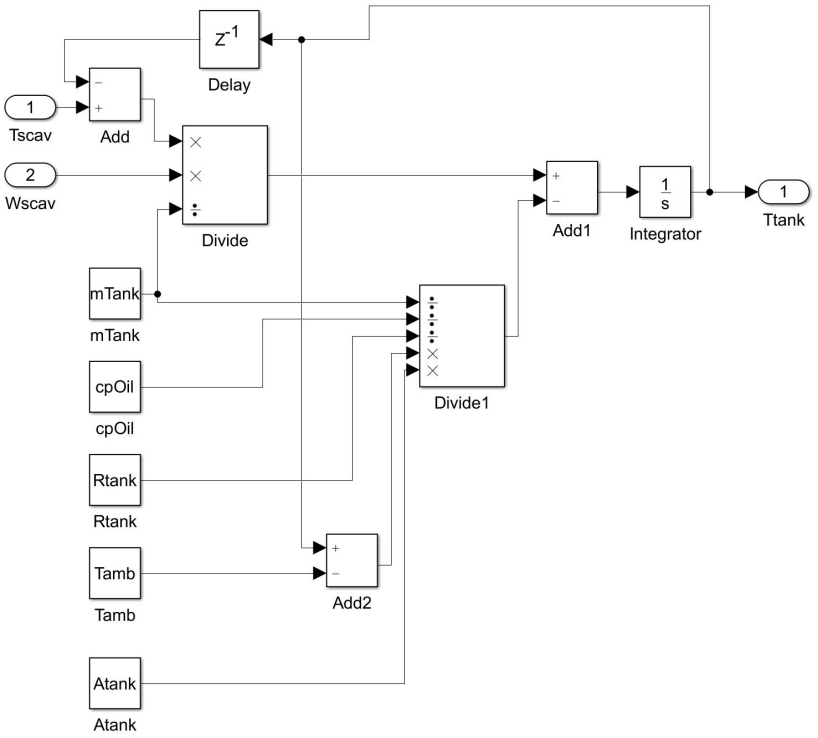


FIGURE B.3: Simulink tank block diagram.

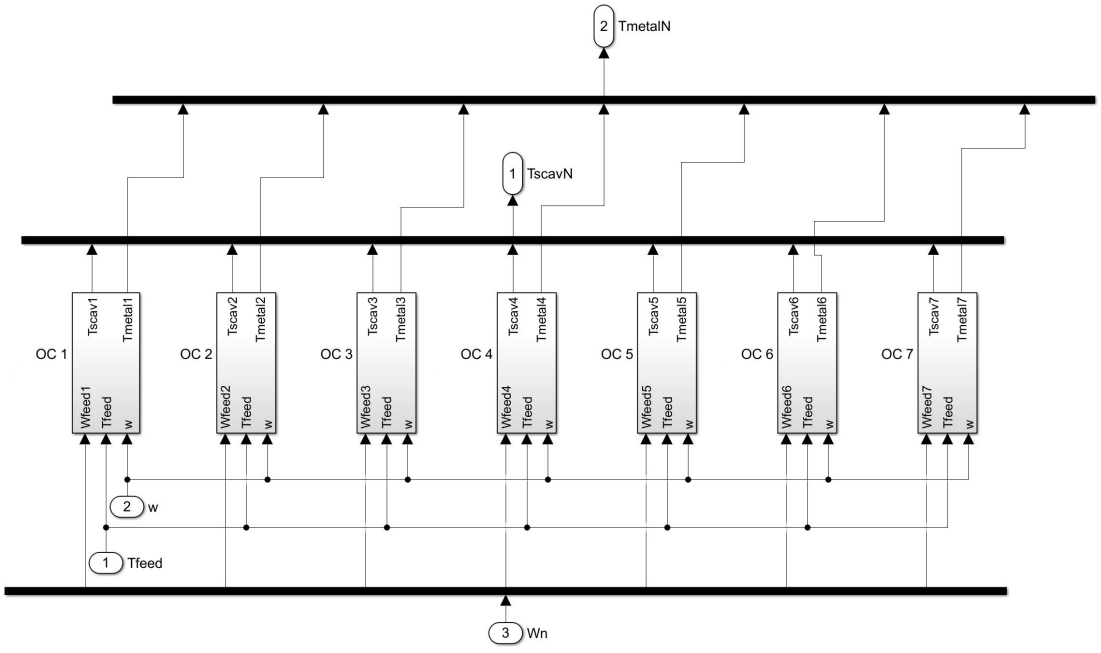


FIGURE B.4: Simulink oil chambers block diagram.

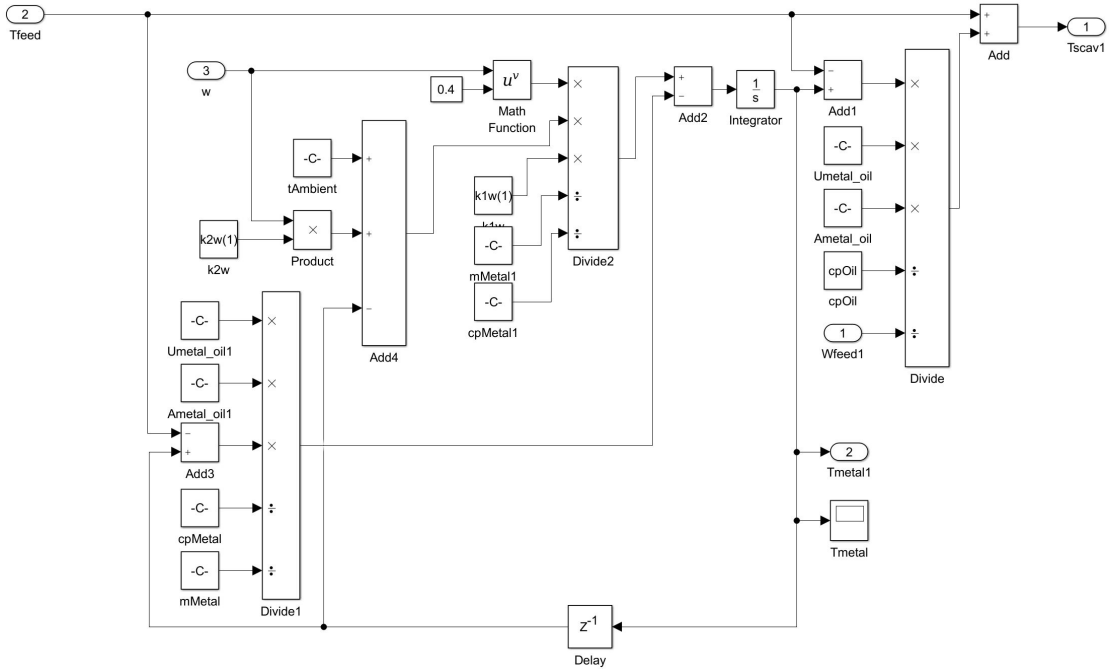


FIGURE B.5: Simulink individual oil chamber block diagram.

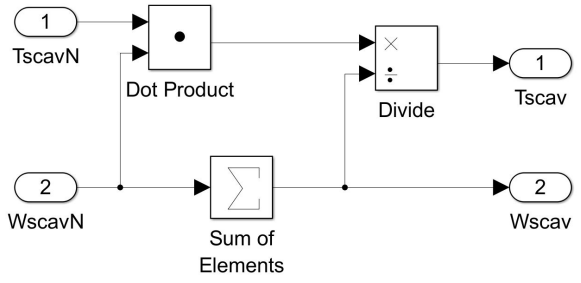


FIGURE B.6: Simulink scavenge combine block diagram.

