Optimal Architecture Selection for an Aircraft Environmental Control System

John Finn
Alberto L. Sangiovanni-Vincentelli

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-28
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-28.html

May 1, 2016

Optimal Architecture Selection for an Aircraft Environmental Control System

by John Butler Finn IV

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

4/24/15

Professor Alberto L. Sangiovanni Vincentelli — Date
Research Advisor

* * * * * * * *

5/4/15

Professor Edward A. Lee — Date
Second Reader

5/8/15

Professor Sanjit A. Seshia — Date
Third Reader

Abstract

Cyber Physical Systems (CPS) are characterized by a tight coupling between the physical and the computational worlds. As system complexity and heterogeneity increase, it is becoming more difficult to perform design space exploration at the system level and analyses between the system architecture and control algorithms. Often, designers are expected to solve combinatorial problems over a large discrete variable space that is coupled to a continuous space, where expensive, high-fidelity simulations must be run to achieve the desired accuracy. In this thesis, I build upon a design methodology, based on the principles of Platform Based Design, to address the challenges associated with next generation CPS. The CPS design methodology is a series of three steps including: Architecture Selection, Control Synthesis and Verification/Optimization. I extend Architecture Selection to support a mixed discrete and continuous design space. The proposed approach is an iterative process, where a discrete architecture SELECTION engine is placed in a loop with a continuous SIZING engine. First, the SELECTION engine proposes a candidate architecture to the SIZING engine. Then, the SIZING engine attempts to optimize continuous parameters subject to formalized design requirements, which are monitored using simulation. If the SIZING engine cannot find a feasible solution, the SELECTION engine is queried for another candidate architecture and the process repeats. I illustrate the methodology on an industrial case study, namely an aircraft Environmental Control System. Finally, I show how balance equations and conservation laws can be used to prune the discrete search space and reduce the number of simulations. In my experiments, I obtain more than an order of magnitude reduction in design runtime.

Acknowledgements

First and foremost, I must sincerely thank my advisor, Prof. Alberto Sangiovanni-Vincentelli, for the opportunity and support during my graduate career at the University of California, Berkeley. His guidance and expertise allowed me to grow as a student and become more independent as an engineer. His exuberant passion for design has most certainly inspired me and has had such a positive influence on my future.

I also sincerely thank Pierluigi Nuzzo for his constant collaboration and insight throughout my stay at Berkeley. He always made himself available to provide helpful discussions and interesting research directions. I very much appreciate all of his support along the way. I must also thank Prof. Edward Lee, Sanjit Seshia and Jaijeet Roychowdhury for offering inspiring and fascinating courses in system design, modeling, simulation, analysis and verification as well as their effective teaching abilities. I would also like to thank Professors Lee and Seshia for taking the time to read this thesis.

I greatly appreciate the support from the MuSyc, iCyPhy and TerraSwarm research centers over the years. I would like to thank the entire iCyPhy community for their help in creating the models used in my thesis. In addition, I must thank United Technologies Corporation for their continued support and internships during my graduate career. I am very grateful for the collaboration with industry, which has been both fun and interesting. Specifically, I would like to thank Jeff Ernst, Earl LaVallee, Eelco Scholte and Clas Jacobson.

I also thank all of the members of the DOP Center for fostering an environment full of collaboration and excitement. I thank my colleagues: Fabio Cremona, Antonio Iannopollo, Marco Marazza, Nikunj Bajaj, Shromona Ghosh, Alberto Puggelli, Baihong Jin, Lianpang Guo, Xuening Sun, Chung-Wei Lin, Michele Lora, Michael Zimmer, Chris Shaver, Marten Lohstroh, Jon Tamir, Zach Wasson, Gage Eads, Nishant Tolta, Mohammad Mozumdar, Mehdi Maasoumy and Alexandre Donze for their friendship, support and collaboration. I thank the EECS staff including Shirley Salanio, Jessica Gamble, Barb Hoversten and Christopher Brooks for all of their help with administrative tasks, organizing meetings and travel. It has been a pleasure to work with all of you.

Finally, I must thank my parents and brother for their unwavering love and support. None of this would have ever been possible without you. Words cannot express my appreciation for all you have done for me.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

In this chapter, I introduce the notion and challenges of Cyber-Physical System design and outline how Platform-Based Design can be used to address such challenges. Then, I provide an overview of aircraft Environmental Control Systems, which is the design driver for this thesis. Finally, I conclude this chapter with a summary of contributions.

1.1 Cyber-Physical Systems

Cyber-Physical Systems (CPS) are characterized by a tight coupling of the physical world to the computational or cyber world. Technology advancements in high performance embedded processors have enabled the "electrification" of systems for improved cost, reliability and efficiency. This trend is very apparent in the automotive and aerospace industries [4] as traditionally mechanical, pneumatic or hydraulic systems are being replaced in part by embedded processors with various sensors and actuators. However, this trend poses significant design challenges, especially for safety critical systems such as cars or airplanes. Challenges arise when designing and verifying large, complex systems with tightly coupled heterogeneous domains. Today, more than ever before, CPS span several domains, such as electrical, mechanical, chemical or computational, and there is little support for designers to analyze tradeoffs between them. For example, designers would like to perform rich trade studies between the design of the control protocol and the architecture. However, this often requires solving combinatorial problems with a large discrete variable space coupled to a continuous space, where expensive, high-fidelity simulations must be run for the desired accuracy.

Traditionally, the design process is carried out according to imprecise requirements, designer experience, previous design iterations or the V-Model [5]. Unfortunately, such practices are typically very manual and ad-hoc, especially at the interface between domains, and no longer scale with the complexity of today's CPS. A severe limitation, which is causing long and costly redesign cycles, is the inability to foresee the impact of design decisions made early in the design cycle on the final implementation. Furthermore, design requirements are generally written in a natural language, and therefore, they are often ambiguous or conflicting, which usually leads to implementations that do not meet the requirements.

1.2 Platform Based Design for Cyber Physical Systems

Platform Based Design (PBD) [6] is a design methodology developed to address the complexity and design challenges associated with CPS. At its core, PBD formalizes the design process, raises the level of abstraction and hides unnecessary lower-level details with abstract models to reduce complexity while enabling efficient design verification early in the design cycle. More concretely, design is carried out as a sequence of refinement steps at well-defined levels of abstraction. At each level of abstraction, a design is selected by a mapping or "meet-in-the-middle" of a top-down process with a bottom-up process. The top-down process maps the specified behavior and function (i.e what the system is supposed to do) to a library of available components. The bottom-up process abstracts away unnecessary details of the library components to implement the desired behavior. Once the function has been coupled to the architecture at a given level of abstraction, the outcome becomes the specification to the next, lower level of abstraction and the process is repeated. Design continues as a sequence of refinement steps until a physical implementation is achieved. PBD has been successfully deployed in the automotive [7] and consumer electronic domains [8]. Furthermore, the iCyPhy consortium has been established to develop and implement PBD methodologies for industrial CPS. In [2], we propose a CPS design methodology to address CPS design challenges and implement the methodology for the design of an aircraft Electric Power System's (EPS) architecture and supervisory control. The goal of this thesis is to extend this work to a more complex physical system, namely an aircraft Environmental Control System.

1.3 Aircraft Environmental Control Systems

An aircraft Environmental Control System (ECS) [9] is responsible for conditioning the high pressure, high temperature engine bleed air to ensure a comfortable and fresh cabin for all passengers and crew. A typical ECS architecture is shown in Figure 1, which is inspired from the patent in [1]. Engine bleed air comes in from valve 1 on the left and is then fed into the primary heat exchanger, which cools the bleed air with ambient ram air from the environment. From the primary heat exchanger, the air is sent to a compressor and then through the secondary heat exchanger, which utilizes the same ambient air for further cooling. Then, to remove moisture in the air, it is passed through a condenser and water separator. After the condenser, the air is sent through a series of turbines. In Figure 1, there are two bypass valves, which control the mixing ratios of air at various temperatures. A mixer combines the output of turbine 2 with the bypass air and recirculated air from the cabin. This mixture is then delivered to the cabin. The control of the valves and

the ambient ram air flow allow the system to maintain comfortable cabin temperatures, pressures and flow rates, while reacting to disturbances and set point changes in a timely manner. Finally, air ducts connect all the ECS components to transport gaseous air from one to another.

In ECS design, the critical design considerations are the individual duct size and material selection, heat exchanger size (length, area and tube diameter), valve positions and ram air flow rate. Typically, ducts are selected from a pre-existing catalog or "off the shelf" as customization is too expensive. In contrast, it is more cost effective and desirable to customize the heat exchanger sizing for the application and design requirements. Therefore, the heat exchanger should be sized on a continuous spectrum. Similarly, the control variables range continuously between pre-defined limits. For example, a valve can range from anything between completely closed to completely open. Hence, for optimal ECS design, designers must simultaneously reason about continuous and discrete choices. Unfortunately, the discrete and continuous choices are strongly coupled, and therefore, it is not effective to separate the two domains. For example, duct sizing is directly related to maximum flow, where flow is controlled by the valves.

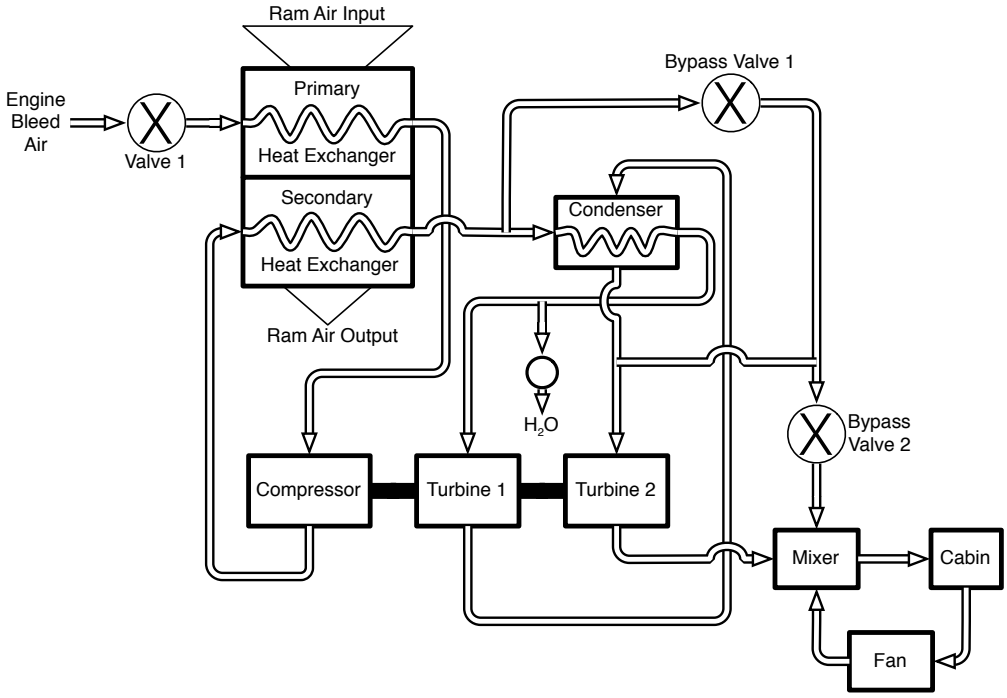

Figure 1: Aircraft ECS Patent Architecture inspired by [1]

1.4 Outline and Contributions

The objective of this thesis is to extend the current CPS design methodology in [2], while also illustrating and implementing the proposed approach on a case study of industrial interest. This thesis is organized as follows. In Section 2, I outline the previous work in both CPS design and aircraft ECS design. Then in Section 3, I outline the proposed methodology for Architecture Selection, while in Section 4, I show how to implement the methodology and formulate the ECS design problem. Section 5 summarizes the experimental results. Finally, I end with concluding remarks and outline future work in Section 6. The contributions of this thesis are as follows:

- Propose an Architecture Selection optimization methodology for problems with a mixed continuous and discrete design space
- Partition the design optimization into separate, but interacting, discrete and continuous problems, while offering a feedback function to prune the discrete space and reduce design runtime
- Implement the methodology on a design driver inspired by industry, namely an aircraft ECS
- Improve upon current ECS design techniques by:
  - Formalizing design requirements
  - Developing dynamic and static multi-view component models
  - Leveraging simulation in the loop to enforce design requirements
  - Optimizing over a large design space with the co-design of the architecture and control

Chapter 2
Previous Work

In this chapter, I outline previous work in CPS with respect to the current CPS design challenges. Then, I give an overview of the current state-of-the-art in aircraft ECS design.

