Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation

Peter Kronström

# Airflow Simulation Tool for Intelligent Air Quality Monitoring System

Master's Thesis
Espoo, August 1, 2016

Supervisor: Ville Kyrki, D.Sc. (Tech.)
Advisor: Juho Pentikäinen, M.Sc. (Tech.)

Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation

ABSTRACT OF
MASTER'S THESIS

| | | | |
|---|---|---|---|
| **Author:** | Peter Kronström | | |
| **Title:** | Airflow Simulation Tool for Intelligent Air Quality Monitoring System | | |
| **Date:** | August 1, 2016 | **Pages:** | viii + 65 |
| **Professorship:** | Automation Technology | **Code:** | AS-84 |
| **Supervisor:** | Ville Kyrki, D.Sc. (Tech.) | | |
| **Advisor:** | Juho Pentikäinen, M.Sc. (Tech.) | | |

Developing an intelligent ventilation system for a passenger ship is an enormous task. Because of the safety critical nature of the product, it is encouraged to use a simulated environment for testing before moving to a real-world pilot phase. This work concerns itself with the specification, design and implementation of a simulation tool that can be used to assist the development process.

The requirements were gathered from the client and based on a literature survey regarding general building airflow simulation. An architectural design was made based on the requirements. Finally, the tool was implemented based on the design, and experimented with, to demonstrate its functionality.

The requirements analysis indicated the need for simulating contaminants and airflow within a single deck's area. Performance at both real-time and expedited simulation speeds was required. The tool needed to be configurable and provide a connection to a CAN bus. An existing, scientifically proven multi-zone simulation engine CONTAM was chosen for executing simulations. A software module was implemented to govern the simulation process and keep track of the internal state of individual air ventilation devices.

The tool was found to work as required even for simulations expedited 2852 times on complex floorplan models. The modeling process was found to require experience in ventilation systems and modeling. In order to verify the simulation models and construct a valid error model for the control subsystem, real world measurements on a pilot environment are needed.

| | |
|---|---|
| **Keywords:** | HVAC, ventilation, simulation, multi-zone, CONTAM |
| **Language:** | English |

Aalto-yliopisto
Sähkötekniikan korkeakoulu
Sähkötekniikan ja automaation laitos

DIPLOMITYÖN
TIIVISTELMÄ

| | | | |
|---|---|---|---|
| **Tekijä:** | Peter Kronström | | |
| **Työn nimi:** | Älykkään ilmanvaihtojärjestelmän simulointityökalu | | |
| **Päivämäärä:** | 1. elokuuta 2016 | **Sivumäärä:** | viii + 65 |
| **Professuuri:** | Automaatiotekniikka | **Koodi:** | AS-84 |
| **Valvoja:** | TkT Ville Kyrki | | |
| **Ohjaaja:** | DI Juho Pentikäinen | | |

Matkustajalaivan älykkään ilmanvaihtojärjestelmän kehittäminen alusta pitäen on työläs prosessi. Tuotteen turvallisuuskriittisyys asettaa haasteita etenkin kehityksen alkuvaiheissa, kun testaukseen tarvittavaa pilottiympäristöä ei ole vielä rakennettu. Tässä työssä kuvataan matkustajalaivan ilmanvaihtojärjestelmän kehitystyötä tukevan simulaatiotyökalun kehitysprosessi vaatimusmäärittelystä alkaen. Vaatimusten pohjalta luodaan suunnitelma ja toteutus.

Vaatimusmäärittely koottiin rakennuksen ilmanvaihdon simulointiin liittyvän kirjallisuustutkielman ja asiakkaan toivomusten perusteella. Vaatimusmäärittelyn pohjalta suunniteltiin arkkitehtuuri, joka oli pohjana työkalun toteutukselle. Työkalu implementoitiin arkkitehtuurikuvausten mukaisesti. Työkalun toiminta testattiin kehitteillä olevaa järjestelmää vasten.

Määrittelyssä tärkeimmäksi vaatimukseksi osoittautui kyky simuloida rakennusten sisäisiä ilmavirtauksia ja kontaminantteja yhden laivankannen alueella. Työkalulta vaadittiin kykyä suoritua sekä reaaliaikaisesta, että nopeutetusta simulaatioista. Lisäksi edellytettiin konfiguroitavuutta ja rajapintaa CAN-väylään. Tieteellisesti tarkaksi todistettu ohjelmisto CONTAM valittiin simulaatiomoottoriksi työkaluun. Erillinen ohjelmistokomponentti toteutettiin ohjaamaan simulaation tilaa ja ylläpitämään CAN-viestintää.

Työkalun todettiin vastaavan sille asetettuja vaatimuksia. Se kykeni jopa 2852-kertaiseen simulaatioon 80 ilmanvaihtolaitteen simulaatiomallilla. Mallinnusprosessin havaittiin vaativan kokemusta niin ilmanvaihtojärjestelmistä kuin mallinnuksestakin. Valmiin simulaatiomallin todentamiseksi ja virhemallin luomiseksi tarvitaan oikeaa mittausdataa pilottiympäristöstä.

| | |
|---|---|
| **Asiasanat:** | HVAC, ilmanvaihto, simulaatio, multi-zone, CONTAM |
| **Kieli:** | Englanti |

# Preface

I remember the moment when I heard being able to do my thesis as a part of the AirD project, feeling both grateful and joyous. At first, I was quite overwhelmed by the vast number of work ahead and the progress seemed slow. Yet, here I am 65 pages later, wondering how fast the time has flown by. As C.S. Lewis put it,

> *"Isn't it funny how day by day nothing changes, but when you look back everything is different."*

I am privileged to have had such an astounding network of friends, colleagues and family members supporting me throughout the process. I would like to thank my supervisor Ville Kyrki for valuable advice and constructive criticism and my advisor Juho Pentikäinen for being there and pushing me forward in times of desperation. I am grateful to Gofore Oy and AirD Finland Oy for enabling my thesis. Thank you AirD team for the encouraging atmosphere.

Mom and Dad, thank you for being there for me throughout my studies. Me being here couldn't be possible without all the support and love. Thank you, Ian for motivating gym and jogging sessions, and Susanna for emotional support. I would like to thank my lovely roommates for providing a habitat that I feel happy to wake up in every morning. Thank you, my friends from SIK-speksi and Nerdclub. Keep up the awesome.

Helsinki, August 1st, 2016

Peter Kronström

# Abbreviations

| | |
|---|---|
| ACH | Air Changes per Hour |
| API | Application Programming Interface |
| BCVTB | Building Controls Virtual Test Bed |
| BSD | Berkeley Software Distribution |
| CAD | Computer-aided Design |
| CAN | Control Area Network |
| CFD | Computational Fluid Dynamics |
| CO | Carbon Monoxide |
| $CO_2$ | Carbon Dioxide |
| CPU | Central Processing Unit |
| CRC | Cyclic Redundancy Check (CAN) |
| DCV | Demand Controlled Ventilation |
| DEVOPS | Development Operations |
| DNS | Direction Numerical Simulation |
| DSM | Design Science Methodology |
| ELA4 | Effective Leakage Area at 4 Pa |
| GPL | GNU General Public License |
| GUI | Graphical User Interface |
| HVAC | Heating, Ventilation and Air Conditioning |
| IAQ | Indoor Air Quality |
| IAV | Intelligent Air Vent |
| LES | Large-eddy Simulation |
| N-R | Newton-Raphson Method |
| OS | Operating System |
| RANS | Reynolds-averaged Navier-Stokes |
| SRS | Software Requirements Specification |
| UML | Unified Modeling Language |

# Symbols

| | |
|---|---|
| $B_i$ | flow element $i$ |
| $\overline{B}$ | column vector of total flow |
| $C_{\alpha,i}$ | concentration mass fraction of contaminant $\alpha$ |
| $C_{flow}$ | flow coefficient |
| $\overline{C}$ | correction vector |
| $F$ | airflow rate |
| $F_{ji}$ | airflow rate between zones $i$ and $j$ |
| $G$ | contaminant generation rate |
| $g$ | acceleration of gravity |
| $J$ | square Jacobian matrix |
| $m$ | mass |
| $P_i$ | pressure of zone $i$ |
| $P_1, P_2$ | entry, exit static pressures |
| $\overline{P}$ | zone pressure estimate vector |
| $\Delta P$ | pressure drop across an opening |
| $Q$ | volumetric airflow rate |
| $R_{air}$ | the gas constant of air |
| $S_\phi$ | source or sink term |
| $T_i$ | temperature of zone $i$ |
| $u_j$ | velocity (of air) in direction $j$ |
| $V_i$ | zone $i$ volume |
| $v_1, v_2$ | entry, exit velocity |
| $x_j$ | distance in direction $j$ |
| $z_1, z_2$ | entry, exit elevation |
| $\alpha$ | contaminant |
| $\Gamma_{\phi,eff}$ | effective diffusion coefficient |
| $\kappa$ | kinetic reaction coefficient |
| $\rho_{air}$ | density of air |
| $\phi$ | transport property |

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

A typical cruise ship consists of several hundred cabins and public spaces, each having their own indoor air conditions and preferred air quality. Often these spaces are simply connected to a group of manual air ventilation systems or systems with relatively limited control capabilities. This leads to inefficient HVAC (Heating, Ventilation, Air Conditioning) control leading to potential energy loss and discomforted customers.

This thesis is conducted as a part of a commercial product development project at Gofore Oy comprising of an intelligent HVAC control system for a passenger ship. The product consists of of a group of centralized control servers gathering air quality sensor data from a number of intelligent intake and exhaust actuator vents. In order to manage a large number of controllable air vents in the ship's premises, the data is used to pinpoint both the areas requiring boosted as well as reduced ventilation. Ideally, this localized air ventilation strategy leads to minimal ventilation costs while still maintaining indoor comfort.

The software needs to be tested in a realistic setting during development to verify its functionality. One of the major concerns is the development of the control subsystem which is responsible for directing fresh air into areas that require it. The task is complex and has a number of unknown variables – ventilation control is a safety critical system and needs to be able to perform correctly in every imaginable scenario. Some devices might break during operation which needs to be detected and handled accordingly.

This poses a challenge since the hardware for the intelligent air vents is being developed alongside the server software itself. A single deck of a

passenger ship can contain hundreds of custom air vents. Any sort of pilot testing is out of the question before the software functionality is verified.

## 1.2 Research Problem and Thesis Scope

In order to ensure the functionality of various areas of the server software before any real-scale pilot projects, a tool for simulating passenger ship indoor air conditioning is developed. The simulator's data is used to test the control algorithms, system configuration process, and the administration user interface of the main system. This makes it easy for the development team to test various aspects of software and innovate new solutions without utilizing any sort of physical HVAC hardware. Simulation parameters are easily modified online and the software can be tested with various simulation models representing different physical environments.

Several research questions are studied in this thesis:

- What are the requirements for the software based on client needs?
- What is the architectural and functional design of the simulation tool based on the requirements?
- How does the implementation of the tool function along with the main system?

The specific requirements for the simulation tool are not known beforehand, hence they will be created based on literature review and client needs. Several different strategies for simulating indoor ventilation exist. Choosing a strategy that offers a good balance between simulation details and computational performance is important since computational resources are limited to a modern Macbook Pro laptop. The chosen strategy also directly affects the time spent on creating the simulation model. More complex simulation strategies take more time to be implemented, require special knowledge about the field of building ventilation and the usage of special CAD software.

The design of the tool depends on the software selected for performing the simulation (simulation engine). In addition to supporting all the required simulated physical properties, the software needs to be interfaceable with an external system, being capable of interacting with the control subsystem and user interface. Special attention in design needs to be paid to the Application Programming Interface (API) and the possibility of both reading

the simulation state as well as performing adjustments to the state during simulation runtime.

The functionality of the implementation is demonstrated on the existing main system to ensure its functionality. A simple simulation model is run on the simulation tool and its behavior observed through the administration user interface. Finally, the initial requirements are verified step by step.

## 1.3 Research Methodology and Evaluation

Design Science methodology (DSM) by von Alan et al. [1] will be used as a guiding framework for this thesis. The DSM offers a concise framework for the process of problem-based research in information technology. It can be condensed into the process of identifying problems, solving them with artifacts, and finally evaluating them through various means. Peffers et al. [2] describes six steps for the Design Science process: problem identification and motivation, the definition of objectives for a solution, design, and development, demonstration, evaluation, and communication.

Four artifacts will be created following the guidelines of the DSM. The *simulation tool requirements* are conducted based on the identified problems and project motivation in collaboration with the stakeholder. The *design artifact* is created from the requirements based on the research work done in the literature survey. The *implementation artifact* follows the design taking relevant tools and scope into consideration. Finally, these artifacts *will be evaluated* using the DSM guidelines, by showing that it adheres the requirements and use cases [1]. The chosen simulation strategy is validated through previous research done and the data sanity is tested against the control system UI.

Figure 1.1 depicts the steps undergone in this thesis. Each step will act as a basis for the following ones, leading to a logical chain of research. Based on the thesis objective, a literary review is produced. A requirements specification is created from the knowledge gained on the literature review part. The requirements specification leads to a generic design specification of the simulation tool, working as a high-level but complete description for implementation. Implementation and experimentation blocks act as validation steps, supporting the design. Finally, conclusions are drawn from preceding chain of steps, answering the research questions set by the initial objective.

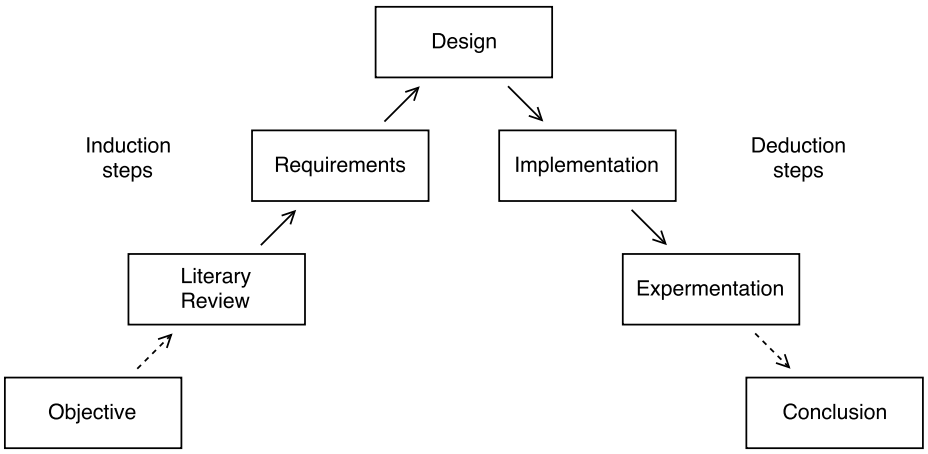


Figure 1.1: The Design Science process.

## 1.4 Thesis Outline

The thesis is divided into seven chapters going through the development process of the simulation tool. The first chapters will work towards a design by the means of literary research and requirements analysis. The tool's design will be validated by implementing a working prototype and demonstrating its usage in the latter chapters.