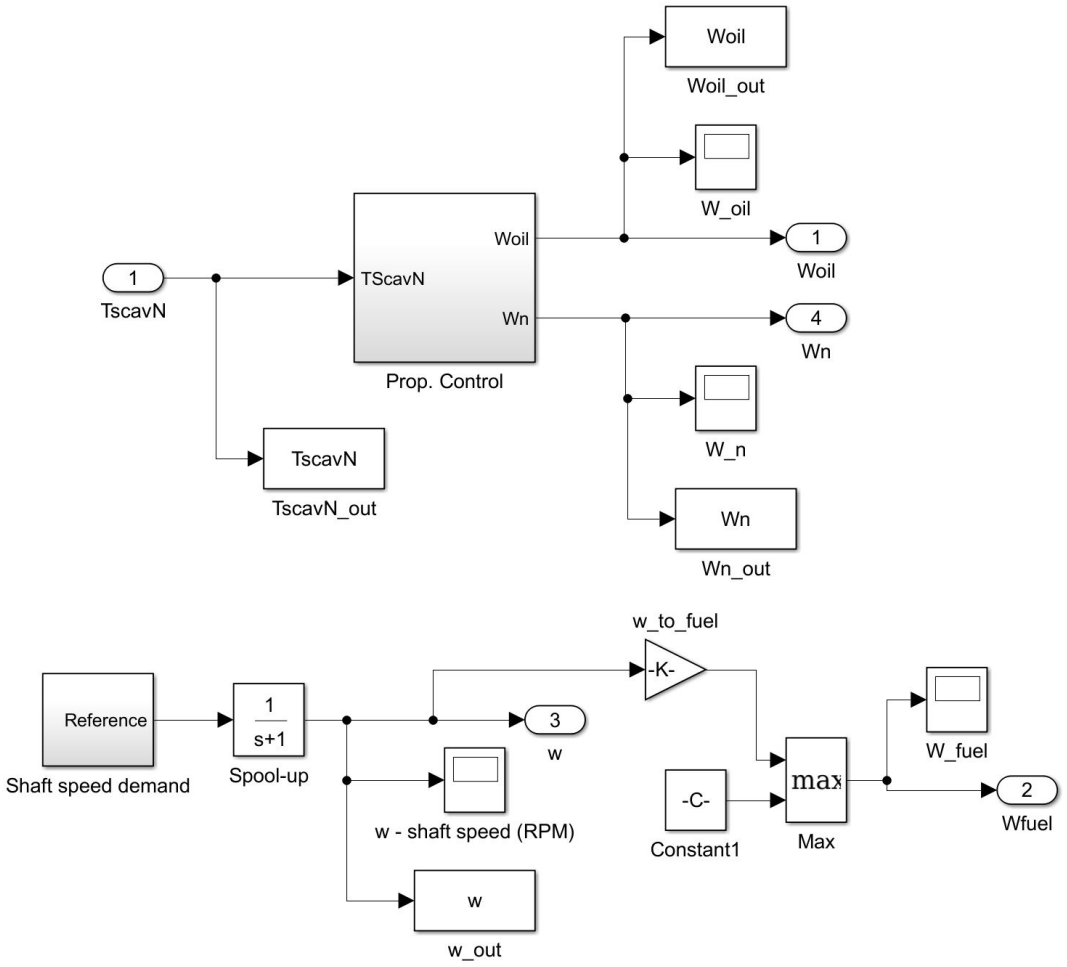


FIGURE B.7: Simulink control block diagram.

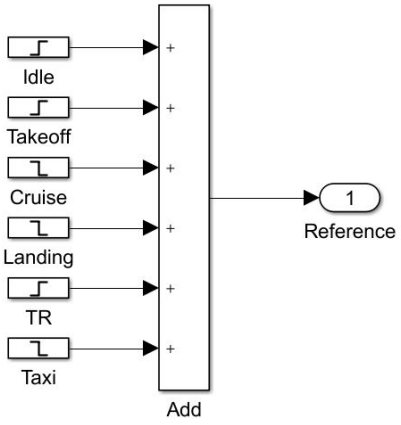


FIGURE B.8: Simulink shaft speed references block diagram.

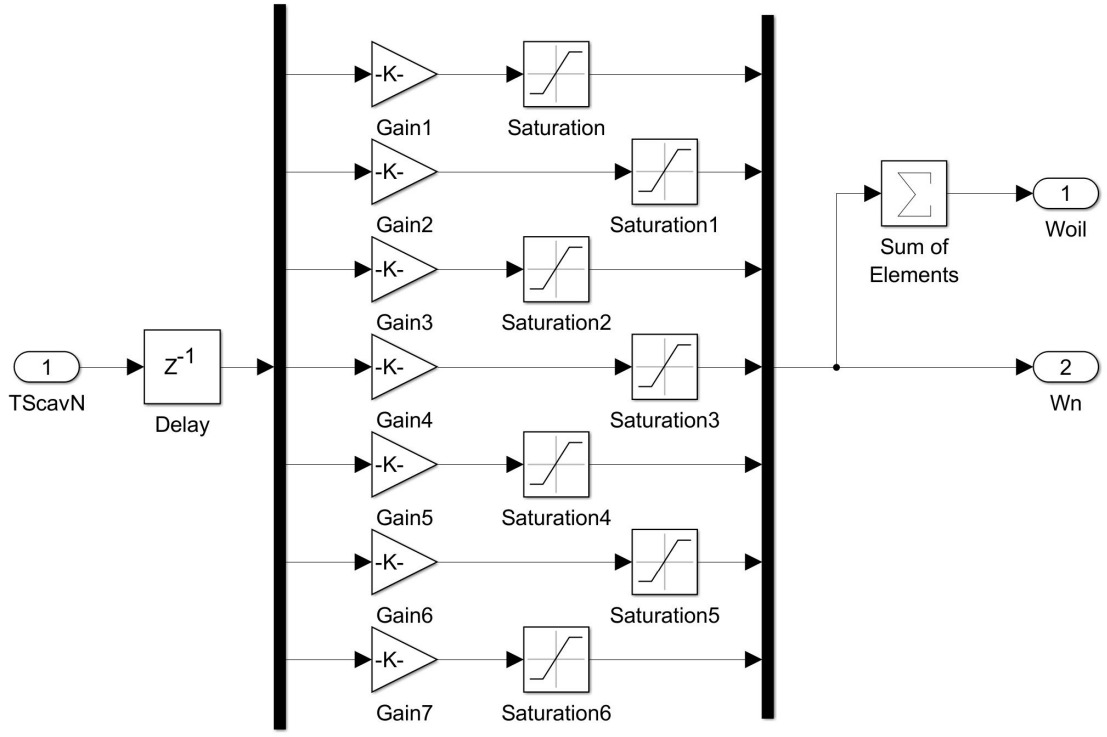


FIGURE B.9: Simulink proportional control block diagram.

The MATLAB code used to set up the parameters for the Simulink model is shown in code block B.1.

```
clear all

%% General variables
cpOil = 2.3; % kj/kg K

%% Tank Variables
mTank = 20;      % kg
k = 0.03;        % W/m.K
L = 0.02;        % m
Atank = 3;       % m^2
Rtank = L/k;         % m^2K/W
Tamb = 293.7;    % K - room temp

%% Heat Exchanger Variables
Uoil_fuel = 1;
Aoil_fuel = 0.75;
Fc = 0.85;
cpFuel = 2.01; % kJ/kg K
TfuelIn = Tamb-40; % assume cold for hi altitude
TfuelOutInitial = TfuelIn+20; % for initial conditions
TfeedInitial = Tamb-30; % for initial conditions
w_fuel_min = 0.1;

%% Oil Chamber Variables
TambOC = [270 380 480 700 600 330 270]; % K
Ametal_oil = [0.15 0.15 0.22 0.15 0.15 0.10 0.10];
Umetal_oil = [1 1 1 1 1 1 1];
```

```
cpMetal = [0.5 0.5 0.5 0.5 0.5 0.5 0.5]; % kJ/kg K
mMetal = [100 100 150 100 100 60 60]; % kg
k1w = [0.1 0.1 0.1 0.1 0.1 0.1 0.1];
k2w = [0.03 0.03 0.035 0.03 0.03 0.025 0.025];

%% Valve variables
w_feed_i_min = 0.07; % lower saturation limit
w_feed_i_max = 0.4; % upper saturation limit
```

Listing B.1: Simulation parameter definitions

Appendix C

Oil System Assume/Guarantee Contract Composition

Thermal Requirements
R1: Oil temperature shall not exceed 220°C at the scavenging side at any time.
R2: All oil flows from the oil system to the oil chambers shall have a temperature in the range 10°C to 120°C for all operating conditions above idle.

Flow Requirements
R3: All individual oil flows from the oil system to the oil chambers shall have a maximum flow rate of 0.4 kg s^{-1} at any time.
R4: The minimum oil feed flow to the chambers shall be 0.07 kg s^{-1} at any time.
R5: The oil system shall have a scavenge ratio of 1.5 to the oil flowrate supplied to each chamber, to ensure no dangerous build-up of pressure in the oil chambers.