2.1 Cyber Physical System Design

The work of this thesis follows and fits into the PBD design methodology for CPS proposed in [2] and [3], which is illustrated in Figure 2 and represents the design process at one level of abstraction. In this work, we propose a PBD approach to address the challenges of CPS design and illustrate the approach on an aircraft Electric Power System (EPS).

First, the designer must formalize the high-level system requirements, which could be accomplished using assume/guarantee contracts [10], temporal logics (i.e. LTL [11] or STL [12]) or mathematical constraints. By using formal languages, requirements no longer suffer from the ambiguity that arises in natural languages. Requirements formalized in LTL allow automatic, correct-by-construction, control synthesis [13], while STL allows automatic requirement monitoring during simulation [14]. Once the requirements are formalized, the design process becomes a sequence of steps including: Architecture Selection, Control Synthesis and Simulation-Based Optimization/Verification. Each step leverages a parametrized component library of various dynamic and static multi-view models, which enable more efficient design algorithms by hiding unnecessary details. Once complete, the result becomes the set of requirements for the next (lower) level of abstraction as the design moves toward an implementation. This principal is known as the fractal nature of design.

The Architecture Selection step aims at synthesizing a plant topology, i.e. an interconnection of library components, as well as determining optimal parameter values, i.e. sizing. For the EPS case study in [2], this step is concerned with topology synthesis (populating an adjacency matrix) while satisfying reliability constraints. Next, the Control Synthesis step looks to synthesize the high-level supervisory control logic such as a finite state machine. Within the EPS, the supervisory control is responsible for the switching logic to route power from source to load under various fault scenarios and was synthesized from requirements formalized in LTL. Lastly, the Simulation-based Optimization/Verification looks to verify or optimize other model constraints or parameters using simulation once the selected architecture has

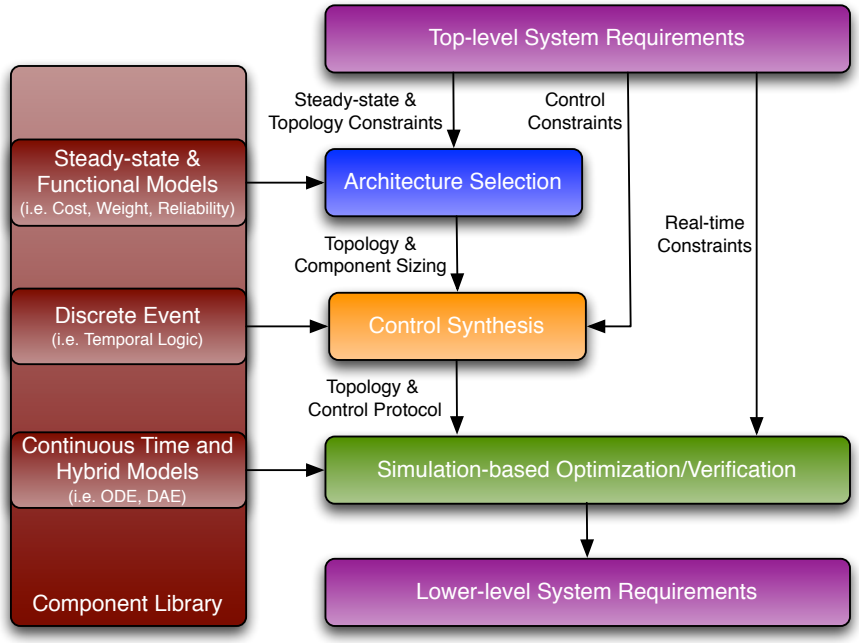

Figure 2: Design Methodology for Cyber-Physical Systems inspired by [2] and [3]

been coupled to the supervisory control. For the EPS, the tradeoff between the responsiveness of the control and switch delay was characterized. After Simulation-based Optimization/Verification, the selected design become the specification of the next level of abstraction toward an implementation. In this thesis, I follow this methodology and extend this work by adding continuous design variables to the Architecture Selection step and deploy the approach on an aircraft ECS.

2.2 Aircraft Environment Control Systems

The current state of aircraft ECS design is very ad-hoc and heavily relies on designer experience and previous design iterations. However, with more strict performance requirements and the increasing complexity of more electric architectures, such design practices do not scale to meet cost and time-to-market requirements. Current design techniques suffer from informal design requirements and do not address the co-design of the control protocol with the physical plant. Furthermore, these approaches only consider a small design space, which prevents rich trade studies, overall cost minimization and higher performance. Instead, the result can be long design cycles, cost overruns and implementations that may not even meet the requirements.

Shang et al., in [15], evaluate three ECS architectures for temperature control and efficiency, while also comparing the results to experimental data. Similarly, Tu et al., in [16], also look at the ECS control problem and propose a protocol based on fuzzy logic. However, in both cases, they only consider design of the control protocol without any consideration for sizing components (i.e. heat exchanger sizing) nor are system requirements rigorously handled in the design process. In [17], Mickelin et al. synthesize a correct-by-construction supervisory control protocol from requirements formalized in LTL along with a state-estimation algorithm, but again, architecture sizing is not considered. In contrast, Vargas et al., in [18], look to size the heat exchanger to maximize fuel efficiency, however pay little attention to control design. In other works such as [19] and [20], the focus is on developing realistic models of the ECS and validation with experimental data. In what follows, the heat exchanger model was inspired by [20], while the cabin and valve models are similar to those in [17]. The most closely related work to this is in [21], where Perez-Grande et al. look to determine the heat exchanger size along with the bypass ratio, which is a control variable, to optimize efficiency and volume. As in the previous cases, the design process is ad-hoc relying on designer experience to make important decisions at the various steps of the design. In summary, the previous work in ECS design explores a small design space in a very manual approach.

In contrast, my proposed approach first formulates an optimization subject to formal design requirements. Then, I optimize ECS design and take into account the tradeoffs between the control and architecture. The optimization leverages multi-view models (i.e. cost, weight, safety and behavioral) with simulation in the loop to ensure design requirements are satisfied, while optimizing cost. To do this, the combination of discrete design choices, including duct material and size selection, and continuous design variables, such as heat exchanger size and valve position, are considered. The proposed approach formalizes the design process without relying on designer experience or past designs.

Chapter 3
Methodology

This chapter is meant to be a very high-level introduction to the overall methodology for mixed discrete and continuous Architecture Selection. The actual implementation (i.e. cost functions, requirements, models, etc.) of the methodology is application dependent. I also offer insight to requirement formalization and system modeling. The next chapter goes into the details on how to implement such a methodology on an real, industrial case study.

3.1 Optimization Mapping Framework

The proposed approach is an interaction between a discrete architecture SELECTION engine and a continuous SIZING optimization engine with simulation. The overall idea is illustrated in Figure 3. As an input, the design optimization takes system and component requirements, behavioral and static component models and a cost function. First, the SELECTION engine optimizes the cost function and proposes a candidate architecture to the SIZING engine. The candidate architecture or topology is an interconnection of the library components according to composition rules and system requirements. Then, the SIZING engine attempts to determine the optimal value of the continuous design parameters associated with the selected architecture such as the length of a heat exchanger or position of a valve. Similarly, the SIZING engine aims to optimize the cost function subject to design specifications and the selected component's requirements. If unfeasible, the SIZING engine provides feedback or a compact reason for unfeasibility to aid the SELECTION optimizer in finding another candidate solution. Otherwise, the SIZING engine returns a solution. At this point, the algorithm may or may not terminate. The termination criterion will depend on the structure of the optimization problem and the application.

It can be very useful to formalize certain domain knowledge to improve the interaction and feedback between the two optimizations. Being able to extract information from the SIZING engine to help prune the SELECTION optimization can drastically improve runtime for large discrete design spaces. For example, I give an axiom based on conservation of mass to prune the discrete SELECTION search space in the ECS case study. Similarly, it may also be possible to export information from the SELECTION optimization to help shrink the continuous design space.

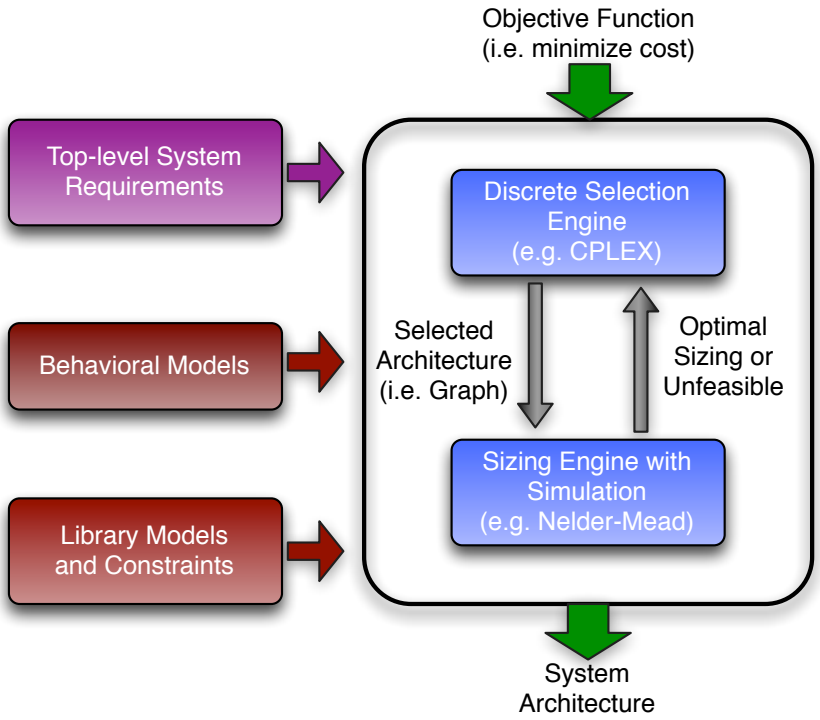

Figure 3: Architecture Selection

3.2 Requirement Formalization

Signal Temporal Logic: Signal Temporal Logic (STL) [12] is an extension of Linear Temporal Logic (LTL) [11] and gives the ability to formally reason about time and simulation traces, which makes it very useful within simulation-based methodologies. Specifically, STL extends LTL by annotating operators with a time interval. More formally, STL is a logic defined over real-valued, time-stamped signals, and I use the STL grammar defined below, which is formally defined in [12].

$$\varphi := True \mid \mu \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \, \mathcal{U}_I \, \varphi_2 \mid$$

Constraints on a real-valued, time-stamped signal $s(t)$, or *predicates*, can be reduced to the form $\mu = f(s(t)) \sim \pi$. Let $f : \mathbb{R}^m \to \mathbb{R}$ be a multidimensional function that evaluates to a scalar, $\sim \in \{<, \leq, >, \geq, =, \neq\}$ be an operator, $\pi \in \mathbb{R}$ and $\mu : \mathbb{R} \to \mathbb{B}$ be a mapping from a scalar to boolean signal. For example, the signal s satisfies $\mu = f(s(t)) > 10$ and evaluates to *true* at time t, written $(s, t) \models \mu$, if and only if $f(s(t)) > 10$, otherwise it is *false*. The grammar also includes the traditional *negation* and *disjunction* operations. The *until* operator has

an interval of the form $I = [a, b]$ associated with it, where $0 \leq a < b$. Formally, $(s, t) \models p\,\mathcal{U}_{[a,\ b]}\ q$ if and only if $\exists t' \in [t+a, t+b]$ s.t. $(s, t') \models p \wedge q$, and $\forall t'' \in [t, t']$, $(s, t'') \models p$. Furthermore, the traditional *future* and *global* operators can be defined as $\Diamond_I p = T\,\mathcal{U}_I\ p$ and $\Box_I p = \neg\Diamond_I \neg p$, respectively.