**Chapter 2** gives an introduction to building simulation, its advantages and presents the most commonly used simulation models and their uses. A recommended modeling process is briefly discussed. **Chapter 3** defines the requirements for a simulation tool in the context of simulating passenger ship's ventilation systems for HVAC control purposes. The chapter begins with a system overview, followed by a general explanation of the requirements and the functional and non-functional requirements listings. **Chapter 4** concerns itself with the design of the tool, including the architecture descriptions and run-time process flow. The simulation engine choice is discussed here. **Chapter 5** describes the prototype implementation following the design guidelines. Also, the choice of a simulation engine and the simulation modeling process is briefly described. **Chapter 6** demonstrates the implementation and presents verification against the requirements. **Chapter 7** concludes the thesis beginning with a summary, conclusions and further suggestions for improvements.

# Chapter 2

# Building Airflow Simulation

This chapter will give an introduction to the main methods for indoor *airflow simulation*. Since the functionality of the tool developed in the thesis relies heavily on indoor airflow and contaminant simulation, only methods for airflow and ventilation simulation are assessed in this literary review.

Section 2.1 will give insight to various advantages in using simulation in product development. Section 2.2 begins with an overview to indoor airflow simulation in general. Sections 2.3 and 2.4 introduce two of the most relevant simulation methods in greater detail, *multi-zone* and *computational fluid dynamics* (CFD).

## 2.1 Advantages of Simulation

There are several advantages to simulating a system over implementing one. Computer simulation offers *a way to study the behavior* of a system without building it first, thus reducing the risk of a faulty, or even a dangerous implementation [3, p. 7]. Often the risk reductions reflect financially, allowing a stakeholder to save money and time by validating certain scenarios before making a decision about the implementation. Alternative system designs can be compared to each other by running the same simulation with slightly altered models and with varying parameters [4, p. 115].

Developing a robust control strategy capable of controlling hundreds of devices throughout a single passenger ship's floor in real-time requires a lot of testing in various environments. Already the vast physical size of the system sets restrictions on the process: in order to test out the system in a real-scale pilot environment, hundreds of air ventilation devices are needed to be manufactured and assembled. In addition, the system installation process

takes time and ties down resources, even before knowing how the control strategy will behave in a real-size environment. The simulation allows the system developers to *iteratively scale up* the model in order to execute a lean, iterative top-down development process for the control strategy, beginning with a small, generalized simulation model and slowly scaling it up to cover real-scale representations of the physical environment.

*A support for expedited simulations* allows for quick investigation of the behavior in broader timeframes [3, p. 7]. This is highly useful for estimating the system's long-term stability or seasonal environmental schedules. In the case of passenger ship's indoor air quality (IAQ) control system development, it is highly advantageous to be able to simulate airflows over several hours within a timeframe of a few minutes, in order to detect notable changes in control variables.

A simulated system enables for *easy manipulation of the model*, allowing the users to adjust the model parameters at will, even out of the typical real world ranges [3, p. 7]. This makes it easy to test out the system in slightly different environments, such as evaluating the system behavior in a metal-walled cabin compared to a composite-walled one, for example. Also the amounts of windows, air ducts, and even complete air handling systems can be easily modified and experimented with.

Also, the imminent *suppression of real-world disturbances* is useful in order to understand the underlying physical principles better [3, p. 7]. By simplifying the model and reducing the number of affecting variables, the debugging process of a hypothetical fault e.g. in the control system becomes easier. This abstraction makes demonstrating the system behavior and comparing various simulation context easier. An animation can be used instead of being limited to less effective numerical and textual presentations [5, Sec. 1.5.3].

## 2.2 Overview of Simulation Methods

The building simulation field offers a wide spectrum of methods and tools for assessing building performance from energy and mass flow to durability, aging and egress simulations. The early work in the branch tackled problems related to energy performance, and HVAC and airflow simulation quickly followed. Recent additions include combined moisture and heat transfer, acoustics evaluation, control systems and various microclimate simulations. [6, p. 4] Barták et al. [7] demonstrate a typical building simuation case in

their research paper. The paper used airflow simulation to validate HVAC capacity for a church that was to be converted to a concert hall.

A large amount of simulation software has been developed along the years, ranging from highly specific solutions to generalized building simulation software. The simulated environment needs to be modeled based on the chosen simulation method before the simulation itself can be done. Some of the software offers an integrated graphical user interface (GUI) for model creation and editing, but the software might also support a textual interface or external CAD software.

Typical simulated variables include *air speed*, *air temperature* and *relative humidity* [8]. In addition to simulating airflow magnitudes in various parts of a building and ductwork, also simulating transport process of *various contaminants* might be of interest. Estimating carbon dioxide ($CO_2$) concentrations in a building might be relevant for assessing HVAC system capacities in terms of inhabitants, for example. A detailed carbon monoxide transport prediction might ensure that the construct abides safety standards. Some simulation software also supports simulation of germs, smoke, and radioactive particles.

Building airflow simulation models can be coarsely categorized based on the level of detail, from the macroscopic to the most microscopic view: *analytical and empirical models*, *multi-zone and zonal models* and *computational fluid dynamics* (CFD), respectively [6]. In general, the simulated building or a part of it needs to be represented as a group of related mathematical abstractions based on the type of an element being modeled. Typically an iterative approach is used to find a solution to a set of equations based on the given parameters. Figure 2.1 visualizes how accurately various models depict the airflows within a building.

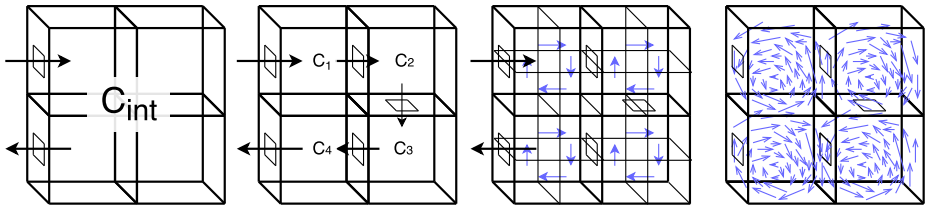


Figure 2.1: Simulation models can be divided into four categories based on the level of model abstraction. From left to right: *analytical and empirical*, *multi-zone*, *zonal* and *computational fluid dynamics* models.

The *analytical and empirical* models are mostly used for tasks that do not require greater detail. They are derived from the fundamental equations of fluid and heat transfer and often rely on geometric simplifications [8]. A widely used analytical model is a set of equations developed by Fitzgerald and Woods [9], which is used to calculate the temperature elevation and flow rate in a room. In some cases, additional experimental measurement or simulation data is used to obtain relevant coefficients to make the models work. A good example of such an empirical model is the set of equations developed by Cho et al. [10] for determining the jet behavior in terms velocity profiles, the spreading rate of jets on the surface, and jets decay. The equation includes an empirically obtained constant. These methods for building modeling are great for finding solutions to simple problems. However, being heavily case dependent, they can not be used for more generic cases. Thus they will not be perused further in this survey.

Multi-zone models interpret the airflow rates in building's openings and contaminant concentrations in a room-scale detail. They depict the building and related ventilation systems as a network of volumetric nodes, representing rooms or parts of rooms. The nodes are connected to each other by flow paths, depicting door cracks, window cracks, and duct terminals.

The most detailed modeling approach utilizes CFD dividing the rooms into a quantity of cells, simulating individual air and contaminant *flow fields within the room* in high detail. Table 2.1 summarizes the most used simulation models in terms of suitable applications, based on [6].

*Multi-zone* models yield a macroscopic view of the building complex, solving a network of mass-balance equations. These result in average concentrations of contaminants and airflows between rooms, rather than exact flow vector fields. Running the simulations for a multi-zone model can be done at a relatively low computational cost. This permits for expedited simulations up to a timeframe spanning several months in a few minutes of time, depending on the model complexity [11, Sec. 13]. Since the simulated quantities are assumed to be well-mixed throughout the control volume, larger premises cause inaccuracies in results. *Zonal* models have been developed to deal with this problem by dividing larger spaces into a limited number of cells, typically less than 1000 per room [8]. Chen et al. [8] however state, that only a handful practical zonal model applications have been done regarding zonal simulation. Comparing zonal models to coarse grid CFD, no significant computing performance was achieved.

CFD simulation solves Navier-Stokes equations resulting in a highly de-

| | Simplified methods | Zonal models | CFD |
|---|---|---|---|
| Basic building air change rate for sizing and energy analysis | YES | YES | YES |
| Calculation of airflow rate through envelope openings | NO | YES | YES |
| Simple combined ventilation or thermal heat loss modeling | NO | YES | YES |
| Average pollutant concentration in buildings | NO | YES | YES |
| Calculation of airflow through multizone structures | NO | YES | YES |
| Airflow and pollutant transport between rooms | NO | YES | YES |
| Input for complex combined ventilation and thermal models | NO | YES | YES |
| Room airflow | NO | NO | YES |
| Room temperature distribution | NO | NO | YES |
| Room pollutant distribution | NO | NO | YES |
| Ventilation/pollutant efficiency parameters | NO | NO | YES |
| Wind pressure distribution | NO | NO | YES |

Table 2.1: Summary of typical airflow simulation model capabilities. Based on [6, p. 88]

tailed view of airflow fields within a building's individual rooms or even ventilation components, including the exact transport paths for contaminants. One of its major advantages is gaining highly accurate simulation results for airflows and contaminant concentrations. However, building the detailed models for CFD simulation takes time, the process often including characterizing of furniture and duct terminal shapes. The increased model complexity requires vast amounts of computational resources compared to the other approaches, and often the gains in detail might be inconsequential considering the simulation motives. [11, Sec. 13] Image 2.2 represents a typical result of a CFD simulation, resulting in highly detailed information about the simulated control volume.

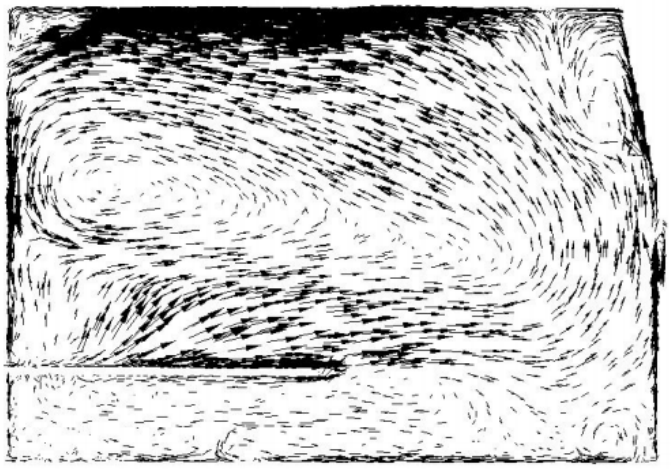

Figure 2.2: ESP-r software generated CFD prediction of an air velocity field in a historical church viewed from above. The results were used to evaluate if the building could be safely converted into a concert hall sustaining an estimated number of visitors. [7]

The computational complexity of the simulation model grows along the number of details present in the model [6]. This means that choosing the most suitable simulation model depends heavily on the use case and available computational resources and is often a trade-off between performance and result detail. When choosing a simulation modeling method, it is important to know which physical quantities are relevant to the situation. For example, a CFD simulation model does not scale up well computationally on trying to mimic real-time behavior of airflows in apartment complexes, whereas a multi-zone or zonal simulation might give performant, but detailed enough

results for the same use case.

## 2.3 Multi-Zone Models

Multi-zone simulation models idealize a building as one or more *nodes* connected to each other by *flow elements*. The nodes, so-called zones, can represent room air volumes within a building, a duct system or an ambient environment. Each node contains simulation-specific state variables, such as pressure, temperature and contaminant concentrations including $CO_2$, smoke, and pollutants, whose values are calculated according to the simulator's mathematical models. The flow elements that connect the nodes to each other represent pathways air can flow through – open windows and doors, cracks in building's construction, staircases, elevator shafts or air ducts and fans, for example.

The flow elements connected to the nodes are characterized by their flow properties, including possible airflow resistance caused by, e.g. an air duct's internal surface roughness or orifice size and shape. The flow rates in each of these links are determined based on the differential pressure across the links, which are typically caused by mechanical ventilation, buoyancy effects or wind. This network of links and zones is internally described as a series of equations, which are solved by the simulator iteratively to provide a mass conserving solution [11, Sec. 13.15].

A multi-zone airflow network model is analogous to an electrical network [11, Sec. 13.14]. Airflow corresponds to electrical current, whereas air pressure is characterized by voltage differences. Flow paths can be characterized by resistors hindering the airflow between nodes, whereas active elements (e.g. fans) as voltage sources introducing air pressure to the network. Figure 2.3 depicts a simple multi-zone model represented by an electrical network. [11, Fig. 13]

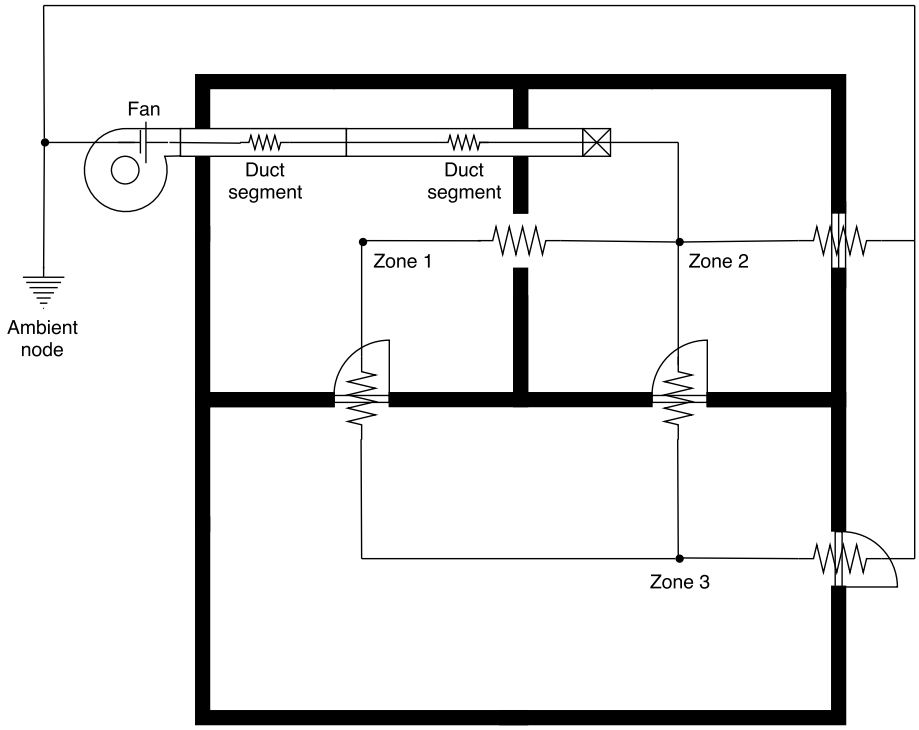


Figure 2.3: Apartment's airflows can be represented as an electrical network, where resistors characterize flowpaths, batteries represent fans and network nodes zones. Based on [11, Sec. 13.14].

The airflow simulation problem is often decoupled from the thermal and pollutant transport problems in multi-zone simulators. Such a model assumes a homogeneous contaminant, heat and pressure distribution within the room, as opposed to more detailed simulation models, such as computational fluid dynamics (CFD), presented later in this chapter. The simplicity of such a node and flow based multizone model has a direct positive effect on computational performance, and makes it feasible to simulate airflows and contaminant transport within a building with relatively low computational resources, even on wider time ranges, such as on a monthly or annual scale. Scaling the model up in terms of rooms is much more effective compared to CFD simulation.

