Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelor-Thesis

Erik Suer

Concept Development of a Cessna 150 Digital Twin using Multi-Agent-Systems

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Erik Suer

Concept Development of a Cessna 150 Digital Twin using Multi-Agent-Systems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Flugzeugbau
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
MARS Group
Hamburg University of Applied Sciences
Department of Computer Sciences
Berliner Tor 7, 20099 Hamburg, Germany

Erstprüfer: Prof Dr.-Ing. Detlef Schulze
Zweitprüfer: Prof. Dr. Thomas Clemen

Abgabedatum: 06.03.2019

Abstract

Erik Suer

Title of the paper

Concept Development of a Cessna 150 Digital Twin using Multi-Agent-Systems

Keywords

Digital Twin, (Multi-) Agent-System, Industry 4.0, Cessna, Aerospace Engineering, Digitalization, Simulation, Pilot, Air Traffic Controller, Aircraft

Abstract

In order to digitize and improve operational procedures with interacting digital twins in aviation industry, new approaches are necessary. One approach is the use of multi-agent systems. The objective of this thesis is to develop a concept for a digital twin using a multi-agent system, which models a Cessna 150 as well as its pilot and the air traffic controller. The first implementation of this system is based on the "MARS" framework. The MARS group represents all “Multi-Agent Research and Simulation” activities at the University of Applied Sciences Hamburg at the Department of Computer Science. The developed and implemented concept demonstrates a first simulation approach, which aims to model the flight operation from preflight check to takeoff. This concept and its first implementation are presented and evaluated. Afterwards, with exploring the potential of multi-agent systems, further recommendations for the application along the flight operation stages are done. The result of this thesis offers a better understanding of multi-agent system in aviation industry and acts as a first foundation for future research with MARS in this area.

Thema der Bachelorthesis

Konzeptentwicklung eines digitalen Zwillings einer Cessna 150 mit Multi-Agenten-Systemen

Stichworte

Digitaler Zwilling, (Multi-) Agenten Systeme, Industrie 4.0, Cessna, Luft- und Raumfahrttechnik, Digitalisierung, Simulation, Pilot, Fluglotse, Flugzeug

Kurzzusammenfassung

Um die Digitalisierung und Verbesserung der Betriebsabläufe mit interagierenden digitalen Zwillingen in der Luftfahrtindustrie zu ermöglichen, sind neue Ansätze erforderlich. Ein Ansatz ist die Verwendung eines Multi-Agenten Systems sein. Ziel dieser Arbeit ist es, das Konzept eines digitalen Zwillings unter Verwendung eines Multi-Agent Systems für eine Cessna 150 mit einem Piloten und einem Fluglotsen zu entwickeln und eine erste Implementierung mit Hilfe des „MARS“ - Frameworks umzusetzen. Die MARS-Gruppe repräsentiert alle „Multi-Agent Research and Simulation“ Aktivitäten innerhalb der Hochschule für angewandte Wissenschaften Hamburg am Institut für Informatik. Das entwickelte und implementierte Konzept demonstriert einen ersten Ansatz realistischer Simulationen für den Flugbetrieb vom Preflight-Check bis zum Takeoff. Dieses Konzept und seine erste Implementierung werden vorgestellt, bewertet und anschließend mit der Entdeckung des Potenzials von Multi-Agenten Systemen weitere Empfehlungen für die Anwendung entlang der Flugbetriebsphasen gegeben. Das Ergebnis dieser Arbeit bietet ein besseres Verständnis für Multi-Agenten Systeme in der Luftfahrtindustrie und stellt eine erste Grundlage für zukünftige Arbeiten mit MARS in diesem Gebiet dar.

Acknowledgements

Even though this is only the beginning of my journey and hopefully not my last scientific research, I benefited from the support of so many people and I would like to show my gratitude.

First, I have to thank my mother, who has dedicated her life to her children. We are very grateful for her unconditional love and support. Also I would like to thank my brothers Eddi and Kilian for always having my back, and my father, who may rest in peace, for showing me the important things that matter in life.

Thank you Teila, for all your love and emotional strength.

This interdisciplinary thesis would have not been possible without the support and guidance of my supervisors Prof. Dr.-Ing. Detlef Schulze and Prof. Dr. Thomas Clemen. Thank you for always providing me assistance and for insightful, memorable and entertaining discussions about a very exciting topic.

I owe special thanks to the MARS members who were always assisting me during coding or were there for an exchange. Especially I would like to thank Daniel and Julius for their valuable input.

Last but not least, I want to thank Sabine, Marwin and Riccardo for the time spent on reading this thesis. Also thanks for the great semesters, Marwin!

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

AOC	**A**irline **O**peration **C**ontrol
ATC	**A**ir **T**raffic **C**ontroller
ATM	**A**ir **T**raffic **M**anagement
CAD	**C**omputer **A**ided **D**esign
CAS	**C**alibrated **A**ir **S**peed
CS	**C**oordinate **S**ystem
DSL	**D**omain **S**pecific **L**anguage
EAS	**E**quivalent **A**ir **S**peed
GIS	**G**eo **I**nformation **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
GS	**G**round **S**peed
IAS	**I**ndicated **A**ir **S**peed
IoT	**I**nternet **o**f **T**hings
ISA	**I**nternational **S**tandard **A**tmosphere
LWT	**L**eft **W**ing **T**ank
MARS	**M**ulti **A**gent **R**esearch and **S**imulation
MRO	**M**aintenance, **R**epair and **O**perations
MSL	**M**ean **S**ea **L**evel
MTOW	**M**aximum **T**ake **O**ff **W**eight
NACA	**N**ational **A**dvisory **C**ommitee for **A**eronautics
PD	**P**roportional **D**erivative
PLM	**P**roduct **L**ifecycle **M**anagement
PPL	**P**rivate **P**ilot **L**icense
ROC	**R**ate **O**f **C**limb
RPM	**R**evolution **P**er **M**inute
RWT	**R**ight **W**ing **T**ank
TAS	**T**rue **A**ir **S**peed

List of Symbols

Symbol	Name	SI Unit
a	Acceleration	m/s^2
b	Wing span	m
c	Distance coefficient	m
C_{D0}	Zero lift drag coefficient	-
C_{Di}	Induced drag coefficient	-
C_L	Lift coefficient	-
$C_{L\alpha}$	Lift curve slope	1/rad
C_P	Power coefficient	-
C_T	Thrust coefficient	-
d	(Propeller-)diameter	m
d_{hav}	Haversine distance	m
D	Drag	N
e	Oswald factor	-
E	Glide Ratio	-
F_f	Friction force	N
F_N	Normal force	N
g	Gravitational acceleration	m/s^2
h	height	m
J	Advance ratio	-
L	Lift	N
L_c	Lapse rate	K/m
lat	Latitude	◦
lon	Longitude	◦
m	Mass	kg
n	Revolution	1/s
p	Pressure	Pa
P	Power	W
r	Radius	m
R	Gas constant	J/(kg K)
Re	Reynolds number	-
S	Wing area	m^2
T	Thrust	N
T_K	Temperature	K
V	Speed	m/s
W	Weight	N
α	Angle of attack	rad
α_0	Zero lift angle	rad
Δ	Angle difference	-

η	Efficiency	-
Λ	Aspect ratio	-
φ	Angle	rad ($\circ$)
ρ	Density	kg/m^3
θ	Pitch angle	rad

Chapter 1

Introduction

1.1 Motivation

The use of buzzwords like "Industry 4.0", "Big Data" and "Internet of Things" is extremely fashionable nowadays. The enormous impact on airlines- and aviation industry (Mattig and Hausweiler, 2017; Singh, 2019) through the advancement of digital transformation is visible. Innovative business models like e.g. the "Digital Cabin" (Altran, 2019), "Digital Twin" or "Remote Tower Control" (Scheurle, 2019) are fostered by multiple aviation companies. Especially the "Digital Twin" is an exciting topic in aviation industry with platforms like "Skywise" by Airbus (Airbus Group, 2019a) or "Aviatar" by Lufthansa Technik (Lufthansa Technik AG, 2019). Further on, aviation industry has been a pioneer in digital twin technologies, although tackling big challenges in the IT-landscape due to documentation requirements (Wendenburg, 2017).

Researching on digital twins and discovering a new conceptual approach with "Multi-Agent-Systems" (MAS) is a great opportunity to contribute to a current major topic in aviation industry. Working within the "MARS Group" (1.5) at the University of Applied Sciences Hamburg for designing a digital twin of a Cessna 150, pilot and an air traffic controller along the flight operations is a great opportunity. Furthermore, working on state of the art analysis to enlighten actual multi-agent system applications in aviation industry, implementing first lines of code of the developed concept within the "MARS Framework" and to work among an ambitious student team is a pleasure.

1.2 Objectives

The main objective of this thesis is to develop a concept of a digital twin using a multi-agent system for a Cessna 150, pilot and air traffic controller. Additionally, a review after implementing the first program architecture within the “MARS DSL” from the University of Applied Sciences Hamburg is done. The developed concept shall demonstrate a first approach of a realistic digital representation for flight operations and include the flight operation stages from the preflight check to takeoff. Furthermore, the implementation of the developed concept and its review are supposed to offer a first foundation for future work by exploring the usability and potential of MAS with the MARS DSL for the different flight operation stages. Hence a structured assessment and sustainable architecture will be established. Additionally, this research reviews the state of the art and the usability of this concept with MARS for further applications beyond this thesis in aviation industry.

1.3 Structure

Depending on the readers experience with multi-agent systems, aviation and its mathematical foundation, the chapters 2 to 4 provide a good introduction and overview to these topics, but also all the necessary theoretical background reference for the developed concept. Afterwards, the chapters 5 to 6 present the developed concept, its first implementation, the first results and the review. In chapter 7 an extensive outlook and suggestions for future work are given.

Chapter 2	State of the art analysis of current technologies with digital twins and multi-agent-systems, introduction to MARS DSL
Chapter 3	Theoretical aviation foundation for the agents and scope used in this thesis
Chapter 4	Mathematical and physical foundation for equations and models used in the developed concept
Chapter 5	Presenting the developed concept and technical model, from overall architecture to agents and further features, present the implementation of the developed concept and the integration into the MARS DSL framework
Chapter 6	Presenting the simulation and implementation results, evaluation of the results, achievements with the implementation, usability review and further challenges
Chapter 7	Conclusion and outlook for future work
Appendix A	Guideline for running the simulation, GitHub reference and external variable input overview
Appendix B	Overview of implemented and further features
Appendix C	Content overview of enclosed CD

1.4 Literature overview

For exploring multi-agent-system in general, Uhrmacher and Weyns (2009) and Wooldridge (2002) provide great information, especially Wooldridge (2002) is said to be one of the most important pioneer for MAS. Important work in the aviation industry using MAS is provided by Bouarfa (2016) with simulations on airport performance, safety assessment of active runway crossing operation or airline operations. Especially important for this work is the “Pilot’s Operating Handbook” (Cessna Aircraft Company, 1977), which provides a lot of information about the procedures and the aircraft used in this thesis.

1.5 MARS Group

The MARS group represents all "**M**ulti-**A**gent **R**esearch and **S**imulation" activities at the University of Applied Sciences Hamburg in Germany at the Department of Computer Science. As a scientific research project, the first concepts of the MARS framework arose in 2014 due to the need for a highly scalable, generic and high-performance framework for multi-agent simulation (Hüning et al., 2014). With the goal to make the MARS framework more accessible for people with various backgrounds, the MARS DSL emerged, which is used in this thesis. In 2020, the MARS DSL provides a high-level agent-based modeling language and it aims to achieve an easy implementation service for domain experts through a model-to-code generator with a possible user-friendly web interface. (Hüning et al., 2016; Glake et al., 2017).

More information on current projects, workshops and research papers can be found at https://mars-group.org/.

Chapter 2

State of the Art

This chapter is designed to provide first insights on digital twins and multi-agent systems and to distinguish between both of them. Each topic is illustrated with current definitions and industry applications, especially in aviation industry. Additionally, multi-agent systems using MARS is briefly introduced.

2.1 Digital Twin

2.1.1 Definition & Explanation

There are many different definitions for digital twins out there. For example Wikipedia suggests nine different definitions as stated in various scientific literature (Wikpedia, 2019). But put simply *it's a virtual representation of anything from the real world* (Kuhn, 2017; Software Education, 2018). This could be:

- a physical object, e.g. a car or an aircraft
- a process, e.g. manufacturing process

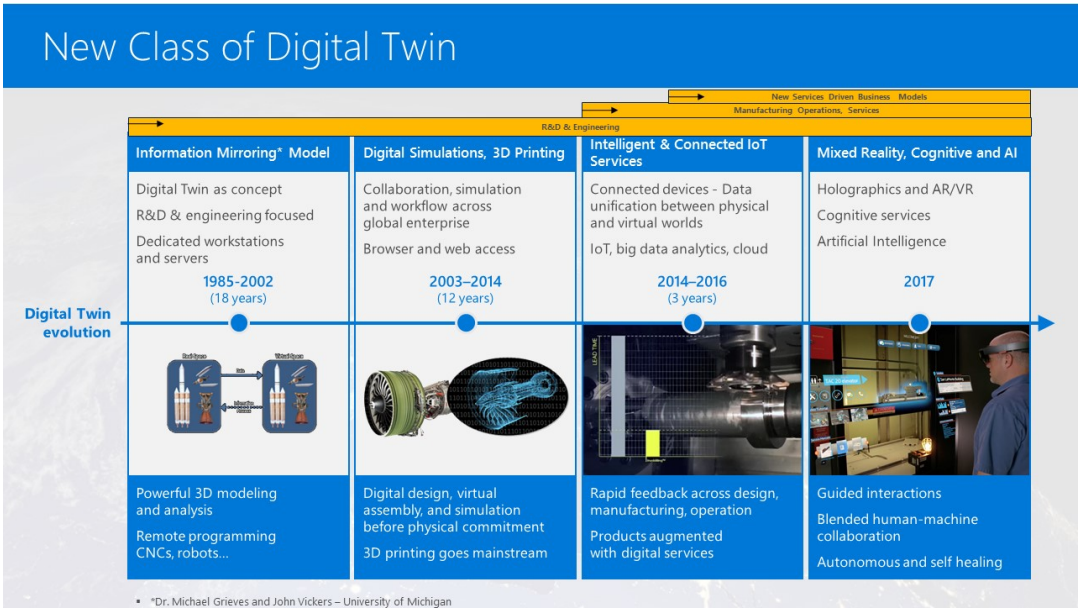

FIGURE 2.1: The evolution of a digital twin (Miskinis, 2018)

The concept of a digital twin has been around since the start of the new millennium and it‘s widely acknowledged that it is originated by Michael Grieves in 2002 (Thilmany, 2017). Figure 2.1 outlines the stages of the evolution from digital twins and shows, that engineers have been using product model technologies for decades, since e.g. every CAD model itself is partially a digital twin. However, "Internet of Things" (IoT) technologies (proposed beginnings in 2014 in figure 2.1) enabled a synchronization between the physical model and the corresponding virtual model by gathering real-time data through sensors. This leads to many more innovative business models to arise by improving operations and increasing efficiency with the aid of analyzing real-time data and thus current operating conditions (Marr, 2019). It should be mentioned that it doesn't matter for digital twins, whether the real counter part exists or not, since digital twins are also used for fast prototyping, production planing and simulation in a virtual environment.

2.1.2 Applications of Digital Twins

Which industries and applications benefit from the use of digital twins and how do they generate value? Some examples are given below:

Manufacturing Industry – The digital twin serves as a virtual replica of the actual occurrences on the factory floor in real time. Aside from the actual product, thousands of sensors enable collecting data like behavioral characteristics of production machines and environmental conditions. This data can be explored and gives insights into unfavorable trends or performance issues, which can be further investigated and solved before potential loss of production (Parrott and Warshaw, 2017).

Healthcare Industry – The ultimate goal of the healthcare industry is having a digital twin of each human, in order to help doctors and medical centers provide better medical advice to the patient based on his individual medical history. Further projects of an even more detailed human digital twin "organ by organ" are being discussed (Bland, 2018).

General Products – The digital twin revolutionized the concept of the "Product - Lifecycle - Management" (PLM). With the aid of the digital thread (Rüchardt, 2019), which covers data of every stage of the lifecycle of the product, new important insights are achieved. E.g. these insights could be operational inefficiencies of a wind farm or an engine, which allows for an improved design in the future.

2.1.3 Digital Twin in Aviation Industry

What are the benefits of a digital twin for the aviation industry? The aviation industry has been a forerunner of digital twin technology with using digital models of commercial aircraft to keep track of aircraft across the world. Nowadays there are different applications that should be mentioned.

One of the first use cases was the reduction of certification costs and time with the aid of simulation models (as in other engineering disciplines), where already first parts of the aircraft were described in form of multiple digital twins, which can be simulated within a given environment. Another early business case was a digital twin for a jet engine (compare General Eletric (2019) and Rolls Royce (2019)). As described in section 2.1.2, this leads to improved operational insights and lower maintenance cost through predictive maintenance and engine degradation prediction. In general,

digital twins are shaping the future of "Maintenance, Repair and Operations" (MRO) services. In Europe, two big players in aviation industry are developing platforms for digital twins: Airbus with "Skywise" (view figure 2.2) and Lufthansa Technik AG with "Aviatar" (view figure 2.3). These platforms set their goal to gather all produced data of an aircraft during its entire life cycle and generate a digital twin for every component within each aircraft. Some of the provided benefits are supposed to be (Airbus Group, 2019b):

- "Improved fleet operational reliability and efficiency through predictive and preventative maintenance"
- "Optimised aircraft performance through flight operations data analytics"
- "Rapid root-cause analysis of in-service issues"

According to Airbus, an A320 on a flight produces 10 gigabytes of data per flight hour with 24000 parameters for their platform "Skywise" (Airbus Group, 2018).

FIGURE 2.2: Skywise
Skywise by Airbus (Airbus Group, 2019a)

FIGURE 2.3: Aviatar
Aviatar by Lufthansa Technik (Lufthansa Technik AG, 2019)

FIGURE 2.4: Flight Simulator of an A320 (Flugsimulator-Vergleich, 2019)

Another early form of a digital twin in the aviation industry is represented by a flight simulator. The goal of an flight simulator is the representation of the actual aircraft. As pilots are trained continuously in these flight simulators, the cockpit layout, aircraft systems, procedures and flight characteristics need to be as realistic as possible. In figure 2.4 an exemplary flight simulator for an A320 is shown.

2.2 Multi-Agent-System

2.2.1 Definition & Explanation of an Agent

In general agents are a concept with diverse forms, thus there is no sole definition. One of the most widely used definition with an overall consensus is given by Wooldridge (2002):

> "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." (Wooldridge, 2002, p.15)

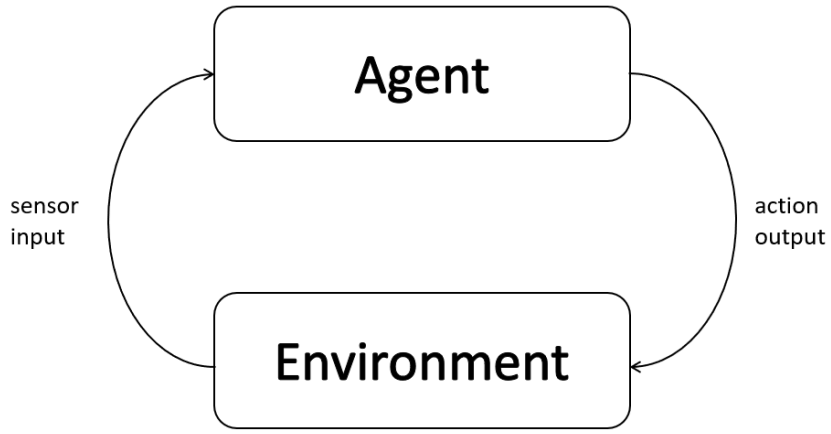

FIGURE 2.5: An agent interacting with the environment (after Wooldridge (2002, p.16))

In figure 2.5, the general concept is visualized. It is shown that alongside the agent the environment plays an important role. Russell and Norvig (2003) provide several proposed properties and modeling approaches regarding the characteristics and dynamics of an environment, which are beyond the scope of this work (see also Dorri, Kanhere, and Jurdak (2018)). One of the simplest forms of an agent acting in its environment is a control system, e.g. a thermostat. It makes use of a very simple decision structure to control the environment:

$$\text{too cold} \rightarrow \text{heating on}$$
$$\text{temperature OK} \rightarrow \text{heating off}$$

Nevertheless, the following properties distinguish "intelligent agents" from other concepts like objects, control systems as the mentioned thermostat or expert systems (Wooldridge, 2002, p.25-30).

- **Autonomy** – intelligent agents operate without direct input and have possibilities to control their actions and internal state

- **Social ability** – intelligent agents have the ability to interact with other agents to reach their design goal
- **Reactivity** – intelligent agents can recognize their environment and react to changes to achieve their goals
- **Proactiveness** – intelligent agents act determined and are able to take the initiative

Furthermore, different architectures for agents like the logic, reactive or cognitive agent are discussed in several academic resources like Uhrmacher and Weyns (2009) or Wooldridge (2002). The belief-desire-intention model for intelligent agents is one of the more notable models. These architectures and properties of an intelligent (cognitive) agent provide several possibilities and application areas, especially when using multiple coupled agents within a multi-agent system.

2.2.2 Definition & Explanation of a Multi-Agent System

The real benefit of agents is their ability to interact with each other in an environment to solve complex problems. This is known as a multi-agent system. In figure 2.6, a small multi-agent system example with two autonomous robots is visualized. As it is visualized, these robots communicate with each other within an environment, take actions, have their own goals and also have their own perception and representation of the world. Each agent type is programmed with a set of rules and behavior. Now thousands of them interacting within an predefined environment can be used to study collective behaviors. One very simple example is a predator-prey model. The predator hunts the prey and the prey is searching for food in the environment.

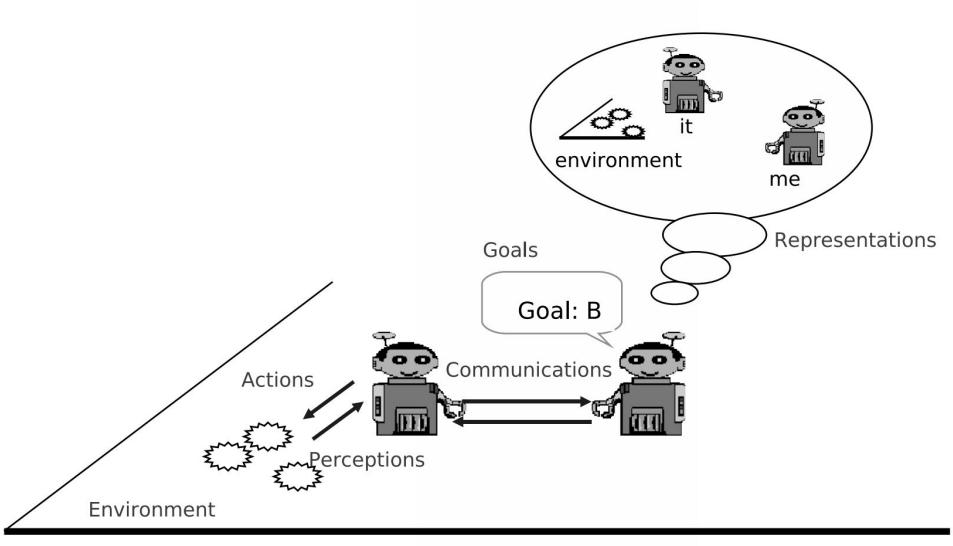

FIGURE 2.6: Example with robots of a multi-agent world (Uhrmacher and Weyns, 2009, p.14)

Regarding to actions on the environment and perceptions from the environment, every agent has its own sphere of influence (Wooldridge, 2002, p.105). That means for instance, that both robots from figure 2.6 might not be able to stand on the same place or perceive the same part of the environment. However, the communication

between the agents can vary from simple forms like direct messages with simple signals to more sophisticated ones like a blackboard, where agents can leave messages for each other (Garro et al., 2018).

2.2.3 Applications of Multi-Agent-Systems

A few notable applications for multi-agent systems in science and industry are listed below:

Manufacturing Industry – Multi-agent systems are used at manufacturing and production facilities to increase supply chain stability and production volume. For instance, when parts of the production chain are damaged, instead of continuing to run at full capacity, the individual production machines communicate with each other and lower the production rate automatically to prevent a major loss of production capacity or produce further malfunction (Tanajura, Oliveira, and Lepikson, 2015).

Social Sciences and Ecology (plus Biology and Chemistry) – One of the biggest strengths of multi-agent systems is the simulation of complex behavior of the human being and the nature and the processes occurring within humans and nature. Previous research within the MARS-group has demonstrated these possibilities and topics are ranging from “Krüger National Park Ecologists Support” (Lenfers et al., 2015) to “Firewood Collection in South Africa” (Lenfers, Weyl, and Clemen, 2018).

City and Built Environments – One major research topic is the optimization of transport systems in metropolitan areas through traffic simulation (also current research topic within the MARS group). Tasks range from simulating traffic flow with autonomous self communicating cars, self organized drones to traffic light pattern optimization (Uhrmacher and Weyns, 2009; Dorri, Kanhere, and Jurdak, 2018).

2.2.4 Multi-Agent-System in Aviation Industry

How are multi-agent systems (MAS) used in aviation industry so far? There are some applications, which automatically result from benefits in socio-technical systems such as simulating human behavior in cases like:

- Simulation of safety e.g. deplaning or an airport emergency (Bicharra et al., 2013)
- Simulation for boarding (Delcea, Cotfas, and Paun, 2018)
- Airport performance (Bouarfa, Blom, and Curran, 2012)

Especially in air traffic management (ATM), research regarding several topics has been conducted with the aid of multi-agent systems. As mentioned in chapter 1.4, Bouarfa (2016) did research on "emergent safety risk of an active runway crossing operation" and "Airline Operations Control (AOC) resilience" partly sponsored by "Single European Sky ATM Research" (SESAR) joint undertaking. Many other papers deal with air traffic management systems optimization, which is also supported by SESAR (e.g. Molina, Carrasco, and Martin (2014) and Micciche et al. (2013)).

Further applications for multi-agent systems in aviation industry are pilot assistants (Pieniazek, 2019) and workload measurement of air traffic controllers during "Human in the Loop" simulations (Dausch et al., 2019).