Mathematical Inequalities: Requirements with no dependence on time can be formalized using inequality constraints.

3.3 System and Component Modeling

In following the PBD paradigm, system modeling is critical for evaluating design alternatives or tradeoffs early in the design phase, especially in a simulation-based framework. The benefits of modeling and simulation for complex, heterogenous systems are invaluable to reduce design time and cost, while ensuring requirements are satisfied. However, it is of upmost importance to characterize accurate and multi-view models for design. The need for accurate models is obvious, especially in safety critical systems. Multi-view modeling encourages designers to reduce model complexity and improve runtime performance by orthogonalizing concerns. For example, a component may have a separate cost and behavioral model, and when evaluating functionality, only the behavioral model is necessary. Similarly, the behavioral model can be abstracted away when performing a cost analysis. In this methodology, I construct models at multiple views to characterize equation-based behavioral models for simulation and static performance models to build a library of components.

Dynamic models such as Differential Algebraic Equations (DAE) are generally more complex than static models, but allow designers to develop low-level control protocols, evaluate transients and real-time performance. Today, Matlab - Simulink [23] is the most common system modeling tool used in industry. Simulink is a data-flow modeling and simulation environment with toolboxes for many physical domains. As an alternative, Modelica [24] is gaining traction and interest from industry. Modelica is an object-oriented, acausal modeling language, which allows designers to declare equations as they appear in physics. Furthermore, since Modelica is just a standardized language, it is tool independent, however, there are a number of tools that support the language such as Dymola [25], JModelica [26], Mathematica [27] and OpenModelica [28]. In this thesis, I develop Modelica models for ECS simulation using JModelica to monitor requirements.

Static modeling allows designers to quickly characterize or abstract components using relevant metrics, such as cost, weight or performance, for example. This type of modeling generally reduces complexity, while allowing designers to evaluate high-level tradeoffs more efficiently. In the proposed approach, I create a library

of components characterized by performance, safety and physical attributes. Therefore, part of the overall design problem is to select the optimal set of components, with respect to a cost function, while satisfying the requirements.

The critical point in modeling is defining appropriate abstraction layers in order to efficiently reason between design alternatives, however this is very application dependent. In the remainder of this thesis, I validate and show how these techniques can be effectively and efficiently used on an aircraft ECS.

Chapter 4
ECS Problem Formulation

In this chapter, I formulate the ECS design problem. Specifically, I start by describing the multi-view component models, which include a static library of weight, cost and safety models with the component behavioral equations implemented in Modelica. Then, I formalize the system and component requirements using STL. Lastly, I formulate the optimization problem and show how to separate the SIZING and SELECTION optimization problems, while also proposing a feedback function to prune the discrete SELECTION design space.

4.1 System Modeling

Dynamic Models

A simplified ECS topology is shown in Figure 4, and in what follows, I describe the component equations that make up the overall ECS Modelica model. Components are connected together with a Modelica *connector*, and in the ECS, this is called a port, which defines the interface between components. The components are connected pneumatically, and therefore, transfer gaseous air at various temperatures [K], pressures [Pa] and mass flow rates [kg/s]. The *connector* construct includes *flow* variables such that the sum of these variables equals zero, which, in the ECS, is used to enforce the mass flow rate entering a port equals the mass flow rate leaving the port (with opposite sign). I use the convention that flows entering a port are positive, while flows leaving a port are negative. A *connector* also allows non-flow variables, which simply enforce equality. The ECS model uses these variables to enforce the temperature and pressure leaving one component equals the temperature and pressure entering the connected component. Multiple connections can be made to single *connector*.

By using ports, each component is described by DAEs, which describe the relationships between the temperature, pressure and mass flow rate for each *port* of that component. In what follows, let variables with subscript o represent outputs and subscript i represent inputs, according to the arrows in Figure 4. The arrows indicate the intended flow and operation of the ECS assuming no back-flow.

As an input, the ECS takes engine bleed air at a fixed pressure P_e and temperature T_e, which is cooled using a heat exchanger. The heat exchanger takes ambient ram air from the environment at a temperature T_a to cool the bleed air. The

ECS has three control inputs including each valve coefficient, C_1 and C_2, and the ambient air mass flow rate into the heat exchanger, F_a.

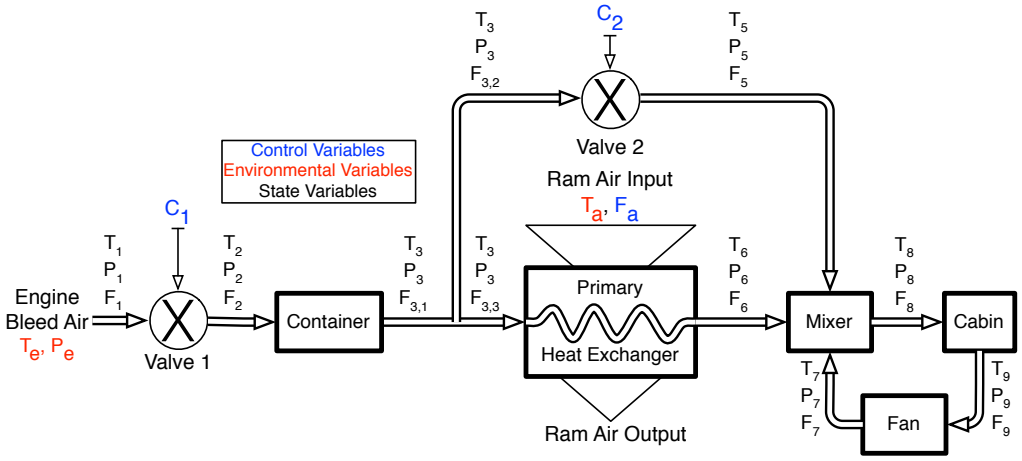

Figure 4: Simplified ECS Architecture

Valve: The valve controls the mass flow rate by restricting the size of the valve opening, which is controlled by C, the valve coefficient. First, the temperature of air leaving the valve equals the temperature entering the valve, as shown by Equation 1. Equation 2 enforces no reverse flow and accounts for choked flow, while describing the relationship between mass flow rate and pressure. In addition, the valve has a maximum coefficient such that $0 \leq C \leq C_{max}$.

$$T = T_i = T_o \tag{1}$$

$$F = F_i = -F_o = \begin{cases} 6.67 \times 10^{-4} C P_i \sqrt{\frac{1}{T}} & P_o < \frac{1}{2} P_i \\ 4.72 \times 10^{-4} C (P_i + 2P_o) \sqrt{\frac{1}{T}(1 - \frac{P_o}{P_i})} & \frac{1}{2} P_i \leq P_o < P_i \\ 0 & P_o \geq P_i \end{cases} \tag{2}$$

Container: The container has a specified volume V [m^3], which is a model parameter. The model also includes the mass m [kg] of air contained and corresponding thermal energy Q_c [J], which are state variables. First, Equation 3 requires equal pressures between the input and output. The output temperature is equal to the air

temperature in the container T, which is also shown in Equation 3. Next, Equation 4 is the Ideal Gas Law relating the pressure, output temperature, mass and volume, where R = 287.058 J/(kg K) is the gas constant for air. Equation 5 describes the thermal energy as a function of the mass, temperature and C_{air} = 1003.5 J/(kg K), which is the specific heat of air.

$$P = P_i = P_o \text{ and } T = T_o \tag{3}$$

$$P = \frac{mTR}{V} \tag{4}$$

$$Q_c = C_{air} mT \tag{5}$$

The following equations describe the dynamic behavior of the container. Equation 6 describes how the mass changes over time as a function of the incoming and outgoing mass flow rates. Similarly, Equation 7 shows how the thermal energy changes as a function of the input and output mass flow rates and temperature. Recall by convention, $F_i \geq 0$ and $F_o \leq 0$, assuming no backward flow.

$$\frac{d}{dt} m = F_i + F_o \tag{6}$$

$$\frac{d}{dt} Q_c = C_{air}(F_i T_i + F_o T_o) \tag{7}$$

Heat Exchanger: The heat exchanger is a component in which two flows of different temperature and at different flow rates are allowed to transfer energy. In this model, the flows are separated and do not mix. This configuration amounts to cooling the hot flow and heating the cold flow. In the context of the ECS, the heat exchanger takes hot air from the engine and cools it with ambient cold air from the ram air door. The model below is based on *effectiveness* [20]. In addition, the pressure drop across the hot flow is also taken into account.

It is highly desirable to optimize the size of the heat exchanger, while still satisfying the design requirements. The parameters to optimize are cross-sectional area A_{HX} [m^2], length L_{HX} [m] and tube diameter D_{HX} [m]. Note, the cross-sectional area is not the area of heat transfer. Instead, the heat transfer area is a function of the number n and length L_{HX} of the tubes as well as the cross-sectional area A_{HX}. Furthermore, one of the control variables of the ECS is the cold air input

flow rate F_a, which is controlled by adjusting the ram air door. In the equations below, a subscript of h denotes the hot side, while a subscript of c denotes the cold side.