### 2.3.1 Mathematical Background

The flow paths provide the most detail to the simulation model in multi-zone airflow models. The airflow rate $F_{i,j}$ between two zones is calculated as a function of pressure drop $P_j - P_i$ along the flow path:

$$F_{i,j} = f(P_j - P_i) \tag{2.1}$$

The function $f(P)$ is heavily dependent on the flow path characteristics, such as orifice area, shape or material roughness. These flow path models are typically presented with various non-linear models, such as the commonly used power law equation:

$$Q = C_{flow}(\Delta P)^n \tag{2.2}$$

where $Q = F/\rho_{air}$ is the volumetric airflow rate in $m^3/s$, $\Delta P$ the pressure drop across the opening, $C_{flow}$ the flow coefficient in $(Pa^{1/n} \cdot m^3)/s$, $n$ the flow exponent and $\rho_{air}$ the density of air in the flow path in $kg/m^3$. The power law model is based on the physical model of orifice flow, but is also used with various other airflow elements, such as duct dampers, bends, and transitions, with the introduction of flow coefficient $C_{flow}$. The power law model can be expanded to account for the laminar effect of low flow rates. In addition, plenty of other models exist for describing the flow through stairwells, various constant and variable flow fans, doorways and windows [12, pp. 79-81] [13, pp. 23-68].

The pressure differences within each airflow element are governed by Bernoulli's equation, which accounts for the air pressure differences in zones caused by air density and height changes. In some cases, static wind causes

additional ambient air pressure, which needs to be accounted for [11, Sec. 13.14]. The equation can be represented in form

$$\Delta P = \left(P_1 + \frac{\rho v_1^2}{2}\right) - \left(P_2 + \frac{\rho v_2^2}{2}\right) + \rho_{air} g(z_1 - z_2) \tag{2.3}$$

where $\Delta P$ is the total pressure drop between points 1 and 2, $P_1, P_2$ are the entry and exit static pressures, $v_1, v_2$ are the entry and exit velocities, $\rho_{air}$ equals air density, $g$ the acceleration of gravity and $z_1, z_2$ the entry and exit elevations [12, p. 256].

Finally, the flow element and zone relations are combined by enforcing the principle of mass conservation at each zone. The mass of the air $m_i$ $(kg)$ in zone $i$ is given by the ideal gas law

$$m_1 = \rho_i V_i = \frac{P_i V_i}{R_{air} T_i} \tag{2.4}$$

where $V_i$ is the zone volume $(m^3)$, $P_i$ is the zone pressure $(Pa)$, $T_i$ is the zone temperature $(K)$ and $R_{air} = 287.055\ J/kg \cdot K$ the gas constant for air. For transient solutions the conservation of mass states that

$$\frac{\partial m_i}{\partial t} = \rho_i \frac{\partial V_i}{\partial t} + V_i \frac{\partial \rho_i}{\partial t} = \sum_j F_{ji} + F_i \tag{2.5}$$

$$\frac{\partial m_i}{\partial t} \approx \frac{1}{\Delta t}\left[\left(\frac{P_i V_i}{R_{air} T_i}\right)_t - (m_i)_{t-\Delta t}\right] \tag{2.6}$$

where $F_{ji}$ is the airflow rate between zones j and i $(kg/s)$, $F_i$ the nonflow processes that add or remove significant quantities of airflows from j to i $(kg/s)$ and $m_i$ the mass of air in zone i $(kg)$. This is, however often simplified assuming steady-state conditions, where the driving forces change slowly compared to the airflow [12, p. 253]. Under such an assumption, the mass conservation in zone i is reduced to

$$\sum_j F_{ji} = 0 \tag{2.7}$$

### 2.3.2 Solving the Airflows Computationally

The simulation computations can be simplified to a three-step process [14]:

1. Find the pressures at each point where a flow element is connected to a node (zone)

2. Find the airflows in each flow path, based on the pressure differences at its terminals

3. Sum up the flows in and out for each zone and calculate a mass balance state by adjusting the reference pressures

This process leads to a set of non-linear mass-balance equations, which must be simultaneously satisfied for any given point in time. This is done iteratively with a non-linear equation solver, such as the commonly used Newton-Raphson (N-R) method [11]. The N-R method calculates an estimate of the zone pressures represented as a vector $\overline{P}^*$ based on the subtraction of the current estimate of the pressures $\overline{P}$ and the correction vector $\overline{C}$

$$\overline{P}^* = \overline{P} - \overline{C} \tag{2.8}$$

the correction vector $\overline{C}$ is computed from the matrix relationship

$$J\overline{C} = \overline{B} \tag{2.9}$$

where $\overline{B}$ is a column vector of total flow into each zone, each element $B_i$ given by the equation

$$B_i = \sum_j F_{ji} \tag{2.10}$$

matrix $J$ being the square Jacobian given by

$$J_{i,j} = \sum_i \frac{\partial F_{j,i}}{\partial P_j} \tag{2.11}$$

$F_{j,i}$ and $\frac{\partial F_{j,i}}{\partial P_j}$ are evaluated using the current estimate of pressure $\overline{P}$ in equations 2.10 and 2.11.

### 2.3.3 Contaminant Simulation

Contaminant concentrations are solved separately from the airflow problem introduced in the previous subsection. Generally, the contaminant transport is addressed by *advection* via flow path airflows. As a comparison to the detailedness of CFD based models, the contaminants in multi-zone models are only represented as well-mixed, uniform concentrations within zones, possibly causing inaccuracies in short-term simulations and volumetrically large, atrium-type zones. In some cases, these inaccuracies can be circumvented by dividing the zone into multiple smaller ones, giving a slightly more

accurate estimate about the transport of contaminants within a zone. The contaminant transport model typically accounts for contaminant generation by various sources and chemical reactions, removal by filtration, radiochemical decay, settling or sorption. [11, Sec. 13.16]

The principle of mass concentration is applied to all species of contaminants within a control volume, such as a zone or a piece of ductwork. Contaminant $\alpha$'s mass in zone $i$ is given as

$$m_{\alpha,i} = m_i C_{\alpha,i} \tag{2.12}$$

where $m_i$ is the mass of air in zone $i$ and $C_{\alpha,i}$ is the concentration mass fraction of contaminant $\alpha$ ($kg$ of $\alpha$ per $kg$ of air). The equation for contaminant dispersal for a given zone $i$ can be represented as [11, Sec. 13.16]:

$$\frac{dm_{\alpha,i}}{dt} = -F_{rem} - \sum F_{out} + \sum F_{in} + \sum F_{react} + F_{gen} \tag{2.13}$$

where the contaminants are added or removed by

- Removal at the rate of $F_{rem} = R_{\alpha,i} C_{\alpha,i}$, where $R_{\alpha,i}$ is the removal coefficient.
- Outward airflows from the zone at the rate of $\sum F_{out} = \sum_j F_{i,j} C_{\alpha,j}$.
- Inward airflows to the zone at the rate of $\sum F_{in} = \sum_j F_{i,j}(1-\eta_{\alpha,j,i}) C_{\alpha,i}$, where $\eta_{\alpha,j,i}$ is the filter efficiency in the path from $i$ to $j$.
- First-order chemical reactions with other contaminants at the rate of $\sum F_{react} = \sum_\beta \kappa_{\alpha,\beta} C_{\beta,i}$, where $\kappa_{\alpha,\beta}$ is the kinetic reaction coefficient between species $\alpha$ and $\beta$.
- Generation at rate of $F_{gen} = G_{\alpha,i}$

Depending on the factors listed above this equation is developed and solved for each contaminant and zone in order to determine contaminants' concentrations. [11, Sec. 13.16]

## 2.4 Computational Fluid Dynamics

The field of *Computational Fluid Dynamics* (CFD) concentrates on modeling thermal and fluid behavior quantitatively. It was originally developed for

modeling aircraft aerodynamics, naval vessel hydrodynamics, meteorology and biomedical engineering. Later on, some of these models have been successfully applied for IAQ simulation [6, p. 4]. The main advantage of CFD simulation is the amount of detail in the results, giving a microscopic view to the examined substance's behavior within the simulated environment. For indoor environment modeling, the CFD can be used to solve air and fluid flow, heat transfer, chemical reactions or thermal stresses, among others.

Airflows within a building can be driven by various forces, such as natural wind phenomena, mechanical fans or thermal buoyancy, creating a combination of complex flow characteristics, including vortices, circulation, impingement, and buoyancy. The CFD tries to model these processes occurring within a fluid by solving a set of coupled, non-linear, partial differential equations, which express the fundamental physical laws governing the conservation of mass, momentum and energy. By applying so-called *Navier-Stokes equations*, the motion of viscous fluid substances within a control volume can be solved. The use of numerical methods is required since an analytical solution for a three-dimensional turbulent flow field is not possible [15].

### 2.4.1 Mathematical Background

The CFD modeling process begins with dividing the control volume into a large number of small cells, also called as *mesh* or *grid*. The size of the grid and the distribution of the cells directly affect the solution's convergence as well as computational complexity and accuracy. Due to the empirical nature of the phenomenon, no general approach have been found to address turbulent flows so far. However, three notable approaches of varying complexity have been developed to address this chaotic motion: *direction numerical simulation* (DNS), *large-eddy simulation* (LES) and *Reynolds-averaged Navier-Stokes* (RANS) [15].

All these turbulence equations can be written in the general form [16]

$$\rho\frac{\partial\bar{\phi}}{\partial t}+\rho\bar{u_j}\frac{\partial\bar{\phi}}{\partial x_j}=\frac{\partial}{\partial x_j}\left[\Gamma_{\phi,eff}\frac{\partial\bar{\phi}}{\partial x_j}\right]+S_\phi \tag{2.14}$$

where $\phi$ represents the transport properties (e.g. air velocity, temperature, contaminant concentration), $\rho$ density ($kg/m^3$), $u_j$ velocity in direction j ($m/s$), $x_j$ distance in direction j ($m$), $\Gamma_{\phi,eff}$ the effective diffusion coefficient and $S_\phi$ the source or sink term of an equation. Equation (2.14) relates the change in time to the amount of flux of a variable at a certain location.

Essentially transient changes and convection equal diffusion and sources [11, Sec. 13.2].

The DNS approach gives the most accurate results, solving the Navier-Stokes equations directly without any approximations. In general, DNS resolves a range of spatial and temporal scales of the turbulence from the smallest to the largest ones, hence requiring a very fine grid resolution and short time steps to compute the flow field. Because a single room might be divided in as large as $1000 \times 1000 \times 1000$ sized grids consisting of up to sub-millimeter sized cells, the DNS also requires the most time to solve computationally [17].

LES was developed based on the Kolmogorov's theory, which states that where large turbulent flows depend on the geometry, the smaller scales are more universal [18]. LES focuses only on large eddies, filtering out the smaller turbulences, which do not have a significant impact on the general flow. Since large scales of motion are mainly responsible for transport processes, no major simulation variables, such as heat or contaminant transfer are ruled out. This approach affects grid size requirements directly, removing the need for a fine spatial grid and small time steps [17].

RANS calculates statistically averaged variables for steady-state and dynamic flows, simulating the turbulence effects on the mean airflow with various approximation models. Many different turbulence models have been developed, the $k$-$\varepsilon$ model with its variations being the most popular [17]. A coarser grid may be applied for the calculation of averaged flow terms, greatly reducing calculation times. RANS has become popular due to its significantly smaller computational requirements. [15]

### 2.4.2 Advantages and Disadvantages

The detailed nature of CFD simulation has many strengths, including gaining elaborate, even microscopic information on airflow fields, temperatures and pollutant concentrations within the simulated environment. However, the accuracy of the simulation depends heavily on the user's knowledge of fluid dynamics and experience in numerical techniques and modeling. Different users may obtain varying results for the same problem even with the same CFD software. [17] Also, building the simulation model and running the simulation takes vast amounts of computing resources, setting strict requirements on the size and detail of the simulation model, as well as on the equipment used to run the simulation. For simulation models bigger than a single room, the number of equations needed to solve each iteration grows

too large to allow for real-time simulation. E.g. for a single-room CFD model divided in $90 \times 90 \times 90$ cells the total number of coupled non-linear differential equations (and unknown variables) that are needed to be solved grows to $5.8 \times 10^6$. The total number of grid point calculations might grow as large as $17 \times 10^9$ since the numerical method can involve up to 3000 iterations to find the total flow field [11, Sec. 13.2].

# Chapter 3

# Requirements

This chapter describes the simulation tool in terms of requirements that the design and implementation must follow. According to Sommerville [19, Sec. 4.5] the software requirements are used to communicate information about the system to different types of readers in an unambiguous and complete way, laying out the constraints on its operation and implementation. By solely following the requirements specification, the user should be able to create a software design and implementation. The Requirements specification in this thesis is adapted from the IEEE 830 software requirements specification (SRS) standard [20] to match the needs of a small-scale software development project, managed, defined and implemented by a single developer.

The IEEE 830 SRS standard [20] lists various benefits that an exhaustive SRS can offer: it works as a mutual contract between the customers and the suppliers on what the software should do. During the development process, a proper SRS reduces the development effort by revealing the inconsistencies, omissions, and misunderstandings already in the early phases of development. Proper requirements help the suppliers to plan out the work by providing a basis for estimating development costs and schedules. After development, as in the case of this thesis, the requirements can be used as a baseline for the validation and verification process of the software product. In addition, SRS helps with the process of transferring the process of usage, development and re-deployment to new users. SRS can also be used as a basis for enhancing the finished product if additional requirements arise.

The IEEE 830 standard [20] lists eight characteristics of a good SRS, which are listed and explained in the following list. These practices are also followed in this thesis.

1. *Correctness.* A correct SRS has only requirements that the software shall meet. The requirements should also be kept up-to-date.

2. *Unambiguousness.* Every requirement should be specified unambiguously so that it has only one interpretation.

3. *Completeness.* A complete SRS includes all the necessary information for the developers to design and implement the software product.

4. *Consistency.* The SRS should be consistent with other higher-level documents, and specified requirements and terms should not conflict with each other.

5. *Proper ranking.* The requirements should be sorted by importance or stability since typically not all the requirements are equally important.

6. *Verifiability.* Every requirement in an SRS should be verifiable with an unambiguous, finite process. Quantitative requirements are recommended for verifiability.

7. *Modifiability.* Changes should be able to be made in an easy manner, which requires non-redundant requirements, coherent organization and separated atomic requirements.

8. *Traceability.* The origin for each of the requirements is required to be traceable to the document from elsewhere, e.g. via unique reference numbers.

In addition, a desirable SRS should answer the following questions (modified from [20]):

1. *Functionality.* What is the software supposed to do?

2. *External Interfaces.* How does the software interact with people, various hardware and other software?

3. *Performance.* What is the speed, availability, response time and recovery time of the software?

4. *Attributes.* What are the portability, correctness, maintainability and security considerations?

5. *Implementation design constraints.* Are there any required standards, programming language, evident database policies, resource limits and operating environments?

The functional and non-functional requirements for the simulation tool were compiled by using the requirements elicitation and analysis process defined by Sommerville [19, Sec. 4.5]. It follows a four-step process beginning with (1) the requirements discovery, which was executed as a meeting with the stakeholders. The functionality of the simulation tool was discussed and an unsorted group of requirements gathered. These were (2) classified into relevant categories based on the initial tool architecture specification. Finally, the requirements were (3) prioritized and (4) specified as functional requirements.