In Section C.1 to Section C.3 the A/G contracts for these subsystems are derived separately, before being combined via the rules of contract composition (see Appendix A) to check compatibility in Section C.5. This is done here to demonstrate how component-level A/G contracts can be composed to give a system-level contract, which would be useful if different components or subsystems were to be produced by different teams or supplier companies.

Note that in this chapter temporal logic symbols $\bigcirc$ (next), $\mathcal{U}$ (until), $\Box$ (always) and $\Diamond$ (eventually) are used in the assumptions and guarantees. Further elaboration on temporal logic is given in Section 2.4.1.

C.1 Heat Exchanger Contracts

Fluid Temperature Contract

$$C_{\text{thermal}_{\text{HE}}} = \begin{cases} A_{\text{thermal}_{\text{HE}}} = \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C})) \\ G_{\text{thermal}_{\text{HE}}} = \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ \qquad\qquad \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C})) \end{cases} \tag{C.1}$$

Note that the assumptions and guarantees highlighted in red in (C.1) do not correspond to requirements R1 and R2. This comes from the fact that the oil system will have to transfer heat to the fuel system and the rate of transfer will be dependent on both T_{tank} and $T_{\text{fuel}_{\text{in}}}$. The fuel tank temperature can drop as low as $-35°$C when flying at max altitude (outside air temp $\approx -60°$C) and reach as high as 55°C when starting on the ground on a hot day [91]. When leaving the heat exchanger the fuel temperature needs to be above 0°C (to avoid ice forming in the fuel filter) and below 165°C (to avoid degradation) [91].

Fluid Flow Contract

$$C_{\text{flow}_{\text{HE}}} = \begin{cases} A_{\text{flow}_{\text{HE}}} = \square\ \big((0.49\,\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\,\text{kgs}^{-1}) \\ \qquad\qquad \wedge (0.1\,\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\,\text{kgs}^{-1})\big) \\ G_{\text{flow}_{\text{HE}}} = \emptyset \end{cases} \tag{C.2}$$

Note that the assumptions highlighted in red in (C.2) are not contained in R3-R5. These come from the fact that the heat transfer is related to W_{fuel} as well as W_{oil}. Typical fuel flow values for a turbofan engine are around 0.5 kg s^{-1} [108] therefore upper/lower limits are set as 0.8 kg s^{-1} and 0.1 kg s^{-1}. The flow guarantees are empty, since the heat exchanger has no control over any of the flows entering/leaving the subsystem.

Alphabet Equalization

Consider the following alphabets for the two contracts:

$$\begin{aligned} \Sigma_{\text{thermal}_{\text{HE}}} &= \{T_{\text{tank}}, T_{\text{fuel}_{\text{in}}}, T_{\text{feed}}, T_{\text{fuel}_{\text{out}}}\} \\ \Sigma_{\text{flow}_{\text{HE}}} &= \{W_{\text{oil}}, W_{\text{fuel}}, W_{\text{feed}_i}\} \end{aligned} \tag{C.3}$$

To combine the contracts we need a common alphabet as shown in (C.4).

$$\Sigma = \{T_{\text{tank}}, T_{\text{fuel}_{\text{in}}}, T_{\text{feed}}, T_{\text{fuel}_{\text{out}}}, W_{\text{oil}}, W_{\text{fuel}}, W_{\text{feed}_i}\} \tag{C.4}$$

This is because of the $\cap$ operator which is applied in composition and conjunction operations. If the contracts do not have a common alphabet then the intersection will yield $\emptyset$. The alphabet-equalized contracts are shown in equations (C.5) and (C.6). The added assumptions/guarantees highlighted in red do not change the individual contracts. They extend the contracts to include the new variables whilst offering no assumptions/guarantees about their behaviour. Therefore the contracts effectively remain the same.

$$C^*_{\text{thermal}_{\text{HE}}} = \begin{cases} A^*_{\text{thermal}_{\text{HE}}} = \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ \qquad\qquad \wedge (W_{\text{oil}} \in \mathbb{R}) \wedge (W_{\text{fuel}} \in \mathbb{R})) \\ G^*_{\text{thermal}_{\text{HE}}} = \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ \qquad\qquad \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C})) \end{cases} \tag{C.5}$$

$$C^*_{\text{flow}_{\text{HE}}} = \begin{cases} A^*_{\text{flow}_{\text{HE}}} = \square\ \big((0.49\,\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\,\text{kgs}^{-1}) \\ \qquad\qquad \wedge (0.1\,\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\,\text{kgs}^{-1}) \\ \qquad\qquad \wedge (T_{\text{tank}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{in}}} \in \mathbb{R})) \\ G^*_{\text{flow}_{\text{HE}}} = \square\ ((T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R})) \end{cases} \tag{C.6}$$

Conjunction of the thermal and flow viewpoints

In order to get an overall contract for the heat exchanger, the thermal and fluid flow contracts need to be combined via the conjunction operator defined as:

$$C_{\text{HE}} = C^*_{\text{thermal}_{\text{HE}}} \wedge C^*_{\text{flow}_{\text{HE}}} = (A^*_{\text{thermal}_{\text{HE}}} \cup A^*_{\text{flow}_{\text{HE}}}, G^*_{\text{thermal}_{\text{HE}}} \cap G^*_{\text{flow}_{\text{HE}}}) \tag{C.7}$$

Note that the assumptions are joined via the union operator. When using the alphabet-equalized assumptions defined in equations (C.5) and (C.6) this union operator yields an assumption:

$$A_{\text{HE}} = \square\ ((T_{\text{tank}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{in}}} \in \mathbb{R}) \wedge (W_{\text{oil}} \in \mathbb{R}) \wedge (W_{\text{fuel}} \in \mathbb{R})) \tag{C.8}$$

This is far too broad an assumption as it means the system has to be designed to fit any possible environment of heat exchanger flow and temperature values. Therefore we need to use the modified equations (C.9) and (C.10), which only perform alphabet equalization on the guarantees.

$$C^*_{\text{thermal}_{\text{HE}}} = \begin{cases} A^*_{\text{thermal}_{\text{HE}}} = \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C})) \\ G^*_{\text{thermal}_{\text{HE}}} = \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ \qquad\qquad \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C})) \end{cases} \tag{C.9}$$

$$C^*_{\text{flow}_{\text{HE}}} = \begin{cases} A^*_{\text{flow}_{\text{HE}}} = \square\ \big((0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ \qquad\qquad \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1})\big) \\ G^*_{\text{flow}_{\text{HE}}} = \square\ ((T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R})) \end{cases} \tag{C.10}$$

This allows a more reasonable set of assumptions to be generated:

$$\begin{aligned} A_{\text{HE}} =& A^*_{\text{thermal}_{\text{HE}}} \cup A^*_{\text{flow}_{\text{HE}}} \\ =& \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ & \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1}))\big) \end{aligned} \tag{C.11}$$

The corresponding guarantees for the contract joined via conjunction are:

$$\begin{aligned} G_{\text{HE}} &= G'^*_{\text{thermal}_{\text{HE}}} \cap G'^*_{\text{flow}_{\text{HE}}} \\ &= \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C})) \end{aligned} \tag{C.12}$$

C.2 Oil Chambers/Valves Contracts

This section will present the oil chambers/valves contracts more quickly than in section Section C.1 as the principle alphabet equalization has already been explained.