First, what flows into the heat exchanger's hot/cold side also flows out, so there is no accumulation of mass, as shown in Equation 8. The control input F_a is equal to the cold flow input $F_{c,i}$. The number of tubes is shown in Equation 9, which is approximated as a real number. The area for heat transfer A_{HT} is shown in Equation 10.

$$F_{h,i} + F_{h,o} = 0 \text{ and } F_{c,i} + F_{c,o} = 0 \tag{8}$$

$$n = \frac{A_{HX}}{D_{HX}^2} \tag{9}$$

$$A_{HT} = n\pi D_{HX} L_{HX} \tag{10}$$

Next, Equations 11 through 18 describe the algebraic equations of the heat exchanger, where C_{air} is the specific heat of air and C_m [J/(kg K)] is the specific heat of the metal separating the two flows. Using empirical data, the heat transfer coefficients have been approximated by the linear relationships $h_h = a_h F_{h,i} + b_h$ and $h_c = a_c F_{c,i} + b_c$, where $[a_h, b_h, a_c, b_c] = [41.987, 23.049, 6.787, 19.409] \times 10^3$ assuming $T_{h,i} \approx 450$K and $T_{c,i} \approx 240$K. UA is the overall heat exchanger heat transfer coefficient multiplied by the area with units of [W/K], and NTU is the number of transfer units with no units. Finally, the heat exchanger *effectiveness* is calculated in Equation 18.

$$F_{max} = \max(F_{h,i}, F_{c,i}) \tag{11}$$

$$F_{min} = \min(F_{h,i}, F_{c,i}) \tag{12}$$

$$T_{hs} = T_{h,i} - \epsilon \frac{F_{min}(T_{h,i} - T_{c,i})}{F_{h,i}} \tag{13}$$

$$T_{cs} = T_{c,i} + \epsilon \frac{F_{min}(T_{h,i} - T_{c,i})}{F_{c,i}} \tag{14}$$

$$\frac{1}{UA} = \frac{1}{h_h A_{HT}} + \frac{1}{h_c A_{HT}} \tag{15}$$

$$NTU = \frac{UA}{F_{min}C_{air}} \tag{16}$$

$$C_r = \frac{F_{min}}{F_{max}} \tag{17}$$

$$\epsilon = 1 - \exp\left(\frac{NTU^{0.22}}{C_r}\left(e^{-C_r NTU^{0.78}} - 1\right)\right) \tag{18}$$

Next, Equations 19 and 20 describe the heat exchanger dynamics, where M_{HX} [kg] is the mass of the heat exchanger. Note, the mass M_{HX} and specific heat C_m depend on the material selected, which is a discrete design variable.

$$\frac{d}{dt}T_{h,o} + \frac{UA}{M_{HX}C_m}(T_{h,o} - T_{hs}) = 0 \tag{19}$$

$$\frac{d}{dt}T_{c,o} + \frac{UA}{M_{HX}C_m}(T_{c,o} - T_{cs}) = 0 \tag{20}$$

Lastly, the pressure drop across the hot side of the heat exchanger is modeled as defined below. A *headloss* [29] model calculates the pressure drop as a function of headloss h_l and density ρ, which is shown in Equation 21.

$$\Delta P = P_{h,i} - P_{h,o} = h_l \rho \tag{21}$$

The headloss h_l is calculated by Equation 22, where f is the friction coefficient and $V_{h,i}$ is the velocity [m/s] of the hot flow.

$$h_l = \frac{f L_{HX} V_{h,i}^2}{2D_{HX}} \tag{22}$$

The velocity is a function of the mass ($W_{h,i}$) and volumetric ($Q_{h.i}$) flow rates and density as shown in Equation 23.

$$V_{h,i} = \frac{4Q_{h,i}}{\pi D_{HX}^2} = \frac{4F_{h,i}}{\pi \rho D_{HX}^2} \tag{23}$$

In general, the density of air can be calculated as $\rho = \frac{P}{RT}$, where P is the pressure and T is the temperature. An acceptable practice is to average the hot side inlet and outlet densities as $\rho = \frac{\rho_{h,i} + \rho_{h,o}}{2}$ in Equations 21 and 23. Next,

the Reynolds number and *friction coefficient* equations are shown in Equations 24 and 25, respectively. ν is the *kinematic viscosity*, which is 2.5×10^{-5} [m/s^2].

$$Re = \frac{V_{h,i} D_{HX}}{\nu} \tag{24}$$

$$f = \begin{cases} \frac{64}{Re} & 0 < Re < 2300 \\ \frac{0.316}{Re^{0.25}} & 2300 \leq Re < 10^5 \end{cases} \tag{25}$$

In the context of the ECS design problem, L_{HX}, A_{HX}, D_{HX} and $F_{c,i}$ are continuous design parameters, while C_m is a discrete design choice. Note, M_{HX} is a function of L_{HX}, A_{HX}, D_{HX} and ρ_m, which is the density of the material selected.

Mixer: The mixer allows multiple input ports to be mixed into a single output port. Equations 26 through 28 define the algebraic equations of the mixer, where the input ports are indexed by k. The mixer requires all ports to have the same pressure and also balances the sum of flows and thermal energy.

$$P_o = P_i[k], \forall k \tag{26}$$

$$F_o = -\sum_k F_i[k] \tag{27}$$

$$F_o T_o = -\sum_k F_i[k] T_i[k] \tag{28}$$

Cabin: The goal of the ECS is to maintain a comfortable environment in the cabin. The cabin is parameterized by the volume V, thermal energy per passenger Q_{pass}, the number of passengers n and the heat gain from the environment dQ. The cabin is also parameterized at a constant pressure P_c, which is acceptable since there is a release valve to ensure the desired pressure.

$$P_c = P_i = P_o \tag{29}$$

Just like the container, the Ideal Gas Law also holds as shown in Equation 30. Let T denote the temperature in the cabin and m the mass of the air within the cabin. Furthermore, the output cabin temperature T_o is equal to T.

$$P_c = \frac{mTR}{V} \tag{30}$$

$$T_o = T \tag{31}$$

The thermal energy of the cabin Q_{cab} is shown in Equation 32, which is constant as enforced by Equation 33. F_l is the air lost to the environment to maintain pressure.

$$Q_{cab} = C_{air} m T = \frac{C_{air} P_c V}{R} = \text{constant} \tag{32}$$

$$0 = \frac{d}{dt} Q_{cab} = C_{air} \Big(F_i T_i + F_o T_o + F_l T_o \Big) + n Q_{pass} + dQ \tag{33}$$

Lastly, the cabin mass m obeys the following differential equation.

$$\frac{d}{dt} m = F_i + F_o + F_l \tag{34}$$

Fan: A fan enforces a specified flow rate, which is given as a parameter F_f. Furthermore, the input and output temperatures are equal.

$$F_f = F_i = -F_o \tag{35}$$

$$T_o = T_i \tag{36}$$

Static Models

Ducts and Materials: The air ducts allow gaseous air to flow throughout the system by connecting each component together. In order to reduce complexity of the dynamic Modelica model described above, abstraction is used to simplify the ducts. Specifically, a static library of ducts with various sizes and materials is created. Each duct j in the library $\mathcal{D}$ is represented by the following tuple: $\{c_j,\ w_j,\ r_j,\ T_{j,max},\ F_{j,max},\ SP_j\ \}$, which denotes the cost per area, weight per area, radius, max temperature, max flow rate and specific heat, respectively. It is assumed that the maximum flow rate is directly related to the duct radius. This assumption simplifies the plant model by abstracting away the pressure drop across each duct. Let $\mathcal{M}$ be heat exchanger material library, which may consist of the unique materials (i.e. unique pairs of $\{w_j, SP_j\}$) in $\mathcal{D}$.

Furthermore, in order to ensure the size of the ECS is reasonable, the physical structure as shown in Figures 5 and 6 is assumed. Let $\mathcal{T}$ denote the set of ducts $\{1, 2, 3, 4, 5\}$ in the topology. The lengths of ducts 1, 2 and 5 are given as l_1, l_2 and l_5, respectively. However, since the length of the heat exchanger L_{HX} is a design parameter, the lengths of ducts 3 and 4 are initially unknown. It is assumed that $l_3 = l_4 = (l_2 - L_{HX})/2$.

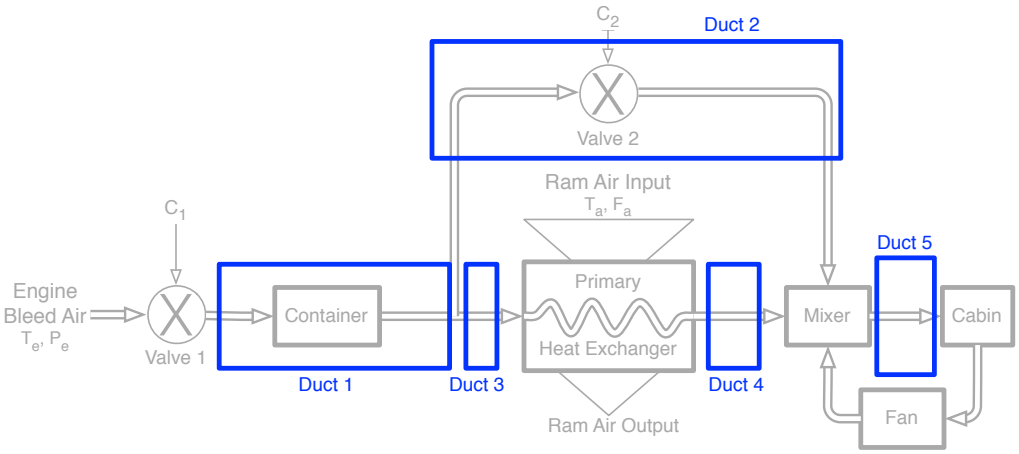

Figure 5: ECS Ducts

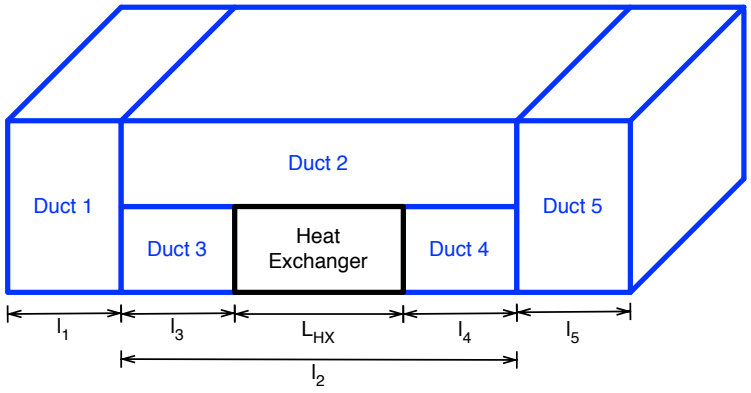

Figure 6: ECS Physical Structure and Ducts

4.2 Simulation-based Requirement Formalization

Steady-State Requirements: There are a few key requirements in ECS architecture design [9]. First, to ensure cabin comfort, it is required that the cabin reaches a comfortable steady-state temperature and maintains a desirable flow of fresh air. Furthermore for safety, the heat exchanger shall not freeze. Finally, the duct temperature and flow rate must not exceed a maximum value. In order to ensure steady-state is achieved, a quasi steady-state, where the magnitude of the time-derivatives are less than σ for $0 < \sigma << 1$, is defined. Note, the time derivative of a state variable is very easy to obtain in Modelica using the *der()* function. Furthermore, it is required that this quasi steady-state is achieved within 15 minutes of startup. Finally, the requirements are written as safety properties, meaning they should never be true.

1.0 - The cabin temperature $T_c(t)$ shall reach the interval [294K, 300K] and steady-state within 15 minutes of startup. The first requirement ensures a comfortable temperature is reached, while the second ensures steady-state.

$$\neg\Diamond_{[15min,\infty]}\big[T_c(t) < 294K \vee T_c(t) > 300K\big] \tag{37}$$

$$\neg\Diamond_{[15min,\infty]}\big[|\dot{T}_c(t)| > \sigma\big] \tag{38}$$

2.0 - The cabin input flow rate $F_c(t)$ shall reach the interval [0.8 kg/s, 1.2kg/s] and steady-state within 15 minutes of startup. The first requirement ensures an adequate flow rate is reached, while the second ensures steady-state.

$$\neg\Diamond_{[15min,\infty]}\big[F_c(t) < 0.8kg/s \vee F_c(t) > 1.2kg/s\big] \tag{39}$$

$$\neg\Diamond_{[15min,\infty]}\big[|\dot{F}_c(t)| > \sigma\big] \tag{40}$$

3.0 - The heat exchanger temperature $T_x(t)$ shall not fall below freezing in steady-state.

$$\neg\Diamond_{[15min,\infty]}\big[T_x(t) \leq 273K\big] \tag{41}$$

$$\neg\Diamond_{[15min,\infty]}\big[|\dot{T}_x(t)| > \sigma\big] \tag{42}$$

4.0 - Each duct temperature $T_i(t)$ and flow rate $F_i(t)$ shall not exceed the maximum allowed in steady-state. These requirements ensure the safety of each duct, where $i \in \mathcal{T}$ represents a duct in the system, $T_{i,max}$ represents a corresponding maximum temperature and $F_{i,max}$ represents a corresponding maximum flow rate.

$$\neg\Diamond_{[15min,\infty]}\big[T_i(t) > T_{i,max}\big], \forall i \in \mathcal{T} \tag{43}$$

$$\neg\Diamond_{[15min,\infty]}\big[F_i(t) > F_{i,max}\big], \forall i \in \mathcal{T} \tag{44}$$

4.3 Optimization Problem

Next, I formulate an optimization problem for the steady-state, open-loop design space exploration of the ECS. The overall design problem is a minimization of monetary cost, where the optimizer decides the optimal set of ducts, heat exchanger size and steady-state control subject to the component and system requirements. The control parameters of interest are the positions of each valve (C_1 and C_2) and the necessary amount of ambient ram air (F_a). The high-level SELECTION problem selects the ducts from a library as well as the heat exchanger material to minimize cost subject to a weight constraint. The low-level SIZING optimization sizes the heat exchanger (L_{HX}, A_{HX} and D_{HX}) and decides the steady-state control values subject to the system requirements and the behavioral model, while also minimizing cost. Figure 3 gives a high-level overview of the interaction between the two optimization problems. In this section, I first propose the entire optimization problem as one, and then describe how to separate it into two interacting subproblems.