2.3 And now a Digital Twin using a Multi-Agent-System?

Joining both previous sections 2.1 and 2.2 lead to the assumption, that multi-agent systems and digital twins applications are overlapping, since every multi-agent system is somehow a virtual representation of a process in the real world. One could say “every multi-agent system is digital twin, but not every digital twin is a multi-agent system”. When looking back at an example from the manufacturing industry, both areas can be distinguished: Digital twins can be the engineered product itself, a representation of the supply chain or the manufacturing tools and the digital trace of a product (refer to chapter 2.1.2). This digital twin can be enhanced by a multi-agent system, where units like e.g. the manufacturing tools are represented by an agent and can communicate with other units autonomously in a cooperative way. This can lead to an optimized supply chain stability and production rate (compare chapter 2.2.3).

Thus the focus of this work is to create a digital twin of a Cessna150 with a multi-agent system incorporating a pilot and an air traffic controller. The focus is to simulate the relations between these agents during flight operations and develop a first concept.

Nowadays digital twins of aircraft are mainly providing predictive analysis with the aid of historical information/ context of components or assembly units (see section 2.1.3). Implementing an aircraft in a multi-agent system with other cognitive agents like the pilot and the air traffic controller could lead to an even more precise and realistic analysis in multiple socio-technical disciplines. The aim here is to provide an approach to optimize processes occurring during flight operations. Some of the optimized disciplines can be:

- flight performance analysis
- flight operations safety analysis
- ground operation performance

Another benefit is the possibility to simulate multiple scenarios in e.g. air traffic management or ground control (with single pilot aircraft), which can not be done in real life, as various characteristics for pilots and air traffic controllers can be included. These simulations can contain “Humans in the loop” (compare Dausch et al. (2019)) for even more sophisticated trials.

Beyond the scope of this work multiple layers of digital twins like flight simulation software, product lifecycle management applications and other physical and cognitive models in combination with multi-agent systems are conceivable. These ideas will be discussed at the end of this thesis.

2.4 Multi-Agent Systems (MAS) with MARS

The developed concept is implemented with the MARS domain specific language (DSL). Therefore this section provides a short introduction of the functionality offered by the MARS DSL. For a full overview the MARS handbook (MARS-Group, 2019) should be consulted. A short introduction of the MARS group is given in section 1.5.

For the time being MARS DSL provides many basic features for an easy and fast implementation of MAS. The approach is an individual agent based model, contrary to a swarm based approach. The big advantage is its accessibility through easy to understand paradigms for people with various backgrounds. It generates the code in C# and makes it executable with *dotnet* installed. The output is a csv-file for each agent with each tagged variable in the frequency of the defined ticksize. The so called “ticksize” defines the magnitude of the discrete time steps the simulation will run with, this could be e.g. one second. A running simulation guide is provided in the appendix A.

The key components of each model are agents and layers. Therefore the main structure of the program will look like in listing 2.1.

```
1 model cessna_digital_twin
2
3 layer AgentLayer as agentlayer{}
4
5 agent Pilot on AgentLayer{}
6
7 agent Aircraft on AgentLayer{}
8
9 agent AirTrafficController on AgentLayer{}
```

LISTING 2.1: MARS model basic structure

The model in listing 2.1 would use GPS as the coordinate reference system with longitude and latitude values and distances defined in meters. However, the other option would be to use a grid-system with a grid-layer.

2.4.1 Layers

There are several different types of layers for agent and data management. The following overview is given by MARS-Group (2019):

1. Layer: Used in GPS-based models to manage agents
2. Grid-Layers: Used for grid-based models to manage agents and to represent rasterized data
3. GIS Raster Layers: Used to provide access to rasterized GIS data, cannot manage agents
4. GIS Vector Layers: Offer access to vector GIS data and time-series data, cannot manage agents

During evaluation in chapter 6, the potential of these types of layers are discussed for further aviation applications.

2.4.2 Agents

Agents are the key actors in the MAS. Each individual type of agent has to be defined with a set of rules and behaviors. Hence, the two key methods are explained here. The first method is the initialize method, while the other one is the tick method. An example is shown in listing 2.2.

The initialize method is executed before the tick method for initializing the agent’s variables. Then the actions for each of the agents per simulation step is defined

using the tick method. One simulation step has the dimension of the previously explained ticksize.

```
agent Pilot on AgentLayer{
  initialize{
    // initialize variables before tick method
    }
  tick{
    // do something each tick
    }
}
```

LISTING 2.2: MARS agents basic structure

A short overview of other methods is given in listing 2.3. When necessary, the methods are explained in more detail as they are introduced. It should be noted, that other basic programming features and some libraries are accessible via MARS as well.

```
passive  // tag for function to be passive and usable for other agents
explore  // explore other agents
nearest  // explore nearest agent
move // move to another location
pos at  // teleport to location
simtime  // returns the actual simulation time
external  // tag for external variables
observe // tag for variables to observe
kill  // remove agent from simulation
xcor  // returns longitude value
ycor  // returns latitude value
```

LISTING 2.3: MARS methods overview

Chapter 3

Aviation Foundation

The aviation foundation chapter offers the background knowledge for the MAS. This does not only include a short summary about the three agents aircraft, pilot and air traffic controller, but also an overview of flight operations procedure and the airport used in this thesis.

3.1 Aircraft - Cessna 150M

FIGURE 3.1: A Cessna 150M in its parking position (Stroes, 2019)

In figure 3.1 a Cessna 150M is displayed, which is used as a reference in this thesis for the MAS. Only the key facts and a short introduction of the aircraft will be given in the following. Further information is provided in the “Pilot’s Operating Handbook” (Cessna Aircraft Company, 1977) for the Cessna 150M Commuter aircraft as mentioned in section 1.4.

Section	Parameter / Description	US Unit	SI Unit
General	Maximum Speed at Sea Level	109 knots	202 kilometer per hour / 56 meters per second
	Service Ceiling	14000 feet	4267 meters
	Maximum Takeoff Weight (MTOW)	1600 pounds	725.7 kilograms
	Maximum Range (at 10,000 ft; 22. 5 Gallons Usable Fuel)	420 Nautical miles	777 kilometers
	Maximum Rate of Climb (15°C, 0 ft)	670 feet per minute	204 meters per minute / 3.4 meters per second
Wing	Wing Area	160 square feet	14.86 square meters
	Wing Root Airfoil	NACA 2412 (Lednicer, 2020)	
	Wing Tip Airfoil	NACA 0012 (Lednicer, 2020)	
	Wing Span	33 feet 4 inches	10.16 meters
Engine	Number of Engines	1	
	Engine Manufacturer	Teledyne Continental	
	Horsepower Rating	100 British Horsepower (BHP)	75 kilowatt
	Maximum Engine Speed	2750 Revolution per Minute (RPM)	
	Static RPM range at full throttle during Takeoff	2460 to 2560 RPM	
	Maximum Oil Temperature	240° Fahrenheit	116° Celsius
	Oil Pressure Minimum	10 Pound-force per square inch (psi)	690 hektoPascal (hPa)
	Maximum Fuel Consumption (Full Throttle)	7.5 gallon per hour	28.4 litres per hour
Propeller	Propeller Manufacturer	McCauley Accessory Division	
	Number of Blades	2	
	Propeller Diameter	69 inches	1.75 meters
	Propeller Type	Fixed pitch	
Fuel Tanks	Total Capacity Each Tank	13 gallons	49.2 liters
Landing Gear	Main Wheel Diameter	6 inches	0.15 meters

TABLE 3.1: Cessna 150M Characteristics, summarized from Cessna Aircraft Company (1977)

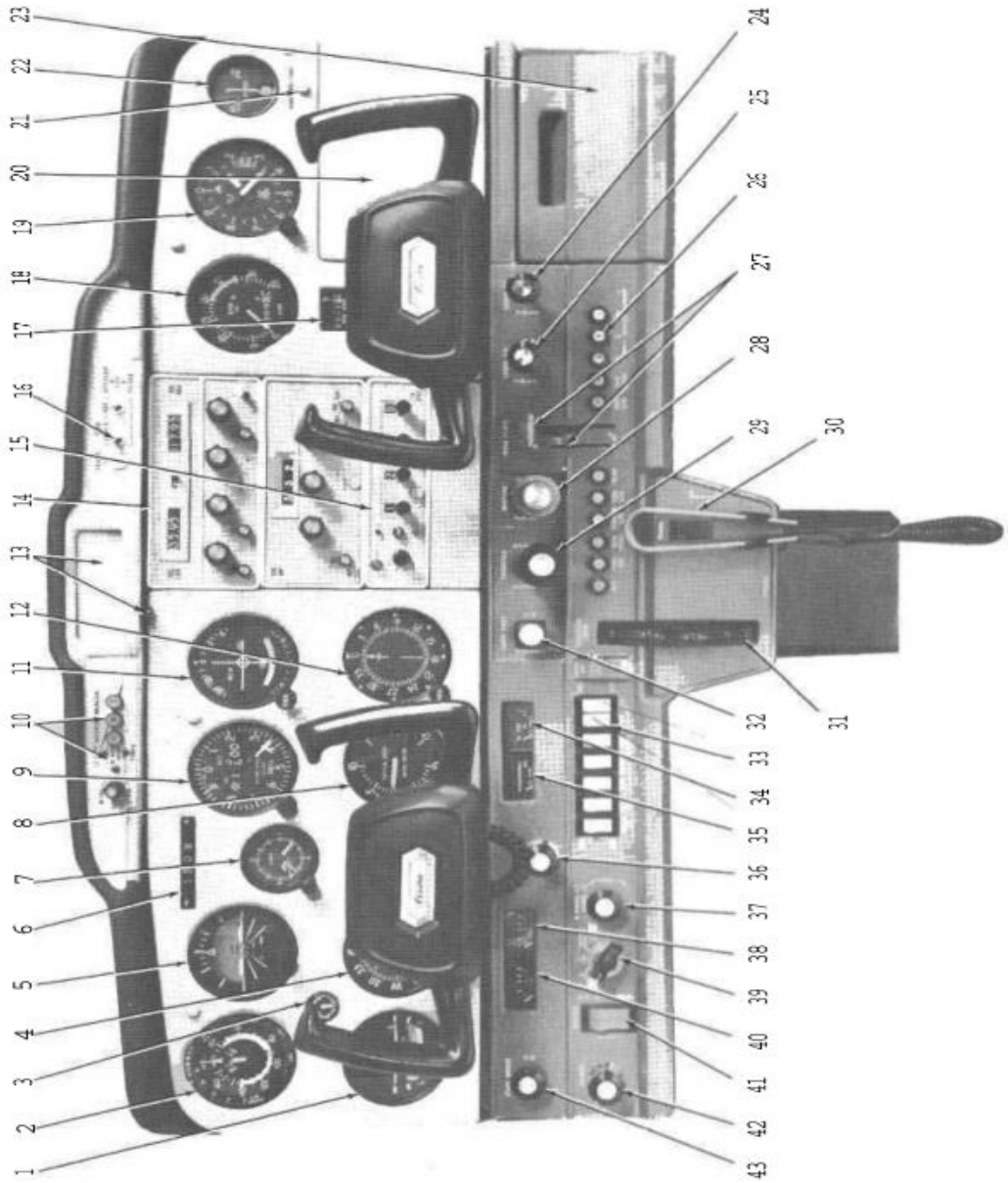

FIGURE 3.2: Cessna 150M Instrument Panel (Cessna Aircraft Company, 1977, p.7-6)

The Cessna 150M is a two-seat tricycle gear lightweight commuter aircraft. Within 27 years of production (1958 - 1985), 31.533 Cessna 150/152s have been manufactured mainly in France and the US. The suffix “M” of the Cessna 150M stands for the final version and design changes of the Cessna 150 made in 1977, where a total of 31 design changes were made within the 27 years of history (compare Cessna-150-152-Club (2019)).

The aircraft was primarily chosen due to its simplicity. Nevertheless, the model for the aircraft will be even further simplified for its initial digital representation to manage the complexity of the model, since the main focus of this thesis lies on general flight operations and a first implementation in a MAS.

Number	Instrument
2	Airspeed Indicator
4	Directional Indicator
5	Artificial Horizon (Attitude Indicator)
8	Rate-of-Climb Indicator
9	Altimeter
18	Tachometer
28	Mixture Control Knob
29	Throttle
34	Oil Pressure Gage
35	Oil Temperature Gage
38	Right Tank Fuel Quantity Indicator
39	Ignition Switch
40	Left Tank Fuel Quantity Indicator
41	Master Switch
43	Parking Brake Knob

TABLE 3.2: Cessna 150M Instrument Panel labeling for figure 3.2

A few basic facts and characteristics of the airplane are relevant in this work for generating a simplified digital twin. An overview of data compiled from Cessna Aircraft Company (1977) is presented in table 3.1 (other sources are stated explicitly). In figure 3.2, the instrument panel of a Cessna 150M is displayed. Some of the instruments that are relevant for this thesis are labeled in table 3.2. Further information like a sample problem for takeoff condition, takeoff and landing distances, stall speeds, cruise performance chart etc. is also available in the handbook (Cessna Aircraft Company, 1977).

3.2 Pilot

The pilot is the person who controls and maneuvers the aircraft and ensures a safe flight operation. For safe operation the job of a pilot begins long before the actual flight with flight planning and preflight inspection (more in section 3.4). In this case, the pilot is a private pilot with a “Private Pilot License” (PPL(A)) for aircraft with a “Maximum Take Off Weight” (MTOW) of 5700 kgs. Thus, it’s important to mention, that in this thesis only single pilot operations will be represented in the MAS for simplicity reasons, since multi-crew operations involving two pilots would make the system a lot more complex.

A pilot is a human being, therefore it offers for the agent concept later in section 5.2 multiple possibilities for descriptive characteristics, since developing cognitive agents is a research field itself. In this thesis only some very simplified assumptions are done.

3.3 Air Traffic Controller

The air traffic controller (ATC) is responsible for the flow of air traffic and gives instructions to each pilot or handles a pilot’s request. For simplification reasons the

assumption is made, that all incoming and outgoing traffic as well as all of the communication on ground is handled by one sole tower ATC.

The most important tasks worth mentioning for this thesis are:

- give taxi instructions
- takeoff and landing clearance
- takeoff preparation clearance
- hand over to next frequency / control center
- decide takeoff heading

Just like the pilot the ATC is a human being as well. This once again allows many different modeling approaches. For complexity reasons, only very simplified assumptions are made here likewise to the pilot.

3.4 Flight Operations

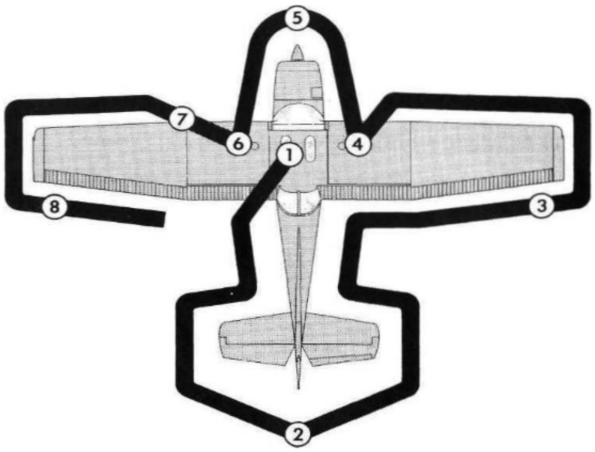

FIGURE 3.3: Preflight Check Procedure (Cessna Aircraft Company, 1977, p.4-4)

This section provides an overview of the flight operations and procedures. For every pilot each flight starts long before the actual flight with the flight planning. This includes e.g. the route and weather planning, airworthiness documentation and weight & balance check (compare Thom (1987, p.11-16)). The procedures after the flight planning are presented in figure 3.4. Every action is accessible in a so called “Checklist Procedure” and is necessary to ensure safe operations. The handbook of the Cessna 150M (Cessna Aircraft Company, 1977, p.4-1ff.) includes these checklists. For instance, the head of the preflight inspection checklist looks like this:

(1) **CABIN**

(1) Control Wheel Lock – REMOVE.

(2) Ignition Switch – OFF.

(3) Master Switch – ON.

(4) Fuel Quantity Indicators – CHECK QUANTITY.

(5) Master Switch – OFF.

(6) Fuel Shutoff Valve – ON.

(2) **EMPENNAGE**

(1) Rudder Gust Lock - REMOVE.

(2) ...

In the following a short introduction to each procedure is presented.

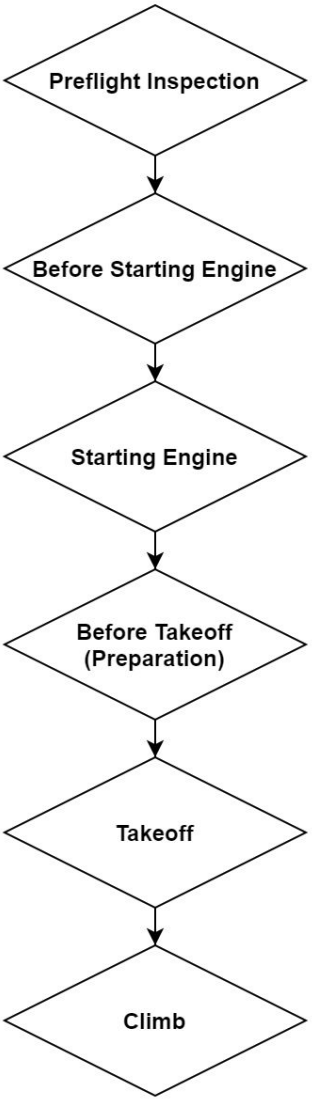

FIGURE 3.4: Cessna 150M Checklist Procedures (Cessna Aircraft Company, 1977)

Preflight Inspection

The preflight inspection is a visual check for an airplane's general condition before every flight. A typical walk-around inspection is performed similar to the scheme in figure 3.3. Very important actions are e.g. checking the engine oil, fuel quantity and possible water sediments in the fuel tanks. Furthermore, the preflight inspection should ensure the equal condition documented during the flight planning before.

Before Starting Engine & Starting Engine

The checklists previous to and for starting the engine make sure, that all the required settings in the instrument panel are set prior to pushing the ignition switch.

Before Takeoff (Preparation)

Before attempting takeoff the takeoff preparation takes place on the taxiway during taxiing. The ATC needs to give permission for that action, since during takeoff preparation, moderate thrust will be applied to check the aircraft's condition. Further checks include examining the free and correct movement of flight controls, engine and the condition of light.

Takeoff

After the takeoff preparation has taken place, the pilot has to to request further taxiing instructions or for takeoff clearance. The ATC can give one of the following instructions.

- "Hold short of runway xx" : Wait on the taxiway for runway and takeoff clearance
- "Line up and wait runway xx" : Line up on the runway, but yet no takeoff clearance
- "Cleared for takeoff on runway xx" (plus wind information) : Cleared for takeoff

When cleared for takeoff, the pilot lines up on the runway, applies full thrust and lifts the nose wheel by applying pitch at approximately 50 knots / 90 kilometers per hour true air speed for the Cessna 150M.

Enroute Climb

During climb the pilot shall continue to apply full thrust. After reaching a specific height, the ATC instructs the pilot to change frequency to contact the next control center.

For more details on flight operations, the handbook (Cessna Aircraft Company, 1977) or the flying manual Thom (1987) should be consulted.

3.5 Airport

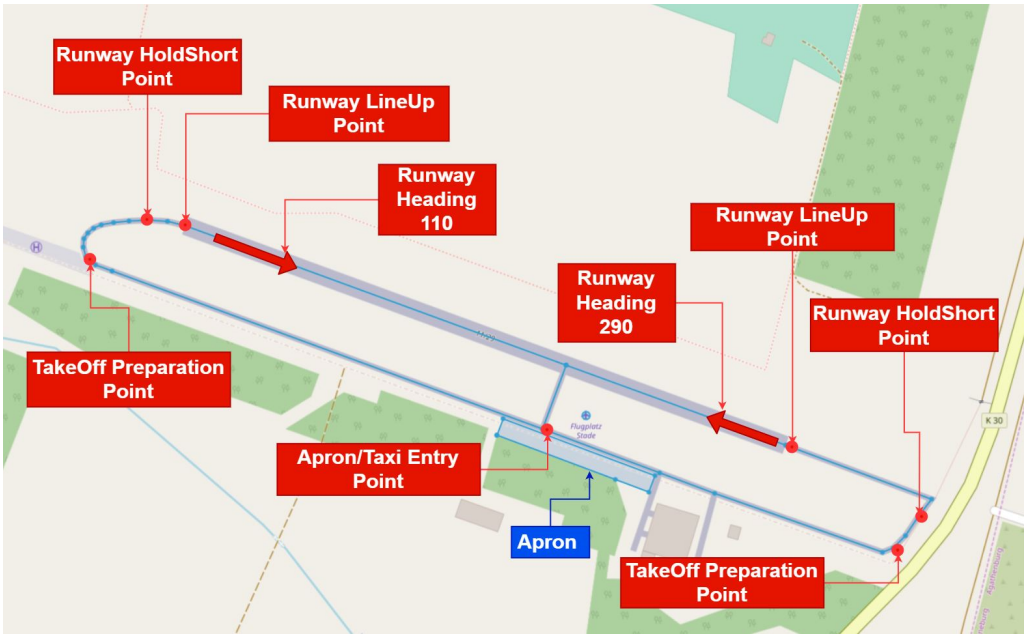

FIGURE 3.5: Airport Stade (data from https://www.openstreetmap.de/ and edited view from https://www.geojson.io/)

Parameter	Value / Description
Airport Elevation	0 meter/feet (adjusted from 19 meters above MSL)
Runway Length	650 meters / 2130 feet
Runway Heading	[110, 290], adjusted value for visualization (official are [107, 287])
Runway Surface	Concrete
Runway Hold Short Point	Point on the taxiway, where pilot wait for takeoff clearance and line up permission
Runway Lineup Point	Point on the runway, where pilot start to take off after takeoff clearance
Takeoff Preparation Point	Point on the taxiway, where pilots executes the takeoff preparation procedure
Apron	Starting and end location for pilots and aircraft

TABLE 3.3: Airport Stade Key Characteristics and Explanations

As it is outlined in section 2.2, the environment plays an important role in a MAS. Thus a short introduction for the airport follows.

The airport used in this thesis is the “Airport Stade” near Hamburg (http://www.flugplatz-stade.de/). It provides all necessary infrastructure for normal flight operations, this includes an apron, taxiways and a runway. In figure 3.5 the airport structure is visualized. Important points and segments are already systematically labeled on that map in figure 3.5 for the concept development chapter 5 later. The explanation and the most important key facts are compiled in table 3.3.

Chapter 4

Physical Background

The modeling of the aircraft agent with an interdependent environment in chapter 5 requires physical background in these disciplines. Hence, this chapter provides all mathematical equations and explanations used in the MAS model.

4.1 Aircraft Physics

All of the aircraft physics used in this thesis is described in the following sections. Further literature like Scheiderer (2008) and Young (2001) should be consulted for a in depth illustration and explanation of aircraft physics and its equations.

Only a simplified two-dimensional physical model is developed and used in chapter 5. In figures 4.1 and 4.2 the forces acting on the aircraft during climb phase and on-ground phase are visualized.

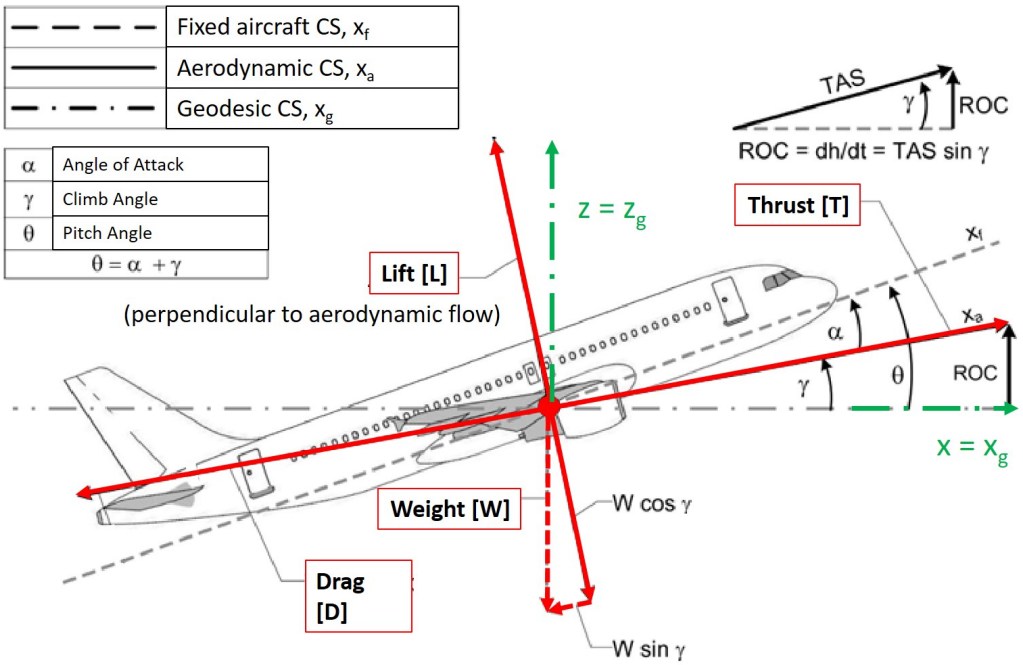

FIGURE 4.1: Force diagram during climb phase (adapted from Scheiderer (2008, p.227))

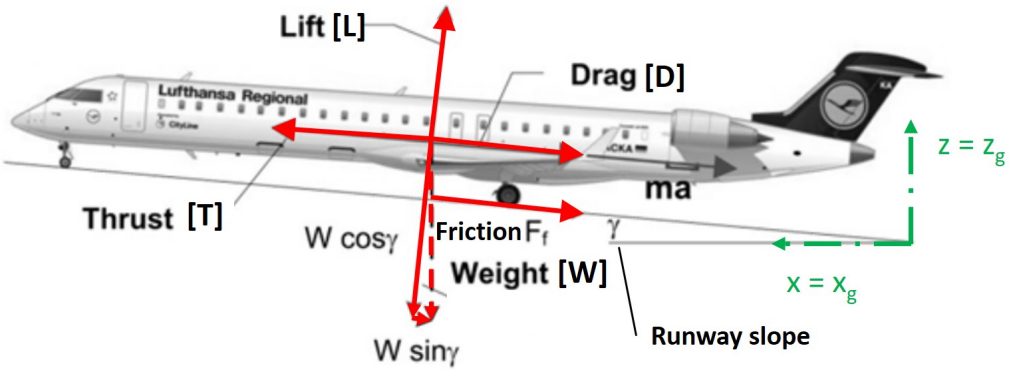

FIGURE 4.2: Force diagram during on-ground phase (adapted from Scheiderer (2008, p.146))

The equations 4.1 to 4.5 are representing the equations of motion for figure 4.1 and 4.2 and are referenced to the geodesic coordinate system x_g and z_g, which is also used in the model later on. Thus, the indices x and z used always refer to the geodesic coordinate system. In the two dimensional x and z coordinate system z is pointing upwards and the angle of attack α, the climb angle (or runway slope, but will be neglected) γ and the pitch angle θ are defined positive as in figure 4.1.