Fluid Temperature Contract

The thermal contract based on requirements R1 and R2 is:

$$C_{\text{thermal}_{\text{OCV}}} = \begin{cases} A_{\text{thermal}_{\text{OCV}}} = \square\,(\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ G_{\text{thermal}_{\text{OCV}}} = \square\,(T_{\text{scav}_i} \leq 220°\text{C}) \end{cases} \tag{C.13}$$

Fluid Flow Contract

The flow contract based on requirements R3 and R4 is given by:

$$C_{\text{flow}_{\text{OCV}}} = \begin{cases} A_{\text{flow}_{\text{OCV}}} = \square\,(0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ G_{\text{flow}_{\text{OCV}}} = \square\,(0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \end{cases} \tag{C.14}$$

To combine $C_{\text{thermal}_{\text{OCV}}}$ and $C_{\text{flow}_{\text{OCV}}}$ these need to be alphabet-equalized as highlighted in red:

$$C^*_{\text{thermal}_{\text{OCV}}} = \begin{cases} A^*_{\text{thermal}_{\text{OCV}}} = \square\,(\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ G^*_{\text{thermal}_{\text{OCV}}} = \square\,((T_{\text{scav}_i} \leq 220°\text{C}) \wedge (W_{\text{feed}_i} \in \mathbb{R})) \end{cases} \tag{C.15}$$

$$C^*_{\text{flow}_{\text{OCV}}} = \begin{cases} A^*_{\text{flow}_{\text{OCV}}} = \square\,(0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ G^*_{\text{flow}_{\text{OCV}}} = \square\,\big((0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \wedge (T_{\text{scav}_i} \in \mathbb{R})\big) \end{cases} \tag{C.16}$$

Conjunction of the thermal and flow viewpoints

The assumptions of the overall bearing chamber contract can be calculated via the union operator:

$$\begin{aligned} A_{\text{OCV}} =& A^*_{\text{thermal}_{\text{OCV}}} \cup A^*_{\text{flow}_{\text{OCV}}} \\ =& \square\,((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ & \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1})\big) \end{aligned} \tag{C.17}$$

The guarantees come from the intersection operator:

$$\begin{aligned} G_{\text{OCV}} &= G^*_{\text{thermal}_{\text{OCV}}} \cap G^*_{\text{flow}_{\text{OCV}}} \\ &= \square\,\big((T_{\text{scav}_i} \leq 220°\text{C}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1})\big) \end{aligned} \tag{C.18}$$

C.3 Pumping and Storage System Contracts

Fluid Temperature Contract

The thermal contract for the pumping and storage system is very simple since we assume that no heat is generated and that all elements are thermally insulated.

Therefore the maximum temperature will be that of T_{scav_i}.

$$C_{\text{thermal}_{\text{PS}}} = \begin{cases} A_{\text{thermal}_{\text{PS}}} = \square\,(T_{\text{scav}_i} \leq 220°\text{C}) \\ G_{\text{thermal}_{\text{PS}}} = \square\,(T_{\text{tank}} \leq 220°\text{C}) \end{cases} \tag{C.19}$$

Fluid Flow Contract

The pumping and storage unit is responsible for two sets of flows:

1. The oil feed flow (via heat exchanger) to the variable restrictor valves - provided by the feed pump.
2. The scavenge oil flows provided by the individual scavenge pumps. This needs to satisfy the flow requirement R5.

Therefore the flow contract for the pumping and storage system will offer guarantees regarding both W_{scav_i} and W_{oil}.

$$C_{\text{flow}_{\text{PS}}} = \begin{cases} A_{\text{flow}_{\text{PS}}} = \square\,(0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \\ G_{\text{flow}_{\text{PS}}} = \square\,\big((0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ \qquad\qquad \wedge (0.105\text{kgs}^{-1} \leq W_{\text{scav}_i} \leq 0.525\text{kgs}^{-1})\big) \end{cases} \tag{C.20}$$

Conjunction of the thermal and flow viewpoints

Following an alphabet equalization process (which has not been shown), the assumptions of the overall pumping and storage system contract can be calculated via the union operator:

$$\begin{aligned} A_{\text{PS}} &= A^*_{\text{thermal}_{\text{PS}}} \cup A^*_{\text{flow}_{\text{PS}}} \\ &= \square\,\big((T_{\text{scav}_i} \leq 220°\text{C}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1})\big) \end{aligned} \tag{C.21}$$

and the guarantees come from the intersection (note that without the alphabet equalization step explicitly shown here ($G^*_{\text{thermal}_{\text{PS}}}$ and $G^*_{\text{flow}_{\text{PS}}}$), the intersection looks like a union operation).

$$\begin{aligned} G_{\text{PS}} =& G^*_{\text{thermal}_{\text{PS}}} \cap G^*_{\text{flow}_{\text{PS}}} \\ =& \square\,\big((T_{\text{tank}} \leq 220°\text{C}) \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ & \wedge (0.105\text{kgs}^{-1} \leq W_{\text{scav}_i} \leq 0.525\text{kgs}^{-1})\big) \end{aligned} \tag{C.22}$$

C.4 A Note on the Control System

In Figure 5.4 the control system is shown as a separate block with links to:

1. Oil chambers/valves - sensing of scavenge temperatures and actuation of valves.
2. Pumping & storage system - actuation of pumps for modulating the feed and scavenge flows.

Therefore it is the controller designed in Section 5.4 which is responsible for the guarantees on these flows/temperatures.

C.5 Combination via Composition

To check that the contracts derived in sections Section C.1 to Section C.3 are compatible, we need to calculate the composition of the contracts which can be calculated iteratively via:

$$\begin{aligned} C_{\text{HE-OCV}} &= C_{\text{HE}} \otimes C_{\text{OCV}} \\ C_{\text{sys}} &= C_{\text{PS}} \otimes C_{\text{HE-OCV}} \end{aligned} \tag{C.23}$$

First Iteration - Combining the HE and OCV Contracts

The first composition $C_{\text{HE-OCV}}$ can be calculated via:

$$C_{\text{HE}} \otimes C_{\text{OCV}} = (A_{\text{HE-OCV}}, G_{\text{HE-OCV}})$$

where

$$\begin{aligned} A_{\text{HE-OCV}} &= (A_{\text{HE}} \cap A_{\text{OCV}}) \cup \neg(G_{\text{HE}} \cap G_{\text{OCV}}) \\ G_{\text{HE-OCV}} &= G_{\text{HE}} \cap G_{\text{OCV}} \end{aligned} \tag{C.24}$$

First we need the alphabet-equalized contracts. Note that since the assumptions consist of the intersection of the individual assumptions and the union of the guarantees, the assumptions need to be alphabet equalized for all variables contained

in C_{HE} and C_{OCV}.

$$C^*_{\text{HE}} = \begin{cases} A^*_{\text{HE}} = \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ \qquad \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1}) \\ \qquad \wedge (T_{\text{feed}} \in \mathbb{R}) \wedge (W_{\text{feed}_i} \in \mathbb{R}) \wedge (T_{\text{scav}_i} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R})) \\ G^*_{\text{HE}} = \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C}) \\ \qquad \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \wedge (T_{\text{scav}_i} \in \mathbb{R})) \end{cases} \tag{C.25}$$

$$C^*_{\text{OCV}} = \begin{cases} A^*_{\text{OCV}} = \square\ ((10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \\ \qquad \wedge (T_{\text{tank}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{in}}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \\ \qquad \wedge (W_{\text{oil}} \in \mathbb{R}) \wedge (W_{\text{fuel}} \in \mathbb{R}) \wedge (T_{\text{scav}_i} \in \mathbb{R})) \\ G^*_{\text{OCV}} = \square\ ((T_{\text{scav}_i} \leq 220°\text{C}) \wedge (T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \wedge (W_{\text{feed}_i} \in \mathbb{R})) \end{cases} \tag{C.26}$$

Then we can calculate the guarantees:

$$\begin{aligned} G_{\text{HE-OCV}} =& G^*_{\text{HE}} \cap G^*_{\text{OCV}} \\ =& \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (T_{\text{scav}_i} \leq 220°\text{C}) \\ & \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1})) \end{aligned} \tag{C.27}$$

For the calculation of the assumptions we need $\neg G^*_{\text{HE}} \cap G^*_{\text{OCV}}$ i.e.:

$$\begin{aligned} \neg G_{\text{HE-OCV}} =& \Diamond\ ((\neg\text{Idle} \wedge T_{\text{feed}} < 10°\text{C}) \vee (\neg\text{Idle} \wedge T_{\text{feed}} > 120°\text{C}) \vee (T_{\text{scav}_i} > 220°\text{C}) \\ & \vee (T_{\text{fuel}_{\text{out}}} < 10°\text{C}) \vee (T_{\text{fuel}_{\text{out}}} > 120°\text{C}) \vee (W_{\text{feed}_i} < 0.07\text{kgs}^{-1}) \\ & \vee (W_{\text{feed}_i} > 0.4\text{kgs}^{-1})) \end{aligned} \tag{C.28}$$