ECS Optimization Problem: For duct selection, let $s_{i,j} = 1$ *iff* library duct $j \in \mathcal{D}$ is selected for a duct $i \in \mathcal{T}$ of the architecture, otherwise $s_{i,j} = 0$. Let, $s_{hx,j}$ denote the heat exchanger material decision variable for $j \in \mathcal{M}$, and let S denote the set of all binary design variables. Furthermore, the cost of a duct C_i for $i \in \mathcal{T}$ is shown in Equation 45, and similarly, the weight of a duct W_i is shown in Equation 46. The equations simply determine the area of material and multiply by the cost or weight per area of the selected material, assuming an appropriate thickness. Note, C_3 and C_4 are also functions of L_{HX} since $l_3 = l_4 = (l_2 - L_{HX})/2$, and for readability this is generalized for all C_i. The same is true for the weight.

$$C_i(S, L_{HX}) = 2\pi l_i \sum_{j\in\mathcal{D}} s_{i,j} c_j r_j \tag{45}$$

$$W_i(S, L_{HX}) = 2\pi l_i \sum_{j\in\mathcal{D}} s_{i,j} w_j r_j \tag{46}$$

Similarly, the cost and weight of the heat exchanger is defined in Equations 47 and 48 for the selection variable $s_{hx,j}$. In this equation, the area of a single tube $\pi D_{HX} L_{HX}$ is multiplied by the number of tubes A_{HX}/D_{HX}^2.

$$C_{HX}(S, A_{HX}, D_{HX}, L_{HX}) = \pi L_{HX} \frac{A_{HX}}{D_{HX}} \sum_{j\in\mathcal{M}} s_{hx,j} c_j \tag{47}$$

$$W_{HX}(S, A_{HX}, D_{HX}, L_{HX}) = \pi L_{HX} \frac{A_{HX}}{D_{HX}} \sum_{j \in \mathcal{M}} s_{hx,j} w_j \tag{48}$$

A_{HX}, D_{HX} and L_{HX} are continuous design variables along with the control variables C_1, C_2 and F_a, which makes up the continuous parameter space $\mathcal{P}$. Let $p = \{A_{HX}, D_{HX}, L_{HX}, C_1, C_2, F_a\}$ denote an assignment to each continuous design variable. Maximum and minimum bounds define $\mathcal{P}$ as a multi-dimensional box. Let $A_{HX,min}$, $D_{HX,min}$, $L_{HX,min}$, $C_{1,min}$, $C_{2,min}$ and $F_{a,min}$ define the lower bounds, while $A_{HX,max}$, $D_{HX,max}$, $L_{HX,max}$, $C_{1,max}$, $C_{2,max}$ and $F_{a,max}$ define the upper bounds. Let the cost function be the summation of the cost of each duct and the heat exchanger as shown in Equations 49.

$$C(S, A_{HX}, D_{HX}, L_{HX}) = C_{HX}(S, A_{HX}, D_{HX}, L_{HX}) + \sum_{i \in \mathcal{T}} C_i(S, L_{HX}) \tag{49}$$

Next, I discuss the constraints. First, the optimization must satisfy the Modelica behavioral model by $\mathcal{F}(\dot{x}, x, y, S, p) = 0$, where x is the set of differential variables, y is the set of algebraic variables, S is an assignment to the discrete variables and p is an assignment to the continuous parameters. Because it is known experimentally that the behavioral model achieves steady-state within 15 minutes of simulation time, the requirements can be simplified. Therefore, let $T_{c,ss} = T_c(15min)$, $T_{x,ss} = T_x(15min)$ and $F_{c,ss} = F_c(15min)$ denote the steady state cabin temperature, heat exchanger temperature and cabin flow rate, respectively and in accordance with the steady-state requirements formalized in STL. Similarly, let $T_{i,ss}$ and $F_{i,ss}$ represent the steady-state temperature and flow rate of duct i, $\forall\, i \in \mathcal{T}$. The set of steady-state requirements are shown in Equation 50.

$$
\begin{aligned}
&T_{c,ss} - 300 \leq 0 && (1)\\
&294 - T_{c,ss} \leq 0 && (2)\\
&F_{c,ss} - 1.2 \leq 0 && (3)\\
&0.8 - F_{c,ss} \leq 0 && (4)\\
&273 - T_{x,ss} \leq 0 && (5)\\
&T_{x,ss} - \sum_{j \in \mathcal{M}} s_{hx,j} T_{j,max} \leq 0 && (6)\\
&T_{i,ss} \leq \sum_{j \in \mathcal{D}} s_{i,j} T_{j,max}, \ \ \forall i \in \mathcal{T} && (8)\\
&F_{i,ss} \leq \sum_{j \in \mathcal{D}} s_{i,j} F_{j,max}, \ \ \forall i \in \mathcal{T} && (9)
\end{aligned}
\tag{50}
$$

Specifically, (1) and (2) specify the cabin must settle to a comfortable temperature within $[294, 300]K$. Requirements (3) and (4) specify the mass flow rate into the cabin must be within $[0.8, 1.2]kg/s$, while (5) and (6) specify the heat exchanger must not freeze or exceed the maximum temperature allowed by the selected material. Requirements (7) and (8) ensure the maximum temperature and flow rate of each topology duct is not exceeded. The weight requirement is formalized below given a maximum weight W_{max}.

$$
W(S, A_{HX}, D_{HX}, L_{HX}) = W_{HX}(S, A_{HX}, D_{HX}, L_{HX}) + \sum_{i \in \mathcal{T}} W_i(S, L_{HX}) \leq W_{max}
$$

Lastly, another set of constraints must be added to ensure a duct is selected for each duct in the topology $\mathcal{T}$ and a material for the heat exchanger, which is shown below in Equation 51.

$$
\sum_{j \in \mathcal{D}} s_{i,j} = 1, \ \ \forall i \in \mathcal{T} \ \text{ and } \ \sum_{j \in \mathcal{M}} s_{hx,j} = 1 \tag{51}
$$

Now, the optimization problem for ECS design is formally stated in Equation 52.

$$
\begin{aligned}
\underset{p\in\mathcal{P},S\in\{0,1\}^{(|\mathcal{T}|+1)\times|\mathcal{D}|}}{\text{minimize}} \quad & C(S, A_{HX}, D_{HX}, L_{HX}) \\
\text{subject to} \quad & \mathcal{F}(\dot{x}, x, y, S, p) = 0 \\
& W(S, A_{HX}, D_{HX}, L_{HX}) \leq W_{max} \\
& T_{c,ss} - 300 \leq 0 \\
& 294 - T_{c,ss} \leq 0 \\
& F_{c,ss} - 1.2 \leq 0 \\
& 0.8 - F_{c,ss} \leq 0 \\
& 273 - T_{x,ss} \leq 0 \\
& T_{x,ss} - \sum_{j\in\mathcal{M}} s_{hx,j} T_{j,max} \leq 0 \\
& T_{i,ss} \leq \sum_{j\in\mathcal{D}} s_{i,j} T_{j,max}, \;\; \forall i \in \mathcal{T} \\
& F_{i,ss} \leq \sum_{j\in\mathcal{D}} s_{i,j} F_{j,max}, \;\; \forall i \in \mathcal{T} \\
& \sum_{j\in\mathcal{D}} s_{i,j} = 1, \;\; \forall i \in \mathcal{T} \\
& \sum_{j\in\mathcal{M}} s_{hx,j} = 1
\end{aligned}
\tag{52}
$$

Next, I present a scheme to solve this nonlinear, mixed integer program with simulation in the loop. Specifically, I separate the problem into a binary architecture SELECTION problem and a continuous SIZING optimization. The SELECTION engine will solve for the binary matrix S and propose a candidate solution to the continuous SIZING engine to determine an assignment of $p \in \mathcal{P}$. In order to split the problem, the cost and weight functions are rewritten as shown in Equations 53 and 54, respectively.

$$
C(S, A_{HX}, D_{HX}, L_{HX}) = \frac{A_{HX} L_{HX}}{D_{HX}} C_{f1}(S) + L_{HX} C_{f2}(S) + C_{f3} \tag{53}
$$

$$
W(S, A_{HX}, D_{HX}, L_{HX}) = \frac{A_{HX} L_{HX}}{D_{HX}} W_{f1}(S) + L_{HX} W_{f2}(S) + W_{f3} \tag{54}
$$

First, I leverage a very convenient property of the cost and weight functions in Equations 53 and 54. It can be shown for our continuos design space $\mathcal{P}$ and component libraries $\mathcal{D}$ and $\mathcal{M}$, that the cost and weight functions in Equations 53 and 54 are monotonically increasing with L_{HX} and A_{HX} and monotonically decreasing in D_{HX}. Therefore, cost and weight are minimized when $A_{HX} = A_{HX,min}$, $L_{HX} = L_{HX,min}$ and $D_{HX} = D_{HX,max}$. However, it is unknown if this assignment satisfies the steady-state requirements, which requires simulation. Therefore, I propose the following setup for the bi-level optimization algorithm.

The process begins by solving the SELECTION problem assuming the best case scenario for minimal cost and weight, $A_{HX} = A_{HX,min}$, $L_{HX} = L_{HX,min}$ and $D_{HX} = D_{HX,max}$, which is denoted as $\hat{p}$. At iteration k, the SELECTION engine proposes the candidate solution S_k^* with cost C_k^*, along with the duct and system constraints, to the continuous SIZING engine to find a feasible heat exchanger size and steady state control. After executing the SIZING engine, one of three things can happen: (1) the SIZING engine returns no solution; (2) the SIZING engine returns a less than ideal solution i.e. at least one of the following is true $A_{HX}^* > A_{HX,min}$, $L_{HX}^* > L_{HX,min}$ or $D_{HX}^* < D_{HX,max}$; or (3) the SIZING returns the ideal solution i.e. $A_{HX}^* = A_{HX,min}$, $L_{HX}^* = L_{HX,min}$ and $D_{HX}^* = D_{HX,max}$.