An introduction to the overall system will be given in the subsection 3.1 as a high-level architecture description, abstract user requirements given in natural statements in 3.2 and both the functional and non-functional requirements related to the simulation tool will be expanded upon in the following subsections 3.3.1 and 3.3.2.

## 3.1 System Overview

This section will give a short introduction to the context this thesis is conducted in. The end product is a complete system capable of monitoring the IAQ and passenger occupancy within a passenger ship's premises. The data is used for real-time air ventilation control in order to yield monetary savings in HVAC costs. This demand controlled ventilation (DCV) system comprises of physical intelligent devices placed within each ship's air vent, sending IAQ measurements to distributed control servers. Each control server manages a group of devices with a suitable control strategy preserving the required air quality criteria within ship's premises. The overall high-level architecture for the end product is depicted on Figure 3.1.

The first block in Figure 3.1 depicts groups of intelligent, IAQ measuring air vents, sending measurement data to local control servers at specific intervals via an industrial CAN bus. Each device is mapped to the passenger ship's premises in system's configuration by a unique ID. This block will be completely replaced by the tool, allowing for simulation complex physical scenarios.

The second block indicates a group of local control servers governing each separate group of IAV's, receiving IAQ data and error messages from

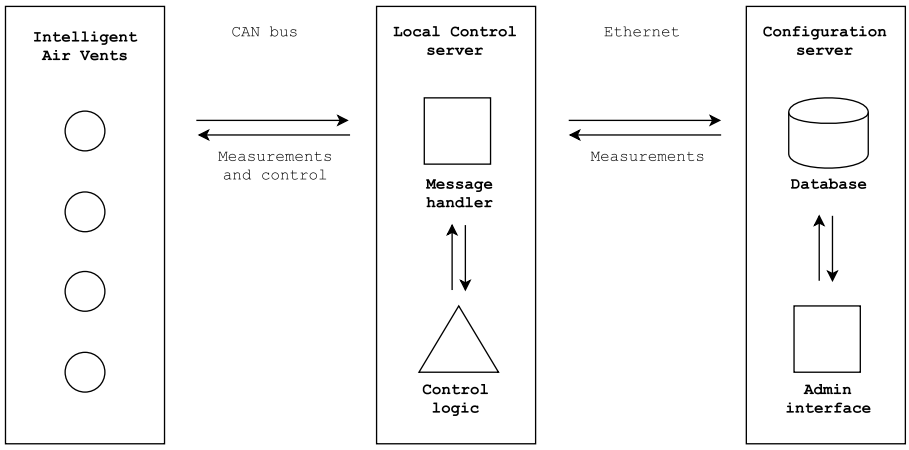


Figure 3.1: High-level main system architecture. The end product consists of three major parts: the configuration server acting as a unified database for the IAV data, several local servers taking care of IAV control groups and a number of IAV devices in ship's premises.

the CAN bus, and forwarding them to the system's time series database. The CAN messages are locally translated by the handler in local control servers to analyze the IAV group's performance, detect faults, and (notably) to execute a continuous control strategy for device.

Finally, the third block in Figure 3.1 acts as a central hub for managing all the separate HVAC subsystems within a passenger ship. It receives IAQ measurements and detected fault messages from local control servers through TCP/IP and stores them into a database for further use. In addition, the configuration server runs an instance of an administrator interface, drawing visualizations about the relevant data and giving access to system configuration for users.

## 3.2 User Requirements

This subsection will provide a high-level abstract illustration about the system. Sommerville [19, Sec. 4.5] describes user requirements as a mix of natural language statements and diagrams, giving a higher level understanding of the services the system is expected to provide and constraints under which it has to operate.

### 3.2.1 Tool Scope and Constraints

The simulation tool implemented in this thesis acts as a replacement for the first block (from the left) represented in Figure 3.1, capable of simulating groups of IAV's within given environment. The simulator should be able to *simulate selected physical quantities* related to IAQ, such as airflow and $CO_2$ levels within ship's premises and report the measurements via a CAN bus to the real physical system. Furthermore, the simulator should *react to control messages* from the local control server in a predetermined way. Figure 3.2 demonstrates the data flow between the simulator and the system. The measurements and control messages have a causal relationship, meaning that the control signals are adjusted depending on the measurements received from the simulator. The timeframe of this data flow loop should be adjustable in order to handle various simulation speeds and the measurement interval.

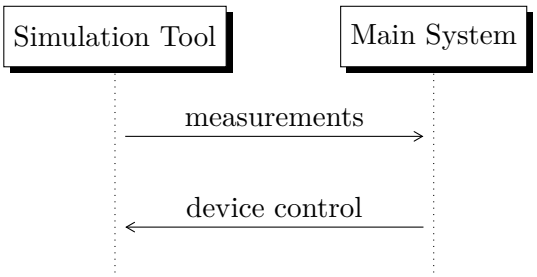


Figure 3.2: High-level representation of the dataflow between the simulator and the system. The simulation tool sends new simulated IAV measurements to the system and adjusts the simulation according to received device control messages. The data exchange happens via CAN bus.

The tool's functionality is purely confined to the airflow simulation within a passenger ship's premises. It should not simulate extra IAV functionalities, such as autonomous HVAC control, automatic discovery or error procedures. No graphical user interface needs to be developed for the simulation tool. The simulation tool should work as a separate entity, capable of hot-plugging in and out of the CAN bus at will. This means that no databases for simulation history will be used.

The tool should be an independent entity, capable of running separately from the rest of the host system (blocks 2 and 3 in Figure 3.1). It should support the same operating system (OS) as the host (64-bit headless Ubuntu server). This limitation does not have a strict restriction on the OS used for the simulator, since the only communication interface the simulation tool

depends on is the CAN bus. However, running the simulator under the same operation system enables taking advantage of the same development operations (DEVOPS) tools as used by the host. Also, being able to connect the simulation tool to the same virtual CAN bus removes the need for a real, physical CAN bus.

### 3.2.2 Interfaces

#### System Interfaces

The communication between the intelligent air vents and control servers are handled via an industrial CAN bus. It is an industrial multi-master protocol for data transmission defined by the ISO 11898-1 standard. Physically the CAN bus consists of a varying length of twisted pair wire with terminating resistors at ends to prevent signal reflections. The communication between the devices happens by using specified CAN message frames, which are small packets of data including packet header fields and a maximum of eight bytes worth of data. The data is broadcast to the entire network, and read by accountable devices based on the message identifier.

The CAN standard is a robust protocol both in terms of the physical and software implementations. Each message frame is protected from accidental bit flips and interruptions by Cyclic Redundancy Checks (CRC), and the error scheme defined by the protocol can handle various hardware and software faults [21].

Data exchange between devices happens by broadcasting short CAN message frames to the bus. Successful delivery relies on the devices having to filter and parses the properly addressed messages themselves, either in hardware or software [22]. A custom light-weight message frame protocol was developed on top of the software layer for the system to support the messaging needs. The simulator should connect to the CAN bus and be capable of receiving and interpreting IAV-specific control messages and write the IAQ measurements from each IAV device to the CAN bus, respectively.

#### User Interfaces

The simulation tool is made for developers to be used during the research and development phase of the project. The tool should be configurable in a way that does not require programming skills from the user. At least a modifiable text file with sufficient variable explanations should be realized. The simulation models should be *easily configurable* and have *a high flexibility*

in terms of possible modeled environments. A non-acquainted user to the system should be able to design and implement a model to be used with the simulation tool. A user-friendly graphical interface and proper user guide *for the modeling process* is recommended.

## 3.3 System Requirements

As stated by Sommerville [19, Sec. 4.5], the system requirements are detailed descriptions of the system's functions, services, and operational constraints. This subsection will explain the simulation tool functionality precisely, as an indexed requirements list. The requirements will be split into two categories – functional and non-functional requirements, stating what the system should do and what constraints the requirements have, respectively.

### 3.3.1 Functional Requirements

**R1** The tool must be able to simulate physical quantities relevant to the ventilation control.

- **R1.1** The simulation models must be based on and verified by established research.
- **R1.2** The $CO_2$ and CO concentrations must be distinguishable in simulated premises.
- **R1.3** The airflow magnitude must be distinguishable in duct terminal points.

**R2** The simulator must support a limited IAV device functionality.

- **R2.1** Each IAV device must have a unique ID that matches the system ID.
- **R2.2** The IAV devices must be mapped to the simulation model's rooms.
- **R2.3** The IAV devices must be adjustable via control messages interpreted from the CAN bus.
- **R2.4** Each configured IAV device must report measurement data to be sent to CAN bus.

**R3** The simulation tool must have a connection to the system's CAN interface.

**R3.1** The tool must be able to receive and recognize custom control message frames.

**R3.2** The tool must be able to curate and send custom measurement message frames.

**R3.3** The tool must support both virtual CAN and physical CAN interfaces.

**R4** The simulator must support an ongoing simulation, with transient sensor updates at specified intervals.

**R4.1** The simulation must work continuously at normal real-time speed.

**R4.2** The measurement sending intervals must configurable.

**R5** The simulation must function at different speeds.

**R5.1** The simulator must be capable of real-time simulation at least at a frequency 0.2 Hz.

**R5.2** The simulator must have a functionality for expedited simulations and add relevant timestamps to the measurement messages.

**R6** The simulation model must be configurable and offer a sufficient level of detail.

**R6.1** The model must be dividable in multiple rooms.

**R6.2** The model's physical parameters must be configurable (physical dimensions, air leakage properties).

**R6.3** The model must support freely different material properties, such as metallic wall structures and composites.

**R6.4** The model must support elements relevant to the system: air ducts, air handling systems, windows, doors.

**R7** The simulation tool must support exporting models to a file for saving.

**R8** The simulator should be able to run from 64-bit Ubuntu command line.

### 3.3.2 Non-Functional Requirements

**R9** Simple simulation models should be implementable in less than an hour for an unacquainted user.

**R9.1** The modeling instructions should be exhaustive.

**R9.2** The model element modification (add, remove, duplicate) should be supported by the editor.

**R9.3** Graphical user interface for building models should be available.

**R10** The simulator must support at least 80 simultaneous IAV devices.

**R11** The simulator must be able to run the predefined simulation models in a contemporary laptop.

**R12** External software used should be easily available.

**R12.1** It must be free for commercial use.

**R12.2** It should be actively maintained.

**R12.3** It should be well documented.

# Chapter 4

# Design

The following chapter discusses the design of the simulation tool according to the design science paradigm by von Alan et al. [1]. The application design will be derived from the requirements introduced in the previous chapter, followed by the structural choices supporting the high-level design. Krutchen's 4+1 view model [23] will be used as a guideline to present the software architecture. The model divides the system in four main views: *physical view*, *logical view*, *process view* and *development view* – as follows:

1. *Physical View.* Describes the component distribution on a physical system, including how the software maps to hardware.
2. *Logical View.* Describes the system's key abstractions on a modular level.
3. *Process View.* Describes the activities and dataflow in the system during run-time.
4. *Development View.* Describes the software component breakdown from a development point of view, enabling a lean development process.

The first three views will be introduced in respective subsections 4.1, 4.2 and 4.3. The fourth view is omitted, since the software is developed by a single developer.

## 4.1 Physical View

Figure 4.1 depicts the role of the simulation tool as a part of the whole application. As stated in section 3.2.1, the simulation tool will behave as

a complete replacement for the physical IAVs, having to be able to work as an independent entity communicating with the client only via an external interface, namely the CAN standard. The CAN message frame data types are predefined.

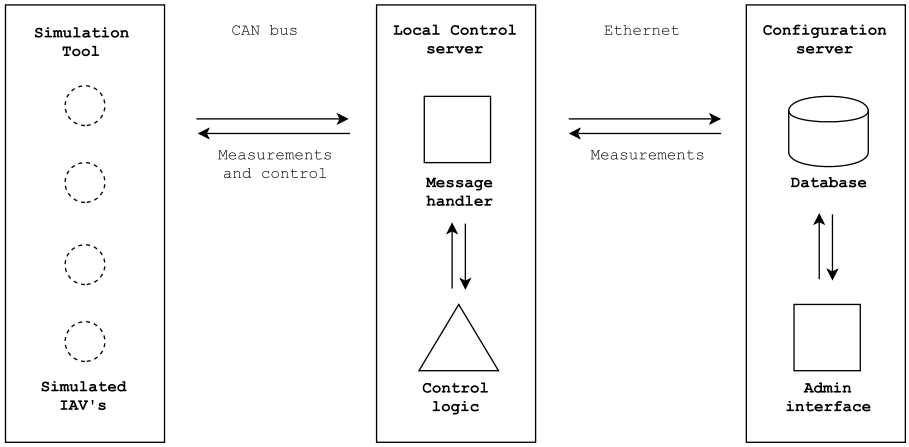


Figure 4.1: The simulation tool as a part of the main system architecture.

Throughout the development phase, the local control server and configuration server logic will be run on the same hardware to enable faster development process. The simulation tool will be run on the same hardware. Initially, a virtual CAN bus interface running on top of SocketCAN drivers will be used instead of a physical one.

## 4.2 Logical View

Figure 4.2 depicts the simulation tool's internal structure on a modular level. The tool can coarsely be divided into two main entities, the *simulation engine* and the *simulation server*. The simulation engine is solely responsible of running the simulation based on the model and configuration it has been given. On preset intervals, it reports simulation results, which are interpreted by the simulation server. The simulation engine receives asynchronous *device control messages*, which it needs to appropriately respond to by adjusting the simulator's internal state.

The simulation server is responsible for initialization of the simulation engine with the appropriate *configuration* and *simulation model*. Upon re-

ceiving new simulation results from the engine on each simulation iteration, the server interprets the results using an internal IAV mapping and sends the results forward to the CAN bus, which is observed by the *main system*.

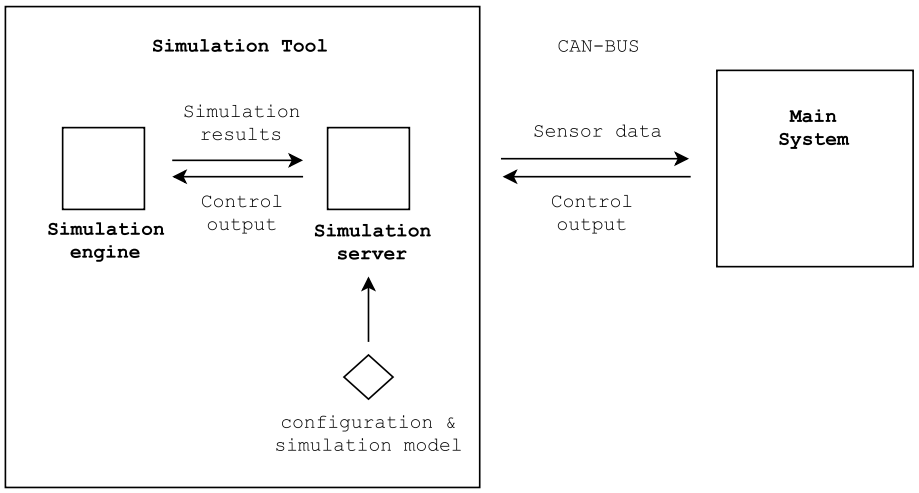


Figure 4.2: Internal architecture of the simulation tool. The simulation execution is governed by the simulation server, which receives the simulation model and configuration files on initialization. Measurement and control messages are handled via CAN bus.