And the intersection of the assumptions:

$$\begin{aligned} A^*_{\text{HE}} \cap A^*_{\text{OCV}} =& \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ & \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1}) \\ & \wedge (\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (T_{\text{scav}_i} \in \mathbb{R}) \\ & \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1})) \end{aligned} \tag{C.29}$$

Notice that when the assumptions are calculated via $A_{\text{HE-OCV}} = (A^*_{\text{HE}} \cap A^*_{\text{OCV}}) \cup \neg G_{\text{HE-OCV}}$ the constraints highlighted in green, blue, orange and magenta in (C.28) and (C.29) effectively state that $\square\,((T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \wedge (W_{\text{feed}_i} \in$

$\mathbb{R}) \wedge (T_{\text{scav}_i} \in \mathbb{R}))$. So overall the assumptions reduce to:

$$\begin{aligned} A_{\text{HE-OCV}} = \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1}))) \end{aligned} \tag{C.30}$$

Second Iteration - Combining the PS Contract With the HE-OCV Composition

As stated in (C.23) we have $C_{\text{sys}} = C_{\text{PS}} \otimes C_{\text{HE-OCV}}$. Again we need the alphabet equalized contracts:

$$C^*_{\text{HE-OCV}} = \begin{cases} A^*_{\text{HE-OCV}} = \square\ ((T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ \qquad \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ \qquad \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1}) \\ \qquad \wedge (T_{\text{scav}_i} \in \mathbb{R}) \wedge (T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \\ \qquad \wedge (W_{\text{feed}_i} \in \mathbb{R}) \wedge (W_{\text{scav}_i} \in \mathbb{R})) \\ G^*_{\text{HE-OCV}} = \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \\ \qquad \wedge (T_{\text{scav}_i} \leq 220°\text{C}) \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C}) \\ \qquad \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \\ \qquad \wedge (T_{\text{tank}} \in \mathbb{R}) \wedge (W_{\text{oil}} \in \mathbb{R}) \wedge (W_{\text{scav}_i} \in \mathbb{R})) \end{cases} \tag{C.31}$$

$$C^*_{\text{PS}} = \begin{cases} A^*_{\text{PS}} = \square\ \big((T_{\text{scav}_i} \leq 220°\text{C}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \\ \qquad \wedge (T_{\text{tank}} \in \mathbb{R}) \wedge\ (T_{\text{fuel}_{\text{in}}} \in \mathbb{R}) \wedge (T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \\ \qquad \wedge (W_{\text{fuel}} \in \mathbb{R}) \wedge (W_{\text{oil}} \in \mathbb{R}) \wedge (W_{\text{scav}_i} \in \mathbb{R})) \\ G^*_{\text{PS}} = \square\ \big((T_{\text{tank}} \leq 220°\text{C}) \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ \qquad \wedge (0.105\text{kgs}^{-1} \leq W_{\text{scav}_i} \leq 0.525\text{kgs}^{-1}) \wedge (T_{\text{feed}} \in \mathbb{R}) \\ \qquad \wedge (T_{\text{scav}_i} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \wedge (W_{\text{feed}_i} \in \mathbb{R})) \end{cases} \tag{C.32}$$

Then we can calculate the guarantees:

$$\begin{aligned} G_{\text{sys}} &= G^*_{\text{HE-OCV}} \cap G^*_{\text{PS}} \\ &= \square\,((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (T_{\text{scav}_i} \leq 220°\text{C}) \wedge (T_{\text{tank}} \leq 220°\text{C}) \\ &\quad \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C}) \wedge (0.105\text{kgs}^{-1} \leq f_{\text{scav}} \leq 0.525\text{kgs}^{-1}) \\ &\quad \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1})) \end{aligned} \tag{C.33}$$

For the calculation of the assumptions we need $\neg G^*_{\text{HE-OCV}} \cap G^*_{\text{PS}}$ i.e.:

$$\begin{aligned} \neg G_{\text{sys}} = \Diamond\,((\neg\text{Idle} \wedge T_{\text{feed}} < 10°\text{C}) \vee (\neg\text{Idle} \wedge T_{\text{feed}} > 120°\text{C}) \vee (T_{\text{scav}_i} > 220°\text{C}) \\ \vee (T_{\text{tank}} > 220°\text{C}) \vee (T_{\text{fuel}_{\text{out}}} < 10°\text{C}) \vee (T_{\text{fuel}_{\text{out}}} > 120°\text{C}) \\ \vee (W_{\text{scav}_i} < 0.105) \vee (W_{\text{scav}_i} > 0.525\text{kgs}^{-1}) \vee (W_{\text{oil}} < 0.49\text{kgs}^{-1}) \\ \vee (W_{\text{oil}} > 2.8\text{kgs}^{-1}) \vee (W_{\text{feed}_i} < 0.07\text{kgs}^{-1}) \vee (W_{\text{feed}_i} > 0.4\text{kgs}^{-1}))) \end{aligned} \tag{C.34}$$

And the intersection of the assumptions:

$$\begin{aligned} A^*_{\text{HE-OCV}} \cap A^*_{\text{PS}} = \square\,((T_{\text{scav}_i} \leq 220°\text{C}) \wedge (T_{\text{tank}} \leq 220°\text{C}) \wedge (-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \\ \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1}) \\ \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1}) \wedge (T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \\ \wedge (W_{\text{scav}_i} \in \mathbb{R})) \end{aligned} \tag{C.35}$$

Notice that when the assumptions are calculated via $A_{\text{sys}} = (A *_{\text{HE-OCV}} \cap A^*_{\text{PS}}) \cup \neg G_{\text{sys}}$ the constraints highlighted in orange, magenta, red, brown, violet, blue and green in (C.34) and (C.35) effectively state that $\square\,((T_{\text{scav}_i} \in \mathbb{R}) \wedge (T_{\text{tank}} \in \mathbb{R}) \wedge (T_{\text{feed}} \in \mathbb{R}) \wedge (T_{\text{fuel}_{\text{out}}} \in \mathbb{R}) \wedge (W_{\text{scav}_i} \in \mathbb{R}) \wedge (W_{\text{oil}} \in \mathbb{R}) \wedge (W_{\text{feed}_i} \in \mathbb{R}))$. So overall the assumptions reduce to:

$$A_{\text{sys}} = \square\,\big((-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1})\big) \tag{C.36}$$

So the final contract is given by:

$$C_{\text{sys}} = \begin{cases} A_{\text{sys}} = \square\ \left((-35°\text{C} \leq T_{\text{fuel}_{\text{in}}} \leq 55°\text{C}) \wedge (0.1\text{kgs}^{-1} \leq W_{\text{fuel}} \leq 0.8\text{kgs}^{-1})\right) \\ \\ G_{\text{sys}} = \square\ ((\neg\text{Idle} \implies 10°\text{C} \leq T_{\text{feed}} \leq 120°\text{C}) \wedge (T_{\text{scav}_i} \leq 220°\text{C}) \\ \qquad \wedge (T_{\text{tank}} \leq 220°\text{C}) \wedge (0°\text{C} \leq T_{\text{fuel}_{\text{out}}} \leq 165°\text{C}) \\ \qquad \wedge (0.105\text{kgs}^{-1} \leq W_{\text{scav}_i} \leq 0.525\text{kgs}^{-1}) \\ \qquad \wedge (0.49\text{kgs}^{-1} \leq W_{\text{oil}} \leq 2.8\text{kgs}^{-1}) \\ \qquad \wedge (0.07\text{kgs}^{-1} \leq W_{\text{feed}_i} \leq 0.4\text{kgs}^{-1})) \end{cases} \tag{C.37}$$

This makes sense since the only variables we are making assumptions on are those that come from outside the modulated oil system. At the same time, all of the requirements R1-R5 are contained in the guarantees. This is what we expected and if we were to find anything different with the final C_{sys} it would indicate that there is some sort of integration issue. However, since $C_{\text{HE}} \otimes C_{\text{OCV}} \otimes C_{\text{PS}}$ satisfies our specification, then any set of subsytems which satisfy the individual contracts will satisfy the whole-system specification. This means we can take away the individual contracts and use them as specifications for design of the different subsystems.