Under case (1), the SIZING engine returns no solution and it must ask the SELECTION engine for another candidate solution and rerun the continuous optimization on the new solution. In this case, a super set of the proposed solution S_k^* is added to a set of constraints $\mathcal{U}$ enforcing the rejected assignment is not found again. In addition to simply adding the exact SELECTION solution to the set of rejected assignments and constraining against it in the next iteration, it would be very useful to explore different FEEDBACK functions to reject multiple discrete assignments at each SELECTION iteration. If case (2) occurs, the cost must re-evaluated since at least one of the following are A_{HX}^*, L_{HX}^* or D_{HX}^* are no longer ideal. Let this updated cost from the sizing engine be $C_k^{**} > C_k^*$, and let C_{best} be the current lowest cost found. If this solution is the current lowest cost option found, i.e. $C_k^{**} < C_{best}$, it may not be the overall optimal solution and termination is not safe. In general, there may be a more expensive discrete assignment, i.e $C_{k+z}^* \geq C_k^*$ for some $z > 0$, but the SIZING engine is able to further reduce the cost, i.e $C_{k+z}^{**} < C_k^{**}$. Therefore, the SELECTION engine must be queried for another solution by adding S_k^* to $\mathcal{U}$. This process repeats until the SELECTION engine returns a best case cost that is more expensive than the current lowest cost solution, i.e. $C_k^* > C_{best}$, or if Case (3) $C_k^* = C_k^{**}$ occurs. At this point, a near optimal solution has been found, and it is safe to terminate. The algorithm is shown explicitly in Algorithm 1.

It can be proven that if $C_k^* > C_{best}$, it is safe to terminate the SELECTION optimizer early. Because the SELECTION function assigns the best case assignment

($L_{HX} = L_{HX,min}, A_{HX} = A_{HX,min}, D_{HX} = D_{HX,max}$) and $C(S, A_{HX}, D_{HX}, L_{HX})$ is monotonically increasing in A_{HX} and L_{HX} and decreasing in D_{HX}, the continuous problem cannot decrease the cost i.e. $C_k^* \leq C_k^{**} \; \forall \; k \geq 1$. Therefore, since $C_k^* > C_{best}$ is assumed, $C_k^{**} > C_{best}$ is also true, and it is safe to terminate. Similarly, it is safe to terminate if $C_k^* = C_k^{**}$ because further iterations can never decrease the cost. Next, I go into the exact details of the SELECTION and SIZING optimization engines within Algorithm 1.

Algorithm 1 *ECS Architecture Exploration*

Input: Duct/Material Library $\mathcal{D}/\mathcal{M}$, Steady State Requirements $\mathcal{R}$, Modelica Model $\mathcal{F}$, Topology $\mathcal{T}$, W_{max}

Output: Duct and material assignment matrix S_{best}, cost C_{best} and parameters p_{best}

$\mathcal{U} \leftarrow \{\}$
$C_{best} \leftarrow \infty$
$S_{best} \leftarrow \{\}$
$p_{best} \leftarrow \{\}$
$k \leftarrow 1$
while true **do**
  $(S_k^*, C_k^*) \leftarrow$ SELECTION($\mathcal{D}, \mathcal{M}, \mathcal{U}, \mathcal{T}, W_{max}, \hat{p}$)
  if $S_k^* == \{\}$ and $C_{best} == \infty$ **then**
    return NO SOLUTION
  else if $C_k^* \geq C_{best}$ or $S_k^* == \{\}$ **then**
    return ($S_{best}, p_{best}, C_{best}$)
  $(p_k^*, C_k^{**}) \leftarrow$ SIZING($\mathcal{F}, \mathcal{R}, W_{max}, \mathcal{D}, \mathcal{M}, S_k^*$)
  if $C_k^{**} < C_{best}$ **then**
    $C_{best} \leftarrow C_k^{**}$
    $S_{best} \leftarrow S_k^*$
    $p_{best} \leftarrow p_k^*$
  if $C_k^* == C_k^{**}$ **then**
    return ($S_{best}, p_{best}, C_{best}$)
  $\mathcal{U} = \mathcal{U} \cup$ FEEDBACK($S_k^*, p_k^*, \mathcal{F}, \mathcal{D}, \mathcal{M}$)
  $k \leftarrow k + 1$

SELECTION: The discrete SELECTION engine selects ducts and the heat exchanger material from a library using the various component attributes including weight, cost, radius, max temperature and max flow rate, which is shown in Equation 55 for a given maximum weight W_{max} and duct lengths l_i for $i \in \{1, 2, 5\}$.

$$
\begin{aligned}
\underset{S \in \{0,1\}^{(|\mathcal{T}|+1)\times|\mathcal{D}|}}{\text{minimize}} \quad & C(S, A_{HX,min}, D_{HX,max}, L_{HX,min}) \\
\text{subject to} \quad & W(S, A_{HX,min}, D_{HX,max}, L_{HX,min}) \leq W_{max} && (1) \\
& \sum_{j \in \mathcal{D}} s_{3,j} F_{j,max} - s_{4,j} F_{j,max} = 0 && (2) \\
& \sum_{j \in \mathcal{D}} s_{1,j} F_{j,max} - s_{5,j} F_{j,max} \leq 0 && (3) \\
& \sum_{j \in \mathcal{D}} s_{i,j} T_{j,max} \geq T_e, \forall i \in \{1, 2, 3\} && (4) \\
& \sum_{j \in \mathcal{D}} s_{i,j} = 1, \ \ \forall i \in \mathcal{T} && (5) \\
& \sum_{j \in \mathcal{M}} s_{hx,j} = 1 && (6) \\
& f_u(S, \mathcal{D}, \mathcal{M}) \leq 0, \ \ \forall \, u \in \mathcal{U} && (7)
\end{aligned}
\tag{55}
$$

Constraint (1) ensures the system does not exceed the maximum weight. Constraints (2) and (3) help prune the search space by taking advantage of the balance equations enforcing conservation of flow. Such constraints can be found by pre-processing the topology. For example, the flow through duct 3 must equal the flow through duct 4 because no mass is accumulated in the heat exchanger. Therefore, the size of duct 3 should be equal to the size of duct 4. A similar relationship can be found for ducts 1 and 5, however, since duct 5 must be large enough for the recirculated air from the cabin, it becomes an inequality. Furthermore, the topology and component models can also be pre-processed to realize the valves do not change the air temperature. Therefore ducts 1, 2 and 3, must have a maximum temperature to accommodate the input environment temperature T_e, which is shown in constraint (4). Constraints (5) and (6) ensure a duct is selected for each duct in the topology as well as a material for the heat exchanger. Lastly, Constraint (7) ensures that rejected assignments are not found again. Let $\mathcal{U}$ represent a set of integer-linear constraints that rule out rejected assignments from the SIZING engine. Then, let $f_u(S, \mathcal{D}, \mathcal{M}) \leq 0 \in \mathcal{U}$ denote a constraint that enforces a previous solution is not found again. For example, $f_u(S, \mathcal{D}, \mathcal{M}) = s_{1,1}$ would ensure library duct 1, is never again selected for topology duct 1. Furthermore,

$$f_u(S, \mathcal{D}, \mathcal{M}) = 1.0 - \sum_{j \in \mathcal{D}} s_{1,j} F_{j,max}$$

would ensure the max flow through topology duct 1 is at least. 1.0 kg/s. The rejected assignments come from the SIZING engine if a solution that satisfies the constraints cannot be found. This problem can be efficiently solved in IBM CPLEX [30].

SIZING: The SIZING engine looks to determine the heat exchanger's cross-sectional area (A_{HX}), length (L_{HX}) and tube diameter (D_{HX}) as well as determining the steady control parameters including C_1, C_2 and F_a. The optimization aims to minimize the overall cost of the ECS, and therefore, it is sufficient to minimize $C(S^*$, A_{HX}, D_{HX}, $L_{HX})$, where S^* is the solution found by the SELECTION engine. Recall, $p = \{A_{HX}, L_{HX}, D_{HX}, F_a, C_1, C_2\}$ is a configuration assignment to the continuous design variables within the box $\mathcal{P}$. The optimization problem is shown in Equation 56

$$\begin{aligned}
\underset{p \in \mathcal{P}}{\text{minimize}} \quad & C(S^*, A_{HX}, D_{HX}, L_{HX}) \\
\text{subject to} \quad & \mathcal{F}(\dot{x}, x, y, S, p) = 0 \\
& T_{c,ss} - 300 \leq 0 && (1) \\
& 294 - T_{c,ss} \leq 0 && (2) \\
& F_{c,ss} - 1.2 \leq 0 && (3) \\
& 0.8 - F_{c,ss} \leq 0 && (4) \\
& 273 - T_{x,ss} \leq 0 && (5) \\
& T_{x,ss} - \sum_{j \in \mathcal{M}} s^*_{hx,j} T_{j,max} \leq 0 && (6) \\
& T_{i,ss} - \sum_{j \in \mathcal{D}} s^*_{i,j} T_{j,max} \leq 0, \ \ \forall i \in \mathcal{T} && (7) \\
& F_{i,ss} - \sum_{j \in \mathcal{D}} s^*_{i,j} F_{j,max} \leq 0, \ \ \forall i \in \mathcal{T} && (8) \\
& W(S^*, A_{HX}, D_{HX}, L_{HX}) - W_{max} \leq 0 && (9)
\end{aligned} \tag{56}$$

Constraints (1) - (9), denoted by the set $\mathcal{R}$, specify the system requirements. Specifically, (1) and (2) specify the cabin must settle to a comfortable temperature within $[294, 300]K$. Requirements (3) and (4) specify the mass flow rate into the cabin must be within $[0.8, 1.2]kg/s$, while (5) and (6) specify the heat exchanger

must not freeze or exceed the maximum temperature. Requirements (7) and (8) ensure the maximum temperature and pressure of each topology duct is not exceeded. Finally, requirement (9) ensures the weight requirement is still met.

To solve this problem, I take advantage of the Nelder-Mead [31] direct search algorithm, which is already built into JModelica. Nelder-Mead is a very convenient optimization algorithm for black-box functions and simulation in the loop because it does not require derivatives. However, since Nelder-Mead does not support explicit constraints, I resort to adding them as a penalty to the objective function of an unconstrained optimization. Let $f_r(x, y, S, p)$ for $r \in \mathcal{R}$ denote the set of constraints such that $f_r(x, y, S, p) \leq 0 \ \ \forall r \in \mathcal{R}$. Let λ_r be the penalty associated with constraint r. An advantage to using a penalty function is that it is easy to obtain a reason for unfeasibility and which constraints are violated. The unconstrained optimization subject to the behavioral model becomes the following:

$$\begin{aligned} \underset{p \in \mathcal{P}}{\text{minimize}} \quad & C(S_k^*, A_{HX}, D_{HX}, L_{HX}) + \sum_{r \in \mathcal{R}} \lambda_r max\{0, f_r(x, y, S, p)\} \\ \text{subject to} \quad & \mathcal{F}(\dot{x}, x, y, S, p) = 0 \end{aligned} \tag{57}$$

Non-convex, black-box optimizations can be very sensitive to the initial guess. Therefore, the sizing algorithm is run on a user-defined number of iterations and initial guesses, which will largely depend on the structure of the Modelica behavioral model and the continuous design space.

If the SIZING engine returns an unfeasible solution, the current assignment $\mathcal{S}$ is sent to a FEEDBACK function and the output is added to the rejected set, i.e $\mathcal{U} \leftarrow \text{FEEDBACK}(S_k^*, p_k, \mathcal{F}, \mathcal{D}, \mathcal{M}) \cup \mathcal{U}$, of the SELECTION optimization problem. Note, the FEEDBACK function may add more than one constraint to $\mathcal{U}$ at each iteration. Then, the SELECTION optimization is executed again and followed by another SIZING optimization. The process continues until an optimal solution is found. In what follows, two different FEEDBACK functions are evaluated.