Regarding the equations of motion for climb phase it is important to note the following general assumptions and information:

- Weight is always perpendicular to the earth's surface and thus parallel to the z-direction
- Curvelinear motion around the earth is neglected
- Thrust vector is simplified and parallel with drag
- Lift L is perpendicular and the drag D parallel to the direction of air flow and thus to the climb angle γ
- Pitch angle θ and climb angle γ always refer to the geodesic coordinate system
- Angle of attack α is derived from equation 4.3
- Rate of climb $ROC = V_z$

Using these assumptions and figure 4.1 the equations of motion for climb phase are (compare Scheiderer (2008, p.228) and Young (2001, p.5-5)):

$$m\frac{dV_x}{dt} = T\cos(\gamma) - L\sin(\gamma) - D\cos(\gamma) \tag{4.1}$$

$$m\frac{dROC}{dt} = L\cos(\gamma) - W - D\sin(\gamma) + T\sin(\gamma) \tag{4.2}$$

And the following relation between the angles applies:

$$\theta = \alpha + \gamma \tag{4.3}$$

Where the climb angle γ is derived from the speed components (top right in figure 4.1), since it specifies the direction of the resultant true air speed.

$$\gamma = \sin^{-1}\left(\frac{ROC}{TAS}\right) \tag{4.4}$$

The equations from 4.1 to 4.4 are also valid for descend flight phase.

For the equation of motion during on-ground phase the runway slope γ is neglected and thus using the relations shown in figure 4.2 it can be derived as:

$$m\frac{dV_x}{dt} = T - D - F_f(L) \tag{4.5}$$

Where the friction force F_f is lift dependent and explained in section 4.1.5 later.

Furthermore, the true airspeed (TAS) simply derives from

$$TAS = \sqrt{V_x^2 + ROC^2} \tag{4.6}$$

and shows the dependency between 4.1 and 4.2. The weight W is constant for the aircraft mass m and the gravitational constant g (mass loss due to fuel consumption is neglected):

$$W = mg \tag{4.7}$$

The equations of motions 4.1 (or 4.5) and 4.2 are solved with a discrete approximation and the first simple choice is the forward euler method as described in equation 4.8 (compare Grüne (2008, p.12)).

$$y_{i+1} = y_i + h_i\, f(t_i,\, y_i) \quad \text{with} \quad h_i = t_{i+1} - t_i \tag{4.8}$$

The other components of the equations 4.1 to 4.5 are explained in more detail in the next sections.

4.1.1 Aircraft Speed

In aviation a distinction between various aircraft speeds is essential. The equations presented in this chapter use the true airspeed (TAS), the speed at which the aircraft is moving relatively to the air flow. That e.g. means with a strong headwind, the TAS can be positive while the aircraft is not moving relative to the ground (see section 4.2.3).

Figure 4.3 shows an overview of speed variations and how they relate to each other.

- Indicated Airspeed (IAS): That is the air speed indicator reading, which is not yet calibrated
- Calibrated Airspeed (CAS): Calibrated IAS for instrument, total pressure and/ or position error
- Equivilant Airspeed (EAS): Air compressibility corrected equivalent speed at sea level for the same dynamic pressure as in TAS

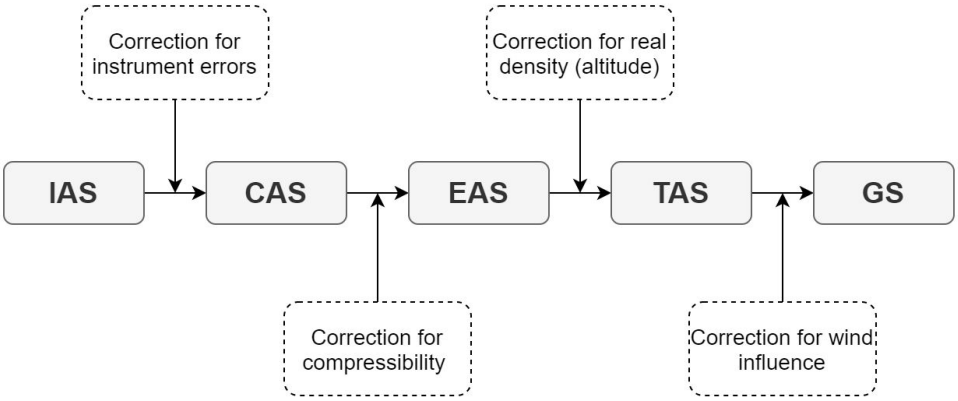

FIGURE 4.3: Aircraft airspeed variations and relations

- True Airspeed (TAS): Relative speed to the ambient air
- Ground Speed (GS): Actual speed relative to the ground

The transformation equations between each speed are not documented, as the aircraft speed system will be simplified and the wind influence is documented in section 4.2.3.

4.1.2 Lift

The lift is produced by the wings and can be calculated using the dynamic pressure $(\rho/2)V^2$, the wing area S and the dimensionless lift coefficient C_L as in equation 4.9. The speed V is the true air speed (TAS).

$$L = C_L \frac{\rho V^2}{2} S \tag{4.9}$$

The lift coefficient C_L depends on the wing airfoil, reynolds number and angle of attack α. In figure 4.4 the lift coefficient of the airfoil NACA 2412 for low reynolds numbers (as they would appear during the flight of the Cessna 150M) versus the angle of attack α is displayed. As documented in table 3.1, the airfoil NACA2412 is used at the root of the wing of the Cessna 150M.

The lift coefficient C_L at a certain angle of attack α can be described using the zero lift angle α_0 and the lift curve slope $C_{L\alpha}$. This is presented in equation 4.10.

$$C_L(\alpha) = C_{L\alpha}(\alpha - \alpha_0) \tag{4.10}$$

4.1.3 Drag

The produced drag can be separated into the zero lift drag and the lift dependent drag respectively the induced drag. As a result the drag D is derived using equations 4.11 to 4.14 with the total drag coefficient being C_D:

$$D = C_D \frac{\rho V^2}{2} S \tag{4.11}$$

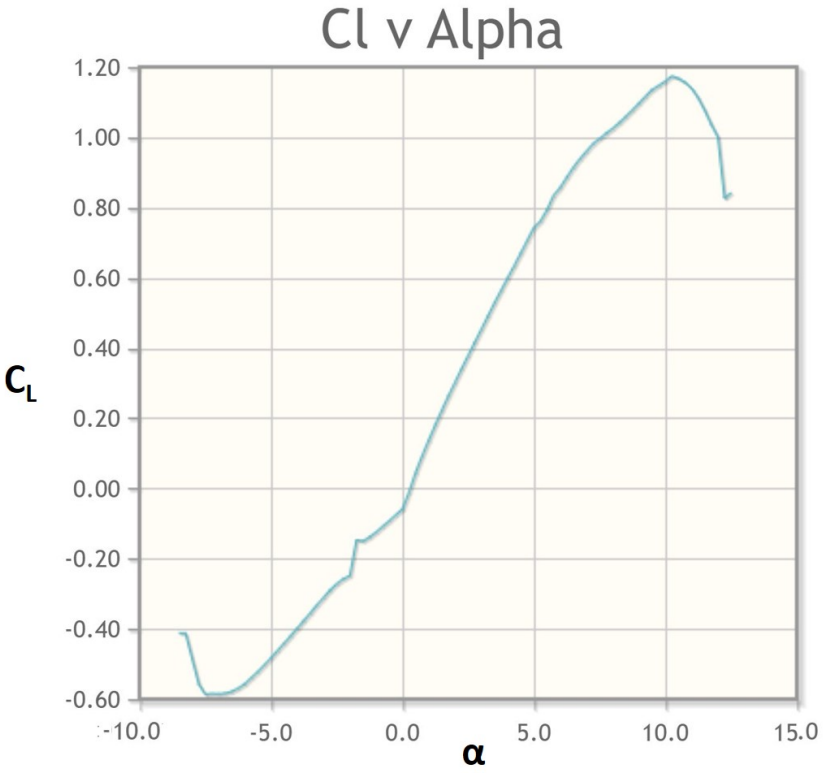

FIGURE 4.4: NACA 2412 - lift coefficient C_L versus AOA α for $Re = 50000$ (AirfoilTools, 2020)

The total drag coefficient C_D is derived from the zero lift drag coefficient C_{D0} and the induced drag coefficient C_{Di}:

$$C_D = C_{D0} + C_{Di} \tag{4.12}$$

Calculating the induced drag coefficient C_{Di} requires the oswald factor e, the aspect ratio Λ and the lift coefficient C_L.

$$C_{Di} = \frac{C_L^2}{\pi \Lambda e} \tag{4.13}$$

Whereby the aspect ratio Λ is composed using the wing span b and wing area S.

$$\Lambda = \frac{b^2}{S} \tag{4.14}$$

4.1.4 Thrust

The thrust equations are used as described in the NASA propeller study (Worobel and Mayo, 1971) or the summary from Schulte (2007). For the thrust calculation the advance ratio J is defined with n being revolution per time unit and d the propeller diameter:

$$J = \frac{V}{n\,d} \tag{4.15}$$

Furthermore the dimensionless thrust coefficient C_T and the power coefficient C_P are determined by:

$$C_T = \frac{T}{\rho n^2 d^4} \tag{4.16}$$

$$C_P = \frac{P}{\rho n^3 d^5} \tag{4.17}$$

The equations 4.16 and 4.17 can be rearranged in such a way, that the thrust coefficient C_T and the power coefficient C_P depend on the advance ratio J. That dependency is described in figure 4.5, where Schulte (2007) provides a diagram for a exemplary constellation.

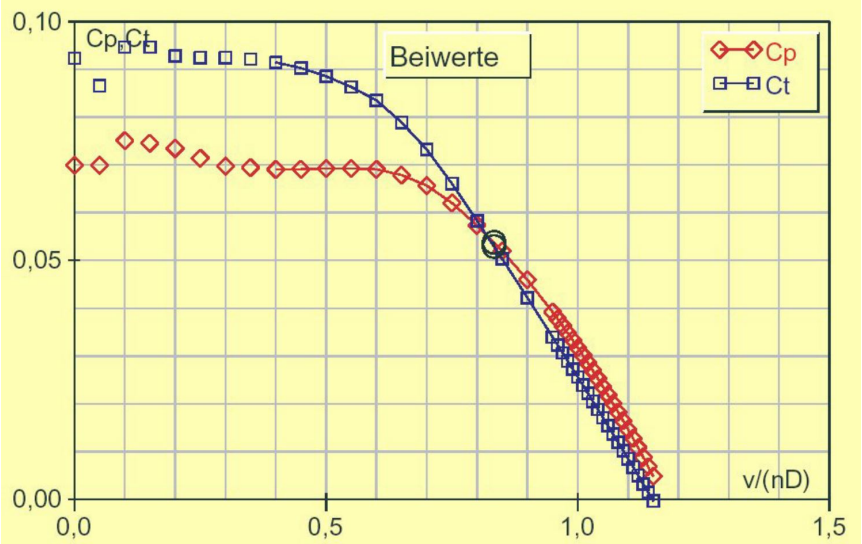

FIGURE 4.5: Example diagram of thrust coefficient C_T and the power coefficient C_P for $P = 115$ kW, $d = 1.9$ m and two propeller blades (Schulte, 2007)

For the sake of completeness the propulsive efficiency η is defined as:

$$\eta = J\frac{C_T}{C_P} \tag{4.18}$$

4.1.5 Ground Friction

The ground friction is described using the equations for “rolling resistance”. The friction force F_f is calculated utilizing distance coefficient c (unit is meter), normal force F_N and radius r as in figure 4.6.

$$F_f = F_N\frac{c}{r} \tag{4.19}$$

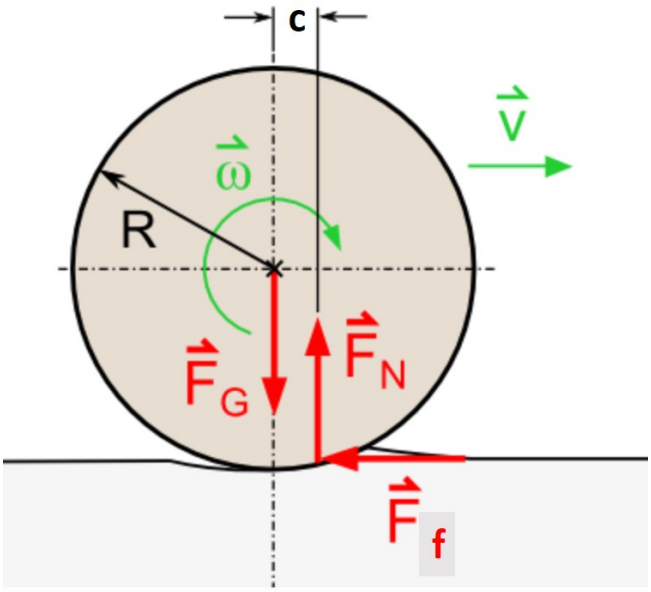

FIGURE 4.6: Forces and friction on rolling tire (edited from Maschinenbau-Community (2009)

The normal force F_N is for e.g. cars equal to its weight W. Since the aircraft is generating lift for a positive true air speed, this needs to be taken into account. Thus the friction force F_f is determined as followed.

$$F_f = (W - L)\frac{c}{r} \quad \text{for} \quad F_f \geq 0 \tag{4.20}$$

In general it should be noted that rolling friction forces are considerably small.

4.2 Environment

4.2.1 Geographic Coordinate System

In MARS the agents are positioned via the geographic coordinate system. The latitudinal and longitudinal values are given in decimal degrees, where the latitude runs east-west and ranges between $-90° < lat < +90°$ and the longitude runs north-south and ranges between $-180° < lon < +180°$. The bearing ranges between $0° < x < 360°$, where 0° is north. The equations used are e.g. accessible at Movable Type Scripts (2020).

For calculating the shortest distance on a sphere (also called “orthodrome” or “great circle distance”) between two points $p_1(lon_1; lat_1)$ and $p_2(lon_2; lat_2)$, the haversine formula is used as in equation 4.23. First the delta values are defined as.

$$\Delta lat = lat2 - lat1 \tag{4.21}$$

$$\Delta lon = lon2 - lon1 \tag{4.22}$$

The haversine formula 4.23 uses two temporary variables and the $arctan2$, a mathematical extension of the arctan, which takes two input variables and delivers the result in a range between $-180° \leq x \leq +180°$ and thus of all four quadrants. The final result is calculated by using the earth radius r_{earth}.

$$\begin{aligned} & d_{hav} = r_{earth}\, a_2 \\ \text{with} \quad & a_2 = 2\, atan2\big(\sqrt{a_1},\ \sqrt{(1-a_1)}\big) \\ \text{and} \quad & a_1 = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2\left(\frac{\Delta lon}{2}\right) \end{aligned} \tag{4.23}$$

For calculating the bearing Φ from one point p_1 to the other point p_2 we determine the so called “initial bearing”. This is precise for short distances, but when following long distances on the great circle path (shortest path between two points) the bearing would constantly vary. The initial bearing formula is defined as:

$$\begin{aligned} & \Phi = atan2(a_1, a_2) \\ \text{with} \quad & a_1 = \sin(\Delta lon) \cdot \cos(lat_2) \\ \text{and} \quad & a_2 = \cos(lat_1) \cdot \sin(lat_2) - \sin(lat_1) \cdot \cos(lat_2) \cdot \cos(\Delta lon) \end{aligned} \tag{4.24}$$

As the result of $arctan2$ and thus bearing Φ value ranges between $-180° \leq x \leq +180°$, these values have to be converted into the range of $0° \leq x \leq 360°$. This is simply done by adding 360° and taking the modulo with 360°.

For some calculations it’s necessary to find the smallest difference between two headings φ_1 and φ_2. The equation 4.25 derives the smallest angle Δ between two headings in the positive value range $0° \leq x \leq 180°$.

$$\Delta = 180 - |(|\varphi_1 - \varphi_2| - 180)| \tag{4.25}$$

4.2.2 ISA Atmosphere

Most of the aircraft physics equations in section 4.1 require the ambient density. In this thesis the “International Standard Atmosphere” (ISA) with non ISA variations is used. For example the term $ISA+20$ means, that a ISA variation with a temperature offset of plus 20°C is valid.

In the following only the basics within the troposphere of the physical ISA model is described (for dry perfect gas). For further reference or ISA tables Scheiderer (2008, p.87), Young (2001, p.1-6) or Sadraey (2017, p.7) are recommended.

The ISA condition at sea level (0 m) is defined as:

- Pressure p_0 = 101325 Pa
- Temperature T_0 = 15°C = 288.15 K
- Density ρ_0 = 1.225 kg/m^3

Whereby the density can also be calculated with the “ideal gas law” and the gas constant R= 287.3 J/(kg K):

$$p = \rho R T_K \tag{4.26}$$

The temperature decreases with height h and is modeled with the lapse rate L_c = 0.0065 K/m by the following equation:

$$T_{ISA} = T_0 - L_c h \tag{4.27}$$

For non ISA conditions it's simply calculated with:

$$T_{NON-ISA} = T_{ISA} + T_{\Delta ISA} \tag{4.28}$$

Figure 4.7 visualizes the temperature curve within the troposphere for better understanding.

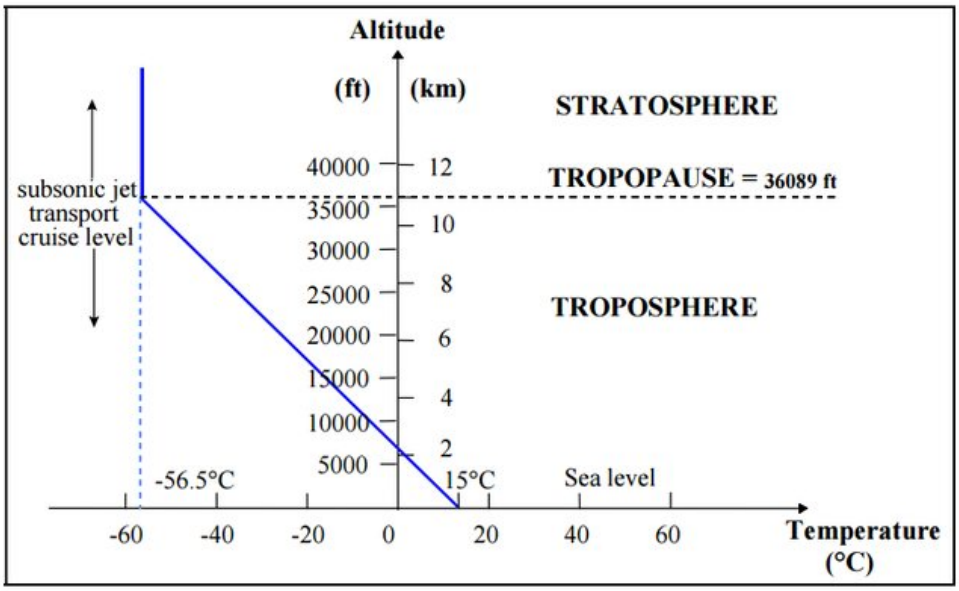

FIGURE 4.7: ISA temperature curve in the troposphere (Prigent, 2015)

Without derivation the following equation and relation applies for the pressure:

$$\frac{p}{p_0} = \left(\frac{T}{T_0}\right)^{\frac{g}{RL_c}} = \left(1 - \frac{L_c h}{T_0}\right)^{\frac{g}{RL_c}} \tag{4.29}$$

For calculations with non ISA conditions in the model later the user is able to enter the QNH p_{QNH} (reference pressure at sea level derived from real pressure at any airport or weather station elevation using ISA) and the temperature $T_{0,NON-ISA}$ at sea level. The necessary steps to determine the density at any altitude of the aircraft using these inputs are the following.

Calculation steps to determine the density for a non ISA variation

Given: p_{QNH}, $T_{0,NON-ISA}$, h

Unknown: density at aircraft height ρ_h

1. Calculate pressure height (delta)

From rearranging equation 4.29 the pressure height h_p equation reads:

$$h_p = (\frac{T_0}{L_c}) \cdot (1 - \left(\frac{p_{QNH}}{p_0}\right)^{\frac{g}{RL_c}} \tag{4.30}$$

2. Determine the ISA variation

Using the pressure offset the overall temperature offset from ISA can be calculated. First the temperature T_{h_p} for the pressure height h_p from step one with the equation 4.27 is derived. Then the temperature delta $T_{\Delta ISA}$ and thus the ISA variation equals:

$$T_{\Delta ISA} = T_{0,NON-ISA} - T_{h_p} \tag{4.31}$$

3. Calculating with ISA variation

With the determined ISA variation it's possible to calculate the density with the ideal-gas equation 4.26 on the basis of the temperature equations 4.27 and 4.28 and the pressure equation 4.29 for any given height h.

4.2.3 Wind

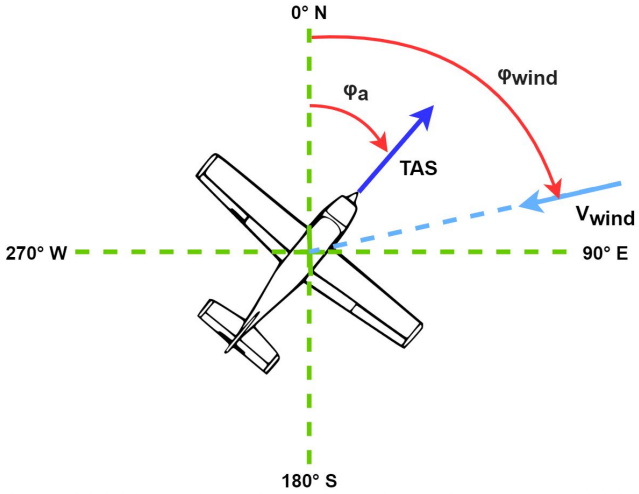

FIGURE 4.8: Aircraft speed calculation with wind

The wind bearing is indicated and defined as the direction the wind is coming from. Figure 4.8 shows an example with an aircraft, where the wind speed results in a cross- and headwind component.

The ground speed (GS) is then simply calculated with the wind speed V_{wind} and by the cosine of the bearing difference between the aircraft bearing φ_a and the wind bearing φ_{wind}. As the crosswind component has no effect in the upcoming model, the calculation of that is left out.

$$GS = TAS - cos(\varphi_a - \varphi_{wind}) \cdot V_{wind} \tag{4.32}$$

Chapter 5

Developed Concept and Implementation

The developed concept and its implementation include the overall architecture, the agents, some general features, the protocol and the environment. It's very important to note that the structure in most of the sections is similar:

1. Showing the full concept and idea with short explanations as a result from the design process
2. Present the part of the (adapted) concept that is implemented within this thesis
3. Comments and presentation on how that simplified, technical model is realized with MARS and explaining decisions, possibilities and obstacles

Step 1 includes the results of a creative design process of concept drawings and of numerous discussions with MARS members. The full concepts are used in chapter 6 and 7 as a reference for evaluation, future work and outlook. This chapter focuses mainly on the implemented part.

The section 5.1 gives an overview of all program components, while the sections 5.2 to 5.6 provide information about each of these components. However, the implemented program's main actions take place along the internal states of the pilot agent. The functionality of each state is explained in section 5.7.

It should be mentioned, that appendix A offers a guide for running the simulation, contains a link to the full program and gives an overview of all possible external variables input, while appendix B offers a full overview of implemented and not implemented features.

5.1 Overall Architecture

Each implemented component of the presented overall architecture in this section is explained in the upcoming sections.

Figure 5.1 presents the full concept idea of the overall architecture. It's built with three layers and for demonstration purposes two pilots and aircraft are displayed.

- Agent Layer: The agents are located here. Within the agent layer a communication layer respectively communication blackboard is defined. This manages the communication between the agents.