Note that in Section 5.4 a simplified version of this is used without the sub-formulae for W_{oil} and T_{tank}. This is because the requirements on W_{oil} are guaranteed by those on W_{feed_i} and the requirements on T_{tank} are guaranteed by those on T_{scav_i}. The simplified version also does not contain the sub-formula for W_{scav_i}. This is because to have a scavenge flow greater than the feed flow requires modeling of the deaerator/breather which has not been performed in Section 5.2. It is therefore assumed that the control system will be capable of satisfying this requirement, which can be verified at a lower-level of fidelity.

C.6 Checking composition of contracts using the OCRA tool

The manual method of checking consistency of contract composition has been demonstrated in section Section C.5. However, in reality this is quite a time-consuming exercise and we ideally want to use some sort of software to do this

automatically. Fortunately this can be done using the OCRA tool [109] The input to OCRA is an *Othello System Specification* file as shown in code C.1.

```
COMPONENT modulated_oil_system_separate_viewpoints system
    INTERFACE
        INPUT PORT T_fuel_in: real;
        INPUT PORT W_fuel: real;
        OUTPUT PORT T_scav_i: real;
        OUTPUT PORT T_tank: real;
        OUTPUT PORT T_feed: real;
        OUTPUT PORT T_fuel_out: real;
        OUTPUT PORT W_scav_i: real;
        OUTPUT PORT W_oil: real;
        OUTPUT PORT W_feed_i: real;

        CONTRACT system
            assume: always ( (T_fuel_in >= -35) and (T_fuel_in <= 55) and (W_fuel >= 0.1) and (
    ↪ W_fuel <= 0.8) ) ;
            guarantee: always ( (T_scav_i <= 220) and (T_tank <= 220) and (T_feed >= 10) and (
    ↪ T_feed <= 120) and (T_fuel_out >= 0) and (T_fuel_out <= 165) and (W_scav_i >= 0.105) and (
    ↪ W_scav_i <= 0.525) and (W_oil >= 0.49) and (W_oil <= 2.8) and (W_feed_i >= 0.07) and (
    ↪ W_feed_i <= 0.4) );

    REFINEMENT
        SUB he: heat_exchanger;
        SUB ocv: oil_chamber_valves;
        SUB ps: pumping_storage;

        CONNECTION he.T_fuel_in := T_fuel_in;
        CONNECTION he.W_fuel := W_fuel;
        CONNECTION he.T_tank := ps.T_tank;
        CONNECTION he.W_oil := ps.W_oil;
        CONNECTION ocv.T_feed := he.T_feed;
        CONNECTION ocv.W_oil := he.W_oil;
        CONNECTION ps.T_scav_i := ocv.T_scav_i;
        CONNECTION ps.W_feed_i := ocv.W_feed_i;
        CONNECTION T_scav_i := ocv.T_scav_i;
        CONNECTION T_tank := ps.T_tank;
        CONNECTION T_feed := he.T_feed;
        CONNECTION T_fuel_out := he.T_fuel_out;
        CONNECTION W_scav_i := ps.W_scav_i;
        CONNECTION W_oil := ps.W_oil;
        CONNECTION W_feed_i := ocv.W_feed_i;

        CONTRACT system REFINEDBY he.thermal, he.flow, ocv.thermal, ocv.flow, ps.thermal, ps.flow;

COMPONENT heat_exchanger
    INTERFACE
        INPUT PORT T_tank: real;
        INPUT PORT T_fuel_in: real;
        INPUT PORT W_oil: real;
        INPUT PORT W_fuel: real;
        OUTPUT PORT T_feed: real;
        OUTPUT PORT T_fuel_out: real;

        CONTRACT thermal
            assume: always ( (T_tank <=220) and (T_fuel_in >= -35) and (T_fuel_in <= 55) );
            guarantee: always ((not Idle implies (T_feed >= 10)) and (not Idle implies (T_feed <=
    ↪ 120)) and (T_fuel_out >= 0) and (T_fuel_out <= 165) );
```

```
        CONTRACT flow
            assume: always ( (W_oil >= 0.49) and (W_oil <= 2.8) and (f_fuel >= 0.1) and (f_fuel <=
    ↪  0.8) );
            guarantee: always true;

COMPONENT oil_chamber_valves
    INTERFACE
        INPUT PORT T_feed: real;
        INPUT PORT W_feed_i: real;
        OUTPUT PORT T_scav_i: real;

        CONTRACT thermal
            assume: always ((not Idle implies (T_feed >= 10)) and (not Idle implies (T_feed <=
    ↪ 120)));
            guarantee: always (T_scav_i <=220);
        CONTRACT flow
            assume: always ( (W_oil >= 0.49) and (W_oil <= 2.8) );
            guarantee: always ( (W_feed_i >= 0.07) and (W_feed_i <= 0.4) );

COMPONENT pumping_storage
    INTERFACE
        INPUT PORT T_scav: real;
        INPUT PORT f_feed: real;
        OUTPUT PORT T_pump: real;
        OUTPUT PORT f_scav: real;
        OUTPUT PORT f_pump: real;

        CONTRACT thermal
            assume: always ( (T_scav_i <=220) );
            guarantee: always ( (T_tank <=220) );
        CONTRACT flow
            assume: always ( (W_feed_i >= 0.07) and (W_feed_i <= 0.4) );
            guarantee: always ( (W_oil >= 0.49) and (W_oil <= 2.8) and (W_scav_i >= 0.105) and (
    ↪ W_scav_i <= 0.525) );
```

Listing C.1: An othello system specification for the modulated oil system

This code has been used along with commands in OCRA to check consistency of contracts (check that the sub-system contracts compose properly with no contradiction between the assumptions and guarantees).

Bibliography

[1] M. Couture, "Complexity and Chaos - State-of-the-Art; Formulations and Measures of Complexity," Defence R&D Canada, Valcartier, Tech. Rep. September, 2007.

[2] L. Amaral and J. Ottino, "Complex networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 147–162, 2004.

[3] A. A. Minai, D. Braha, and Y. Bar-Yam, "Complex engineered systems: A new paradigm," in *Complex Engineered Systems: Science Meets Technology*. Berlin Heidelberg: Springer-Verlag, 2006, pp. 1–21.

[4] A. Sangiovanni-Vincentelli, "Quo vadis, SLD? Reasoning about the trends and challenges of system level design," *Proc. IEEE*, vol. 95, no. 3, pp. 467–506, 2007.

[5] *Research Grand Challenges for Systems Engineering Workshop Report*. Systems-NET, June 2015.

[6] A. Benveniste, B. Caillaud, D. Nickovic *et al.*, "Contracts for Systems Design," INRIA, Rennes, France, Tech. Rep., 2012.

[7] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems," *European Journal of Control*, vol. 18, no. 3, pp. 217–238, 2012.

[8] Object Management Group. (2012) OMG SysML V1.3. [Online]. Available: http://sysml.org/ (Accessed on 11/06/2018).

[9] S. Schmerler, S. Fürst, S. Lupp *et al.*, "AUTOSAR – Shaping the Future of a Global Standard," in *5th VDI Congress*, Baden-Baden, Germany, 2012.

[10] V-Modell® XT V1.3 Documentation. [Online]. Available: ftp://ftp.heise.de/pub/ix/projektmanagement/vmodell/V-Modell-XT-Gesamt-Englisch-V1.3.pdf (Accessed on 11/06/2018).

[11] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems.* Society for Industrial and Applied Mathematics, 2005.

[12] Y. Chahlaoui and P. Van Dooren, "A collection of benchmark examples for model reduction of linear time invariant dynamical systems." Université Catholique de Louvain, Tech. Rep., 2002.

[13] C. Gu, "Model order reduction of nonlinear dynamical systems," Ph.D. dissertation, University of California at Berkeley, 2011.

[14] S. W. Miller, T. W. Simpson, and M. Yukish, "Design as a sequential decision process: a method for reducing design set space using models to bound objectives," in *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Volume 2A: 41st Design Automation Conference*, 2015.

[15] A. Sangiovanni-Vincentelli, "Defining platform-based design," *EE Times*, pp. 1–20, 2002.

[16] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[17] P. Nuzzo, J. Finn, M. Mozumdar, and A. Sangiovanni-vincentelli, "Platform-Based Design Methodology and Modeling for Aircraft Electric Power Systems," in *Proceedings of Green Energy and Systems Conference*, 2013, pp. 1–7.