FEEDBACK: The most simple FEEDBACK function is to only enforce the exact previous discrete assignment is not found again, which is referred to as NAIVE feedback. Let $\mathcal{A}(S) = \{s_{i,j}, s_{hx,l} \mid s_{i,j} = 1, \ s_{hx,l} = 1, i \in \mathcal{T}, j \in \mathcal{D}, l \in \mathcal{M}\}$

represent a SELECTION assignment, i.e. only the variables in S that are assigned to one. More specifically, NAIVE feedback is shown in Equation 58.

$$\begin{aligned} &\text{NAIVE}(S) = \sum_{s \in \mathcal{A}(S)} s \leq |\mathcal{A}(S)| - 1 \\ &s.t.\ \mathcal{A}(S) = \{s_{i,j}, s_{hx,l} \mid s_{i,j} = 1,\ s_{hx,l} = 1, i \in \mathcal{T}, j \in \mathcal{D}, l \in \mathcal{M}\} \end{aligned} \tag{58}$$

However, this method will only eliminate one assignment at a time, which is inefficient, especially if the simulations are expensive or the discrete SELECTION space is large. If instead, a smaller, more compact set can be obtained, many discrete assignments can be eliminated at each iteration, which will drastically improve runtime. Next, I will describe one method to help prune the discrete search space. Furthermore, I will leave other methods as future work.

One way of eliminating multiple discrete assignments at each iteration would be to enforce only the combination of selected components whose requirements are violated in the SIZING engine are not repeated. I will refer to this approach as IMPATIENT feedback and will describe an approach with this idea in mind. In general, this approach may rule out feasible solutions, especially if the root cause of unfeasibility is not due to the components whose requirements are violated. However, I define an axiom related to duct size and maximum flow, which rules out the risk of missing feasible solutions due to conservation of mass. Since cost is minimized, this approach amounts to increasing the duct size, which increases cost. Intuitively, if the flow constraint on Duct 1 in Figure 5 is violated, increasing the maximum flow (i.e. duct size) for any of the other ducts will not fix the violation on Duct 1. Therefore, it is far more efficient to only constrain against this current size for Duct 1 instead of the entire discrete solution. A similar thought process can be used for the other ducts and the temperature requirements to define FEEDBACK rules. In fact, the only way increasing the size of a duct, whose constraints are satisfied, may help push another duct to feasibility, is if the two ducts are in parallel. For example, lets assume ducts A and B are in parallel and duct A's flow requirement is violated. Assuming conservation of mass, increasing the maximum flow through duct B gives the control the ability to route more air through duct B and reduce the amount through duct A.

More formally, let $F^k_{i,ss}$ represent the steady-state flow rate through duct $i \in \mathcal{T}$ at SELECTION iteration k, and let $\mathcal{V}^k_F$ be the corresponding set of ducts with

violated flow constraints, which is shown in Equation 59. Let the superscript k of $s_{i,j}^k$ represent the SELECTION solution at iteration k.

$$\mathcal{V}_F^k = \{i \in \mathcal{T} \mid F_{i,ss}^k > \sum_{j \in \mathcal{D}} s_{i,j}^k F_{j,max}, i \in \mathcal{T}, j \in \mathcal{D}\} \tag{59}$$

Let $\mathcal{H}_i$ denote the set of ducts in parallel with duct i, including duct i. The set $\mathcal{H}_i$ can be determined from the balance equations of the topology. For example, in this balance equation of flow variables, $f_1 - f_2 - f_3 = 0$, it is known f_2 and f_3 are in parallel and both are in series with f_1. For ducts without any other duct in parallel (i.e. $\mathcal{H}_i = \{i\}$), a constraint is added to $\mathcal{U}$ to increase its duct size as shown in Equation 60, and let $\mathcal{K}_{\mathcal{FS}}$ denote the set of such constants. Let the superscript $k+1$ of $s_{i,j}^{k+1}$ represent the next iteration's SELECTION variables. Note, at iteration $k+1$, the solution at the previous iteration k is constant. Therefore, constraints are added to eliminate all assignments with the previous violated maximum flow rates. Intuitively, temperature constraints behave the same way as series flow constraints. Therefore, let $\mathcal{V}_T$ represent the ducts with temperature violations and $\mathcal{K}_T$ represent this set of constraints.

$$\mathcal{K}_{FS} = \bigcup_{i \in \mathcal{V}_F^k \;\; s.t. \;\; \mathcal{H}_i = \{i\}} \sum_{j \in \mathcal{D}} s_{i,j}^{k+1} F_{j,max} > \sum_{j \in \mathcal{D}} s_{i,j}^k F_{j,max} \tag{60}$$

For ducts in parallel, one of them must increase its size, which is shown in Equation 61. Let this set of constraints be $\mathcal{K}_{\mathcal{FP}}$.

$$\mathcal{K}_{\mathcal{FP}} = \bigcup_{n \in \mathcal{V}_F^k \;\; s.t. \;\; \mathcal{H}_n \neq \{n\}} \sum_{i \in \mathcal{H}_n} \sum_{j \in \mathcal{D}} s_{i,j}^{k+1} F_{j,max} > \sum_{i \in \mathcal{H}_n} \sum_{j \in \mathcal{D}} s_{i,j}^k F_{j,max} \tag{61}$$

Finally, the IMPATIENT feedback method is shown in Equation 62. Note, IMPATIENT depends on the selected sizing p, behavioral model $\mathcal{F}$ and the libraries $\mathcal{D}$ and $\mathcal{M}$ because it must check (simulate) for violations. If there are no duct violations, the NAIVE approach is used. In the Experimental Results section, I evaluate this heuristic for the ECS.

$$\text{IMPATIENT}(S, p, \mathcal{F}, \mathcal{D}, \mathcal{M}) = \begin{cases} \mathcal{K}_T \cup \mathcal{K}_{FS} \cup \mathcal{K}_{FP} & \text{if } \mathcal{V}_T \cup \mathcal{V}_F \neq \{\} \\ \text{NAIVE}(S) & \text{else} \end{cases} \tag{62}$$

4.4 Algorithm Properties

Unfortunately, the SIZING function implemented with Nelder-Mead is neither sound nor complete because it is dealing with an infinite design space with black-box functions. However, the methodology in no way restricts users to particular optimization algorithms. In fact, designers are encouraged to exploit the structure of the problem and leverage appropriate optimization algorithms or approximations for better guarantees and faster results. I outline this as a future work for the ECS.

Chapter 5
Experimental Results

In this section, I illustrate the experimental results from implementing the Architecture Selection methodology on the ECS. First, the problem is setup with the various constants within the ECS Modelica model and the component library. The duct library is shown in Table 1. Note, the same library is used for the heat exchanger material selection. Materials 1, 4, 7, 10 and 13 are aluminum, 2, 5, 8, 11 and 14 are steel and 3, 6, 9, 12 and 15 are composite. Let the heat exchanger material library $\mathcal{M}$ consist of materials 1 and 2, only. Next, a table of the fixed constants and parameters within the Modelica behavioral model are shown in Table 2. Note, the heat exchanger mass M_{HX} and specific heat C_m depend on the material selected. Table 3 shows the continuous design space $\mathcal{P}$, and a weight requirement of $w_{max} = 44.5\ kg$ is used. Lastly, the duct lengths are: $l_1 = 0.1\ m$, $l_2 = 0.5\ m$ and $l_5 = 0.25\ m$.

ID	r_j (m)	$F_{j,max}$ (kg/s)	$T_{j,max}$ (K)	c_j(\$/m^2)	w_j (kg/m^2)	SP_j(J/(kg K))
1	0.1660	0.400	455	5.50	2.17	900
2	0.1660	0.400	800	3.00	3.85	450
3	0.1660	0.400	330	41.50	0.75	NA
4	0.2630	1.250	455	5.50	2.17	900
5	0.2630	1.250	800	3.00	3.85	450
6	0.2630	1.250	330	41.50	0.75	NA
7	0.2286	0.870	455	5.50	2.17	900
8	0.2286	0.870	800	3.00	3.85	450
9	0.2286	0.870	330	41.50	0.75	NA
10	0.2032	0.695	455	5.50	2.17	900
11	0.2032	0.695	800	3.00	3.85	450
12	0.2032	0.695	330	41.50	0.75	NA
13	0.1534	0.304	455	5.50	2.17	900
14	0.1534	0.304	800	3.00	3.85	450
15	0.1534	0.304	330	41.50	0.75	NA

Table 1: Duct Library $\mathcal{D}$. Heat exchanger material library $\mathcal{M}$ consists of materials 1 and 2.

The boolean SELECTION problem is solved very quickly using IBM CPLEX. Over an entire design space exploration execution, the average time to solve a single SELECTION problem in CPLEX is on the order of a few milliseconds. Because JModelica is implemented in Python, I leveraged PuLP [32], which is a Python API for CPLEX, to connect the two optimization problems. Not surprisingly, nearly all

Component	Name	Value	Units
All	C_{air}	1003.5	J/(kg K)
All	R	287.058	J/(kg K)
Container	V	0.0049	m^3
Heat Exchanger	a_h	41.987×10^3	W/(K m^2)
Heat Exchanger	b_h	23.049×10^3	J/(K m^2 Kg)
Heat Exchanger	a_c	6.787×10^3	W/(K m^2)
Heat Exchanger	b_c	19.409×10^3	J/(K m^2 Kg)
Heat Exchanger	ν	2.5×10^{-5}	m/s^2
Cabin	V	141.58	m^3
Cabin	Q_{pass}	90	W
Cabin	n	200	-
Cabin	dq	0	W
Cabin	P	101.325	kPa
Fan	F_f	0.3042	kg/s
Environment	T_e	450	K
Environment	P_e	350	kPa
Environment	T_a	240	K

Table 2: Model Constants

of the computational effort is spent on simulation in the SIZING engine, which is very characteristic of black-box optimization problems with simulation in the loop.

A single SIZING optimization trace using the Nelder-Mead algorithm is shown in Figures 7 and 8 for $\lambda_r = 1000 \; \forall \; r \in \mathcal{R}$. Figure 7 shows how the optimal design parameters vary across each iteration for a given initial guess. Loosely speaking, the goal is to minimize L_{HX} and A_{HX} and maximize D_{HX}, all while determining the steady-state control values of the valves and ram air usage. The red lines indicate the parameter bounds ($\mathcal{P}$), while the blue lines indicate the optimization traces. However, the performance and component requirements must be satisfied as well. Figure 8 shows how the penalty function is able to drive the optimization parameters from a region where the cabin and heat exchanger requirements are all violated into a region where they are satisfied.