### 4.2.1 Simulation Engine

A separate simulation program, namely the *simulation engine*, is used for running the simulation. The tool developed as a part of this thesis will be used to govern the simulation engine, and pass on the simulation results to the main system via the *simulation server*, working as a proxy.

Instead of developing a new simulator, an externally developed application is used as the simulation engine. Building simulation has been an ongoing branch of research for well over 50 years, having had enough time for the best models and practices to mature. A lot of knowledge about the building simulation and computer science is required in order to implement a well-functioning simulator. Using a well-matured, validated simulator the risks related to the code reliability are reduced. A wide range of available simulation software allows for choosing the one that best adheres to the requirements stated in Chapter 3.

The simulation engine module is responsible for maintaining an internal

representation of the simulated environment, adjusting it when necessary. It works in a tight co-operation with the simulation server sending simulation results to the server and receiving control values in return. This is done in short simulation iterations, the engine simulating the model a short time step forward, then adjusting the simulation according to the server's instructions. The section 4.3 gives in-depth explanation of the tool's run-time functionality.

The requirements R1 and R2 specify the quantities relevant to the simulation. The engine should report the following measurements to the simulation server each iteration:

- Distinguishable ID's for each IAV.
- IAV orifice openness representing its relative motor position.
- Contaminants, such as $CO_2$ and CO, for each IAV.
- Airflow magnitude in $l/min$ for each IAV.

After reporting the measurements, the engine waits for instructions from the server, including:

- IAV motor position in percentage for device control.
- Contaminant source adjustments, representing occupancy (inhabitant produced $CO_2$) and contaminant leaks (CO, $SF_8$).

### 4.2.2 Simulation Server

The simulation server acts as the main process for the simulation tool, initializing the simulation engine with a simulation model and settings specified in the configuration. It also maintains an internal device mapping, translating simulation results and control values between the two interfaces. The main tasks are:

- Initializing the simulation engine and establishing a connection to it.
- Running the simulation at a pre-configured speed, in short time-steps.
- Acting as a proxy (translator) between the simulation engine and the CAN bus.
- Handling the connection to the simulation engine, listening for new updates and forwarding control messages.

- Handling the connection to the CAN bus, listening for control messages and forwarding device measurement data.
- Maintaining an internal IAV mapping, translating the simulation model's air duct terminal points and zone measurements to IAV specific messages.

Multi-zone simulators track contaminant concentrations, temperatures and pressures in zone-sized control volumes. The airflows are often reported for each flow element, such as an orifice or duct terminal. Since the measurement messages required by the main system are reported on an IAV basis, a mapping is needed to handle the translation from zonal contaminant concentrations and terminal specific airflow readings to IAV measurements. Depending on the building size, a single room might contain several IAV devices. The simulation server's task is to act as a proxy, internally translating the engine's simulated values to CAN messages for each individual IAV by using internal mappings. Such mappings should connect the main system's IAV IDs to zones and air duct terminals in the specific simulation model used. The readings required from the simulation engine are listed in section 4.2.1.

Each simulated IAV is controlled by the main system sending specific control message frames from the CAN bus in short intervals. The simulation server should interpret the control messages and use the same internal device mapping to adjust the simulation accordingly, i.e. changing the duct terminal's opening size in the simulation model according to the control value. Figure 4.3 depicts the flow of data between the modules.

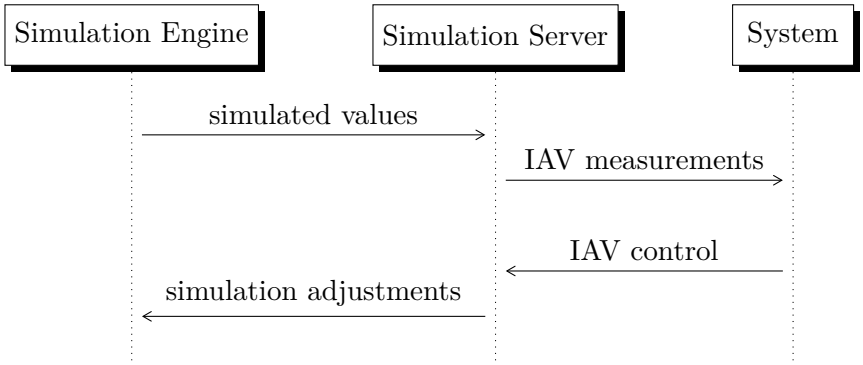


Figure 4.3: The flow of information between the simulation server and the main system. The simulation engine reports the simulation results to the simulation server module, which interprets and forwards them to the system. The system responds with a control message, which is used to adjust the simulated IAVs.

### 4.2.3 Model and Configuration

The simulation engine is be given a simulation model file containing the parameterized environment. A simulation multi-zone model file typically includes at least the building floorplan with volumetric dimensions, ductwork, and contaminant sources. Depending on the engine, the model file is created with an included floorplan editor or supported CAD software.

The configuration file includes at minimum the settings required for running the simulation. The configuration file is given in a human-readable format allowing anyone to modify the project configuration. At least the following settings should be present:

- The name and location of the simulation model file.
- Model specific mapping struct(s), that map the IAV IDs to the model.
- Simulation iteration timestep setting, to adjust simulation step forward in time on each measurement iteration.
- Simulation sleep timestep in seconds, to adjust the frequency of measurement messages sent to the CAN bus.
- A setting to enable/disable simulated CAN message timestamps at arbitrary time.

## 4.3 Process View

The simulation tool's runtime functionality can be divided into six steps, as depicted in the sequence chart of Figure 4.4. When the simulation server is started, (1) the engine is initialized with the simulation model and starts executing the simulation. The initialized engine (2) reports the initial values that the simulation server should keep track of. (3) A simulation loop is started and the engine reports the simulation results at 0 seconds. After every update step, the simulation engine halts the simulation and waits for a message from the simulation server to proceed with the next time step.

The simulation server receives the measurements from the engine, (4) prepares the measurement CAN messages, and forwards them to the CAN bus. At this point, the simulation server (5) starts to listen for the CAN bus for control messages from the main system's control subsystem. All received control messages (6) are interpreted and forwarded to the simulation engine. After having waited the predefined sleep time, the simulation server sends a message to the simulation engine to continue with the simulation to the next point in time. The engine simulates the timestep and the process starts over from step 3.

As demonstrated in Figure 4.4, the simulation server establishes a continuous loop that handles all the required functions in a pre-defined order between the transient simulation steps. Similar approaches are used in *coupled simulation*, where two multi-zone simulators handle the airflow and heat simulations in separate processes [6, Sec. 4.3].

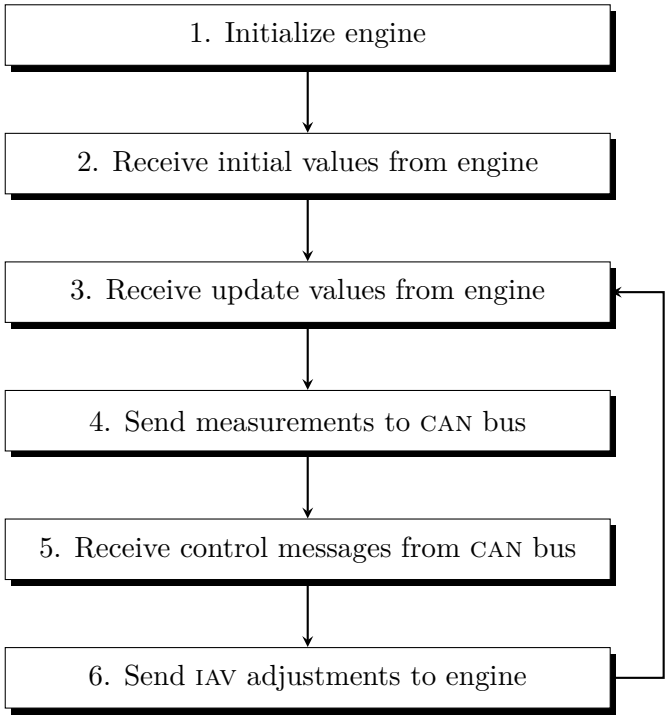


Figure 4.4: The simulation server's sequential runtime functionality. The steps 3. - 6. are executed repeatedly in a loop after initialization.

## 4.4 Simulation Software

### 4.4.1 Comparison

A list of the most popular multi-zone building simulation software was gathered based on simulation software reviews by Foucquier et al. [24] and Lorenzetti [14]. The selected software is evaluated against the non-functional requirements for the simulation engine in Table 4.1. The programs BSim, BUILDOPT-VIE and Clim2000 listed by Foucquier et al. [24] were omitted since no data regarding their current state was found.

The *licencing* row is a boolean value indicating whether the software is free for commercial use, as required. The second row, *development*, indicates the maintenance level of the software. This is evaluated using three categories: *active*, *maintained* and *abandoned*. Active means that recent updates have been made within the past rolling year. Maintained means the documentation and software is older than one year and only maintained (i.e. occasional bug fixes). Abandoned means that no recent updates have been

| | Licencing | Development | Documentation |
|---|---|---|---|
| COMIS | Free | Abandoned | Good |
| CONTAM | Free | Active | Exhaustive |
| EnergyPlus | Free | Active | Exhaustive |
| ESP-r | Free | Active | Exhaustive |
| IDA-ICE | Paid | Active | Good |
| TrnSys | Paid | Maintained | Good |

Table 4.1: Comparison of the most popular multi-zone simulation software.

made within the past five years.

The *documentation* row depicts the quality of documentation and tutorials for the software. It is evaluated into three categories: *exhaustive*, *good* and *poor*. Exhaustive documentation includes both the user guide and developer guide for the software, as well as tutorials. Good documentation is lacking in one of the preceding aspects and poor is lacking in more two or more aspects.

Based on the comparison in Table 4.1, IDA-ICE and TrnSys can be left out because of the free licence requirement. Even though COMIS can be used for free, it is not actively developed or maintained. This leaves three suitable alternatives: CONTAM, EnergyPlus and ESP-r. Two aspects will be inspected for each of these three simulation programs, namely the *ease of software coupling* and the *tools provided for modeling*.

One of the primary concerns in the choice of a simulation engine is its ability to couple with the simulation server. In order to control the IAVs in real-time, the simulation engine needs to have an interface for externally managing the simulation during an ongoing simulation. Therefore, the requirements set for the simulation engine specifically apply to the communication of the simulation engine.

The available means for constructing the simulation model should also be assessed since modeling is a crucial aspect of the project. In general, the model creation should be easy to learn, relatively fast and not dependent on external software.

### CONTAM

CONTAM [25] is a multi-zone indoor air quality and ventilation analysis program designed for simulating airflow rates, contaminants and occupancy scenarios within a building. Contam consists of two separate programs,

*ContamW* and *ContamX*. ContamW works as a GUI for creating and editing simulation models, as well as configuring and running simulations. ContamX acts as a separate simulation engine, capable of running independent simulations from the command line, without the GUI.

ContamX supports simulation coupling through a TCP/IP socket interface, allowing for external servers to communicate with the engine during simulation. The socket interface is however limited to reading airflows and contaminant concentrations. Adjusting simulation parameters apply to constant type control nodes only. CONTAM its capable of *transient simulation* via the socket interface, advancing the simulation in short time-steps governed by the simulation tool.

CONTAM comes with a graphical model editor *ContamW*, offering a wide set of features for creating multi-zone simulation models. It also supports running the simulation engine from the user interface, which is useful for testing the implemented model validity. The ContamW model editor only works in Windows environments, as opposed to the engine's required UNIX environment. This necessitates a separate Windows installation for creating and editing models. The simulation process is completely separated from the modeling process, however. A ContamW model can be edited separately in a Windows virtual machine, and the project file transferred to the simulation tool after creation and validation in ContamW.

#### EnergyPlus

EnergyPlus [26] is a modular, console-based, whole building simulation programs designed primarily for building energy simulations. It is capable of simulating both transient heat and mass transfer within a building. The documentation is exhaustive and separate, detailed guides are provided ranging from basic tutorials to advanced use and application development. The intended use for EnergyPlus is to provide a simulation engine around which third-party interfaces can be wrapped. Both the inputs and outputs are easily parseable ASCII text files.

EnergyPlus supports run-time simulation coupling using the external software, called *Building Controls Virtual Test Bed* (BCVTB) [27]. It is an application that can be used to interface EnergyPlus with external components, such as another simulator, hardware or various visualizations and interfaces. BCVTB can be configured to run as a console application. Simulation data and control parameters in BCVTB are communicated via Berkeley Software Distribution (BSD) sockets.

EnergyPlus offers an interface for external user interfaces, enabling a wide range of tools to use it with. IDF Editor is supplied with each EnergyPlus installation to create input files. It is a simple table based editor, allowing the creation and modification of model components using a spreadsheet-like grid as an interface. Several third-party graphical user interfaces have been developed, usually with specific energy simulation use cases in mind. These vary by functionality, usability and licensing.

#### ESP-r

ESP-r [28] is a general-purpose tool for building simulation. It supports thermal, inter-zone and intra-zone airflow, HVAC system and electrical power flow simulations. ESP-r has been under development since 1974, and the source code is freely available under GPL. ESP-r documentation for both learning and developing is thorough. Only UNIX systems are supported. However, the software is able to run on Windows under the Cygwin environment.

Simulations are managed using a *Central Project Manager* which governs the simulation engine, visualizations, reports and database support, among others. Building geometry in an ESP-r model is defined using included or third-party CAD tools, using a "click-on-grid" interface or an imported image. Components are assigned to the geometry using the component library included with the project manager.

ESP-r has been coupled with external software by Jost [29] for TRNSYS. Both required the development of new custom components for both software in order to enable data exchange during transient simulation since ESP-r does not offer communications via other means [30].

### 4.4.2 Engine Choice

CONTAM was chosen as the simulation engine for its good balance between usability and features. The modeling process using CONTAM's included GUI, ContamW is easy and relatively fast to learn following a guide. ContamW also comes with all the required software packages included. Both EnergyPlus and ESP-r only include very basic interfaces for model creation. External interfaces require existing CAD modeling experience or settling for commercial options.

All three engine alternatives supported simulation coupling – CONTAM through an existing socket interface, EnergyPlus through BCVTB software or plain BSD sockets and ESP-r through offering adequate documentation

for implementing custom components. Out of these, CONTAM has a clear advantage in the simplicity of the coupling process, and EnergyPlus in the amount of variety it offers. Implementing coupling for ESP-r would require time and additional research, which is outside the scope of this thesis.

# Chapter 5

# Implementation

A simulation tool implementation following the design introduced in Chapter 4 was created. Section 5.1 will introduce the physical context the tool was implemented for, followed by a description of the simulation server module and interface implementations in 5.2.

## 5.1 Development Environment

The main system runs on a 64-bit Ubuntu platform, offering a virtual CAN interface for IAV communications. The prototype tool was also implemented on the same operating system, ease the development process. The simulation tool was developed to be run as a separate process, completely independent from the rest of the system. It only communicates via virtual CAN. The choice of virtual CAN interface enables a purely software-based development environment, completely removing the typical hardware needs of a CAN adapter and a physical bus. It also allows for the use of a range of *can-utils* debugging tools made for SocketCAN, such as *candump* and *cangen*.