[18] A. Pinto, S. Becz, and H. Reeve, "Correct-by-Construction Design of Aircraft Electric Power Systems," in *10th AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, 2010.

[19] P. Nuzzo, H. Xu, N. Ozay, J. B. Finn, A. L. Sangiovanni-Vincentelli, R. M. Murray, A. Donze, and S. A. Seshia, "A Contract-Based Methodology for Aircraft Electric Power System Design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[20] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis, "Multiple viewpoint contract-based specification and design," in *Formal Methods for Components and Objects.* Springer, 2008, vol. 5382 LNCS, pp. 200–225.

[21] L. Benvenuti, A. Ferrari, E. Mazzi, and A. L. Sangiovanni-Vincentelli, "Contract-based design for computation and verification of a closed-loop hybrid system," in *Hybrid Systems: Computation and Control*. Springer, 2008, vol. 4981 LNCS, pp. 58–71.

[22] P. Nuzzo, J. B. Finn, A. Iannopollo, and A. L. Sangiovanni-vincentelli, "Contract-Based Design of Control Protocols for Safety-Critical Cyber-Physical Systems," in *ACM/IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014, pp. 1–4.

[23] P. Nuzzo, A. Sangiovanni-Vincentelli, X. Sun, and A. Puggelli, "Methodology for the design of analog integrated interfaces using contracts," *IEEE Sensors Journal*, vol. 12, no. 12, pp. 3329–3345, 2012.

[24] D. G. Firesmith, P. Capell, D. Falkenthal, C. B. Hammons, D. T. Latimer, and T. Merendino, *The Method Framework for Engineering System Architectures*. Boca Raton, FL: CRC Press, 2008.

[25] E. Crawley, B. Cameron, and D. Selva, *System Architecture: Strategy and Product Development for Complex Systems*. Hoboken, NJ: Prentice Hall, 2015.

[26] E. Crawley, O. de Weck, S. Eppinger *et al.*, "The influence of architecture in engineering systems," in *Engineering Systems Monograph*. Cambridge, MA: Massachusetts Institute of Technology, 2004, pp. 1–30.

[27] INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 2006, vol. Version 3.

[28] D. Selva, B. Cameron, and E. Crawley, "Patterns in System Architecture Decisions," *Systems Engineering*, vol. 19, no. 6, pp. 477–497, 2016.

[29] J. P. Elm, D. R. Goldenson, K. E. Emam, N. Donitelli, and A. Neisa, "A Survey of Systems Engineering Effectiveness - Initial Results (with detailed survey response data)," Software Engineering Institute, Carnegie Mellon University, Tech. Rep., 2008.

[30] H. Gustavsson and J. Axelsson, "Architecting complex embedded systems: An industrial case study," in *IEEE Int. Systems Conf.*, 2011, pp. 472–478.

[31] D. R. Georgiadis, T. A. Mazzuchi, and S. Sarkani, "Using Multi Criteria Decision Making in Analysis of Alternatives for Selection of Enabling Technology," *Syst. Eng.*, vol. 16, no. 3, pp. 287–303, 2013.

[32] J. Fröberg, S. Larsson, S. Dersten, and P. Å. Nordlander, "Defining a method for identifying architectural candidates as part of engineering a system architecture," in *IEEE 8th Annual International Systems Conference*, 2014, pp. 266–271.

[33] N. Bajaj, P. Nuzzo, M. Masin, and A. Sangiovanni-Vincentelli, "Optimized selection of reliable and cost-effective cyber-physical system architectures," in *Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 561–566.

[34] J. Finn, P. Nuzzo, and A. Sangiovanni-vincentelli, "A Mixed Discrete-Continuous Optimization Scheme for Cyber-Physical System Architecture Exploration," in *2015 IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 216–223.

[35] O. Hammami and M. Houllier, "Rationalizing approaches to multi-objective optimization in systems architecture design," in *IEEE Int. Systems Conf.*, 2014, pp. 407–410.

[36] O. Hammami, "Architecture frameworks, multiobjective optimization and multiphysics simulation: Challenges and opportunities," in *IEEE 9th Annual International Systems Conference*, 2015, pp. 546–553.

[37] S. V. Subramanian and D. A. DeLaurentis, "Application of Multidisciplinary Systems-of-Systems Optimization to an Aircraft Design Problem," *Systems Engineering*, vol. 19, no. 3, pp. 235–251, 2016.

[38] R. E. Thompson, J. M. Colombi, J. Black, and B. J. Ayres, "Disaggregated Space System Concept Optimization: Model-Based Conceptual Design Methods," *Systems Engineering*, vol. 18, no. 6, pp. 549–567, 2015.

[39] A. Wichmann, S. Jäger, T. Jungebloud, R. Maschotta, and A. Zimmermann, "System architecture optimization with runtime reconfiguration of simulation models," in *IEEE 9th Annual International Systems Conference*, 2015, pp. 660–667.

[40] S. Pugh, *Total Design: Integrated Methods for Successful Product Engineering.* Addison-Wesley Publishing Company, 1991.

[41] D. G. Jansson and S. M. Smith, "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, 1991.

[42] S. J. Rekuc, J. M. Aughenbaugh, M. Bruns, and C. J. J. Paredis, "Eliminating Design Alternatives Based on Imprecise Information," SAE International, Tech. Rep., 2006.

[43] R. Ullah, D.-Q. Zhou, P. Zhou, M. Hussain, and M. A. Sohail, "An approach for space launch vehicle conceptual design and multi-attribute evaluation," *Aerospace science and technology*, vol. 25, no. 1, pp. 65–74, 2013.

[44] J. Craisse, S. Krüger, Y. J. Kim, I. Chakraborty, S. Briceno, Y. Li, E. Garcia, and D. Mavris, "Creation of a decision-support methodology for selecting more-electric aircraft subsystem technologies," in *IEEE 10th Annual International Systems Conference*. IEEE, 2016, pp. 1–7.

[45] S. Burge, "The systems engineering tool box," Burge Hughes Walsh, Tech. Rep., 2014.

[46] M. Schmit, S. Briceno, K. Collins, D. Mavris, K. Lynch, and G. Ball, "Semantic Design Space Refinement for Model-Based Systems Engineering," in *IEEE 10th Annual International Systems Conference*, 2016, pp. 443–450.

[47] A. Agarwal, G. L. Hamza-Lup, and T. M. Khoshgoftaar, "A system-level modeling methodology for performance-driven component selection in multicore architectures," *IEEE Systems Journal*, vol. 6, no. 2, pp. 317–328, 2012.

[48] C. Calvert, G. L. Hamza-Lup, A. Agarwal, and B. Alhalabi, "An integrated component selection framework for system-level design," in *IEEE 5th Annual International Systems Conference*, 2011, pp. 261–266.

[49] W. Chapman, J. Rozenblit, and A. Bahill, "System Design is an NP-Complete Problem," *Syst. Eng.*, vol. 4, no. 3, pp. 222–229, 2001.

[50] N. Albarello and J.-B. Welcomme, "A model-based method for the generation and optimization of complex systems architectures," in *IEEE 6th Annual International Systems Conference*, 2012, pp. 1–6.

[51] J. Löfberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design*, 2004, pp. 284–289.

[52] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[53] C. A. Coello Coello, G. B. Lamont, and D. A. van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. New York: Springer, 2007.

[54] J. D. Lohn, G. S. Hornby, and D. S. Linden, "An evolved antenna for deployment on nasa's space technology 5 mission," in *Genetic Programming Theory and Practice II*, U.-M. O'Reilly, T. Yu, R. Riolo, and B. Worzel, Eds. Boston, MA: Springer US, 2005, pp. 301–315.

[55] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 416–423.

[56] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms — part i: A unified formulation," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.

[57] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms — part ii: An application example," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 1, pp. 38–47, 1998.

[58] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe, "Many-Objective Optimization: An Engineering Design Perspective," *Evolutionary Multi-Criterion Optimization*, vol. LNCS 3410, pp. 14–32, 2005.

[59] H. A. Thompson, A. J. Chipperfield, P. J. Fleming, and C. Legge, "Distributed aero-engine control systems architecture selection using multi-objective optimisation," *Control Engineering Practice*, vol. 7, pp. 655–664, 1999.