Name	A_{HX} (m^2)	D_{HX} (m)	L_{HX} (m)	F_4 (kg/s)	C_1 $(msK^{0.5})$	C_2 $(msK^{0.5})$
Lower Bound	0.15	0.001	0.20	0.5	0.01	0.01
Upper Bound	0.30	0.005	0.40	1.0	0.08	0.08

Table 3: Continuous Design Space $\mathcal{P}$

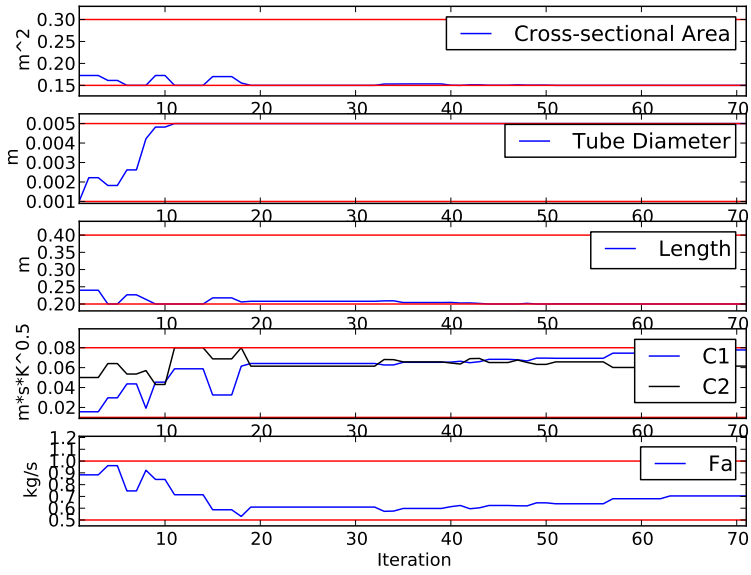

Figure 7: Single Nelder-Mead SIZING Parameter Optimization Trace

Next, the two optimization problems are coupled together to find an optimal ECS architecture using both NAIVE and IMPATIENT feedback. The SIZING engine is executed on three random starting points for each discrete solution. The initial guess includes $A_{HX} = A_{HX,min}$, $L_{HX} = L_{HX,min}$ and $D_{HX} = D_{HX,max}$, while F_a, C_1 and C_2 are randomly chosen from a uniform distribution. Initially, Algorithm 1 was executed on the first six library elements in Table 1. For the NAIVE approach, where the feedback from the SIZING engine only eliminates one discrete assignment at a time, Algorithm 1 converged to a solution after 17 SELECTION iterations and 51 SIZING iterations. The optimization took 3.72 hours to explore 62,496 total configurations. The optimal feasible cost was found to be $112.39, with a SELECTION assignment of $\{s_{1,4}, s_{2,1}, s_{3,4}, s_{4,4}, s_{5,4}, s_{hx,1}\}$ and a SIZING of $\{A_{HX} = 0.15\ m^2$, $D_{HX} = 0.005\ m$, $L_{HX} = 0.20\ m$, $F_a = 0.72\ kg/s$, $C_1 = 0.08\ m \cdot s \cdot K^{0.5}$, $C_2 = 0.08\, m \cdot s \cdot K^{0.5}\}$. Next, the IMPATIENT feedback approach was implemented, which converged to the same optimal cost, but only took 2 SELECTION iterations and 0.36 hours.

Finally, I show how the runtime scales with the size of the library from Table 1 in Table 4. There is a very large reduction in runtime for the IMPATIENT feedback function, especially in cases where the NAIVE function timed out. For

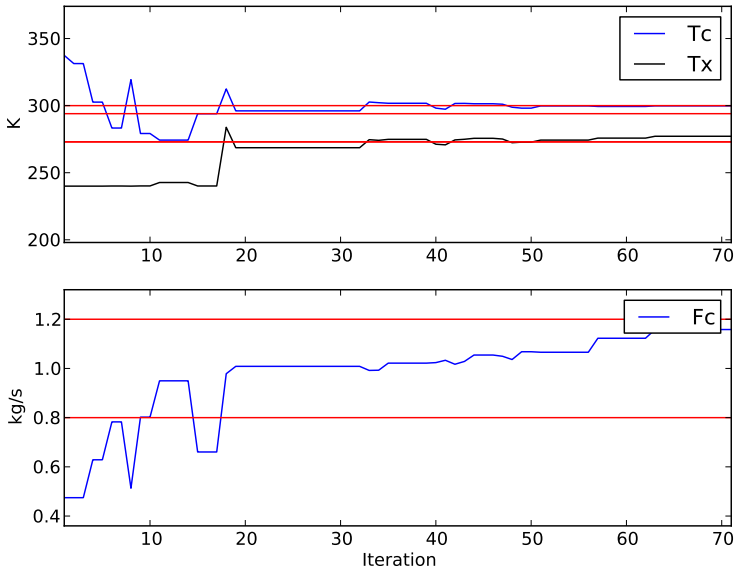

Figure 8: Single Nelder-Mead SIZING Requirement Optimization Trace

each library, there is more than an order of magnitude reduction in design runtime. All tests were executed on a Intel Xeon 3.59 GHz processor with 24 GB of RAM.

Library		**NAIVE**				**IMPATIENT**			
Size	**Discrete Choices**	**Cost**	**Discrete Iterations**	**Simulations**	**Run Time (hr)**	**Cost**	**Discrete Iterations**	**Simulations**	**Run Time (hr)**
6	15,552	$112.39	17	62,496	3.72	$112.39	2	5,812	0.36
9	118, 098	$112.03	72	257,141	15.42	$112.03	4	13,329	0.83
12	497,664	18hr Timeout				$111.63.	6	21,418	1.34
15	1,518,750	21hr Timeout				$111.14	9	31,874	1.91

Table 4: Runtime vs. Library Size

Chapter 6
Conclusion

6.1 Conclusion

In conclusion, I first summarized a design methodology based on principals of PBD to address the challenges associated with next generation CPS. The methodology consists of three steps, each leveraging parameterized, multi-view dynamic and static component models. Then, I proposed an extension with respect to the previous work to support both a discrete and continuous design space for Architecture Selection. The proposed approach is an iterative process between a discrete architecture SELECTION engine and a continuous SIZING engine, which is implemented on an aircraft ECS. The SELECTION engine selects a set of air ducts and a material for the heat exchanger to minimize cost subject to a weight requirement. Then, the SIZING engine attempts to find a heat exchanger size, valve positions and ambient ram flow rate that satisfies the system and component requirements, while also minimizing cost. When the SIZING engine fails to find a feasible solution, I explore an IMPATIENT feedback method to prune out many discrete solutions each at iteration by using conservations laws. I show the IMPATIENT method can improve the design runtime by more than an order of magnitude when deployed on the ECS.

6.2 Future Work

In the future, I plan to extend the design problem to the full patent ECS plant shown in Figure 1, which would include developing turbine, compressor and moisture/humidity models. I also would like to explore the design for transient behavior in addition to steady-state. In this direction, it would be necessary to formalize transient requirements as well as designing the low-level control protocol such as PID control gains to close the loop. Finally, another possible direction is to explore the performance of various optimization algorithms and feedback functions.

References

1. J. Warner, "Environmental Control System Condensing Cycle," Patent EP 0 542 909 B1, 09 28, 1994.
2. P. Nuzzo, M. Xu, N. Ozay, J. B. Finn, A. Sangiovanni-Vincentelli, R. Murray, A. Donze, and S. Seshia, "A Contract-Based Methodology for Aircraft Electric Power System Design," *IEEE Access*, November 2014. [Online]. Available: http://icyphy.org/pubs/35.html
3. P. Nuzzo, A. Sangiovanni-Vincentelli, and R. Murray, "Methodology and Tools for Next Generation Cyber-Physical Systems: The iCyPhy Approach," *INCOSE*, June 2015.
4. A. Fisher, C. Jacobson, E. A. Lee, R. Murray, A. Sangiovanni-Vincentelli, and E. Scholte, "Industrial Cyber-Physical Systems - iCyPhy," in *Proc. Complex Systems Design & Management (CSD&M)*. Springer, December 2013, pp. 21–37, paris, France. [Online]. Available: http://icyphy.org/pubs/34.html
5. K. Forsberg, H. Mooz, D. K. Forsberg, M. Harold, and M. Co-principals, "System Engineering for Faster, Cheaper, Better," 1998.
6. A. Sangiovanni-Vincentelli, "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design," *Proc. IEEE*, no. 3, pp. 467–506, 2007.
7. A. Benveniste, W. Damm, A. Sangiovanni-vincentelli, D. Nickovic, and R. Passerone, "Contracts for the Design of Embedded Systems Part I: Methodology and Use Cases."
8. P. Nuzzo, A. Sangiovanni-Vincentelli, X. Sun, and A. Puggelli, "Methodology for the Design of Analog Integrated Interfaces Using Contracts," *Sensors Journal, IEEE*, vol. 12, no. 12, pp. 3329–3345, Dec 2012.
9. M. Dechow and C. Nurcombe, "Aircraft environmental control systems," in *Air Quality in Airplane Cabins and Similar Enclosed Spaces*, ser. The Handbook of Environmental Chemistry, M. Hocking, Ed. Springer Berlin Heidelberg, 2005, vol. 4H, pp. 3–24. [Online]. Available: http://dx.doi.org/10.1007/b107234
10. A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen, "Contracts for System Design," Research Report RR-8147, Nov. 2012. [Online]. Available: https://hal.inria.fr/hal-00757488
11. A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, ser. SFCS '77. Washington, DC, USA: IEEE Computer Society, 1977, pp. 46–57. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1977.32

12. O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *In: Proceedings of FORMATS-FTRTFT. Volume 3253 of LNCS*. Springer, 2004, pp. 152–166.
13. T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, “TuLiP: A Software Toolbox for Receding Horizon Temporal Logic Planning,” in *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’11. New York, NY, USA: ACM, 2011, pp. 313–314. [Online]. Available: http://doi.acm.org/10.1145/1967701.1967747
14. A. Donzé, “Breach, a Toolbox for Verification and Parameter Synthesis of Hybrid Systems,” in *Proceedings of the 22Nd International Conference on Computer Aided Verification*, ser. CAV’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 167–170. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14295-6_17
15. L. Shang, G. Liu, and P. Hodal, “Development of High Performance Aircraft Bleed Air Temperature Control System With Reduced Ram Air Usage,” *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 438–445, March 2010.
16. Y. Tu and G. P. Lin, “Dynamic Simulation of Aircraft Environmental Control System Based on Flowmaster,” *Journal of Aircraft*, vol. 48, no. 6, pp. 2031–2041, Nov. 2011. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.C031433
17. O. Mickelin, N. Ozay, and R. Murray, “Synthesis of correct-by-construction control protocols for hybrid systems using partial state information,” in *American Control Conference (ACC), 2014*, June 2014, pp. 2305–2311.
18. J. V. C. Vargas and A. Bejan, “Thermodynamic optimization of finned crossflow heat exchangers for aircraft environmental control systems,” vol. 22, pp. 657–665, 2001.
19. R. Romani, L. Carlos, and S. Goes, “CABIN TEMPERATURE CONTROL MODEL FOR COMMERCIAL,” no. Cobem, pp. 9311–9326, 2013.
20. D. Dias, E. L. Zaparoli, M. E. Gomes, W. H. L. Turcio, E. Brasileira, and B. F. Lima, “DYNAMIC MODELING OF AN AERONAUTICAL AIR CONDITIONING SIMPLE AIR CYCLE MACHINE,” no. 1995, 2010.
21. I. Pérez-Grande and T. J. Leo, “Optimization of a commercial aircraft environmental control system,” *Applied Thermal Engineering*, vol. 22, no. 17, pp. 1885–1904, Dec. 2002. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1359431102001308
22. C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

23. Matlab/simulink - simulation and model-based design. [Online]. Available: http://www.mathworks.com/products/simulink
24. Modelica - a unified object-oriented language for systems modeling. [Online]. Available: https://www.modelica.org/
25. Dymola. [Online]. Available: http://www.modelon.com/
26. Jmodelica. [Online]. Available: http://www.jmodelica.org/
27. Mathematica - systemmodeler. [Online]. Available: http://www.wolfram.com/system-modeler/
28. Openmodelica. [Online]. Available: https://www.openmodelica.org/
29. R. Fox and A. McDonald, *Introduction to fluid mechanics*. Wiley, 1985. [Online]. Available: http://books.google.com/books?id=JZlRAAAAMAAJ
30. IBM ILOG CPLEX Optimizer. [Online]. Available: www.ibm.com/software/integration/optimization/cplex-optimizer/
31. J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.
32. *PuLP*. [Online]. Available: https://pypi.python.org/pypi/PuLP