The graphical model editor ContamW is run on a Windows 8.1 virtual machine since only Windows operating systems are currently supported. The created simulation models were transferred to the development environment using a shared folder between the virtual machine and the host.

The Python programming language was used for the development of the simulation server module. The thesis author has experience in Python and furthermore, a Python library for CAN communications had already been created for other purposes, allowing for code re-use in this tool. Python offers great capabilities for TCP/IP socket communications through the means

of a built-in SocketServer standard library, allowing for rapid software development independent of platform.

## 5.2 Simulation Server Implementation

Figure 5.1 demonstrates how the simulation server is divided functionally into three classes. The main script `server.py` is used to launch the simulation tool. It will initialize required loggers and instantiate a socket server using Python's built-in TCPServer class. After the server is successfully running, a ContamX simulation engine client process is started and coupled with the server. A `TCPServer` class is instantiated on program execution. It establishes a TCP Socket connection to the simulation server and starts listening for new socket messages. When a socket message is received, method `handle()` of class `ContamMsgHandler` is invoked. Socket messages are recognized and parsed, as specified in CONTAM documentation [12], using a custom `ContamMsgParser` class.

**TCPServer** handles the connection to the simulation engine. The server functionality is implemented using Python's standard library SocketServer. SocketServer was chosen for its high-quality documentation. The `TCPServer` class is given a `ContamMsgHandler` instance, which is used to handle the arriving socket messages.

**ContamMsgHandler** acts as a handler class implementing the main logic for the simulation server as specified in Figure 4.4. Internally, the `handle()` function is called once for each new connection. Since we want to maintain the socket connection with the engine, an infinite loop is run after initialization. The socket is handled as a byte stream, which means that correctly sized chunks of data need to be read from the stream. The class keeps track of the simulation time in order to set the timestamp in CAN messages properly. Depending on the simulation speed, the internal `simulation_time_counter` variable is incremented by the correct amount of seconds for each simulation iteration.

The message type and size are detected from the socket message header and then the `ContamMsgParser` class is used to parse the data from the chunk. The message handler class works internally as a simple finite state machine (FSM). Depending on the message received, the server either (1) unpacks the data, such as concentrations and airflow measurements, and

continues to receive messages or (2) moves to a sleeping state for a short time period, where it handles CAN communications by sending device measurements and receiving control messages. Two Python libraries are used for CAN messaging. The *python-can* library is used to interface with SocketCAN drivers that can be used to connect with real CAN devices, or with a virtual CAN bus. Mutually, the CanFrame class in `canframe.py` implements methods for reading and building CAN messages in host system specific format.

**The ContamMsgParser** class contains the methods for parsing various socket messages specified in the CONTAM user guide [12]. An internal dictionary is maintained containing the message type as an integer key and a reference to a corresponding parser function as its value. Nine different parser methods have been implemented to parse ContamX's socket messages. Another dictionary variable, `simulation_parameters` temporarily saves the information in the received messages. This solution is sufficient for the purpose, since the simulation data received on each iteration is only slightly re-organized and then passed forward to the CAN bus to be saved in a time series database by the main system. In addition, the simulation tool saves all its actions, including received and sent messages to a log file, that can be used for debugging both the simulation and the message passing.

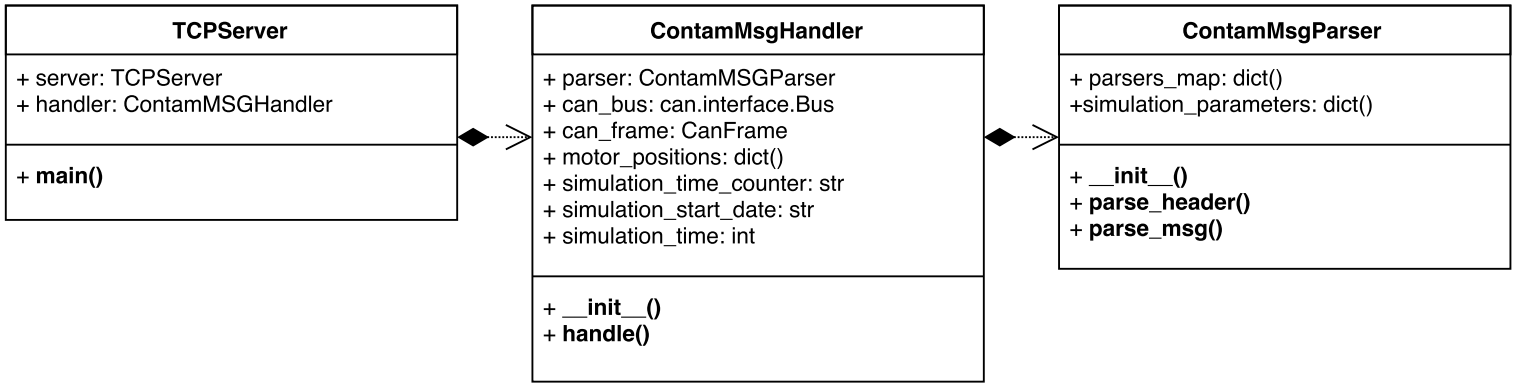


Figure 5.1: A Unified Modeling Language (UML) diagram of the server implementation. A `TCPServer` instance initializes a TCP/IP Socket server, with a custom message handler `ContamMsgHandler`. The socket packages are parsed using a custom `ContamMsgParser` class.

## 5.3 Modeling Process

### 5.3.1 Overview

Law and Kelton [4, pp. 106-109] describe a detailed process for implementing a simulation model. The process consists of seven steps: *planning*, *definition*, *implementaion*, *validation*, *experimentation*, *analysis* and *demonstration*. Because of the general nature of the workflow, most of its steps can be easily adapted to the building simulation study at hand. Although the process is not completely followed in the implementation of the experimental model, its good practices for modeling are used.

The modeling process begins with the careful formulation of the problem. Parameterizing the problem, finding the key constraints and interaction with the stakeholders are advised. If measured data or blueprints are available, they should be used in the process. The modeling process should be started with a less detailed model in mind and supplemented as required, leading to potential improvements both in simulation performance and factors causing model inaccuracies [4, p. 107]. A validation of the model is recommended by comparing to real measurements or by the help of experts (e.g. similar previous research). Frequent validation throughout the process is encouraged [4, p. 108].

### 5.3.2 Model Implementation

The experimental simulation model implemented in Section 5.3.3 was created using CONTAM's model editor *ContamW*. It offers a wide variety of tools for implementing a simulation model. Figure 5.2 portrays this editor. It consists of a single window, containing a large sketchpad area, toolbar, and a status bar. The simulation model is drawn on the sketchpad by using the tools selected from the toolbar. The most important tools available include *walls and boxes* for drawing floorplan layouts, *ducts* for designing ductworks and air handling systems and *controls* for attaching an adjustable control signal to various elements to control their behavior during simulation. [12]

The experimental simulation model was created based on the feedback gathered from the client. The floorplan for the simulation model is shown on Figure 5.3. The floorplan represents passenger ship cabins in terms of physical dimensions and structure. It is a simple layout that can be scaled up to several groups of cabins at ease by copying the given layout when creating the simulation model.

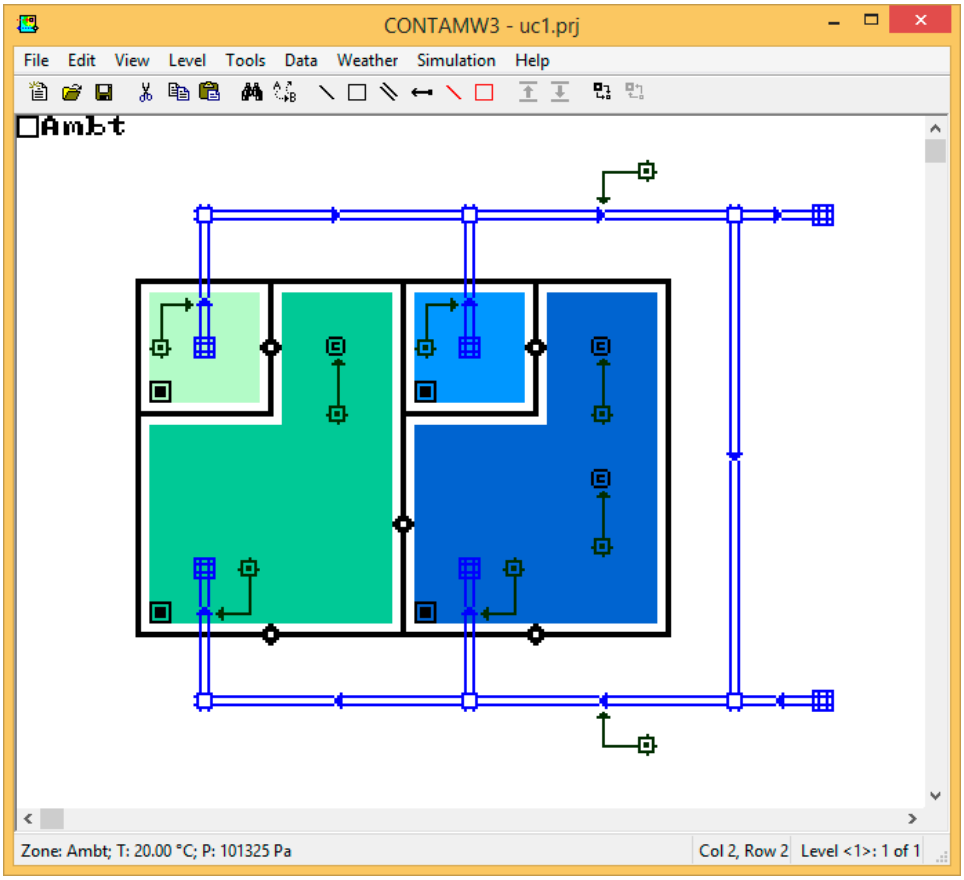


Figure 5.2: ContamW graphical simulation model editor. The experimental simulation model depicts two passenger ship cabins with small toilets, and a complete ventilation ductwork.

Figure 5.2 depicts the final multi-zone simulation model created based on the given floorplan. The model contains two separate cabins with singular intake vents inside each of them. The exhaust vents are situated in the cabin toilets, resulting in a positive airflow towards the exhaust vents.

The modeling process in ContamW begins with *defining zones* for both cabins and toilets. Their physical dimensions, structure, and airflow throughput were estimated by the client since no real passenger ship floorplans were yet available. An individual cabin was estimated to be roughly of size 4.0 m $\times$ 3.0 m $\times$ 2.3 m each.

CONTAM requires at least one continuous leakage path from a zone to ambient zone. Cabin walls were estimated to be made out of material with low leakage properties (metal) and the air either flows through the

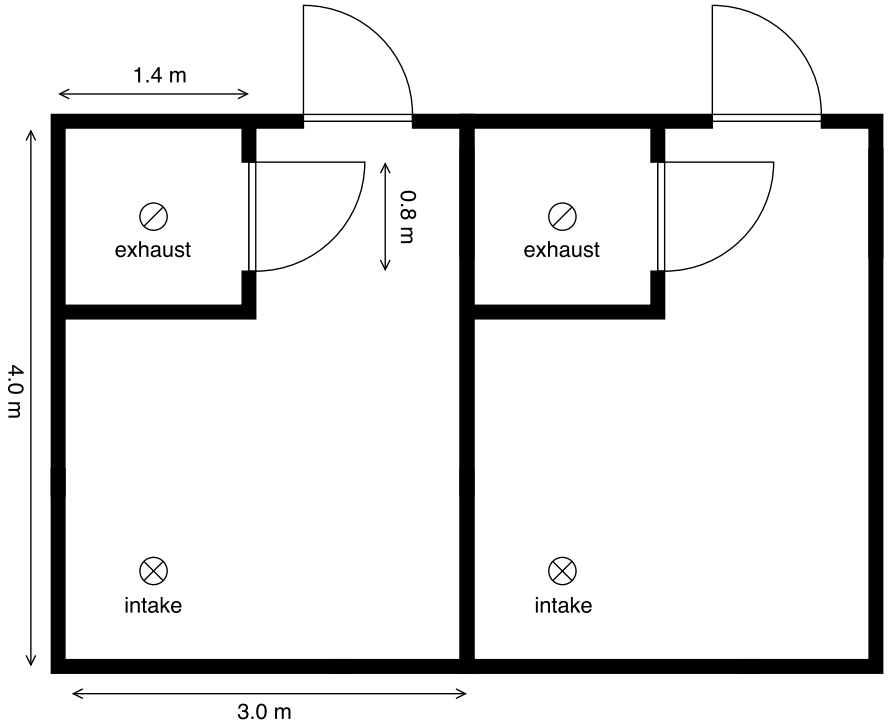


Figure 5.3: Floorplan depicting a small, two-cabin entity in one floor level connected to a singular air handling system within a passenger ship.

open toilet door or a small crack below the closed door to the exhaust vents leading to a closed air circulation. A leakage path following *effective leakage area in 4 Pa pressure* (ELA4) rating [11, Sec. 16.15] was defined for each wall using the `EXT_WALL_METAL` component from CONTAM's supplied airflow element libraries. In addition toilet doors and windows were modeled using a *Wall Leakage Path* element, transferring air from cabin intake vents to toilet exhausts.

Two primary ways for implementing a ventilation system in a simulation model are offered: using an existing *air handling system* component or building the complete ductwork from individual blocks. The latter was chosen for more flexibility in ductwork layout. The intake and exhaust fans need to be defined separately for this strategy. The fans were adjusted for an hourly air change rate (ACH) of 2, meaning that twice the amount of air in the cabins are circulated in the air handling system every hour [31].

CONTAM offers a special *control component* for reading measurements and adjusting various other components. The control components' values

can be adjusted using the simulator's socket interface. In order to achieve freely controllable and readable IAV devices, three separate components were used for each device. For reading IAV data, an air duct *terminal* was used for measurements. A separate control node was attached next to the duct terminal controlling an adjustable, circular shaped *segment airflow element*.

Additional components were added to the model, including controllable contaminant sources for $CO_2$ and CO, and various flow paths representing structural pressure leakage through walls and ceiling. The fans' airflow rates and the amount of contaminant release can be adjusted via normal CAN control messages by using the appropriate ID for the element, which is set during model creation.

### 5.3.3 Model Limitations

De Wit [6, pp 25-56] introduces factors of uncertainty in the building simulation modeling process, representing the factors by which the simulation output can differ from the real result. *Specification uncertainty* is caused by the partial or incomplete relevant information on the modeled environment.

The experimental simulation model for CONTAM was created primarily to test the simulation tool. The largest limiting factor in the modeling process is probably the lack of modeling experience in general. As stated in Section 5.3.1, the careful formulation of the problem is a key factor to achieving accurate models. The lack of actual information on cabin layout, structural choices and ventilation systems is an issue since only conjectures about the structures could be made.

Most probably the experimental model does not directly scale up, due to the ventilation system layout. Ventilation systems vary ship-by-ship and use devices from multiple manufacturers. The level of control offered by the systems vary. The ducts' dimensions and airflow resistance properties require verification.