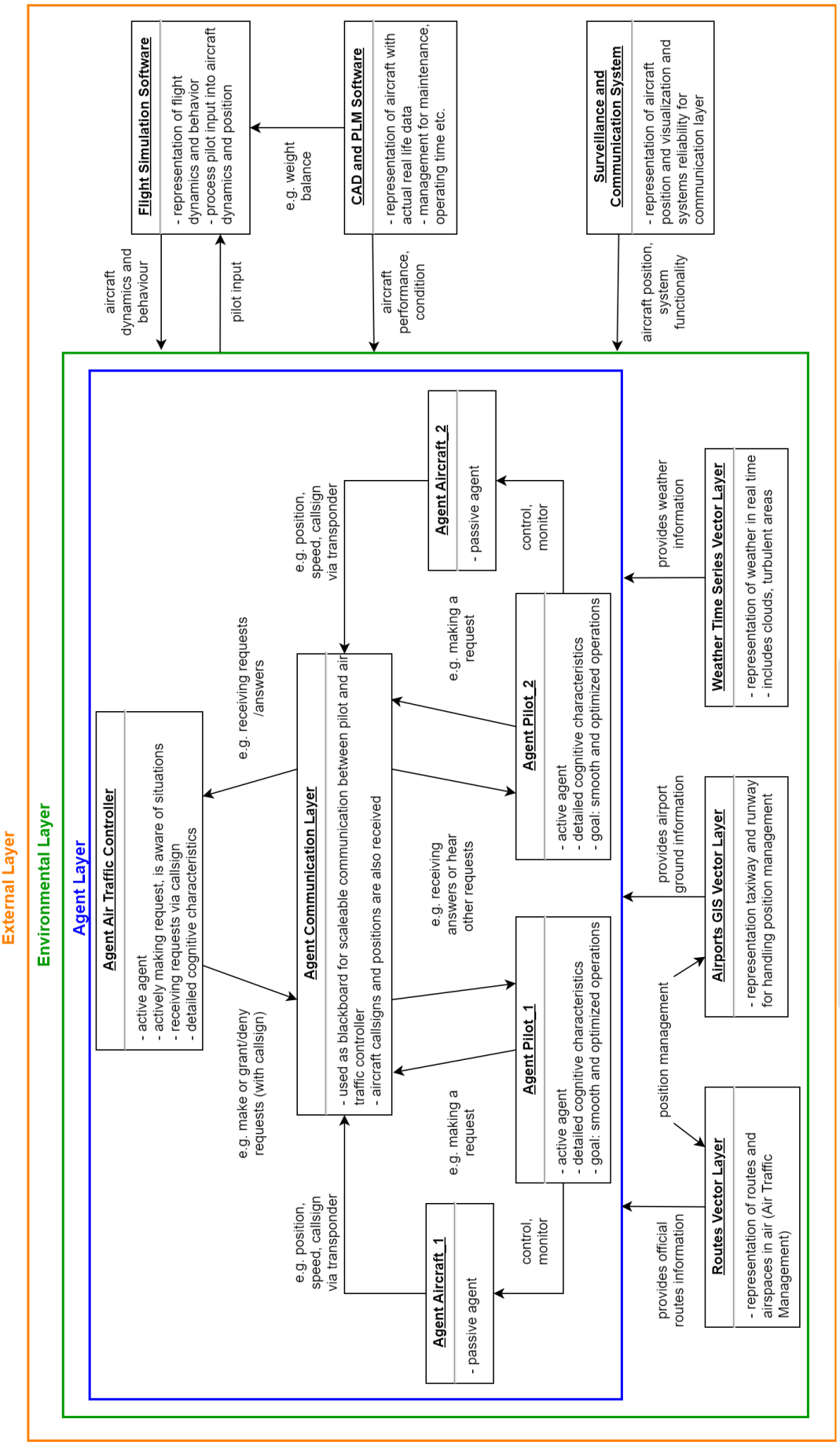

FIGURE 5.1: Overall architecture - full developed concept

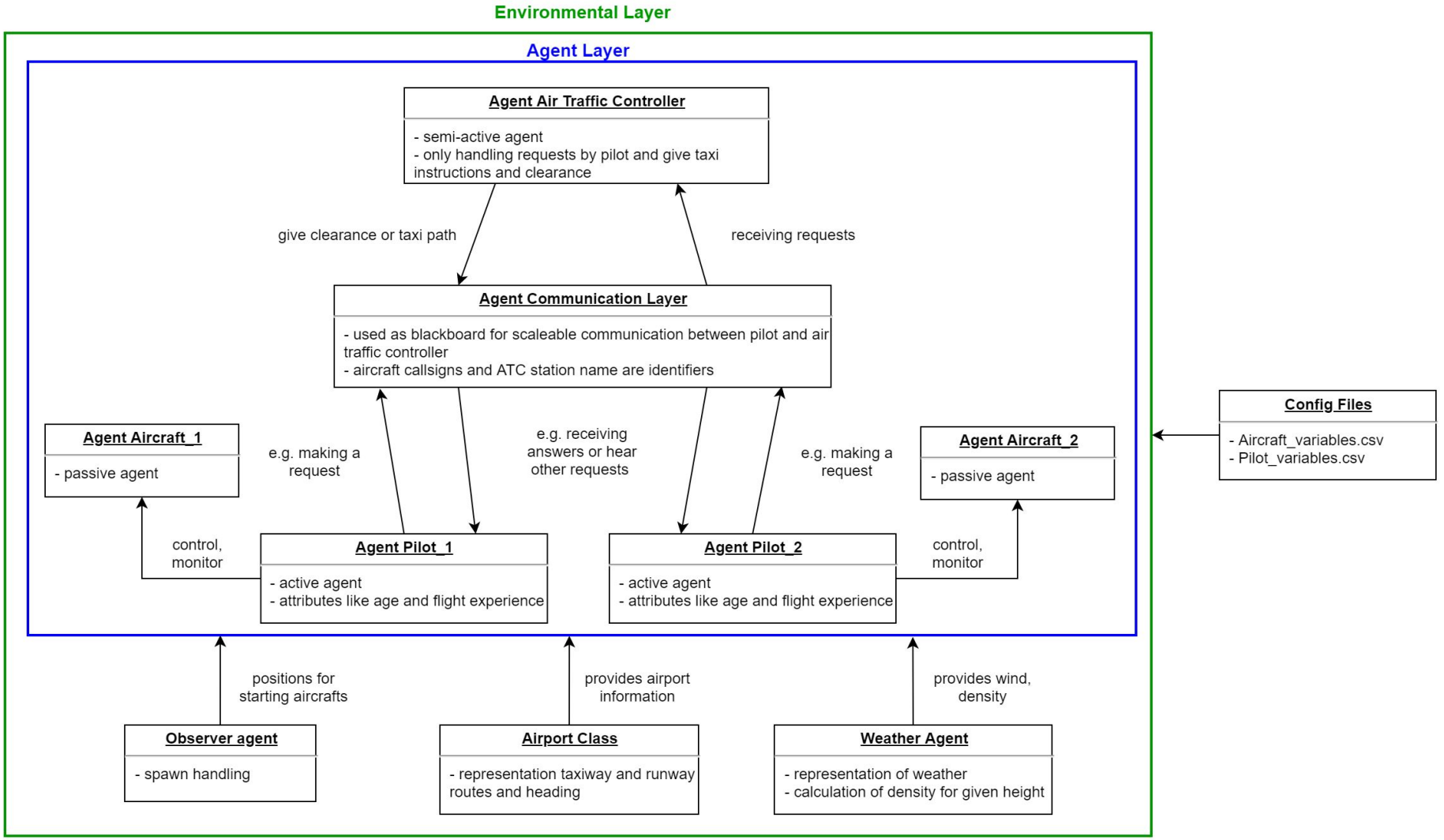

FIGURE 5.2: Overall architecture - reduced implemented concept

- Environmental Layer: The agents have access to airport data, airways and weather data through GIS vector layers
- External Layer: Access to aircraft information or simulation data of third party programs

Due to the complexity of this concept, the reduced concept in figure 5.2 has been implemented in MARS. The key differences and simplifications are:

- Agent Layer: Simplified agent definitions, as they are presented in the following. Aircraft is not contributing to the communication layer
- Environment Layer : Instead of sophisticated layers a weather agent and airport class are used as explained later. Necessity of an observer agent for spawn handling
- External: Possibility to configure most of the pilot's and aircraft's parameters and variables with config files (see appendix A)

The full concept and these simplifications and implementations are documented in the upcoming sections. However, the following general information applies:

- Naming scheme is a combination of snake case and camel case through all agents in the program, since MARS did not allow subagents or subclasses within each agent
- GPS based positioning system is used (compare 4.2.1) and a geodesic reference coordinate system for calculations
- The tick size and thus the simulation step interval is one second, which offers a simplified calculation with SI Units and a reasonable runtime

5.2 Pilot Agent

The pilot agent is by definition an active agent that actively has to control and monitor the aircraft. In figure 5.3 a full internal model for human agents is presented by Bouarfa (2016). This model is used for runway crossing operations safety assessment and thus fully established by Bouarfa (2016) for the study. Since the scope of this thesis ranges from preflight check to takeoff, the pilot internal states and actions only follow some basic rules and protocols described in section 5.7 to achieve his goal to take off.

The possible set of actions to control the aircraft is hereby logically bound to the passive actions of the aircraft in section 5.3. Each of these actions is represented within the pilot agent again to offer the possibility to influence the input or output due to pilot's characteristics, although this is not elaborated further within this thesis. This is implemented as stated in listing 5.1.

```
1 def Apply_Engine__throttle(input : real)
2 {
3   // insert pilot uncertainties here
4   myAircraft.CIP_Apply_Engine__throttle(input)
5 }
```

LISTING 5.1: Aircraft passive actions representation in pilot agent

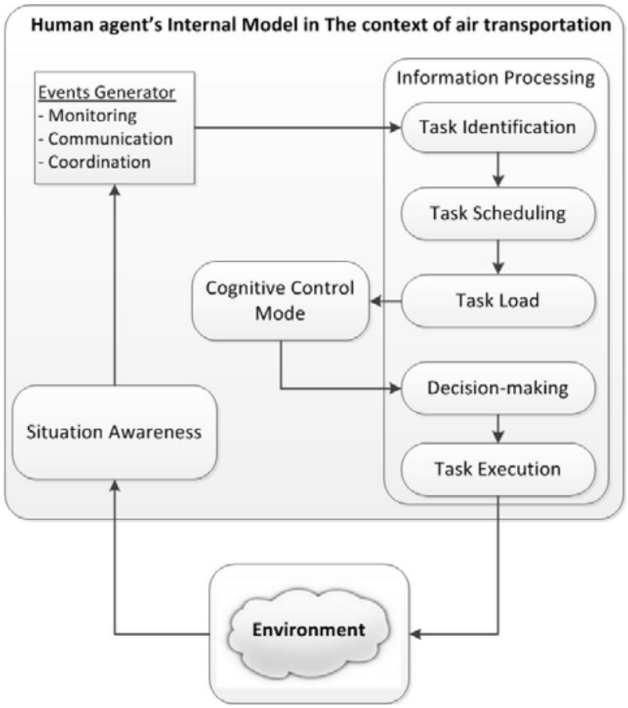

FIGURE 5.3: Agent architecture proposal for human agent (Bouarfa, 2016, p.84)

The only characteristics introduced so far for statistical distributions are the pilot's age and the flight experience, which are randomly generated during initialization (see listing 5.2). Their impact and the initialization process is discussed in section 5.7.

As mentioned earlier, the main actions take place along the internal states of the pilot agent. Thus, the initialize and tick method are implemented as shown in listing 5.2 for the pilot. Within each tick the active state defines the actions executed. In the function "update_general_values" the position of the pilot is aligned with the aircraft's position.

```
initialize
{
  // (...) some unimportant parts left out

  age = random(..)
  flight_experience = random(..)
}

tick
{
  if (state === "Initialization"){
    Second_Initialization()
  }

  else if (state === "PreflightInspection"){
    PreflightInspection_action()
  }

  else if (state === "StartingEngine"){
    StartingEngine_action()
```

```
21    }
22
23
24    // (...) some parts left out for space
25
26
27    else if (state === "TakeOff"){
28      TakeOff_action()
29    }
30
31    else if (state === "LeavingFrequency"){
32      LeavingFrequency_action()
33    }
34
35    update_general_values()
36  }
```

LISTING 5.2: Pilot agent initialize and tick method

Further it's worth mentioning that most of the existing parameters and variables of the pilot agent are accessible via external input and fully listed in the appendix A. All variables impact and meaning are either self explanatory or are explicitly explained within the section 5.7.

5.3 Aircraft Agent

5.3.1 Overview and Architecture

As described in figure 5.2 the aircraft alias Cessna 150M is a fully passive agent. Thus, the goal is to create an aircraft agent, that is accessible and controllable for the pilot agent. Figure 5.4 shows the implemented and developed aircraft agent architecture. A full concept for the aircraft digital twin with e.g. further systems is not presented, since the aircraft's system architecture can be expanded infinitely. The shown concept acts as a base layer for further enhancement.

The internal subcomponents of the aircraft agent are presented within the blue box in figure 5.4. The most important subcomponent is the the aircraft's physics calculation, which relies on the input of other subcomponents. The aircraft physics calculation is summarized in section 5.3.3. It requires further variables, modeling etc. for the Cessna 150M with the theoretical background of chapter 3 and 4. These calculations are used for the motion of the aircraft, which should depend on the input of the pilot.

```
1  // airspeed_indicator
2  passive IP_Get_Aircraft__true_air_speed()
3  {
4    return Aircraft__true_air_speed
5  }
6
7  // control wheel
8  passive CIP_Apply_Aircraft__pitch(input : real)
9  {
10   Aircraft__pitch = input
11 }
```

LISTING 5.3: Aircraft passive actions

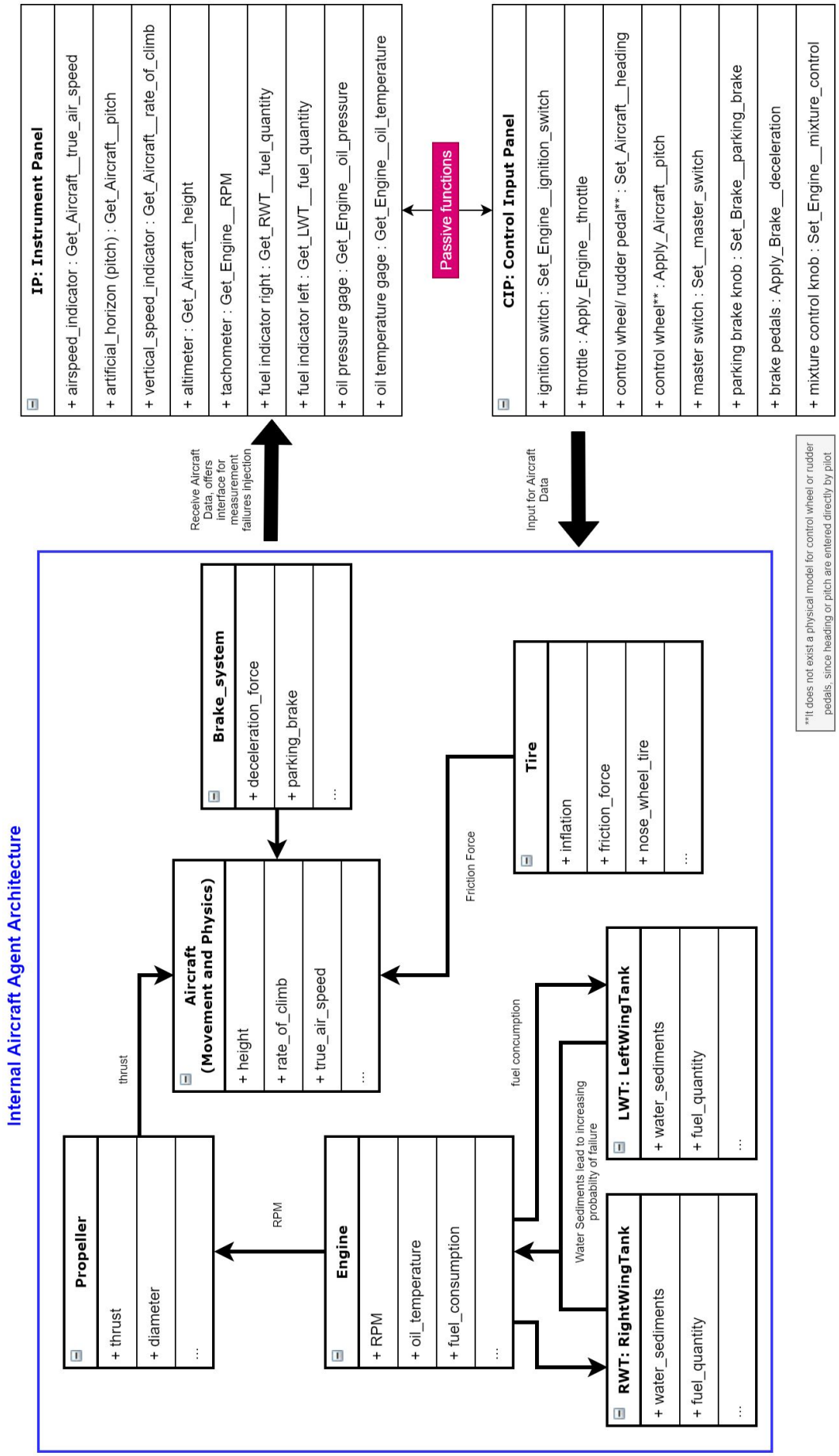

FIGURE 5.4: Implemented and developed aircraft agent architecture

Outside of the blue box in figure 5.4 all passive functions of the aircraft are divided into two groups. Implemented functions in MARS are exemplary shown in listing 5.3.

Instrument Panel (IP): Functions for the pilot to access data of the aircraft systems via the instrument panel. With this structure it's possible to implement instrument malfunctions before transferring data.

Control Input Panel (CIP): Functions for the pilot to enter data via the instrument panel.

The Cessna 150M instruments of both panels are described in figure 3.2 and table 3.2. As both panels are presented, some assumptions can already be derived:

- For speed measurement the simplification applies TAS = CAS (or IAS), which is not true for varying density, which changes with temperature and height (compare 4.1.1)
- The measured height equals the altitude above MSL without a barometer reading with reference pressure
- The pilot is able to control the aircraft's motion by direct throttle, pitch and heading input, hence no roll or yaw is integrated and aircraft physic calculations are only in the x-z-plane (compare figure 4.1)

Consequently the initialize and tick method are aligned with the architecture in figure 5.4 and presented in listing 5.4. The second initialization is explained in section 5.7.

```
initialize
{
  initialize_RightWingTank()
  initialize_LeftWingTank()
  initialize_Tire()
  initialize_Brake()
  initialize_Engine()
  initialize_Propeller()
  initialize_AircraftPhysics()
  initialize_CIP()
}

tick
{
  second_initialize()
  if (second_initialize_by_pilot_bool === true)
  {
    update_Engine()
    update_LeftWingTank()
    update_RightWingTank()
    update_Brake()
    update_AircraftPhysics()  // includes engine_rpm, propeller thrust
    and friction force due to tick size independent calculation
  }
}
```

LISTING 5.4: Aircraft agent initialize and tick method

The aircraft updates its values and thus the input of the pilot every tick. As noted in section 5.1, the tick size is equal to one second. All relevant inputs for the motion

equations 4.1 and 4.2 are updated within “update_AircraftPhysics”. This includes the calculation of the RPM, thrust and friction. A tick size independent numerical approximation of the motion equations is used and explained in section 5.3.3. All other updates and underlying systems are explained in the next section. An overview of all the observable results is presented in chapter 6.

5.3.2 Update Systems

Engine

The system representation of the engine has been developed in more detail as a first approach and implementation with MARS to discover its capabilities. The following system features are included and initialized:

- engine oil pump and its condition
- engine oil leakage possibility
- engine oil quantity, temperature and pressure
- engine mixture control
- engine ignition switch
- engine failure (probability)

Then in “update_Engine” the following summarized steps are done:

1. When engine is off:
   (a) Check if engine is getting turned on
2. When engine is running:
   (a) Check if an engine failure occurs
   (b) Check if engine is turned off (by pilot)
   (c) Check if engine mixture control is set right, else reduce engine power output
   (d) Calculate fuel consumption with simple linear correlation to throttle input
   (e) Determine engine oil pressure and temperature:
      i. Check engine oil quantity and adapt values
      ii. Check oil pump condition and set typical engine oil values

The system behavior is based on symptoms described in Cessna Aircraft Company (1977) and Thom (1987), e.g. low oil pressure and high oil temperature indicate low engine oil quantity. With the aircraft data from table 3.1 and these symptoms simple values and assumptions are done for values like e.g. engine oil temperature or the effect of the wrong mixture control on the power output. The engine failure probability is set and checked with each tick (e.g. 10^{-6}).[1] So far the causes for increasing the engine failure probability are:

[1] Note: The engine failure probability over a time period ($EFPOT$) would be calculated with the engine failure probability per tick EFP and the equation $EFPOT = 1 - ((1 - EFP)^{t_{ges}})$

- water sediments in the tanks
- wrong engine oil quantity
- bad engine oil pump condition

Further functionality is the calculation of the engine RPM with the throttle input and engine power as described in section 5.3.3.

Fuel Tanks

The right (RWT) and left wing fuel tank (LWT) contains the two following features:

- possibility of water sediments (effect on engine failure probability)
- fuel quantity with maximum from table 3.1 and fuel consumption

Brake System

The brake system also contains two features:

- parking brake system
- calculation of brakes applied with a percentual brake input by pilot and a maximum brake force (around 2500N, external variable rational guessed)

5.3.3 Update Aircraft Physics

Within the function “update_AircraftPhysics” from listing 5.4 the motion of the aircraft is determined. The aircraft physics is aligned with the characteristics of the Cessna 150M to the best extent of the used model. In figure 5.5 the architecture of this function is described. The calculation sequence and the use of a tick independent inner loop can be seen. It should be recognized, that each calculation in this figure depends on the TAS. Thus, the calculation of the motion equations 4.1 (or 4.5) and 4.2 with the forward euler method from equation 4.8 is done with a variable time interval of tickSize/n_{cycle}, where n_{cycle} is an external input variable and defines the precision of that inner loop. E.g. with n_{cycle} = 100 the time interval results to 10ms with a ticksize of one second. This has been done due to high inaccuracies and unstable behavior of the aircraft physics system with too high time intervals. This is visualized and explained in chapter 6.

The following subsections present each calculation step and how the digital twin of the Cessna150 is modeled. It should be noted that most of the values are derived from literature and a small iterative process of checking the systems overall behavior. Only major design reasons are documented, but without recording every minor iteration. Furthermore it should be noted that the flight physics are based on a simplified approach and not further optimized, as the primary goal of this thesis is a first working architecture of the MAS. As mentioned before, a list of all external variables of the aircraft can be found in appendix A, which includes most of the parameters derived in the following and thus making it easier for the user to quickly adapt, test and visualize the impact of these parameters.

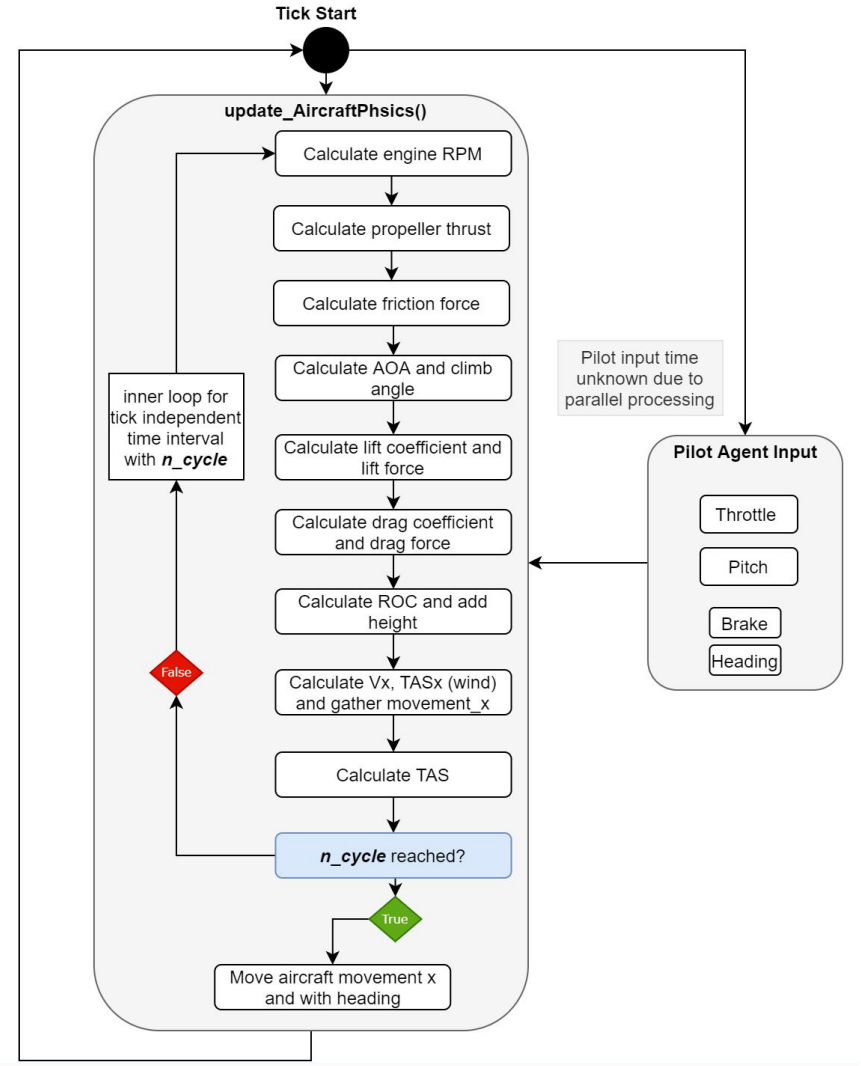

FIGURE 5.5: Aircraft Agent update_AircraftPhysics() function

Caculate Engine RPM

The engine RPM calculation is conducted by changing equation 4.17 to:

$$n = \sqrt[3]{\frac{P}{C_P \rho d^5}} \tag{5.1}$$

The density ρ is given and the diameter d of the propeller is 1.75 m as seen in table 3.1. The power input P results from a simple linear assumption of the maximum power of the engine with P_{max} = 75000W (compare table 3.1) and the throttle input by the pilot. However, the power coefficient C_P varies with changing advance ratio J (see equation 4.15 and figure 4.5). The figure 5.6 shows the used model for the power coefficient C_P. It's aligned with figure 4.5 (which uses a similar setting) and the fact, that the aircraft static full thrust RPM during takeoff ranges between 2460 min^{-1} and 2560 min^{-1} (Cessna Aircraft Company, 1977, p.2-4). The calculated RPM for takeoff at ISA MSL then results to 2526 min^{-1}. A further simplification as seen in figure 5.6 is that the power coefficient C_P decreases only with the TAS, thus interdependencies with the advance ratio J and the need for a numerical solution

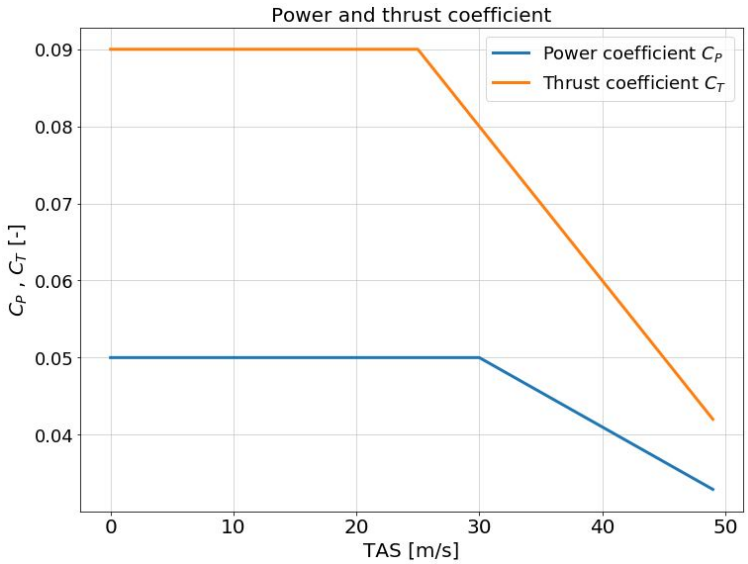

FIGURE 5.6: Aircraft Agent power C_P and thrust coefficient C_T model

for C_P are avoided. This still means an increasing RPM n with decreasing power coefficient C_P, which behavior is correct with lower static load of the propeller at higher speeds (Klesa, 2017).

Caculate propeller thrust

The thrust is calculated by adapting equation 4.16 to:

$$T = C_T \, \rho \, n^2 \, d^4 \tag{5.2}$$

The thrust coefficient C_T follows the same model as the power coefficient, therefore the thrust coefficient C_T is only dependent on speed as well. This model can also be seen in figure 5.6. Once again the model was tuned to fit with the data in figure 4.5 and experimental results by Mingtai (2012), who suggests a static thrust of 1400 N for the Cessna 150 during takeoff. However, the static takeoff thrust with the model in figure 5.6 is actually 1800N, which leads to a better takeoff behavior. Reasons for that discrepancy are discussed in chapter 6.

It should be noted that the power C_P and thrust coefficient C_T are highly dependent on the propeller pitch, which is constant for the Cessna 150M. Even though this is not further analyzed in this model, reasonable results can be archived with this first approach.