[60] R. Kudikala, A. R. Mills, P. J. Fleming, G. F. Tanner, and J. E. Holt, "Aero Engine Health Management System Architecture Design Using Multi-Criteria Optimization," in *GECCO'13*, 2013, pp. 185–186.

[61] R. Kudikala, A. R. Mills, P. J. Fleming, G. F. Tanner, and J. E. Holt, "Real World System Architecture Design Using Multi-Criteria Optimization: A Case Study," in *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*. Heidelberg: Springer International Publishing, 2013, pp. 245–260.

[62] J. Froberg, S. Larsson, and P. A. Nordlander, "A method for analyzing architectural drivers when engineering a system architecture," in *IEEE 7th Annual International Systems Conference*, 2013, pp. 711–717.

[63] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science*, 1977, pp. 46–57.

[64] R. Alur and T. A. Henzinger, "A Really Temporal logic," *Journal of the Association for Computing Machinery*, vol. 41, no. 1, pp. 181–204, 1994.

[65] O. Maler and D. Nickovic, "Monitoring Temporal Properties of Continuous Signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. New York: Springer, 2004, pp. 152 – 166.

[66] J. Liu and N. Ozay, "Abstraction , Discretization , and Robustness in Temporal Logic Control of Dynamical Systems," in *International Conferance on Hybrid Systems: Computation and Control*, 2014, pp. 293–302.

[67] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification*, 2010, vol. 6174 LNCS, pp. 167–170.

[68] A. Donźe, "Breach Toolbox: Instrumenting Simulation and Signal Temporal Logics for the Analysis of Hybrid Systems," *6th Workshop on Numerical Software Verification*, 2013.

[69] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems*, 2010.

[70] G. E. Fainekos and G. J. Pappas, "Robustness of Temporal Logic Specifications for Continuous Time Signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[71] A. Donzé, T. Ferrère, and O. Maler, "Efficient Robust Monitoring for STL," in *Computer Aided Verification*, 2013, vol. 8044 LNCS, pp. 264–279.

[72] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust Online Monitoring of Signal Temporal Logic," *Formal Methods in System Design*, vol. 51, pp. 5–30, 2017.

[73] X. Jin, A. Donzé, and G. Ciardo, "Mining Weighted Requirements from Closed-Loop Control Models," in *Sixth International Workshop on Numerical Software Verification (NSV)*, 2013.

[74] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 81–87.

[75] M. R. Garey and D. S. Johnson, *Computers and intractability*, vol. 29.

[76] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 239–248.

[77] S. S. Farahani, V. Raman, and R. M. Murray, "Robust model predictive control for signal temporal logic synthesis," in *IFAC Conference on Analysis and Design of Hybrid Systems*, vol. 48, no. 27. Elsevier, 2015, pp. 323–328.

[78] A. Donzé and V. Raman, "Blustl: Controller synthesis from signal temporal logic specifications." *EPiC Series in Computer Science*, vol. 34, pp. 160–168, 2015.

[79] C. J. Hambley, A. R. Mills, V. Kadirkamanathan, T. J. Dodd, W. Bradley, and R. Shirtcliffe, "Customer-oriented architecture refinement in multi-criteria synthesis of large-scale system architectures," in *IEEE International Symposium on Systems Engineering (ISSE)*, 2017.

[80] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Heidelberg: Springer, 2010.

[81] V. Shukla, G. Auriol, and K. W. Hipel, "Multicriteria Decision-Making Methodology for Systems Engineering," *IEEE Syst. J.*, vol. 10, no. 1, pp. 4–14, 2016.

[82] A. Inselberg, "The plane with parallel coordinates," *Visual Computer*, vol. 1, pp. 69–91, 1985.

[83] D. Luzeaux, T. Morlaye, and J.-L. Wippler, "Chances are the architecture looks fuzzy: Pitch it right!" in *IEEE International Symposium on Systems Engineering (ISSE)*, 2015.

[84] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *Engineering of Complex Computer Systems, 1998. ICECCS'98. Proceedings. Fourth IEEE International Conference on*. IEEE, 1998, pp. 68–78.

[85] L. Cohen, *Quality function deployment : how to make QFD work for you*. Addison-Wesley Publishing Company, 1995.

[86] J. R. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review*, vol. 66, no. 3, 1988.

[87] R. Kudikala, "System Architecture Design Using Multi-Criteria Optimization," Ph.D. dissertation, University of Sheffield, 2015.

[88] S. R. Goerger, A. M. Madni, and O. J. Eslinger, "Engineered resilient systems: A DoD perspective," *Procedia Computer Science*, vol. 28, pp. 865 – 872, 2014, 2014 Conference on Systems Engineering Research.

[89] J. F. Schank, F. W. Lacroix, R. E. Murphy, C. IP, M. V. Arena, and G. T. Lee, "Learning From Experience: Lessons from the United Kingdom's Astute Submarine Program," RAND National Defense Research Institute, Tech. Rep., 2011.

[90] United States Nuclear Regulatory Commission Technical Training Center, "Pressurized Water Reactor (PWR) Systems," in *Reactor Concepts Manual*. US Nuclear Regulatory Commission, 2016.

[91] Rolls-Royce, *The Jet Engine*. London: Rolls-Royce plc, 2005.

[92] A. Linke-Diesinger, *Systems of Commercial Turbofan Engines: An Introduction to Systems Functions*. Berlin: Springer-Verlag, 2008.

[93] The MathWorks Inc., "MATLAB and Optimization Toolbox Release 2015b," Natick, Massachusetts, United States.

[94] C. J. Hambley, B. L. Jones, I. Griffin, A. R. Mills, and V. Kadirkamanathan, "Optimized synthesis of cost-effective, controllable oil system architectures for turbofan engines," *Systems Engineering*, vol. 21, no. 5, pp. 417–431, 2018.

[95] D. Dori, *Object Process Methodology: A Holistic Systems Paradigm*. Berlin: Springer-Verlag, 2002.

[96] L. Jones, "Battle for the skies," *Engineering & Technology*, pp. 72–75, 2015.

[97] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. Chichester: Wiley, 2005.

[98] Gurobi Optimization Inc., "Gurobi Optimizer 6.5," Beaverton, Oregon, United States, 2016.

[99] R. Mukherjee, "Effectively Design Shell-and-Tube Heat Exchanger," *Chemical Engineering Progress*, vol. 94, no. 2, pp. 21–37, 1998.

[100] TEMA (Tubular Exchanger Manufactures Association), "Standards of the Tubular Exchanger Manufacturers Assocciation," Tech. Rep., 1999.

[101] J. E. Edwards, "Design and rating shell and tube heat exchangers," *P&ID Design Ltd., Teesside, UK*, 2008.

[102] E. J. Roldán-Villasana, Y. Mendoza-Alegría, and I. F. Galindo-García, "Lube Oil Systems Models for Real Time Execution Used on a Training Power Plant Simulator," in *Intelligent Automation and Systems Engineering*, S.-L. Ao, M. Amouzegar, and B. B. Rieger, Eds., 2011, vol. 77.

[103] Y. Mendoza-Alegría, E. J. Roldán-Villasana, I. F. Galindo-García, and J. Zorrilla-Arena, "Oil Systems Modeling for an Operators' Training Combined Cycle Plant Simulator," in *World Congress on Engineering and Computer Science*, vol. II, 2010, pp. 1002–1008.

[104] N. E. Daidzic, "Jet engine thrust ratings," *Professional Pilot*, vol. 46, no. 9, pp. 92–96, 2012.

[105] European Aviation Safety Agency, "Certification specifications for engines," Cologne, Germany, Tech. Rep., 2007.

[106] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of physics*. Wiley New York, 2013, vol. 10th Edition.

[107] *Mobil Jet Oil II: Typical Properties*, ExxonMobil Aviation, 2018.

[108] *747-400 Performance Summary*, StartupBoeing, 2010.

[109] A. Cimatti, M. Dorigatti, and S. Tonetta, "OCRA: A tool for checking the refinement of temporal contracts," in *28th IEEE/ACM International Conference on Automated Software Engineering*, 2013, pp. 702–705.

[110] A. Iannopollo, P. Nuzzo, S. Tripakis, and A. Sangiovanni-vincentelli, "Library-Based Scalable Refinement Checking for Contract-Based Design," in *ACM/IEEE Design, Automation and Test in Europe Conference and Exhibition*, 2014.