The cabin walls, windows, and doors were modeled using existing CONTAM libraries. Real environment measurements should be used to verify their correctness. A single study was found regarding cabin air leakage and fire-proofness by Arvidson et al. [32]. It could be used as a guiding document for structural choices in early development phases. Structural leakage was completely disregarded in the model.

Currently, the IAV device is modeled as a simple segment airflow element representing circular orifice flow using a power law model. However, the device has a slightly downward opening for the air, causing different air-

flow patterns. It is not known whether this will be a significant factor in simulation results.

De Wit's *scenario uncertainty* refers to external conditions imposed on the building. For the implemented experimental model, weather conditions, the heating effect of the Sun and occupant behavior were left out. Although control for $CO_2$ sources was enabled in the models, its scheduled behavior were left to the simulation tool user.

# Chapter 6

# Evaluation

The implementation of the simulation tool detailed in Chapter 5 will be verified against the SRS presented in chapter 3 to show that all the requirements are satisfied. Section 6.1, *Experimentation* works as a demonstration of how the simulation tool works during runtime, which will be used to evaluate the majority of the non-functional requirements. The functional requirements will be reviewed in section 6.2, *Requirements Verification*.

## 6.1 Experimentation

The simulation tool is run with a multi-zone simulation model based on a floorplan representing two small passenger cabins connected to the same air handling system. This is used to demonstrate the functionality of the simulation tool – namely the IAV functionality and the changing occupancy scenarios in cabins ($CO_2$ production) to observe IAQ levels.

The implementation of the simulation tool was run with the main system in an expedited mode. The two-cabin simulation model represented in figure 5.2 was used for experimentation. ID's of the simulated IAV devices were added to the main system configuration in order to monitor the four simulated IAV devices in the GUI. The simulation was adjusted during runtime using control CAN messages to demonstrate that the tool is able to work with the main system as intended.

The simulation tool is run using a command line interface as shown in Figure 6.1. The simulation tool can be configured to print the CAN and socket communications to the command line in order to debug the behavior. Additional details, such as message type and its data are also printed. A log file is created for retrospective analysis.

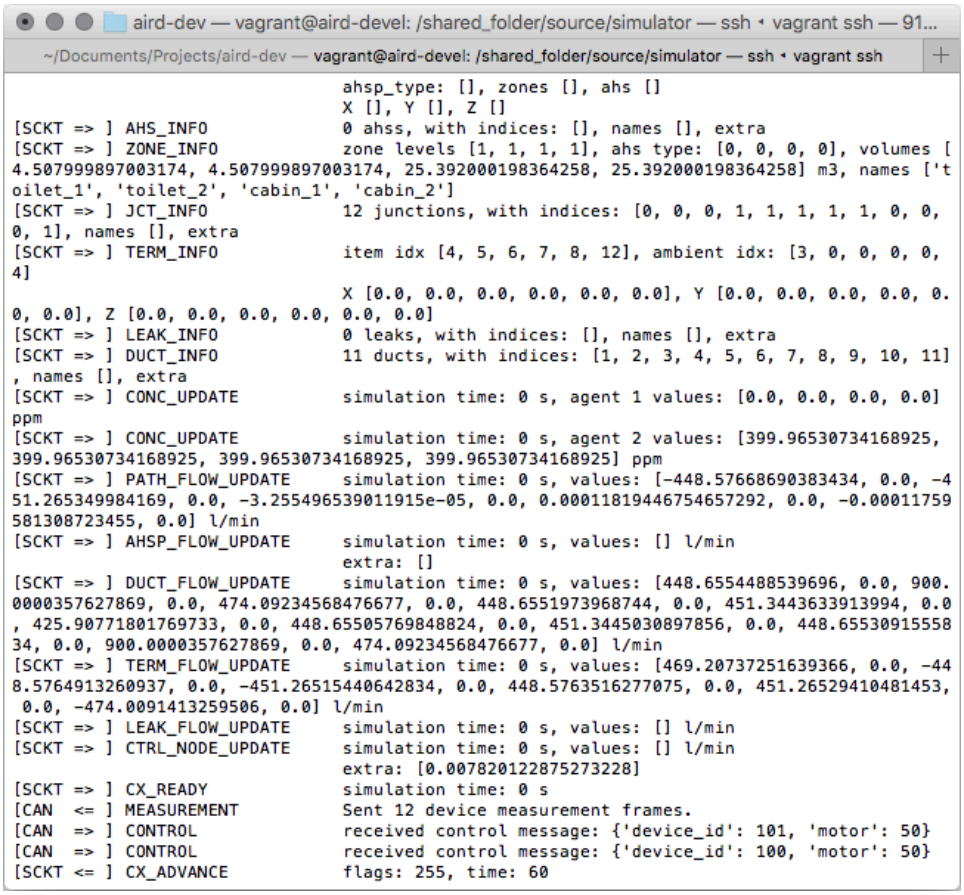


Figure 6.1: Simulation tool running on Ubuntu console. Helpful debug messages are printed to the screen and logged during simulation.

Figure 6.2 shows a screenshot from the main system cabin details GUI during simulation tool execution. The graph represents the simulated measurements of an individual IAV device located in the cabin intake vent. The y-axis is the carbon dioxide concentration detected by the device, the x-axis represents the flow of time. The simulation was run at 60 times expedited speed. Three points have been marked to the graph, representing various events triggered during simulation. All the adjustments to the simulation model were made by sending control messages with the appropriate IDs via a virtual CAN bus.

Initially, all the IAV devices were set to be 50% open and the carbon dioxide source in cabin 1 is set at 10% representing a single occupant in cabin 1. The air handling system intake and exhaust fans were set to blow at a rate of 25% of the maximum, at 135 $m^3$/h. At 9000 seconds (1), the

contaminant source is set to 20% indicating the introduction of a second occupant in cabin 1, which results as a rising curve. At 12 000 seconds (2), both the intake and exhaust IAV devices in cabin 1 were increased to 75% to compensate the second occupant. At 15 000 seconds (3) since simulator initiation, an adjustment message is sent to increase the total flow of air in the ventilation system by doubling the maximum airflow in the air ducts, to 270 $m^3/h$.

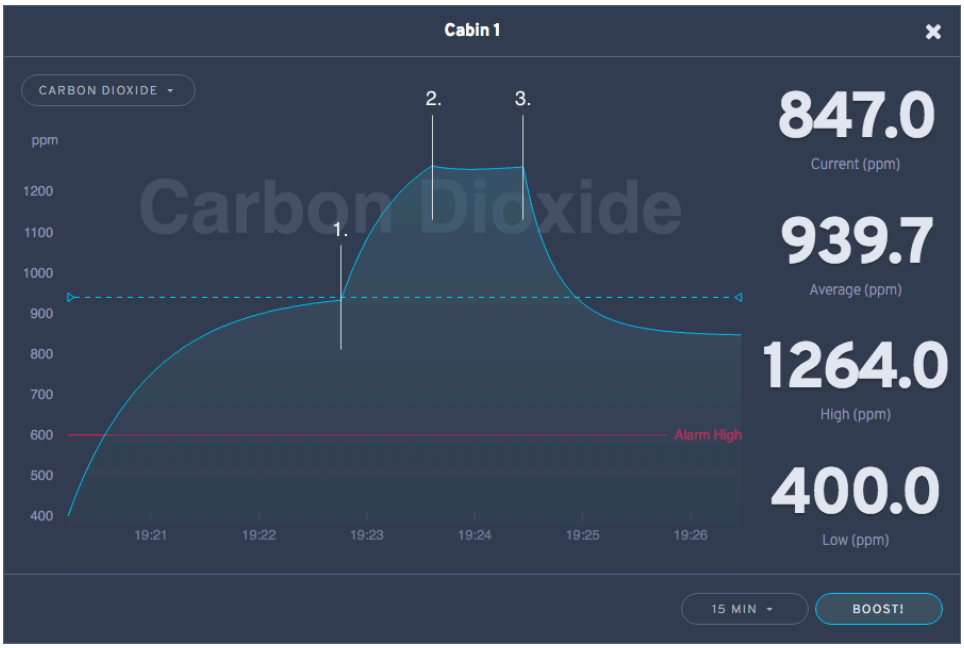


Figure 6.2: A screenshot of the main system GUI showing carbon dioxide measurements for a simulated IAV device situated in the first cabin (dark green room in Figure 5.2). Three adjustments were made to the model during the simulation using CAN control messages.

## 6.2 Requirements Verification

The simulation tool implementation is validated against the previously listed requirements. This section goes through each requirement specified in Sections 3.3.1 and 3.3.2 evaluating the implementation against them.

**R1 - Simulation Engine** Requirement R1 addresses the relevant physical quantities for the main system, including $CO_2$, CO and airflow. This requirement mainly concerns itself with the simulation engine module as

specified in the design. The simulation software *CONTAM* was chosen to run the simulation in the implementation.

The CONTAM model has been widely acknowledged by the scientific community to produce good results, satisfying the requirement R1.1. Emmerich [33] conducted a review on a number of multizone model validation studies. Out of these, eight address the CONTAM simulation model, including a three-phase comparison made by Haghighat and Megri [34] and a two-model comparison conducted by Dutton et al. in 2008 [35], which both show good results.

Emmerich's review [33] shows that all of the reviewed studies except one show a good correlation between the model prediction and empirical measurements with correlation coefficients $R$ from 0.92 to 0.99. The single poor correlation of 0.27 was believed to have been caused by a mix of a measurement error, uniformly assumed leakage and certain coefficients estimated from the literature. In his review, Emmerich states based on the results that *"a knowledgeable user can make reasonable predictions of air change rates and contaminant concentrations for residential-scale buildings"*. However, he also warns against trusting a single multi-zone IAQ model validation study because of the inherent limitations in scope.

CONTAM supports reporting the contaminant concentrations for each contaminant in a zone through the socket API, as required in R1.2. *Contaminant Source* components can be used to introduce new contaminants to the simulation either as a single burst or continuously. Furthermore, an additional *control* element can be attached to the source element to be able to adjust the amount of introduced contaminants via socket API during simulation run-time. Airflows are reported by the CONTAM socket API for each flow path component in the model. This satisfies R1.3.

**R2 - IAV Simulation** R2 sets the requirements for the functionality that the simulation tool must provide in terms of IAV behavior. The IAV's are represented by *airflow terminal* elements in CONTAM's model editor. The terminal elements need to be named as an integer that corresponds to the main system ID in the model editor (R2.1). A separate *duct segment* element with a constant control node is added next to the terminal element, to simulate an adjustable opening for the airflow, satisfying the requirement R2.3. The IAV devices are mapped to the rooms (albeit manually) in the simulation server module, using a `ZONE_MAPPING` configuration value (R2.2). The requirement R2.4 is satisfied: the simulation server sends periodic mea-

surement messages based on the model's air duct terminal points reported values.

**R3 - CAN Interface** A library for both custom control and measurement CAN messages has been implemented using the python-can library. It offers wrappings to transceiving (R3.1, R3.2) messages via SocketCAN interface, which can be mapped to either a virtual bus or a real physical one (R3.3). The interface has been thoroughly tested by a set of unit tests.

**R4, R5 - Transient Simulation** CONTAM supports transient simulations, running the simulation forward one short time step at a time. This is done via the socket API, sending appropriate messages to the simulation engine on each simulation step.

The shortest transient time step is freely configurable in the simulation server, with a minimum span of one second (R5.1) [12]. Device measurements are sent after each simulation step (R4.2). By default, the simulation server sleeps for the duration of a single time step to maintain the time stamp consistency for measurement messages. Even though the simulation is not exactly done in real-time, the discrete nature of the simulation is sufficient, as long as the simulation time step is smaller or equals to the measurement sending interval (R4.1).

The functionality for expedited simulations is implemented in the simulation server (R5.2). By adjusting the ratio between the IAV measurement interval and a single simulation time step, various simulation speeds can be achieved. For example, by progressing the simulation 10 seconds on each step and sending measurement every 2 seconds, the simulation is run at a fivefold speed. The expedited simulation is limited by the computational resources available – a short measurement interval might not leave enough time for the preceding simulation calculations to finish.

The time step length has an indirect effect on simulation results in CONTAM. For simulation models with large or particularly long volumes, such as atriums or hallways, the contaminant concentrations might not agree with real-world measurements – CONTAM instantly iterates the average concentration of contaminants within a volume, even though the transportation process has delay in a real-world environment. The results can be improved by changing the time step scale from seconds to minutes, giving contaminants time to spread.

**R6 - Model Creation** The ContamW model editor supports addition and removal for rooms of various sizes and shapes (R6.1). The total dimensions and volumes are freely modifiable. The CONTAM manual [12] lists 26 different models for an airflow path element to choose from (R6.2). The requirement R6.3 states that the model must support different material properties for a wall. This can be achieved by creating a new airflow element for each wall to model the air leakage rate. Various rates are listed in industry books, such as the *ASHRAE Fundamentals handbook* [11] or from a readily available library distributed on the CONTAM website [36]. CONTAM supports all the elements relevant to the simulation (R6.4): various multi-floor ductwork elements as well as complete air handling systems, windows, doors and a number of specialized elements for runtime control [12].

**R7 - Model Export** The simulation model can be directly saved after creation in ContamW. It also supports loading existing models and external libraries thus fulfilling the requirement R7. The simulation server does not support saving of simulation state, however.

**R8 - Operating System** R8 requires the simulation tool to be able to run from a 64-bit Ubuntu command line. Even though the graphical editor ContamW does not support Linux operating systems, the simulation engine ContamX that is used to run the simulation does. The 64-bit version of the engine executable was used during the development of the simulation server code, but it turned out to contain a bug in the 64-bit implementation of the socket API, providing inconsistent messages to the simulation server module. This is circumvented by enabling 32-bit execution on the Ubuntu server, and running the 32-bit ContamX executable for Linux instead. Thus, the requirement R8 is satisfied.

**R9 - Modeling Process** Requirement R9 demands the modeling process to be well documented and feasible by an unacquainted user after an introduction. The model creation process was tested by giving a developer unacquainted with CONTAM a task to implement a simple simulation model from scratch. The developer was able to implement a functional model for the simulator in roughly 40 minutes following the simulation modeling guide that was made for the tool. The modeler was occasionally assisted in the usage of ContamW in cases where the guide was lacking.

The ContamW GUI satisfies the requirement R9. The unclear parts of the simulation tool modeling guide were written down and accordingly amended to the documentation afterwards R9.1. ContamW editor supports adding, deleting, copying and pasting elements in the drawing pad, satisfying R9.2. The ContamW GUI fully satisfies R9.3.

**R10, R11 - Simulation Performance** The simulation engine's computational performance was evaluated against three different sized models. Separate simulation timing functionality was implemented in order to measure out the computational time for each simulation step. For each measurement, the simulator was run at least 60 seconds (or a minimum of 5 iterations) uninterrupted and an average was calculated from the timings of individual simulation iterations. CAN bus messaging was disabled for the testing. The simulator was run on a 64-bit Ubuntu virtual machine on a mid-2014 Macbook Pro laptop. Table 6.1 presents the results of these performance tests.