Calculate rolling friction force

The rolling friction force is calculated with equation 4.20. This calculation is integrated within the inner calculation cycle, as it depends on the lift and hence TAS. For equation 4.20 the wheel radius is r = 0.075 meter (Cessna Aircraft Company, 1977, p.8-11) and is simplified to be equal size for nose and main wheel tire. The distance coefficient c is chosen to be about 0.003. In combination with the radius of the tire r the roll coefficient c/r for tire on asphalt results to 0.04. This value is higher than the suggested roll coefficient of 0.02 (compare Engineering ToolBox (2008)). This modification is necessary due to the fact that the idle thrust of the propeller would be greater than occurring ground friction, which would force the pilot to constantly apply

breaks. Using this small tweak improves the taxiing behavior. It's not distinguished between static and kinetic coefficients.

When the aircraft is in-air, the friction force is set to zero, this simplifies the used equations of motion later on.

Calculate AOA and climb angle

The climb angle γ is updated with the equation 4.4. Then the angle of attack α can be determined using the pilot pitch input θ and equation 4.3 consequently.

Calculate lift coefficient and lift force

The lift L is calculated with equation 4.9, where the density ρ, TAS and the wing area $S \approx$15 m^2 from table 3.1 is given. On the other hand, the lift coefficient C_L depending on the AOA needs to be modeled. This is done with the following information:

- stall speed of $CAS \approx$ 25 m/s with MTOW = 725 kg at 5500 feet and ISA+16 (Cessna Aircraft Company, 1977, p.5-10)
- airfoil C_L over α plots as in figure 4.4 for the wing root airfoil NACA 2412 and the wing tip airfoil NACA 0012 from AirfoilTools (2020)
- estimated lift curve slope $C_{L\alpha}$ = 4.9 (Lautrup, 2005, p.451)
- maximum glide ratio of $E_{max} \approx$ 9 (Cessna Aircraft Company, 1977, p.3-9)

Since the glide ratio is defined as $E = C_L/C_D$, the model for the lift coefficient C_L over the AOA α in figure 5.7 is an iterative process due to the interdependencies with occurring drag. After reverse engineering the lift coefficient using the constraints from previous bullet points, the resulting lift coefficient C_L over AOA α can be observed in figure 5.7.

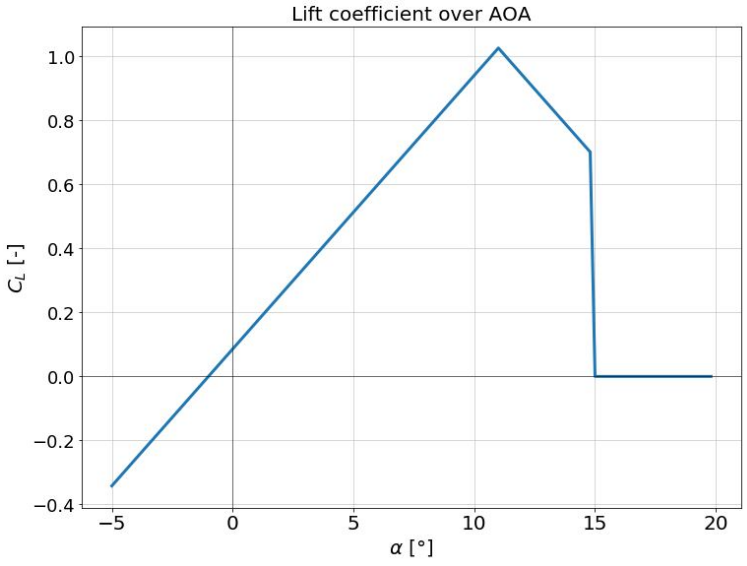

FIGURE 5.7: Aircraft Agent lift coefficient C_L over AOA α with lift curve slope $C_{L\alpha}$=4.9, stall angle α_{stall} = 11° and zero lift angle α_0 = -1°

Calculate drag coefficient and drag force

The drag force is calculated using equation 4.11. All variables are known except the zero-lift drag coefficient C_{D0} and the oswald factor e for equation 4.13. The proposed and chosen values given by Torenbeek (1982) are shown in table 5.1, which are widely acknowledged in aircraft design.

Variable	Proposed value range	Chosen value
zero lift drag coefficient C_{D0}	0.025 - 0.04	0.04
oswald factor e	0.65 - 0.75	0.7

TABLE 5.1: Aircraft Agent drag calculation variables (Torenbeek, 1982, p.149)

Calculate ROC, V_x and the TAS

All variables are determined to calculate the motion equations 4.1 and 4.2. The motion equation on ground 4.5 is not needed, since the wheel friction is set to zero when the aircraft is in air. Moreover, since a geodesic coordinate system is used, there are no different velocity components definitions. That means all variables can be added together in one equation.

The motion equations 4.1 and 4.2 merely represent the acceleration in x and z direction within the geodesic coordinate system. With these calculations the changed aircraft speeds ROC and V_x can be calculated using the artificially reduced time interval, which is controlled by n_{cycle}. The height of the aircraft is then changed instantly while speed V_x needs some distinction.

The speed V_x is the ground speed (in x-direction) GS_x. With the GS_x the real aircraft movement on the GPS based system is calculated, but only summed up and used in the end. This has been done at first due to the unknown input time of the heading by the pilot. However, direct movement is also possible, as no effect on simulation behavior has been discovered so far. In order to determine the total TAS, the true air speed in x direction needs to be computed first. This is done in equation 4.32, since the wind needs to be accounted for.

Afterwards the TAS is updated by using equation 4.6. When the loop is finished, the last statement is the movement of the aircraft, otherwise the calculation starts all over again from “Caculate Engine RPM”.

Move aircraft

After the the inner loop the movement of the aircraft is executed using the movement in the x direction derived from the GS_x and the last heading input of the pilot. In MARS this is done as shown in listing 5.5.

```
1 move me Aircraft__movement_x to Aircraft__heading_bearing
```

LISTING 5.5: Aircraft agent move method in MARS

5.4 Air Traffic Controller Agent

Similar to the pilot agent, there are many possibilities for a human cognitive agent. Also figure 5.3 could be applied to the ATC. The ATC could be a fully active agent, who is being aware of the entire environment and interferes processes or makes instructions on his own (as it is required in the real world). Nevertheless, the ATC agent in its implemented form is a semi active / reactive agent with no characteristics so far. The ATC is always listening to the frequency and when a pilot is making a valid request, the ATC processes the request and returns the determined answer. Thus, the ATC needs a trigger (the pilot's request) to change into an active agent role. This can also be seen in listing 5.6, where the initialize and tick method are presented.

```
initialize
{
  // (...)
  identifier = "Tower"  // only tower available so far
  state = "Listen_on_frequency"
}

tick
{
  if (state === "Listen_on_frequency")
  {
    // Check every few seconds for incoming messages
    // (...) When valid message arrives, change state to "
    Communicate_on_frequency"
  }

  else if (state === "Communicate_on_frequency")
  {
    // (...) handle request, return answer and change state within a
    randomized time range
  }
}
```

LISTING 5.6: ATC agent tick and initialize method

Handling requests could mean giving take off clearance or deriving safe taxi paths for giving taxi instructions. For deriving the correct taxi paths, the ATC is able to calculate the correct runway heading that needs to be used for landing and takeoff due to the wind direction. This is done by choosing the smallest angular difference Δ considering all available runway headings and the current wind bearing through equation 4.25.

In addition for evaluation the ATC can save the request type and the aircraft call sign, which is briefly presented in chapter 6. Further explanation on the airport information retrieval system are given in subsection 5.6.2, while section 5.5 offers more details on the communication system and a visualization of the actions by the ATC.

5.5 Communication Layer

The communication layer works like a blackboard described in section 2.2.2. This structure is used for a scalable communication system between ATCs and pilots instead of direct messaging, just like in the real world where pilots and ATCs have

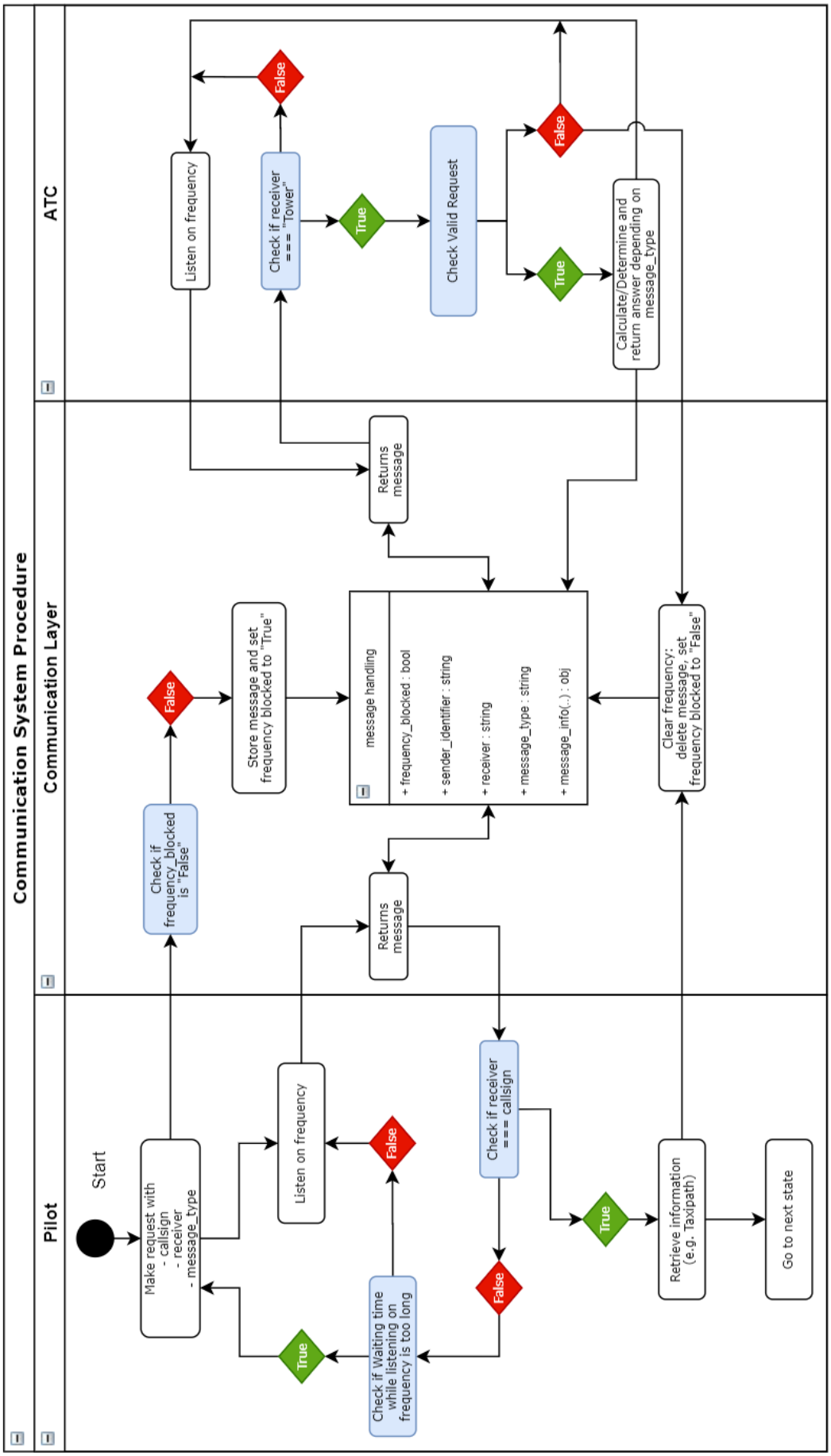

FIGURE 5.8: Implemented and developed communication system

to communicate over a given frequency. As shown in listing 2.1, the agents have to be defined within an “AgentLayer”, thus the communication layer and its functionality are accessible and defined using the “AgentLayer” in MARS. Figure 5.8 presents the communication system procedures in a swimlane diagram as it is implemented and how the agents are involved.

Pilot

When the pilot is in a state where he needs to make a request, he transmits the following data to the communication layer:

- call sign (aircraft)
- receiver (of the message)
- message (request) type

Either his message will reach the ATC via the communication layer or not. However, after making the request he will listen to the frequency and wait for a reply, which includes his call sign. If this takes too long, he will make another communication attempt. If he received his answer, he will go to the next state (for states see section 5.7). Moreover the communication layer is then triggered to delete the information message.

Communication Layer

The communication layer will block further pilot messages, if it is already containing a message. Thus it’s a first attempt to simulate the possibility of blocking the frequency in real aviation communication. It’s functionality breaks down into holding only one information at the time (blocking other messages by pilots at this time) and making it accessible to all agents that listen on the frequency. Please note that the ATC is always able to overwrite the information on the communication layer, while the pilot isn’t.

ATC

As mentioned in section 5.4, the ATC listens to the frequency the entire time, until a valid message with the right receiver and a valid request triggers him to process the request and to determine the answer. He is then always able to communicate back on the communication layer.

5.6 Environment

5.6.1 Weather

The full concept idea of accessing weather data is a so called spatiotemporal layer (in MARS with a time-series raster layer). That means weather data like temperature or wind is made accessible for all agents depending on the current (simulation) time and location. This system could however not be implemented successfully with MARS yet and is subject for further work. A spatial weather layer can be imagined as in figure 5.9.

Consequently another solution for the access of weather data is used: a weather agent. The weather agent has been chosen over a simple class or an integration in the “AgentLayer” for offering the option of external variable input. The following

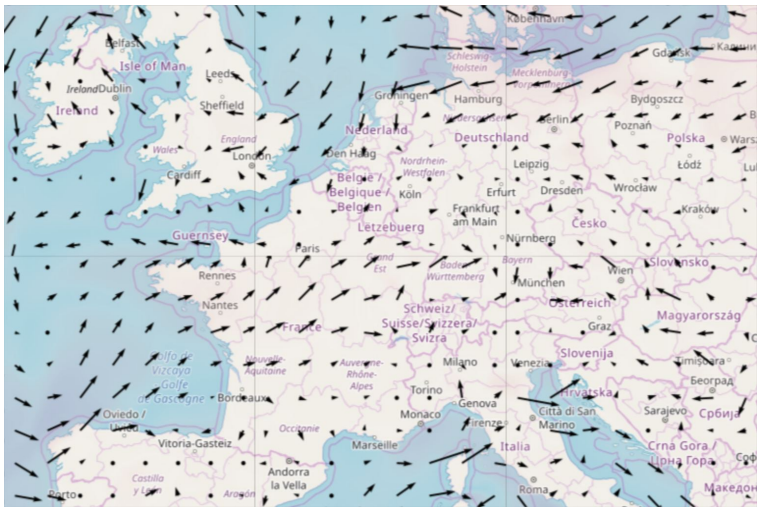

FIGURE 5.9: Spatial weather layer with wind data (OpenWeather, 2020)

accessible weather information, which is constant over the time and location independent, is provided to other agents:

- wind bearing φ_{wind}
- wind speed V_{wind}
- density ρ for a given height, calculated with ISA atmosphere rules as demonstrated in section 4.2.2

Hence the weather agent functions like the ISA atmosphere and provides wind data as presented in section 5.6. The optional external inputs are listed in table 5.2.

Variable Name	Type	Unit	Value Range	Description
gravity	real	m/s^2	[9.81]	Gravitational acceleration
pressure_QNH	real	Pa	[95000; 105000]	QNH, airport ambient pressure since elevation is 0 m
temperature	real	C°	[-10; 50]	Outside air temperature at airport
wind_bearing	real	°	[0; 360]	Bearing the wind is coming from.
wind_speed	real	m/s	[0; 15]	-

TABLE 5.2: Weather Agent external variables

5.6.2 Airport

MARS allows the integration of GeoJSON files with a vector-layer (MARS-Group, 2019, p.51). By extracting a GeoJSON file from https://www.openstreetmap.de/, most of the necessary data for airport Stade (see section 3.5) is available in form of polygons, lines and points. Some points of interests such as e.g. the holding point of the runway have to be added manually, as it is presented in figure 3.5. The challenge is to create a pathfinder algorithm in MARS (since integration of further packages

is not the user's authority) for autonomously detecting the right taxipaths to these points of interest. One possibility would be to implement a pathfinder algorithm, e.g. Liedman (2020) offers a package for the shortest path with weight functions between two points, when they're connected with lines or polygons.

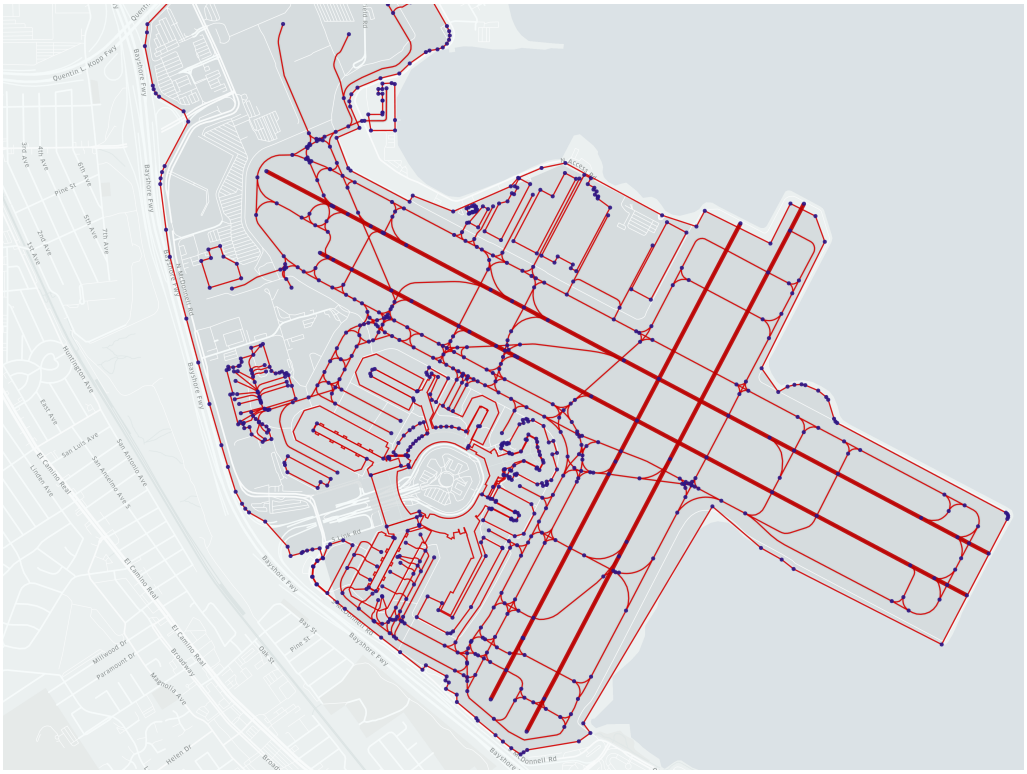

FIGURE 5.10: San Francisco Airport (IATA: SFO) graph visualization (data from Daniel Glake and visualized with https://kepler.gl/)

Another solution would be a graph network approach and working with nodes and edges (that can also be derived from a GeoJSON file), as it is done for the traffic simulation within the MARS Group, where a road network is used (MARS Group, 2020). An examplary visualization is offered in figure 5.10 for San Francisco Airport and displays the possibility with complex airport structures. Here a pathfinder algorithm is implemented with less effort and further functionality like occupying a gate or a point on the taxiway is easier to integrate. This method can be seen in research conducted by the DLR (DLR, 2020) or NASA (Heron Yang and Pasareanu, 2018).

In this thesis a simple solution is implemented. This solution is a class containing all paths, points of interest and runway heading information necessary for the pilot and ATC. The visualization of the airport is presented in figure 3.5. A further simplification is the assumption of an airport elevation of zero.

The implementation is shown in listing 5.7. The taxipaths are a list of coordinates in tuples, which are accessible using the runway headings and points of interest. It has to be noted that the order for coordinates in MARS is defined as: (Longitude, Latitude).

```
class AirportStade
{
  var taxipath: List<Tuple<real, real>>
  var available_runway_heading_list = #[110.0, 290.0]

  def Get_available_runway_heading_list() => return
    available_runway_heading_list

  def Get_taxipath_to_TakeOffPreparationPoint(heading : real)
  {
    if (heading === available_runway_heading_list.Get(0))
    {
      taxipath = #[#(9.4994287,53.5604441), #(9.4934248,53.5617509),
    #(9.4931937,53.5618008), #(9.4930632,53.5618664)]
      return taxipath
    }

    else if (heading === available_runway_heading_list.Get(1))
    {
      taxipath = (..)
      return taxipath
    }
  }

  // (...)
}
```

LISTING 5.7: Airport Stade class implementation

5.7 Protocol, States and Features

As stated before, the core of the simulation are the pilot's states, where some can be seen in listing 5.2. These states and their implemented functionality are documented here.

The states and their individual actions in figure 5.11 are aligned with the flight operations procedure for takeoff presented in section 3.4. The order of each state is shown in this figure for a normal procedure (e.g. without engine failure), from the point of time when the pilot reaches the aircraft for the preflight check until he leaves the airspace. All flight operation procedures are not representable within this thesis, thus as a first approach some actions from e.g. the preflight checklist procedure are chosen and displayed. It should be noted, that these procedures are also limited to the aircraft agent digital twin progress.

Additionally, very simple landing pilots and aircraft are added in the simulation in this thesis as well. The pilot states while landing are presented in figure 5.12 and explained later on.

Each of these states is not presented in full detail, but rather in their functionality and collaboration between the individual agents. Some further integrated features are explained along the way. It's explicitly stated here, that appendix B offers an overview of all features implemented in this thesis and all other (primary) ideas for further enhancements. Some of these enhancement are mentioned, if applicable. A summarized overview of further ideas is also given in section 7.2.

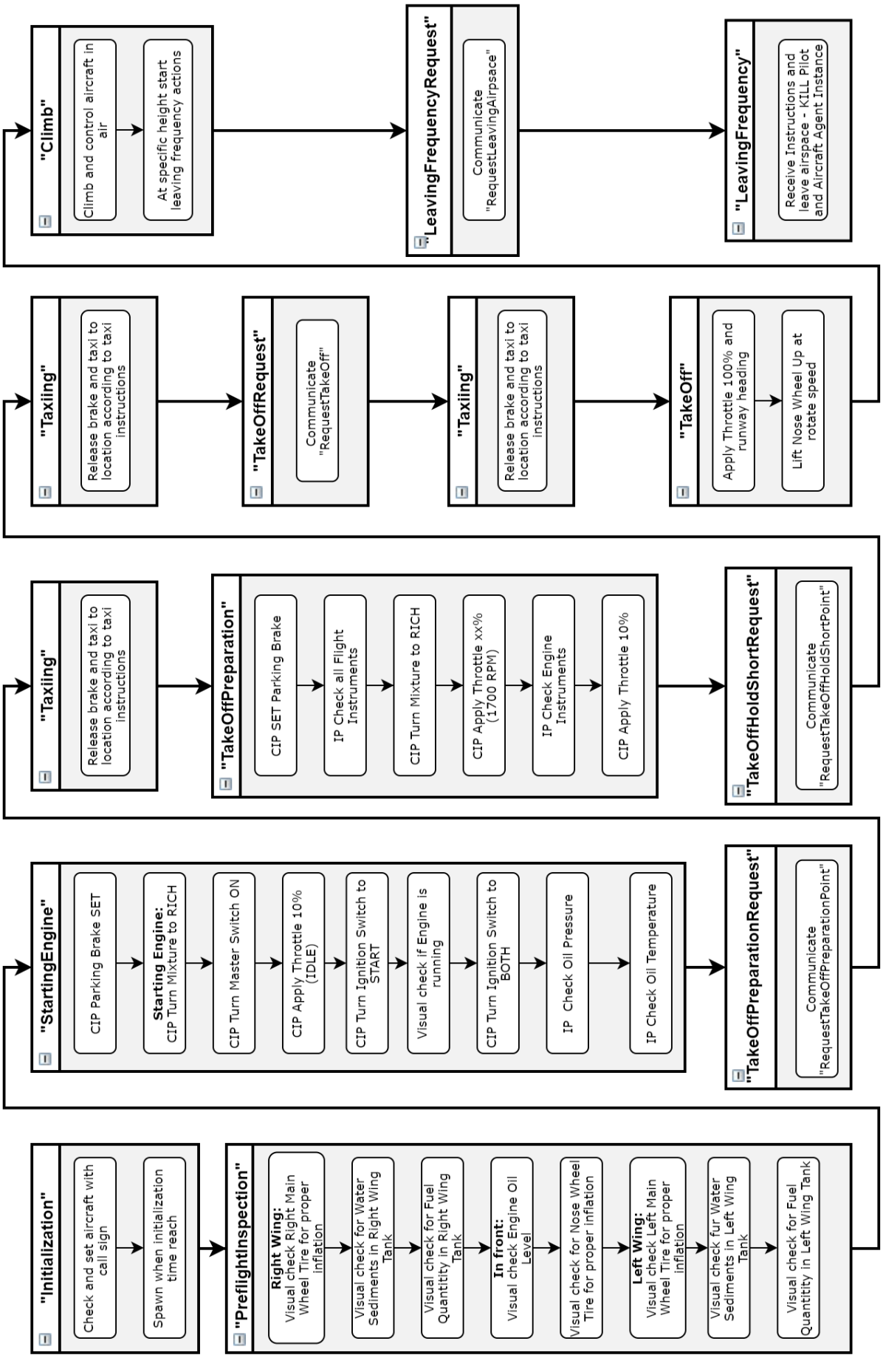

FIGURE 5.11: Developed and implemented pilot states and standard procedure for takeoff

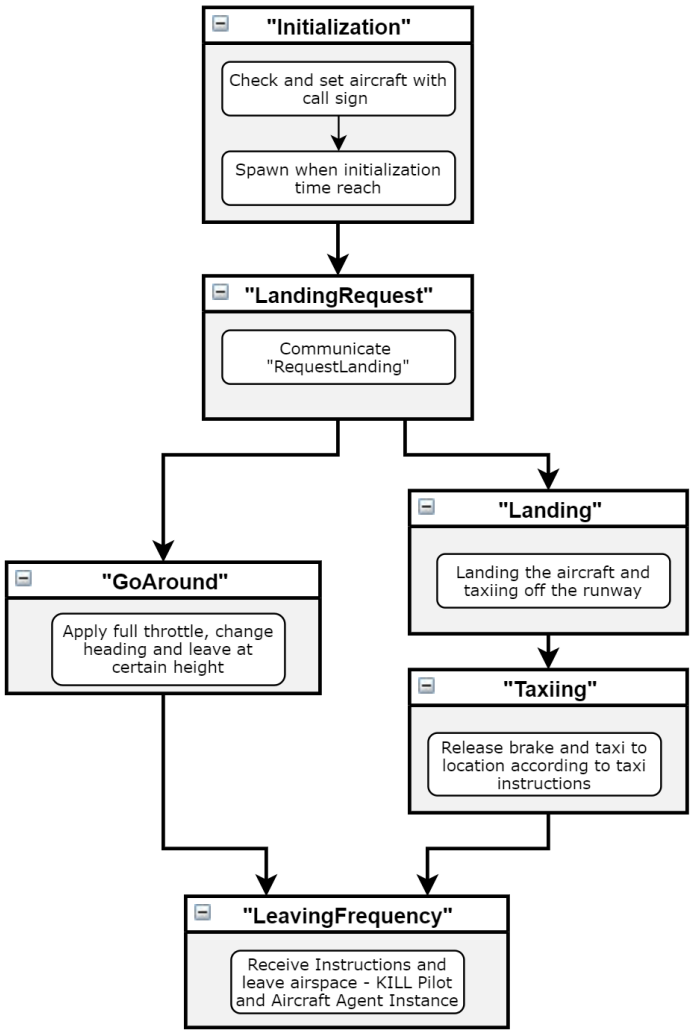

FIGURE 5.12: Developed and implemented pilot states for landing

Initialization

During the initialization state a second initialization of the pilot and the aircraft is triggered. Using the external config files one pilot and one aircraft should be mapped with an assigned call sign. Furthermore, it can be configured whether the pilot is taking off or landing and when the pilot starts with his actions. This is also explained in the appendix A. The second initialization of the pilot includes the following steps:

1. Check if only one aircraft with the same call sign exists, otherwise remove pilot (this is done in MARS with the explore function and a predicate as shown in listing 5.8)
2. When initialization time is reached, initialize takeoff or landing
   (a) Takeoff: Get apron spawn location by observer agent and set state "PreflightInspection" (when no apron spawn point is available, delay second initialization)
   (b) Landing: Get landing spawn location by observer agent, set state "LandingRequest" and trigger landing initialization process from aircraft

```
var myAircraft_array = explore Aircraft where [it => return it.
    Get_callsign() === myAircraft_callsign]
if (length(myAircraft_array) !== 1)
{
  // (...) abort mission and remove me
}
```

LISTING 5.8: MARS code: Pilot checking for aircraft with same call sign

Another reason why the aircraft and the pilot need a second initialization is the instantiating time of the observer and weather agent, since it's possible that during the initialization method the other agents are not yet available. As mentioned here the **observer agent** provides the spawn location information. It handles the space on the apron, where the amount of spawn locations can be controlled using an external variable. Additionally, it prints out the current simulation information in the console.

Preflight Inspection

During preflight inspection the pilot visually checks the health condition of the aircraft. With the aircraft system architecture presented in section 5.3.2, the pilot e.g. needs to remove possible water sediments in tanks or refill engine oil, otherwise this might lead to an engine failure. This is paired with different outcomes as shown in figure 5.13, where a typical visual check procedure is presented. It is visualized, how the pilot might skip a possible repair action. These implementations are done as a first system representation of individual pilot mistakes for possible incidents.

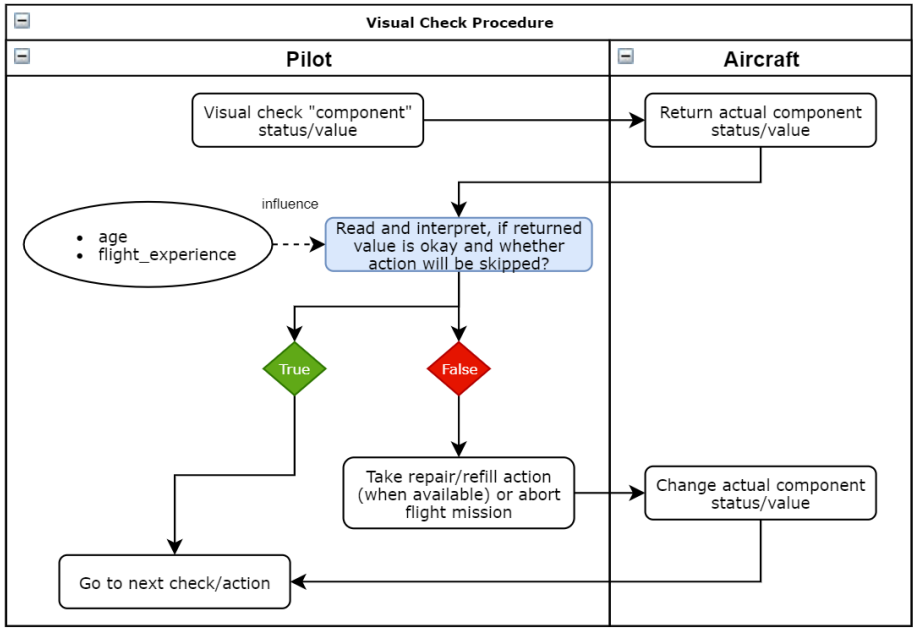

FIGURE 5.13: Preflight Inspection - Visual check procedure

Starting Engine

During the starting engine state the pilot needs to make his first actions via the control input panel presented in section 5.3 to get the engine running. It should be documented that the pilot is making direct inputs and changes the corresponding

aircraft variables. Thus the update time within the tick method of the aircraft is unknown and not controlled. This has already been indicated in section 5.3.3. Worth mentioning here is the alternative solution with a mailbox system, where the aircraft starts with internal updates as soon as it gets the pilot input.

During the starting engine state the pilot also conducts his first instrument check. Instrument checks are kept at a low level in this thesis, thus the pilot checks the value and either ignores failures with a certain probability and continues or goes back and tries to fix the issue with the preflight inspection one more time. The full developed concept idea for instrument checks is presented in figure 5.14, which would also be already feasible with the pilot and aircraft architecture. The red border indicates features that are not yet implemented.

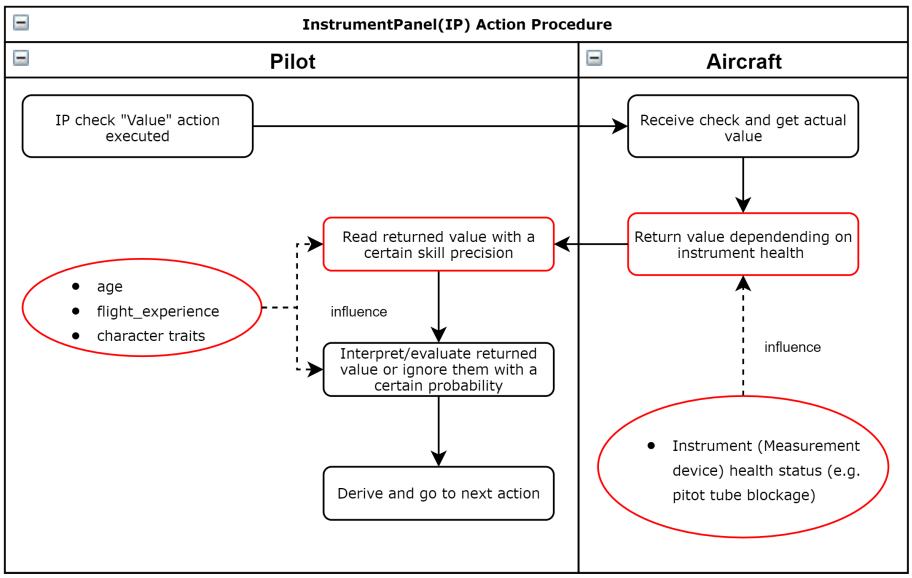

FIGURE 5.14: Instrument check procedure

After the engine started there is the possibility for an engine failure. Thus, it's important to note that there is no engine failure procedure integrated yet, since in various states like e.g. takeoff the complexity is very high. Instead the pilot and aircraft will be removed from the simulation after a short time.

Request Takeoff Preparation

In this state the pilot makes a request for taxi instructions to a point, where the pilot can do his takeoff preparations as shown in section 5.5.

Taxiing

After receiving the taxi information in form of a list of coordinates, the pilot is able to follow these taxi instructions. The implemented tick method of the pilot during the taxi state is visualized in figure 5.15.

The following complementary explanations for figure 5.15 are given:

- For all calculations that require an own position the aircraft position is used (although pilots position gets updated with the aircraft's position)
- For the distance and bearing calculations the equations 4.23 and 4.24 are used

- The pilot is controlling the taxi speed like a simple proportional-derivative (PD) - controller, since he is adapting the speed based on thresholds and a temporary acceleration feeling implemented as a help function

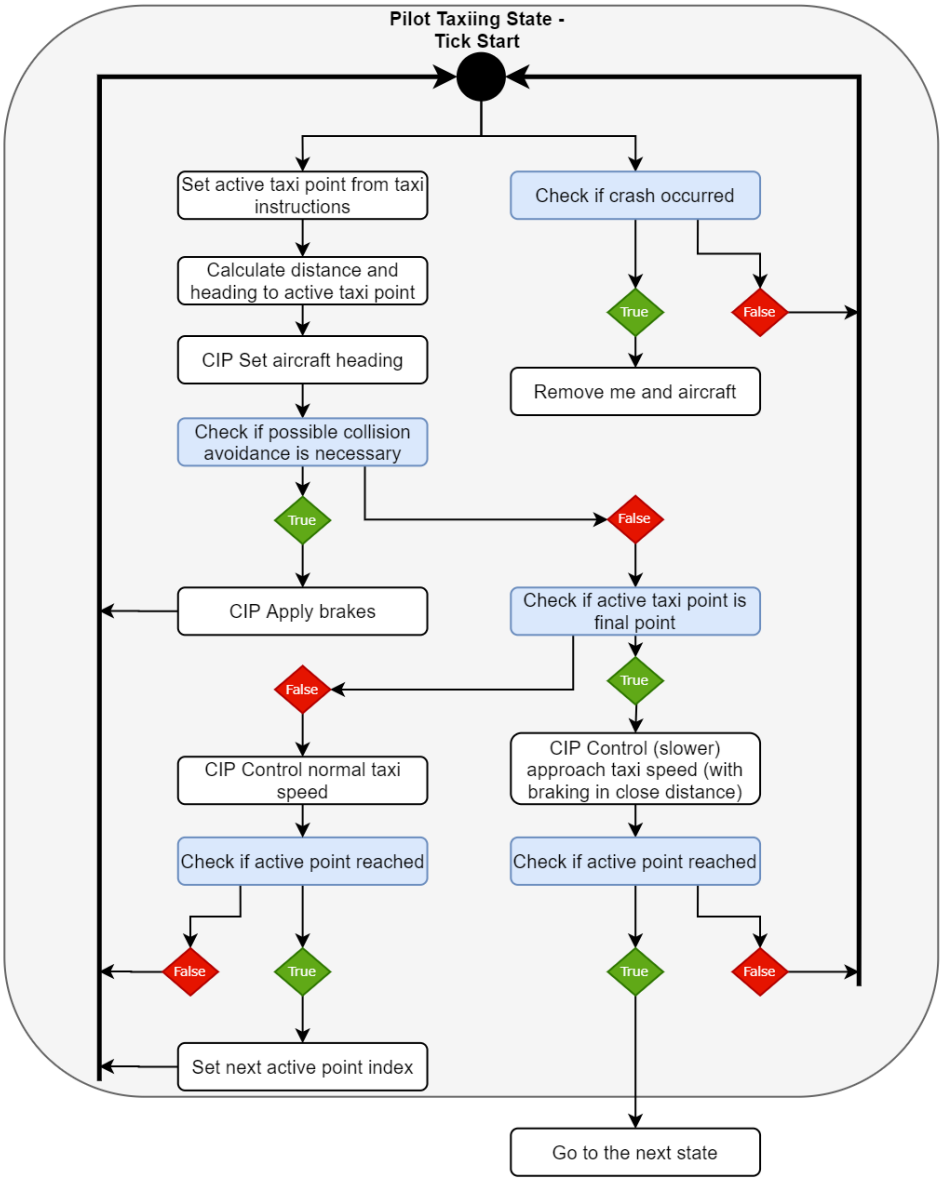

FIGURE 5.15: Pilot tick method in taxiing state

Further on the collision system implemented in the taxiing state can be imagined as seen in figure 5.16. This feature allows the pilot to “see” other aircraft and apply brakes if necessary. The implementation in MARS is demonstrated in listing 5.9, where the explore method is used with a predicate chained with the haversine function from equation 4.23. The cone angle φ_{vision} and the cone radius d_{vision} are also external variables (compare appendix A “Taxiing__angle_of_vision” and “Taxiing__spacing_to_next_aircraft”).

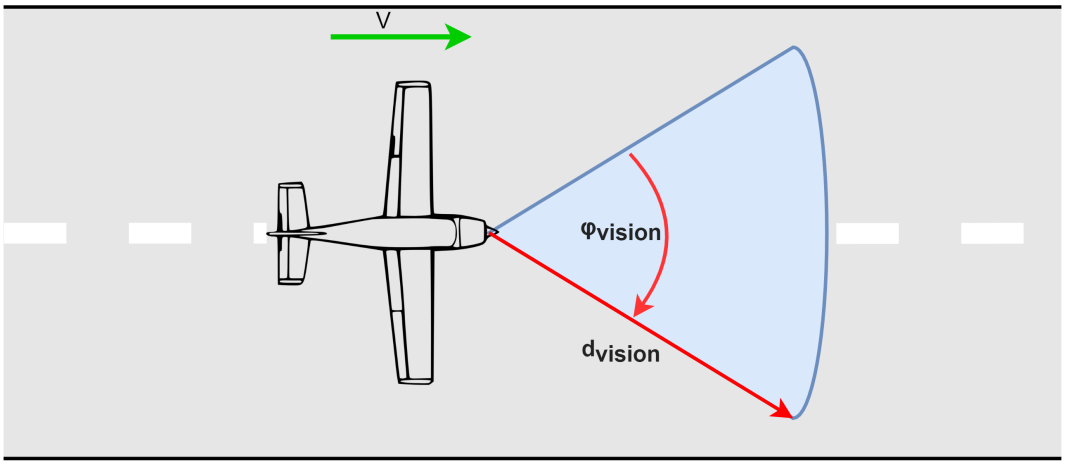

FIGURE 5.16: Collision system - Pilot vision and collision detection

```
1 var aircraft_array = explore Aircraft where [x => return formula.
    haversine(myAircraft.Get_position(), x.Get_position()) <
    Taxiing__spacing_to_next_aircraft]
2 if (length(aircraft_array) > 1) // since list contains ALWAYS the
    airplane controlled by the pilot!
3 {
4   // (...) Check all other aircraft found, whether their in between the
      cone angle of vision and set next action / flag
5 }
```

LISTING 5.9: MARS code: Pilot checking for other aircraft in collision range

Takeoff Preparation and further Taxiing

During takeoff preparation the pilot goes through a long checklist and checks various instruments. This procedure is similar to the mechanisms presented in the starting engine state. Worth mentioning is the application of thrust with parking brake set during takeoff preparation in order to check various engine instruments. This is limited by the current implementation of digital twin of the Cessna 150M presented in section 5.3 (e.g. no magnetometer representation and thus check available). Afterwards the pilot makes a request for the holding point short of the runway, taxi to that point and subsequently asks for takeoff clearance.

Takeoff (and Landing) Request

The difference between takeoff or landing requests to other general requests is an additional check that is performed by the ATC. The ATC checks, whether no other aircraft is currently landing or taking off. Only after detecting no other aircraft the ATC gives clearance to the pilot. Thus, this small feature leads to a regulated takeoff and landing operation at the airport.

Takeoff and Climb

The takeoff and climb process can be broken down into the following steps:

1. Apply full (100%) thrust and set correct runway heading
2. Rotate aircraft (apply pitch) at given rotate speed V_{rotate} (which is an external input, Cessna Aircraft Company (1977) defines $V_{rotate} \approx 25$ m/s)

3. control pitch as a simple PD - controller again (see taxiing state) to maintain a value range of ROC
4. make a request to leave airspace at specific altitude

These are simplified steps for making a representation of the takeoff and climb phase work. The most important part in order to enable the takeoff is the aircraft's architecture and a appropriate physical model as presented in section 5.3. It should be noted that all simplifications made to the aircraft physics in section 5.3 and its calculation in section 4.1 apply and thus e.g. the ground effect is missing. Further improvements on the detail of the model is possible and explained later for e.g. instrument screening or environment perception.

Landing Procedure

For better visualization of the benefits from MAS (later on in chapter 6), the possibility of landing pilots and aircraft is introduced. The pilot states and actions during the landing procedure are presented in figure 5.12 and will be described briefly in the following.

1. Initialization:
   (a) The pilot and the aircraft are spawned at a specific height and distance away from the airport runway
   (b) The aircraft is heading towards the runway and with a landing initialization, the engine is running and and the aircraft is already flying
2. Landing Request: Either get landing clearance and attempt landing or go around
3. Go around: Abort landing procedure, apply full thrust and fly to different heading
4. Landing: As soon as the aircraft touches the ground the pilot applies brakes and taxi to the apron

Please note that the underlying system during landing for controlling the aircraft is as simple as during takeoff. Additionally an intelligent behavior for targeting the runway is missing, since when e.g. the weather temperature or wind is adapted, the control mechanism of the pilot would lead to an unsuccessful landing.

Chapter 6

Results and Review

In this chapter the results of this work are presented and afterwards evaluated. Consequently the objectives of this thesis should be emphasized again. The main objective of this research is developing a concept for a digital twin of the Cessna 150, pilot and ATC with MAS using MARS for the flight operation stages from preflight check to takeoff. Another goal is the implementation of the first parts of this developed concept in order to get a better understanding about the usability of this concept.

First of all the general result of this thesis is the developed concept and its implementation presented in chapter 5 itself. As previously mentioned, it's the result of a concept design phase with numerous discussions and sketches. It offers a basic foundation for future work with specific formulations of questions. Thus, one part of this chapter is section 6.1 with a summary of chapter 5 and its functionality and achievements, but also with its challenges. This is combined with the actual possible visualizations of the simulation output and some insights about simulation behavior.

Consequently another part of the results is the experience made during implementation of features and requirements for the digital twin using MAS with MARS based on criteria like e.g. MAS paradigms benefits. These are documented and then reviewed in the section 6.2 using different criteria. Thus, the gained insights during the literature research, concept development and implementation are combined with the actual simulation results and evaluated with a benefit analysis matrix for MAS with MARS.

6.1 Results

The access to the implemented program with MARS is available in appendix A. As mentioned in 2.4, the output of the simulation is a csv file that contains each observed variable per time step. The available jupyter notebook script (see appendix A) can automatically visualize all outputs in order to provide an easier user experience and help speed up the design process. Within this thesis, no specific investigation is conducted, thus it can rather be seen as a tool that should be utilized during future work. E.g. a investigation on takeoff behavior of the pilot could be done, but further systematical modeling is necessary. Recommendations on a approach to that are given later on.

The most important achievements of the **implemented** program are:

- digital representation of all the flight operation stages from preflight check to takeoff with additional landing pilots and aircraft

- define sustainable architectures for procedures
- first architecture of a controllable, fully passive and digital Cessna 150M agent architecture with system representation and aircraft physics
- active pilot agent is able to navigate the aircraft (taxi, takeoff, landing)
- simplified and accessible environment (weather, airport)
- sustainable communication layer with pilots and ATCs interacting
- Use of MAS paradigms in features like e.g. collision system
- representation for airport operations with adding landing aircraft and ATC checking for clearance

Some of these achievements can be visualized with the jupyter notebook script. For instance figure 6.1 visualizes all the states for a landing and a starting pilot agent over the time for a normal run, which are introduced in section 5.7. This is saved for each pilot agent.

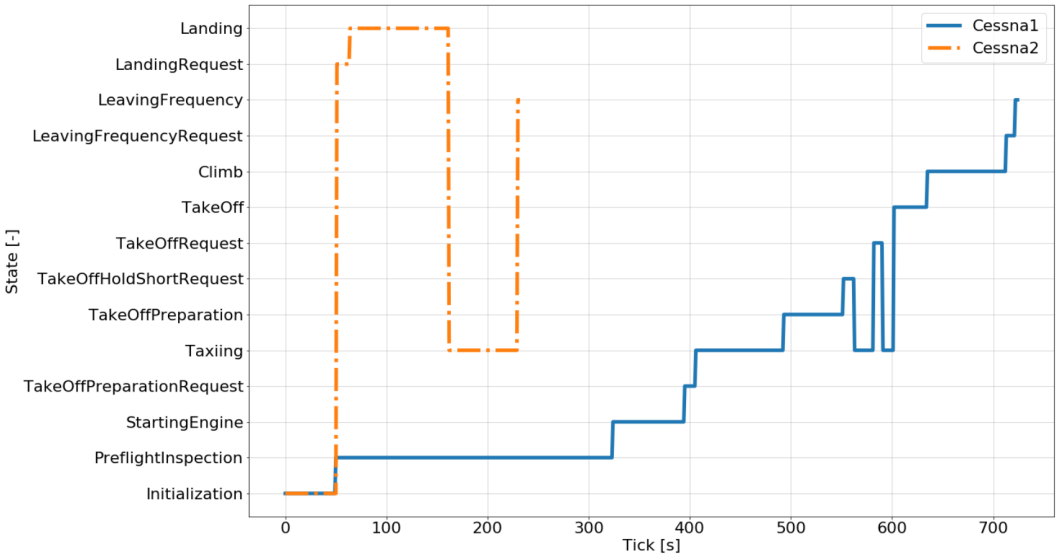

FIGURE 6.1: State overview of a landing (Cessna1) and starting pilot (Cessna2)

Furthermore, all aircraft agent variables can be investigated. This includes variables such as drag, lift, speed, height, acceleration, various angles or engine condition. For demonstration purpose, figure 6.2 visualizes the engine RPM for a landing and a starting aircraft. Here the various states can also be distinguished and compared to figure 6.1, since the RPM changes with the throttle applied by the pilot in his various states, who's control behavior is a simple approach as mentioned in section 5.7. It can also be observed that the RPM increases due to higher TAS during takeoff. The TAS is visualized in figure 6.3 for comparison. For orientation, the takeoff phase of the pilot begins at the simulation tick of 600 seconds as seen in figure 6.1. As previously mentioned, the takeoff state can also be identified in the figures 6.2 and 6.3. Regarding TAS it should be kept in mind, that the TAS depends on the wind speed (which was 5 m/s during the simulation) and its direction. Therefore a change in TAS will be observed when the aircraft is turning. Additionally, the TAS is limited to positive values to avoid the aircraft moving backwards due to negative drag caused

by wind. For further demonstration of the working aircraft physic, figure 6.4 shows the lift and figure 6.5 the thrust along all states for a landing and a starting aircraft.

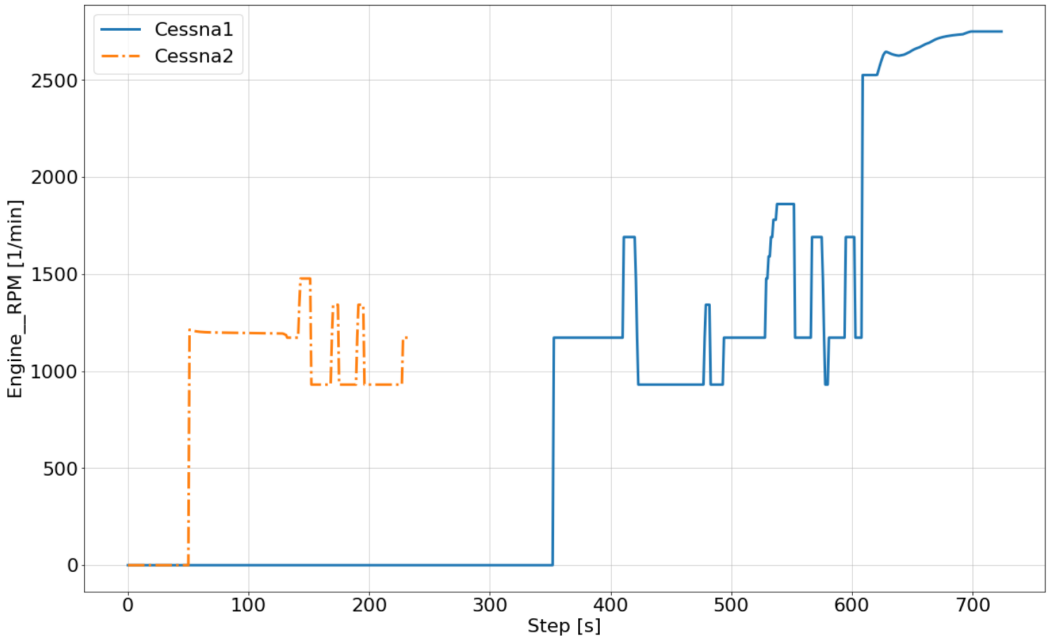

FIGURE 6.2: Engine RPM over the time of a landing (Cessna1) and starting aircraft (Cessna2)

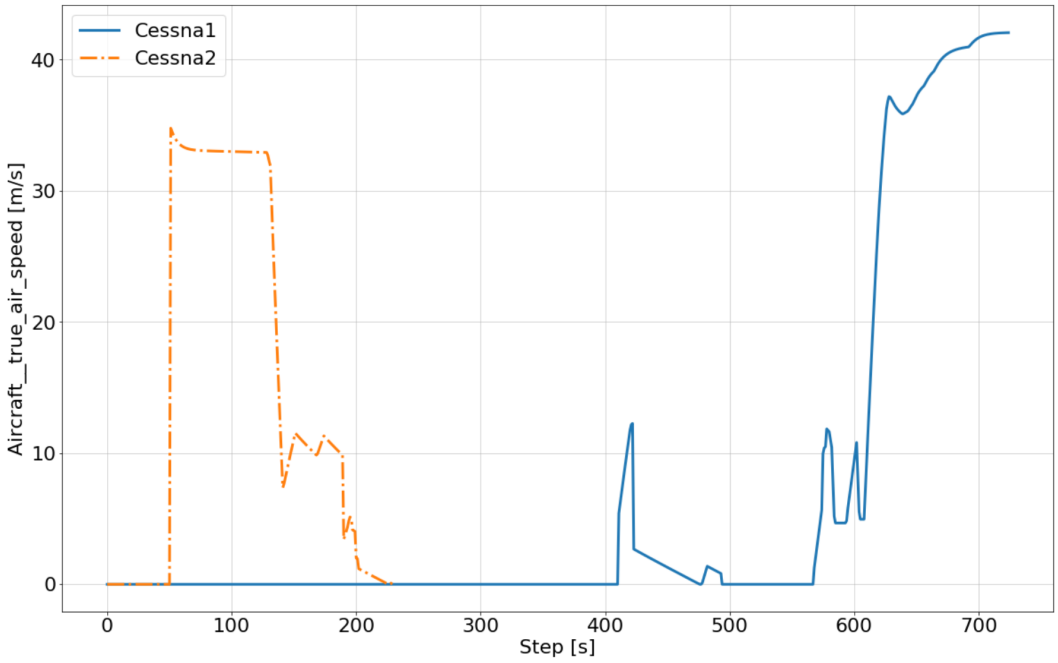

FIGURE 6.3: TAS over the time of a landing (Cessna1) and starting aircraft (Cessna2)