Three simulation models of varying size were used for the engine performance evaluation. The first model of 2 cabins is depicted in figure 5.2. It consists of 17 airflow nodes and 22 airflow paths. The second model is an upscaled version of the first, depicting a group of ten cabins connected into a single air handling system. The second model consists of 66 nodes and 102 connecting paths. The third model depicts an entirety of 40 cabins, including 261 nodes and 417 paths. The number of equations the simulation engine needs to solve for these models is 17, 66 and 261, respectively. Figure 6.3 visualizes the performance of the simulator in terms of simulation time step length. Fitting a line for each measurement set, we can conclude the simulator performance to be linear for each model.

Since a new simulation step is run every five seconds, a simulation step length of 5 s represents the real-time performance, a 60 s step an expedited 12× speedup, and a 1 h step a 720× speed up. The time slept between each simulation iteration is used to receive CAN messages. This means that if the simulation takes longer than the defined sleeping time (5 s by default), the simulation tool is not able to perform as required. Extrapolating from the results, upper limits for simulation speedups with 5-second simulation intervals are roughly 134480× (2 cabins), 23308× (10 cabins) and 2852× (40 cabins).

The requirement R10 is satisfied since the simulator is easily capable of running the simulation for the largest model of 40 cabins (80 IAV devices) in real time, a single iteration taking less than 50 ms on average. It can

also be assumed that the simulation tool is capable of running expedited simulations of up to 14260 seconds (3 h 57 min 36 s) of simulation every 5-second messaging cycle. Concluding from the table 6.1, a solid performance for both real-time and expedited simulations can be assumed, satisfying requirement R11.

It is important to notice that these performance timings are valid only for the specific simulation model and hardware the tests were run with. These results are still valid for estimating the scale of the computational resources required by the simulation. Also, the simulation server's logic and CAN communications both add minor overhead on the total performance of the simulation tool.

| | **2 cabins** | **10 cabins** | **40 cabins** |
|---|---|---|---|
| **5 s** | 0.039126 s | 0.038714 s | 0.041515 s |
| **1 min** | 0.039471 s | 0.042641 s | 0.060337 s |
| **1 h** | 0.066391 s | 0.196984 s | 1.189498 s |
| **1 d** | 0.708617 s | 3.738567 s | 29.714049 s |
| **3 d** | 1.919500 s | 11.188641 s | 86.760934 s |
| **5 d** | 3.262505 s | 18.456103 s | 148.269696 s |
| **7 d** | 4.498375 s | 25.750495 s | 199.282409 s |

Table 6.1: Performance test results for three different simulation models ran with a range of simulation time steps. The results indicate the average computing time for a single simulation step in seconds.

**R12 - Engine Quality and Licensing** CONTAM software suite is distributed under a BSD license, allowing for free redistribution and modification provided the derivative works bear notice about the original author and any modifications (R12.1). The CONTAM software has been under active development since the first version of CONTAM's predecessor AIRNET, released in 1989. Afterward, three major versions of CONTAM have been released, including several smaller updates (R12.2). CONTAM also has an active user group, the developers answering questions related to the software. CONTAM offers exhaustive documentation in the form of a *User Guide and Program Documentation* [12]. Additional information can be found from the website [25] in the form of several tutorials and helper software (R12.3). [12]

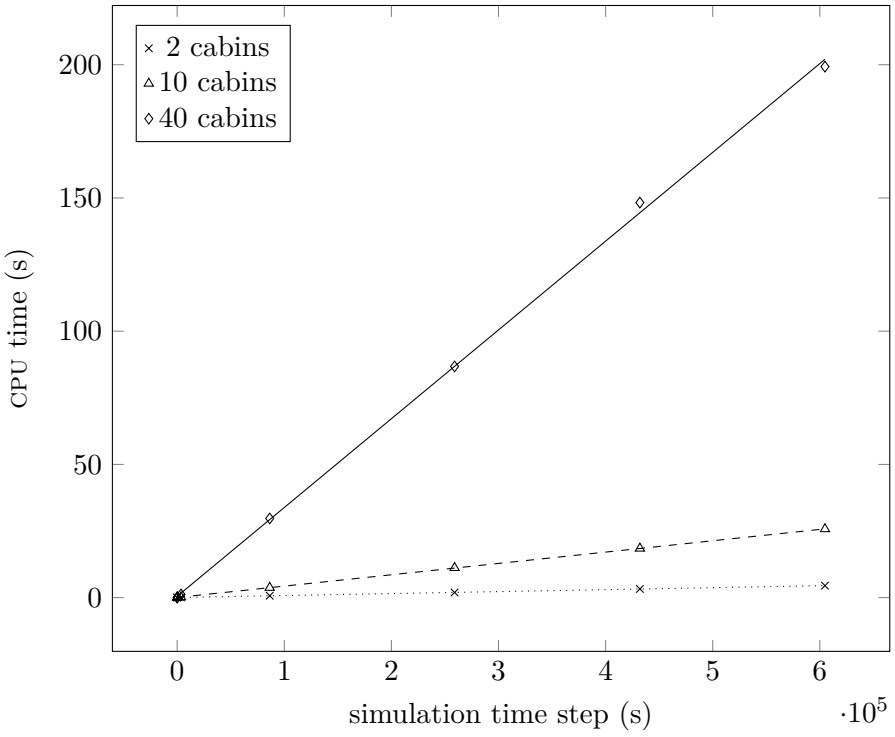


Figure 6.3: Simulator performance visualized in terms of time step length. A linear fit is calculated for each model's measurements. The dotted fit represents the two-cabin model, dashed fit the ten-cabin one and the solid fit the forty-cabin model.

# Chapter 7

# Conclusion

## 7.1 Summary

The objective of this thesis was to define and implement a tool suitable for simulating a smart indoor air quality monitoring system for passenger ships. This was done, among other reasons, in order to enable the further control subsystem development. The development was carried out in four phases, following the design science guidelines by von Alan et al. [1]. A custom software requirements specification based on the IEEE 830 standard [20] was gathered with the stakeholders. Based on the requirements specification, a literature survey on indoor airflow simulation models was made. The knowledge gained was applied in the design that introduces the architectural layout for the software. Finally, an implementation was realized and experimented with to demonstrate operation.

The most important requirements were related to the core functionality of the software: being able to simulate multiple intelligent air ventilation devices in passenger ship's premises robustly. The physical quantities were confined to airflow magnitudes and contaminants such as CO and $CO_2$. In order to quickly test out various control scenarios, it was important that the simulation tool was able to perform in an expedited mode in addition to real-time simulation. The choice of using an existing simulation application instead of developing own was made, since broadly validated software suitable for the use case was found. This also allowed for support from more experienced users of the software.

Several airflow simulation models were evaluated in the literature review. The simple analytical and experimental models were found to be too application-specific. The models utilizing computational fluid dynamics

were found to be too computationally heavy and required vast knowledge of fluid dynamics from the modeler. Multi-zone simulation models were found to be fit for the purpose, offering a good balance between performance and result detail. The simulation software CONTAM was found to offer suitable communication capabilities in order to couple it with the product.

The software design fulfilling the requirements is presented as three architectural views. The tool was designed as a separate, modular entity, able to run without ties to the main system. This was achieved by implementing a CAN bus interface that was used to send measurements to the main system and receive device control messages from the control subsystem. Furthermore, the simulation engine ContamX was decoupled from the simulation server by using socket messages to control the simulation flow. The communication between the server, CAN bus and sockets were handled in a single thread in a sequential manner. This was found to be a performant solution without the problems of messaging concurrency.

The simulation tool implementation was carried out based on the architectural design. The biggest problems with the implementation ended up being caused by a bug in the 64-bit ContamX executable regarding sizes of socket message variables. This was circumvented by using 32-bit executable on the 64-bit server. The implementation was found to fully comply with the requirements specification. The simulation performance was adequate. Even for bigger models spanning 80 simulated IAV devices, a single simulation loop in real time took 0.04 seconds on average running in a modern laptop. A linear extrapolation leads to a maximum of 2852 times expedited simulation speed for the same model for a measurement interval of 5 seconds per device.

## 7.2 Future Work

The ContamW editor offers a set of tools for the quick creation of simulation models, including limited copying and pasting of elements and even whole floors. In order to enhance model accuracy, real measurements need to be made. Multiple resources offer charts for the leakage properties of various construction elements, but for models as specialized as passenger ship cabins, very little existing research was found in terms of structure leakage properties. It is highly recommended to do real airflow measurements in a small-scale pilot environment to evaluate the simulation model, before moving on to testing the control subsystem on larger models.

Several improvements could be made to the simulation model to enhance its functionality and usability. Mapping the model zones and devices in the tool's configuration file was found to be laborious. This could be automated by parsing the CONTAM project file and creating the mappings during software initialization.

Currently, ContamX offers no way of reporting zone pressures using the socket interface. An improvement to the simulation tool could be made that parses the zonal pressures from CONTAM's log files during execution, in order to ensure that the control subsystems do not cause notable pressure differences between various zones. Model generation could also be automated to an extent, since the CONTAM user guide [12] has a thorough documentation of the project file format. However, the apparent complexity of the airflow network with several nodes, airflow elements, control elements and ducts, including keeping track of element indices and connections might cause issues.

The modeling process was found to be limited by factors mainly related to incomplete information on the modeled environment, scaling, and external conditions. Delay, noise, and disturbances were not accounted for in the simulation tool. Using the simulation tool to develop a robust HVAC control system requires implementing a disturbance model for IAV sensors based on measured data. Initiating simulated sensor faults may be of use in the development of error detection systems. A more realistic simulation result could be achieved by simulating delay.

The modular design of the simulation tool enables customization for use cases beyond the scope of this work. The simulation engine can be replaced, e.g. to support CFD or energy simulations. The CAN bus can be replaced or modified to write results directly to a database, enabling data analysis on a larger time scale.

# References


[1] R Hevner von Alan, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

[2] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[3] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 2.1*. John Wiley & Sons, 2010.

[4] Averill M Law and W David Kelton. *Simulation modeling and analysis*, volume 2. McGraw-Hill New York, 1991.

[5] Christopher A Chung. *Simulation modeling handbook: a practical approach*. CRC press, 2003.

[6] Ali Malkawi and Godfried Augenbroe. *Advanced building simulation*. Routledge, 2004.

[7] Martin Barták, Frantisek Drkal, Jan Hensen, Milos Lain, Tomas Matuska, Jan Schwarzer, and Borivoj Sourek. Simulation to support sustainable HVAC design for two historical buildings in Prague. In *Proc. 18th Conference on Passive and Low Energy Architecture, PLEA*, pages 903–908, Florianópolis, Brazil, 2001. Passive and Low Energy Architecture.

[8] Qingyan Chen. Ventilation performance prediction for buildings: A method overview and recent applications. *Building and environment*, 44(4):848–858, 2009.

[9] Shaun D Fitzgerald and Andrew W Woods. The influence of stacks on flow patterns and stratification associated with natural ventilation. *Building and Environment*, 43(10):1719–1733, 2008.

[10] Youngjun Cho, Hazim B Awbi, and Taghi Karimipanah. Theoretical and experimental investigation of wall confluent jets ventilation and comparison with wall displacement ventilation. *Building and Environment*, 43(6):1091–1100, 2008.

[11] ASHRAE. *Fundamentals Handbook, SI Edition.* ASHRAE, 2013.

[12] W. S. Dols and J. Polidoro. Contam user guide and program documentation. *NIST Technical Note 1887*, 2015.

[13] Helmut E Feustel and Alison Rayner-Hooson. Comis fundamentals. *Report LBL-28560, US Department of Energy, Lawrence Berkeley Laboratory*, 1990.

[14] David M Lorenzetti. Assessing multizone airflow simulation software. In *Proceedings of the Indoor Air 2002*, volume 1, pages 267–271, Monterey, California, 2002. Lawrence Berkeley National Laboratory.

[15] Zhengen Ren. *Enhanced modelling of indoor air flows, temperatures, pollutant emission and dispersion by nesting sub-zones within a multizone model.* PhD thesis, Queen's University of Belfast, 2002.

[16] Zhao Zhang, Wei Zhang, Zhiqiang John Zhai, and Qingyan Yan Chen. Evaluation of various turbulence models in predicting airflow and turbulence in enclosed environments by cfd: Part 2—comparison with experimental data from literature. *Hvac&R Research*, 13(6):871–886, 2007.

[17] Zhiqiang John Zhai, Zhao Zhang, Wei Zhang, and Qingyan Yan Chen. Evaluation of various turbulence models in predicting airflow and turbulence in enclosed environments by cfd: Part 1—summary of prevalent turbulence models. *Hvac&R Research*, 13(6):853–870, 2007.

[18] Andrei N Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. In *Dokl. Akad. Nauk SSSR*, volume 30, pages 301–305. JSTOR, 1941.

[19] Ian Sommerville. *Software Engineering (9th Edition).* Pearson, 2010.

[20] IEEE. IEEE Recommended Practice for Software Requirements Specifications. Standard, 1998.

[21] Texas Instruments. Introduction to the Controller Area Network (CAN). `http://www.ti.com/lit/an/sloa101a/sloa101a.pdf`. [Online; accessed 2016, March 21.].

[22] Kvaser. Higher Layer Protocols. `https://www.kvaser.com/about-can/higher-layer-protocols`. [Online; accessed 2016, March 29.].

[23] Philippe Krutchen. The 4+1 model view of architecture. *IEEE Software*, 12(6):42–50, 1995.

[24] Aurélie Foucquier, Sylvain Robert, Frédéric Suard, Louis Stéphan, and Arnaud Jay. State of the art in building modelling and energy performances prediction: A review. *Renewable and Sustainable Energy Reviews*, 23:272–288, 2013.

[25] National Institute of Standards and Technology. CONTAM Website. `http://www.bfrl.nist.gov/IAQanalysis/CONTAM/`. [Online; accessed 2016, May 12.].

[26] National Renewable Energy Laboratory. EnergyPlus Website.

[27] Michael Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 4(3):185–203, 2011.

[28] Energy Systems Research Unit. ESP-r Website.

[29] Romain Jost. *Developing a run-time coupling between ESP-r and TRNSYS*. PhD thesis, École Polytechnique de Montréal, 2012.

[30] J. W. Hand. Esp-r developers guide. *Energy Systems Research Unit of the University of Strathclyde*, 2012.

[31] ASHRAE. Ventilation and Acceptable Indoor Air Quality in Low-Rise Residential Buildings. Standard, 2003.

[32] M Arvidson, J Axelsson, and T Hertzberg. Large-scale fire tests in a passenger cabin. *SP Technical Research Institute of Sweden, Report*, 33:1–100, 2008.

[33] Steven J Emmerich. Validation of multizone iaq modeling of residential-scale buildings: A review/discussion. *Ashrae Transactions*, 107:619, 2001.

[34] Fariborz Haghighat and Ahmed Chérif Megri. A comprehensive validation of two airflow models—comis and contam. *Indoor air*, 6(4):278–288, 1996.

[35] Spencer Dutton, L Shao, and S Riffat. Validation and parametric analysis of EnergyPlus: air flow network model using CONTAM. In *Third national conference of IBPSA-USA*, Berkeley, California, 2008. International Building Performance Simulation Association.

[36] National Institute of Standards and Technology. CONTAM Libraries. `http://www.bfrl.nist.gov/IAQanalysis/CONTAM/libraries.htm`. [Online; accessed 2016, May 12.].