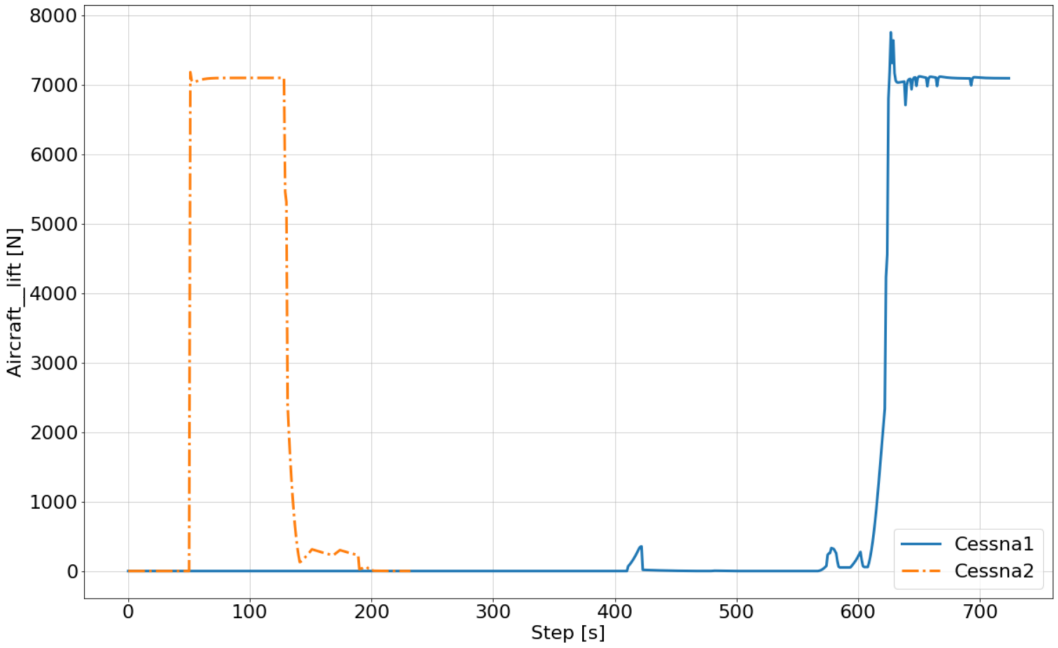

FIGURE 6.4: Lift over the time of a landing (Cessna1) and starting aircraft (Cessna2)

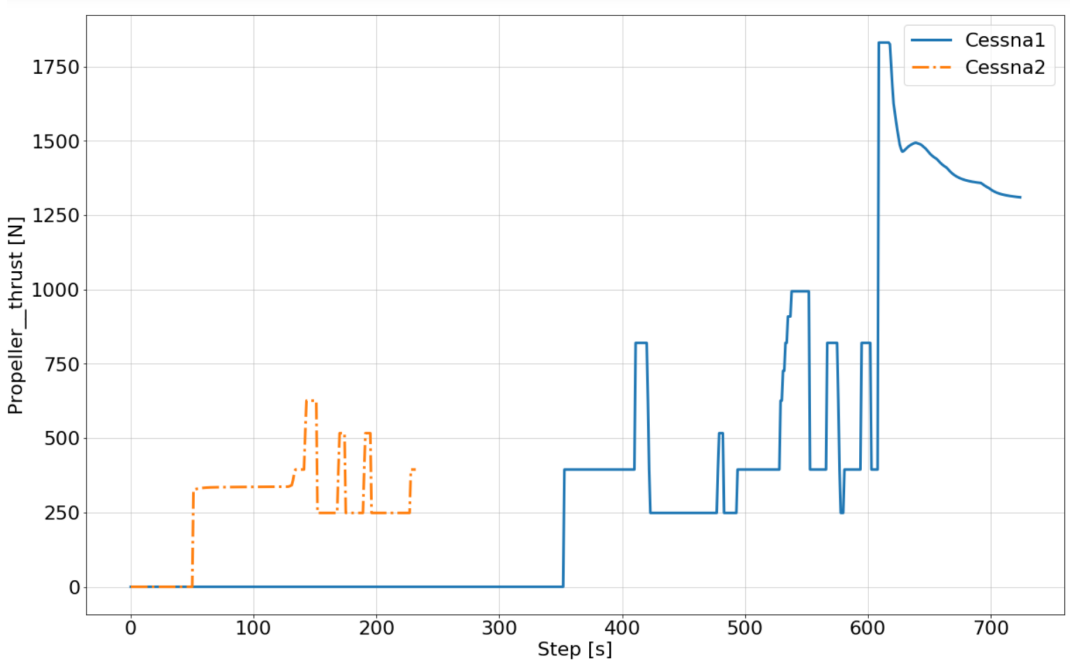

FIGURE 6.5: Thrust over the time of a landing (Cessna1) and starting aircraft (Cessna2)

An example for the aircraft system is given in figure 6.6, where the status of the parking brake can be observed. E.g. during takeoff preparation, the pilot needs to set the parking brake back on again (compare to figure 6.1). The appendix A includes a link to the jupyter notebook for further visualizations of the aircraft.

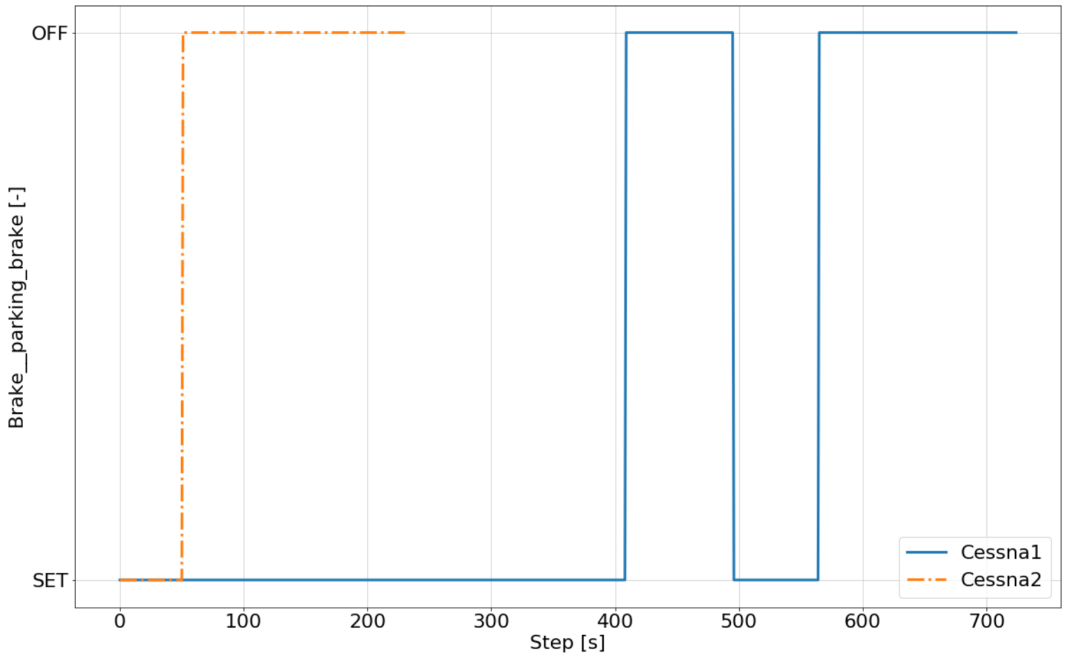

FIGURE 6.6: Parking brake over the time of a landing (Cessna1) and starting aircraft (Cessna2)

It is worth mentioning that the pilot and aircraft agents also include an observed variable with event information, which catches special occurrences like skipping a repair action or an engine failure. This variable allows better backtracking of the simulation outcome and should further be used in future works for extending features. Another notable variable that can be analyzed is also the takeoff distance needed.

Visualizations are also available for the ATC agent. These are e.g. the call sign or request type he received to analyze the communication activities. Figure 6.7 shows the call signs received by the ATC, when a simulation for multiple pilot and aircraft agents is done. This shows the working communication system and its further potential as being discussed in the next section.

Other achievements such as the airport operation representation, pilot taxiing and controlling the aircraft or the collision system can be visualized with “kepler”. Kepler visualization on `https://kepler.gl/` enable a simple animation of the aircraft movements, this is indicated in figure 6.8. A short animation is also available on the accompanying CD or at the link given in the appendix A.

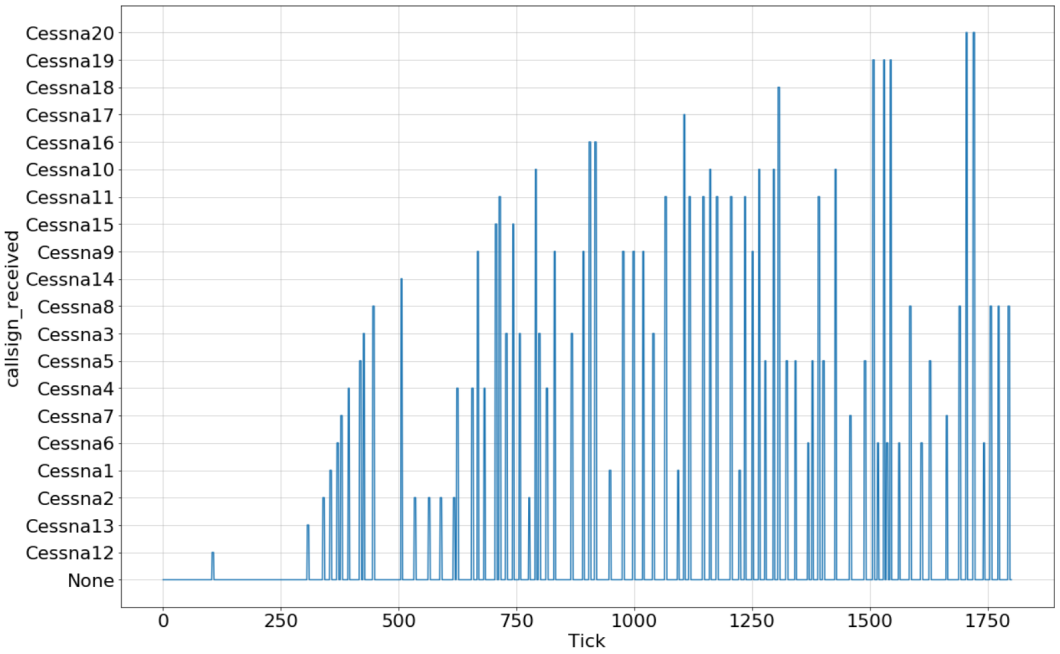

FIGURE 6.7: Call signs received by the ATC in a simulation with multiple pilots and aircraft

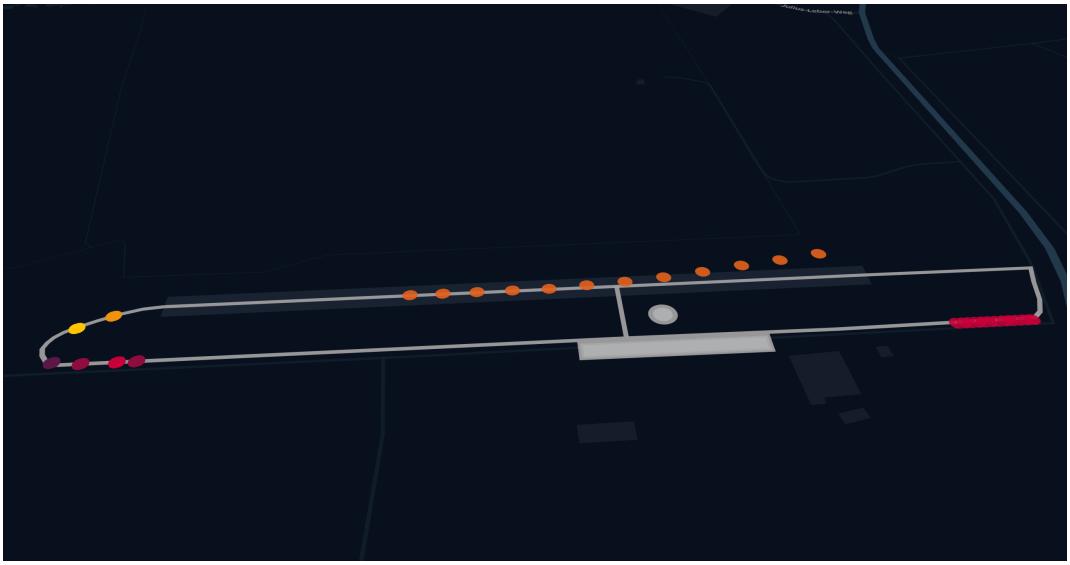

FIGURE 6.8: Kepler visualization of simulation result, where each dot represents an aircraft with pilot (https://kepler.gl/)

Beside the achievements many challenges occurred during the implementation of the concept. One of the main challenge is the representation of the aircraft physics. A possible solution using external flight simulation programs in order to achieve more accurate results is discussed in the section 7.2. Some of the challenges due to the implemented aircraft physics are:

- Rotating the aircraft at takeoff
  - When rotating the aircraft and without having a ground effect lead to sudden increase of drag and thus the airplane slows down (visible in figure 6.3 at around tick 620 seconds)

  - Slowing the aircraft leads to less lift and thus the aircraft can have a negative ROC. The aircraft consequently goes back to ground and friction of the landing gear occurs. This in combination of high drag due to high pitch angle leads to slow acceleration.
  - This issue is temporary solved with a higher thrust as stated in section 5.3.3

- Numerical stability for aircraft physics calculation
  - A high tick size of one second and thus a high time interval for the forward euler method (see equation 4.8) leads to numerical instability that can lead to the simulation crashing (also see Grüne (2008, p.38pp) for numerical stability)
  - Figure 6.9 shows the comparison between a higher and lower time interval for the numerical calculation of the angle of attack (some signal jumps are still caused by the jump inputs for pitch by the pilot)

- pilot taxiing control behavior
  - Due to the tick size of one second a precise stop at a taxipoint along the taxi instructions is difficult at higher speed and leads to overshooting these points
  - With wind speed the taxiing speed could no longer be controlled on TAS, since TAS can be above zero while the aircraft is standing in one place (temporary fix with GS added for this)

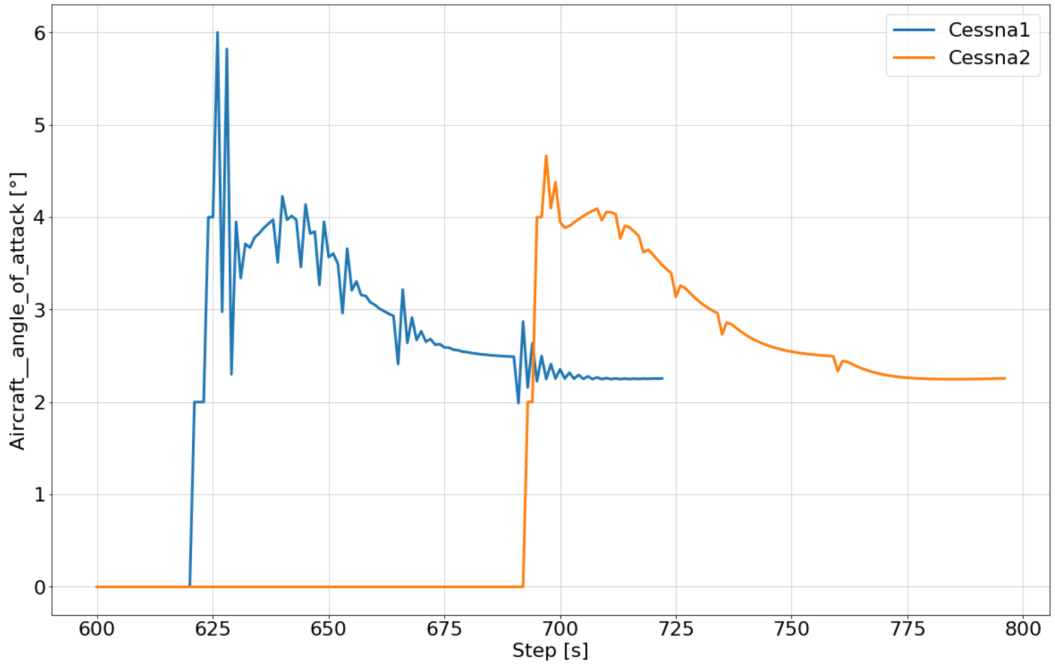

FIGURE 6.9: Numerical stability: angle off attack during takeoff for a time interval of Cessna1 = 1s and Cessna2 = 0.1s

Further challenges regarding the overall concept are discussed in the next section.

6.2 Evaluation

Table 6.1 shows an assessment matrix of MAS with MARS for the application of a Cessna 150, pilot and ATC digital twin for various states from the preflight check to takeoff. The assessment criteria (columns) are derived from Wooldridge (2002), MARS-Group (2019), and Lars Braubach and Lamersdorf (2008). Here the columns show first an overall assessment of MAS and MARS. The following columns after either MAS or MARS include the proposed advantages of each system, where MARS can also be seen as a subdivision of MAS with further functionality and features. Further on the rows include the states, which are grouped and then assessed on how they benefit from MARS and MAS. The performance regarding each criteria is then rated from "- -" (weak) to "++" (strong). This serves as an overview for the upcoming discussion. The criteria are interpreted as follows:

- MAS
  - Collective Behavior: Possibility to study complex collective behavior from individual agent based modeling
  - Autonomy: Autonomous simulation and agent action after defining set of rules and logic
  - Decentralization: Advantage of all agents having impact on the simulation outcome and influence the environment
- MARS
  - Spatiotemporal Layer: Utilization of geospatial and timeseries layer including position control
  - Data input and processing: Handling of input data, processing data (and evaluating data)
  - Simplicity of language: Easy to understand, lightweight but also limited modeling language

State	**MAS**	Collective Behavior	Autonomy	Decentralization	**MARS**	Spatiotemporal layers	Data input and processing	Simplicity of language
Preflight Inspection, Starting Engine and Takeoff Preparation	- -	- -	- -	-	-	- -	o	++
Communication	++	++	o	++	++	o	+	++
Taxiing	++	++	+	++	++	++	++	+
Takeoff, Climb and Landing	+	o	+	+	+	++	+	+

TABLE 6.1: Assessment matrix for the various states

Table 6.1 can be interpreted and summarized as follows:

- Due to the linearity and a less complex system for procedural checks, the states preflight inspection, starting engine and takeoff preparation do not benefit well from neither MAS or MARS
  - Only two agents interacting (aircraft and pilot)
  - Simulation results can be stochastically determined
  - A better solution for the first two states, preflight inspection and starting engine, is a defined aircraft condition representation via an external system for the starting point of the simulation
  - Nevertheless, the definition and interaction between the agents and entities are quickly integrated due to the simple language usability of MARS
- Communication actions do benefit as expected from MAS (compare section 2.2)
  - Especially with the possibility to use the “agentlayer” as a communication layer within MARS
  - Access across all agents for e.g. the ATC with giving takeoff and landing clearance or instructions
  - Impact of (pilot) agents on the environment with e.g. blocking the frequency
- Taxiing (or airport ground ground control) offers the best possibility for autonomous control and research with MAS
  - In contrary to the procedure checks, a set of defined actions (thrust, steering, communication) and rules (collision, taxi path) leads to a good representation of the taxi behavior
  - The collision feature implementation or similar are achieved without big effort
  - With MARS (outside the DSL) the possibility exists to use a graph system as mentioned in figure 5.10 (compare similar research areas like DLR (2020)) for airport performance and process simulation (compare Airport Research Center (2020))
  - A spatiotemporal weather layer can be used to access wind data for even more precise representation
- Takeoff and Landing are very complex procedures, especially the aircraft physics representation with MARS is very limited
  - MARS DSL does not enable the user to include further packages or programs (easily), this was especially for the modeling of the aircraft physics an obstacle
  - During takeoff only high interaction between one pilot and one aircraft (so far)
  - However, spatiotemporal weather layer is a great advantage of MARS in order to include e.g. gusts during takeoff at specific locations

  - Modeling with MARS for answering specific problems with limited boundaries like e.g. safety analysis is better suited (see explanation below)

Within the scope of this thesis a detailed implemented model for all states from preflight check to takeoff is not possible. The concept presented in chapter 5 acts as a foundation and gives insight into how different areas can easily be applied with MAS. To apply these concepts, another approach needs to be chosen. For instance Bouarfa (2016) proposes an approach, which is better suited for the MARS framework. Instead of using a holistic approach, Bouarfa formulates a precise scientific problem like e.g. safety assessment of active runway crossing operation (Bouarfa, 2016, p.80pp). Bouarfa chooses the approach based on one scenario and then models every possible interaction and agent (even the lights on the runway). Each agent is described with a limited set of action that influences the environment. Then Bouarfa runs a Monte Carlo simulation to access the impact of agents individual performance on the simulation outcome. This would also increase the autonomy and the dynamics[1] of the simulation, since the dynamics of this current concept is low due to the huge impact of the pilot on the simulation outcome. This kind of approach would also enable further options utilizing cognitive agents, since the set of actions across all states in this work is too large for the proposed cognitive agent model in figure 5.3. That approach would also allow for the implementation of a detailed engine failure procedure within a limited scenario. Thus, a similar approach could be done for each state like e.g. takeoff procedure itself.

However, to increase the quality of the model for the takeoff state, another solution in order to model the aircraft physics might be beneficial, due to the following disadvantages of MARS (DSL):

- As mentioned before the missing options of using further libraries and the lack of support for matrix operations
- Hence the aircraft system and physics representation have to be built up from scratch (e.g. flaps system still missing)

A better solution would be using an open source library for the flight simulation and trying to include that into an multi-agent environment, which could be conducted in future works. It should be noted, that a more realistic and full aircraft physics simulation is only needed for the takeoff, since for aircraft ground control simulation a simple trajectory model like the current concept in this thesis is good enough. Nevertheless, a more sophisticated physics implementation would also enable further simulations beyond the takeoff phase. Further suggestions are made in section 7.2.

The aircraft physics model of the Cessna 150 developed within this thesis still offers a good foundation for further work, since it provides an interface for agents (in this case the pilot) for taxiing, taking off or landing, although some optimization could be done. One optimization is applying other more precise approximations like the Runge-Kutta method (Grüne, 2008, p.28pp) for numerical approximation (compare section 5.3) instead of using the forward euler method.

Further on the implemented concept lacks the following features, which are subject to further optimization (also see appendix B for all suggestions):

[1] In this context dynamics means the influence of various entities on the environment and the simulation result, similar to decentralization

- Dynamic adaption of the tick size for better movement representation and reaction handling of the pilot
- The use of states does not allow the parallel execution of e.g. communication and taxiing at the same time, thus a workload handling is missing or parallel states execution
  - This can also be accounted as a disadvantage of the states system, but that does not matter when modeling a scenario
- Realistic instrument screening behavior for pilot as environment perception and deriving next action (cognitive agents implementation)

In conclusion the developed concept and its first implementation shows that the representation of the holistic process from preflight check to takeoff with a digital twin of a Cessna 150, pilot and ATC is possible on a low scale, but not the ideal approach for further detailed modeling. The reasons are that some procedures are too linear and and the implementation of all possible procedures along all stages of flight operation as well as the linkage of these procedures is too complex. Even the Cessna 150M, which is a rather simple aircraft, implies a set of possible actions and subsystems that is too big for general modeling. Therefore it would be better to choose another approach like Bouarfa (2016) did and model a single scenario, that utilizes parts of the developed model. Due to the user-friendly MARS DSL, fast results with detailed modeling of situations like takeoff or runway crossing could be achieved. Further on a full working concept in chapter 5 is presented and fully documented for future work.

Chapter 7

Closure

7.1 Conclusion

The main objective of this thesis is to develop a concept of a Cessna 150, pilot and ATC digital twin, implement it with the MARS DSL and review the the application potential. This combination of concept research and first implementation allowed a better understanding and usage of MAS with MARS for various flight operation stages and offers the first basic foundation for further researches at the University of Applied Sciences Hamburg.

The chapters 2 to 4 offer a solid foundation for a vast majority of people in both disciplines, aviation and computer science. Thereafter the holistic approach of representing a full and implemented concept along the flight operations preflight check to takeoff is presented in chapter 5. With its complete documentation and ideas beyond the current implementation, it offers a scalable foundation for formulating specific questions. With the experience made during the concept development, implementation and further literature, the application potential of MAS with MARS has been evaluated. It is explained how e.g. taxiing and airport ground control would highly benefit from MAS and how this should be further examined. This leads to the final conclusion that rather then trying a holistic approach of representing all flight operations, specific problems should be formulated and modeled in more detail. Some further suggestions are given in chapter 6. These suggestions are only partly possible using MARS DSL, since as it is being mentioned the DSL has some limitations and thus further researches should be conducted with standard MARS services.

Nevertheless, within this thesis a fully passive aircraft agent has been developed including aircraft physics and system representation, which can be further used, when the proposed implementation of external flight dynamics model in section 7.2 is postponed. Further on, each flight operation stage is defined within the pilot states and execution of protocols is implemented with sustainable architectures. Additionally, the pilot is able to taxi along given way points with avoiding collision and to take off. A sustainable simplified communication layer is integrated for further enhancement and communication between pilot and ATC. Thus, most parts of this work can further be used for a more detailed modeling of a scenario.

7.2 Outlook

As mentioned previously, it is recommended to limit the scope on the investigated research. Narrow boundaries with implementation in MARS enable quick results.

This scenario should have many agents with a small set of actions. These could even be passive agents like a “light system”. Afterwards the application of this developed architecture or some single features can be checked for reusability.

However, some ideas which emerged during the course of this thesis are documented in the following. As mentioned, figure 5.1 also represents the full architecture concept.

One idea is the implementation of an open source flight dynamics model into MARS for an interface for agents. One open source flight dynamics model is “JBSSim”, which is also used by RWTH Aachen (FlightGear Wiki, 2018). Another alternative is “YASim” (FlightGear Wiki, 2019). Both are used by the open source flight simulator “FlightGear” (https://www.flightgear.org/) and take the geometry of an aircraft as an input to generate flight characteristics (templates also for e.g. Cessna 150 available). Furthermore, they offer a clear interface for external software with e.g. pilot input like throttle, mixture control, flaps etc. and thus also include various system representation. Before using it within a MAS, the interaction and how a cognitive pilot agent interacts with the aircraft needs to be investigated.

Another idea is to make use of an aircraft condition representation via a CAD or PLM software as shown in figure 5.1. Information like flight hours of each part or last maintenance can be included and evaluated for possible system failures. This can be used for the starting point of the simulation.

Further possible research aligned to the actual MARS group traffic researches is the adoption of this existing research on airport ground control handling. This does not necessarily need a sophisticated aircraft physics model rather than a simple trajectory model like the existing developed concept. A possible graph representation of an airport is in figure 5.10.

Other ideas and research questions:

- An application of MAS could also be the investigation of the reduction from a multi-crew cockpit to a single pilot cockpit, as it has been discussed (in the media) for the Boeing 797 and make a first workload assessment
- Detailed modeling of the takeoff and landing procedure and validate crash statistics (for aircraft below 5.7t, https://www.bfu-web.de/)
- ATM or cruise flight performance research with the advantage of the weather layer in MARS
  - E.g. defining air routes and let the pilot fly based on flight performance criteria and weather
  - Self organization of ATM (already existing research clusters at some other universities, compare Molina, Carrasco, and Martin (2014))

Some of these proposed research topics and ideas require experts from both backgrounds, computer science and aeronautical engineering. As developing an agent-based aircraft model from scratch takes a lot of time, the idea to use an external flight dynamics model and make it usable within the MARS environment should be pursued first.

Bibliography

Airbus Group (Feb. 2018). *Big Data: Airbus is mining the wealth of knowledge for aviation*. URL: http://bit.ly/2JPEHyR.

– (2019a). *skywise. The beating heart of aviation*. URL: http://bit.ly/2Nzl7rR.

– (2019b). *Unleashing the potential of data*. URL: http://bit.ly/2rcKJmR.

AirfoilTools (2020). *NACA 2412 (naca2412-il)*. Airfoil Tools. URL: http://bit.ly/2vFOIKF.

Airport Research Center (2020). *CAST Aircraft - Aircraft Traffic and Process Simulation Software*. 2020 Airport Research Center GmbH. URL: http://bit.ly/2PBJKWG.

Altran (2019). *Digital Cabins - 5 Ways it will revolutionize the Passenger Experience*. URL: http://bit.ly/2N9mApX.

Bicharra, Ana Cristina et al. (2013). "Multi-agent simulations for emergency situations in an airport scenario". In: *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* 1.3. ISSN: 2255-2863. URL: http://revistas.usal.es/index.php/2255-2863/article/view/ADCAIJ20121316973.

Bland, Dan (Sept. 2018). *Digital twin tech for improving digital medical records*. Challenge Advisory LLP. URL: http://bit.ly/2pH1Ah3.

Bouarfa, Soufiane (Dec. 2016). "Agent-Based Modelling and Simulation of Safety and Resilience in Air Transportation". PhD thesis. DOI: 10.4233/uuid:b676db6c-ed86-4b42-9940-9b90b94651f1.

Bouarfa, Soufiane, Henk Blom, and Richard Curran (June 2012). *Airport performance modeling using an agent based approach*. Delft University of Technology.

Cessna-150-152-Club (Dec. 2019). *Cessna 150 History*. Cessna 150-152 Club. URL: http://bit.ly/35qLkjP.

Cessna Aircraft Company (1977). "Pilots Operating Handbook - Cessna 150M". In: URL: https://www.cpaviation.com/images/downloads/CESSNA_150_POH.pdf.

Dausch, C. et al. (2019). *Talking with CLAUDI - Using virutal Agents for an immersive Approach to measure Workload and Situation Awareness in Human-in-the-Loop Simulations*. Deutsches Zentrum für Luft und Raumfahrt – Institut für Flugführung.

Delcea, Camelia, Liviu-Adrian Cotfas, and Ramona Paun (2018). *Agent-Based Evaluation of the Airplane Boarding Strategies' Efficiency and Sustainability*. Department of Economic Informatics and Cybernetics, Bucharest University of Economic Studies. DOI: 10.3390/su10061879.

DLR (2020). *Simmod*. Institut für Flugführung. URL: http://bit.ly/38YWwpU.

Dorri, Ali, Salil Kanhere, and Raja Jurdak (Apr. 2018). "Multi-Agent Systems: A survey". In: *IEEE Access*. DOI: 10.1109/ACCESS.2018.2831228.

Engineering ToolBox (2008). *Rolling Resistance*. URL: http://bit.ly/32jXRoK.

FlightGear Wiki (2018). *JSBSim*. http://bit.ly/38cnvx5 and http://jsbsim.sourceforge.net/. Flight Gear Flight Simulator open-source.

– (2019). *YASim*. Flight Gear Flight Simulator open-source. URL: http://bit.ly/2Tn5erm.

Flugsimulator-Vergleich (2019). *Flugsimulator Hamburg - A320*. URL: http://bit.ly/2RyeauT.

Garro, Alfredo et al. (Jan. 2018). *Intelligent Agents: Multi-Agent Systems*. DOI: 10.1016/B978-0-12-809633-8.20328-2.

General Eletric (2019). *Digital Twin Creation*. URL: http://bit.ly/2qh5wVY.

Glake, Daniel et al. (May 2017). *Modeling Through Model Transformation with MARS 2.0*. DOI: 10.22360/springsim.2017.ads.005.

Grüne, Lars (2008). "Numerische Methoden für gewöhnliche Differentialgleichungen (Numerische Mathematik II)". In: URL: http://numerik.mathematik.uni-bayreuth.de/~lgruene/numerik08/skript.pdf.

Heron Yang, Robert Morris and Corina Pasareanu (2018). "Analysing The Effect of Uncertainty in Airport Surface Operations". In: *ISSTA Companion/ECOOP Companion 18*. DOI: 10.1145/3236454.3236510. URL: https://github.com/heronyang/airport-simulation.

Hüning, Christian et al. (Apr. 2014). "MARS - A next-gen multi-agent simulation framework". In: *GI-Workshop Simulation in dem Umwelt- und Geowissenschaften*.

Hüning, Christian et al. (Apr. 2016). *Modeling & Simulation as a Service with the Massive Multi-Agent System MARS*. DOI: 10.22360/SpringSim.2016.ADS.001.

Klesa, J. (2017). "Increase of the Propeller Efficiency by variable RPM". In: URL: https://www.dglr.de/publikationen/2017/450176.pdf.

Kuhn, Thomas (Nov. 2017). *Digitaler Zwilling*. Fraunhofer IESE. URL: http://bit.ly/34n0dmz.

Lars Braubach, Alexander Pokahr and Winfried Lamersdorf (2008). "A Universal Criteria Catalog for Evaluation of Heterogeneous Agent Development Artifacts". In: URL: http://bit.ly/2TbXWHS.

Lautrup, B. (2005). *Physics of Continuous Matter - Exotic and Everyday Phenomena in the Macroscopic World*. ISBN: 0-7503-0752-8. Bristol and Philadelphia: Institude of Physics Publishing Ltd.

Lednicer, David (2020). *Airfoils of US and Canadian Aircraft*. K.O. Eckland. URL: http://bit.ly/2SMc6OQ.

Lenfers, Ulfia et al. (2015). "A Benefit for KNP-Ecologists – Massive Multi-Agent Simulation with MARS". In: *Poster at Savanna Science Network Meeting, Kruger NP, South Africa*.

Lenfers, Ulfia A., Julius Weyl, and Thomas Clemen (2018). *Firewood Collection in South Africa: Adaptive Behavior in Social-Ecological Models*. Vol. 7. 3. Multidisciplinary Digital Publishing Institute. DOI: 10.3390/land7030097.

Liedman, Per (2020). *geojson-path-finder*. Github repository. URL: https://github.com/perliedman/geojson-path-finder.

Lufthansa Technik AG (2019). *AVIATAR - Digital power platform*. URL: https://www.aviatar.com/ (visited on 11/01/2019).

Marr, Bernard (Apr. 2019). *7 Amazing Examples of Digital Twin Technology In Practice*. Forbes. URL: http://bit.ly/2C6GzPJ (visited on 11/03/2019).

MARS-Group (Nov. 2019). *Multi-Agent Modeling with MARS*. Version 1.7.2. URL: https://mars-group.org/modeling-handbook/.

MARS Group (2020). *SmartOpenHamburg - Traffic Simulation and Analysis with a Digital Twin: the MARS Group on the Roads of Hamburg*. URL: http://bit.ly/2SUHDiK.

Maschinenbau-Community (2009). *Rollwiderstand / Rollreibung berechnen*. Maschinenbau-Wissen. URL: http://www.maschinenbau-wissen.de/skript3/mechanik/kinetik/283-rollreibung (visited on 01/20/2020).

Mattig, Nicole and Jowl Hausweiler (2017). *Digitalization and its Impact on Aviation*. Prologis AG. URL: bit.ly/34mqjpH.

Micciche, Salvatore et al. (Nov. 2013). "An Agent Based Model of the Air Traffic Management". In: *SIDs 2013 - Proceedings of the SESAR Innovation Days*.

Mingtai, Chen (2012). "Static Thrust Measurement for Propeller-driven Light Aircraft". In: URL: https://download.atlantis-press.com/article/2629.pdf.

Miskinis, Carlos (Jan. 2018). *Practical examples on how digital twin can be used*. Challenge Advisory LLP. URL: http://bit.ly/2C7tndk.

Molina, Martin, Sergio Carrasco, and Jorge Martin (2014). "Agent-Based Modeling and Simulation for the Design of the Future European Air Traffic Management System: The Experience of CASSIOPEIA". In: *Highlights of Practical Applications of Heterogeneous Multi-Agent Systems* 430. ISBN: 978-3-319-07767-3. DOI: 10.1007/978-3-319-07767-3_3.

Movable Type Scripts (2020). *Calculate distance, bearing and more between Latitude/Longitude points*. URL: http://bit.ly/3bHP1Fz.

OpenWeather (2020). *Weather maps 2.0*. URL: http://bit.ly/2VuvxPd.

Parrott, Aaron and Lane Warshaw (May 2017). *Industry 4.0 and the digital twin - Manufacturing meets its match*. Deloitte. URL: http://bit.ly/2NfBNpi.

Pieniazek, Jacek (2019). "Control and monitoring assitant for pilot". In: *Aircraft Engineering and Aerospace Technology* 91.5, pp. 783–789. DOI: 10.1108/AEAT-01-2018-0012.

Prigent, Sylvain (Sept. 2015). "Innovative and integrated approach for environmentally efficient aircraft design and operations". PhD thesis. DOI: 10.13140/RG.2.2.22230.34884.

Rüchardt, Dominik (Oct. 2019). *Was ist ein „Digital Thread"?* German. PTC. URL: http://bit.ly/36xDqX2.

Rolls Royce (2019). *IntelligentEngine - Our vision for the future*. URL: http://bit.ly/2NgOdyY.

Russell, Stuart Jonathan and Peter Norvig (2003). *Artificial Intelligence: A Modern Approach*. 2nd ed. ISBN: 0137903952. Upper Saddle River, New Jersey: Pearson Education.

Sadraey, Mohammad (2017). *Aircraft Performance : An Engineering Approach*. ISBN: 978-1-498-77655-4. CRC Press Taylor & Francis Group.

Scheiderer, Joachim (2008). *Angewandte Flugleistung - Eine Einführung in die operationelle Flugleistung vom Start bis zur Landung*. ISBN: 978-3-540-72722-4. Springer-Verlag Berlin Heidelberg. DOI: 10.1007/978-3-540-72724-8.

Scheurle, Klaus-Dieter (2019). *A step towards a digital revolution in air traffic control*. Deutsche Flugsicherung (DFS). URL: http://bit.ly/2NuwwsN.

Schulte, Klaus (2007). "Der Propeller – das unverstandene Wesen". In: URL: http://bit.ly/2P0JVKN.

Singh, Sarwant (June 2019). *Airline Digital Transformation Takes Flight*. Forbes. URL: http://bit.ly/2N6S6EY (visited on 11/01/2019).

Software Education (Jan. 2018). *Continuous Engineering with Digital Twin*. Online Video. URL: https://www.youtube.com/watch?v=Ri0TD7kYsIQ&t= (visited on 11/03/2019).

Stroes, Jeroen (Jan. 2019). *JetPhotos Cessna 150M*. JetPhotos. URL: http://bit.ly/36Id4B4.

Tanajura, Ana Paula M., Valdir Leanderson C. Oliveira, and Herman Lepikson (2015). *A Multi-agent Approach for Production Management*. ISBN: 978-3-662-47200-2. DOI: 10.1007/978-3-662-47200-2_8.

Thilmany, Jean (Sept. 2017). *Identical Twins*. ASME - The American Society of Mechanical Engineers. URL: http://bit.ly/32bZZwQ.

Thom, Trevor (1987). *The Air Pilots Manual 1 - Flying Training*. ISBN: 1-85310-014-5. Airlife Publishing Ltd.

Torenbeek, Egbert (1982). *Synthesis of subsonic airplane design*. ISBN: 90-247-2724-3. Delft University Press.

Uhrmacher, Adelinde M. and Danny Weyns (2009). *Multi-Agent Systems Simulation and Applications*. ISBN: 978-1-4200-7023-1. Taylor and Francis Group LLC.

Wendenburg, Michael (Nov. 2017). *Digitaler Zwilling vor dem Rollout bei Airbus*. German. TeDo Verlag GmbH. URL: http://bit.ly/2WyOlN8.

Wikpedia (2019). *Digital Twin*. URL: http://bit.ly/2pBfAsL.

Wooldridge, Michael (2002). *An Introduction to MultiAgent Systems*. ISBN-13: 978-0-471-49691-5. John Wiley & Sons LTD.

Worobel, Rose and Millard Mayo (1971). "Advanced General Aviation Propeller Study". In: URL: https://ntrs.nasa.gov/search.jsp?R=19710025730%20.

Young, Trevor (2001). *Lecture Notes Flight Mechanics*. 6. University of Limerick, Department of Mechanical and Aeronautical Engineering.

Appendix A

Running Simulations Guide

In this appendix a guide on how to setup, run and analyze the simulation results is presented.

Setup

First of all it's recommended to download the "MARS Handbook" (see MARS-Group (2019)) from https://mars-group.org/modeling-handbook/ for a full setup and further reference. In this appendix it's only described what is needed to run the simulation program of this thesis without working on it.

1. For running the simulation install Dotnet Core - SDK (Version 2.2.402) from https://dotnet.microsoft.com/download/dotnet-core/2.2.
2. For accessing this thesis simulation online go to https://github.com/Erikx3/cessna_digital_twin_multi_agent and clone / download the files to your local device.
3. Optional: For visualizing the results with the provided jupyter notebook it's recommended to install the anaconda distribution with Python 3.x from https://www.anaconda.com/distribution/. All necessary packages are included within the Anaconda distribution.

Simulation Settings and Parameters

The option is provided to customize the simulation via external variable input. Please read this section carefully when changing any parameters.

In the *src-gen* folder are the following three files:

- *config.json* : Main config for global parameters, but also for weather and observer variables
- *Aircraft_variables.csv* : Config file for aircraft agents
- *Pilot_variables.csv* : Config file for pilot agents

The existing files are set up for a simulation and analysis with jupyter notebook afterwards. For an analysis with kepler, use the provided input files in the folder *config_kepler_simulation_example* and copy them into the *src-gen* folder.

Please note:

- For input values and output results the decimal point is always in use
- During this work were an existing small bug in the output format of the position results, which lead to the workaround of changing delimiter between a comma (for kepler) and semicolon (for jupyter notebook)

config.json

In the config.json following global parameters within the key “globals” can be changed:

```
"globals": {
        "startTime": "2020-01-01T18:00:00.000Z",
        "endTime": "2020-01-01T18:15:00.000Z",
        "options": {
            "delimiter": ";"
        }
    }
```

LISTING A.1: config.json global parameters

With “startTime” and “endTime” it’s possible to vary the internal time of the simulation. With the delimiter the output csv file format can be adapted. Choose between a comma for kepler or a semicolon for jupyter notebook analysis later on.

Also the input variables for the weather and observer agent can be defined here. Please refer to the last section for the variables definitions.

```
{
    "name":"Weather",
    "count":1,
    "mapping": [
        {
            "parameter" : "wind_bearing",
            "value" : 107.0
        },
        {
            "parameter" : "wind_speed",
            "value" : 5.0
        },
    (...)
    ]
},
```

LISTING A.2: config.json weather agent

For the following sections it’s already important to mention, that the value for the key “count” in listing A.3 for the pilot agent and aircraft in the *config.json* should align with the number of defined agents in the *Aircraft_variables.csv* and *Pilot_variables.csv* later.

```
"agents": [
    {
        "name":"Pilot",
        "count":2,
        "file" : "Pilot_variables.csv"
    },
    {
        "name":"Aircraft",
```

```
            "count":2,
            "file" : "Aircraft_variables.csv"
        },
        (...)
]
```

LISTING A.3: config.json aircraft and pilot agent

Pilot_variables.csv and Aircraft_variables.csv

Here the input values for the pilot and the aircraft can be changed. All important variable descriptions can be found in the last section. Nevertheless, here are some helpful comments:

- Each row after the column row represents ONE agent
- Make sure that the “callsign_number” in each csv has a matching partner
- Choose a reasonable high delta for “time_initialization” between landing aircraft
- The number of agents defined in the csv files should match with the number defined in the *config.json* as mentioned earlier
- Watch out for whitespaces and remove them

Run and Analyze the Simulation

Now simply run the simulation with the *run.sh* shell script from the *src-gen* folder. The output will be a *Aircraft.csv* and *Pilot.csv* file. There are two ways to visualize the results.

Kepler

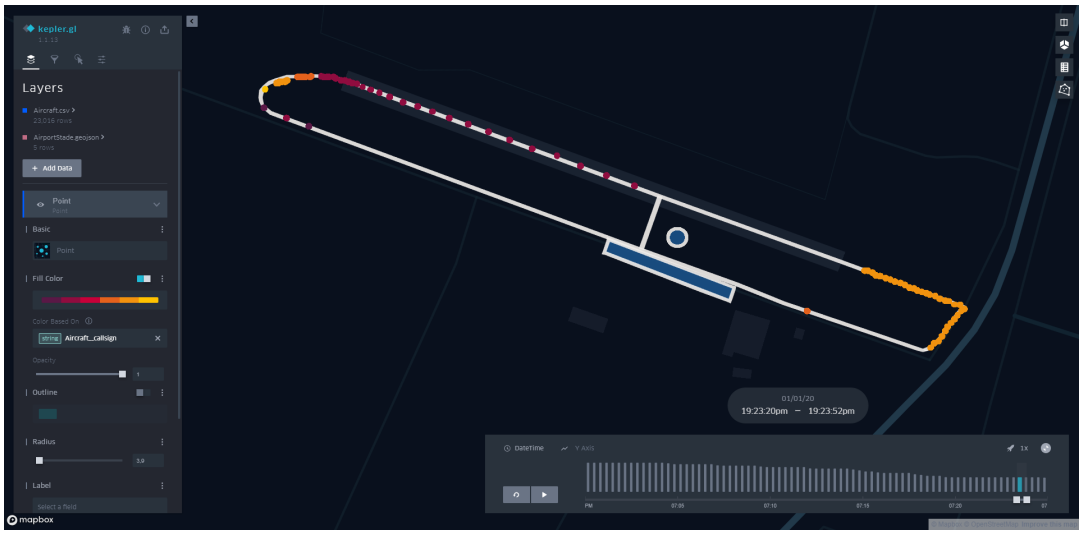

FIGURE A.1: Visualization of results with kepler

Visit `https://kepler.gl/` and upload for instance the *Aircraft.csv*. To get a nice visualization over the internal simulation time as in figure A.1 follow these steps:

1. Go to the options for your “Points”, then open the options for “Fill Color” and select for “Color Based On” the column name “Aircraft__callsign”
2. Optional: Reduce Radius of each point and add a label
3. Go to next tab and choose as a filter the column name “DateTime”, which is already in a format that kepler will provide you a time slider
4. Optional: Upload the *AiportStade.geojson* file from the *src-gen* folder for visualizing the airport paths

Kepler offers even more possibilities for exploring the data, thus only the basics are explained here.

Jupyter Notebook

With the jupyter notebook script *Data Visualization.ipynb* each signal for each aircraft will be plotted. Simply run the script with your jupyter notebook environment. It’s recommended to have a maximum of two to three agents for a good overview of each signal. This script allows a in depth analysis of each action and offers a lot of help during development.

Variable Overview

The following four tables represent all variables, that are able to be customized within each agent. Please note that sometimes the defined column “Value Range” are only reasonable values, which partially are further discussed in this thesis. Also note that some variable names are self explanatory.

Variable Name	Type	Unit	Value Range	Description
gravity	real	m/s^2	[9.81]	Gravitational acceleration
pressure_QNH	real	Pa	[95000; 105000]	QNH, airport ambient pressure since elevation is 0 m
temperature	real	C°	[-10; 50]	Outside air temperature at airport
wind_bearing	real	°	[0; 360]	Bearing the wind is coming from.
wind_speed	real	m/s	[0; 15]	-

TABLE A.1: Weather Agent external variables

Variable Name	Type	Unit	Value Range	Description
number_of_ spawning_points	integer	-	[1; 8]	Number of spawning points on the apron
print_interval	integer	s	[100, inf]	Interval at which the observer prints an overview of the states of all pilots in the command line window

TABLE A.2: Observer Agent external variables

Variable Name	Type	Unit	Value Range	Description
callsign_number	integer	-	[1; inf]	Call sign number, that has to match a call sign number of the aircraft variables table
Check_Engine__oil_ skip_probability	real	-	[0; 1]	-
Check_Instrument_ Engine__oil_ pressure_skip_ probability	real	-	[0; 1]	-
Check_Instrument_ Engine__oil_ temperature_skip_ probability	real	-	[0; 1]	-
Check_LWT__water_ sediments_skip_ probability	real	-	[0; 1]	-
Check_RWT__water_ sediments_skip_ probability	real	-	[0; 1]	-
Climb__pitch_ magnitude	real	◦	[0.1; 1.5]	Pitch magnitude changes applied during climb state
Climb__rate_of_climb_ lower_limit	real	m/s	[0; 3]	Rate of climb control during climb state
Climb__rate_of_climb_ upper_limit	real	m/s	[1; 4]	Rate of climb control during climb state
Set_Engine__mixture_ control_skip_ probability	real	-	[0; 1]	-

Variable Name	Type	Unit	Value Range	Description
TakeOff__rotate_pitch	integer	°	[8; 14]	Aircraft pitch applied after rotating
TakeOff__V_rotate	real	m/s	[25; 35]	Aircraft TAS where pilot rotates aircraft during takeoff
Taxiing__angle_of_vision	real	°	[30; 90]	Angle of vision during taxi. Note: too high or too low values might lead to issues.
Taxiing__collision_crash_distance	real	m	[1; 3]	Distance threshold for crash detecion
Taxiing__final_point_distance	integer	m	[0; 20]	Distance to final point, when pilot applies brake
Taxiing__next_point_distance	integer	m	[0; 20]	Distance threshold to next point on path, when switiching to next point after that one
Taxiing__spacing_to_next_aircraft	real	m	[20; 50]	Distance kept to next aircraft while taxiing.
Taxiing__speed	real	m/s	[4; 8]	Speed during taxi. Note: too high speeds might lead to collisions and overshooting points
time_base_next_communication_attempt	integer	s	[10; inf]	Base time for waiting to make another attempt to communicate to tower after getting no answer
time_extra_next_communication_attempt	integer	s	[10; inf]	Variabel extra time for waiting to make another attempt to communicate to tower after getting no answer
time_initialization	integer	s	[2; inf]	Defines the starting time of the pilot AND its aircraft. Please do NOT use the same time twice, since spawning might be unstable. Please choose a reasonable high delta between landing aircraft.
type_initialization	string	-	[Landing, Takeoff]	Defines whether this pilot is going to be landing or starting from ground

TABLE A.3: Pilot Agent external variables

Variable Name	Type	Unit	Value Range	Description
Aircraft__lift_ coefficient_slope	real	-	[4.9]	-
Aircraft__mass	real	kg	[550; 725]	-
Aircraft__oswald_factor	real	-	[0.7]	-
Aircraft__stall_angle	real	◦	[11]	-
Aircraft__total_stall_ angle	real	◦	[15]	Defines the angle when complete flow separation occurs and no lift will be produced
Aircraft__wing_area	real	m^2	[15]	-
Aircraft__wing_span	real	m	[10.1]	-
Aircraft__zero_lift_ angle	real	◦	[-1.0]	-
Aircraft__zero_lift_ drag_coefficient	real	-	[0.04]	Typical value for small single engine aircraft, fixed gear
Brake__deceleration_ force_max	real	N	[2500; 3000]	Simplified maximum brake force
callsign_number	integer	-	[1; inf]	Call sign number, that has to match a call sign number of the pilot variables table
Engine__failure_ probability_add_ oil_critical_min	real	-	[0; 1]	Probability of an engine failure (per each tick!) due to engine oil below critical minimum
Engine__failure_ probability_add_oil_ min	real	-	[0; 1]	Probability of an engine failure (per each tick!) due to enigne oil below minimum
Engine__failure_ probability_add_oil_ pump_condition	real	-	[0; 1]	Probability of an engine failure (per each tick!) due to faulty engine pump condition
Engine__failure_ probability_add_ water_sediments	real	-	[0; 1]	Probability of an engine failure (per each tick!) due to water sediments
Engine__fuel_ consumption_max	real	l/s	[0.0079]	Fuel consumption for 100% throttle (corresponds to 7.5 gallon per hour)
Engine__oil_leakage_ probability	real	-	[0; 1]	Probability of an engine oil leakage during initialization

Variable Name	Type	Unit	Value Range	Description
Engine__oil_pump_ condition_probability	real	-	[0; 1]	Probability of oil pump condition to be healthy during initialization
Engine__power_ coefficient_constant	real	-	[0.05]	-
Engine__power_ coefficient_slope	real	-	[-0.0009]	-
Engine__power_ coefficient_speed_ constant	real	m/s	[30.0]	Up to which speed the "Engine__power_coefficient_constant" is constant
Engine__power_max	integer	W	[75000]	-
Engine__RPM_max	integer	RPM	[2750]	Maximum mechanical revolution per minute
LWT__total_capacity	integer	liter	[49]	Light wing fuel tank capacity
LWT__water_ sediments_probability	real	-	[0; 1]	Probability of existing water sediments in the left wing tank during initialization
n_cycle	integer	-	[1; 100]	Defines the number of calculation cycles between each tick for the numerical calculation of the aircraft physics (low numbers might lead to unstable behaviour)
Propeller__diameter	real	m	[1.75]	-
Propeller__thrust_ coefficient_constant	real	-	[0.1]	-
Propeller__thrust_ coefficient_slope	real	-	[-0.002]	-
Propeller__thrust_ coefficient_speed_ constant	real	m/s	[25]	Up to which speed the "Propeller__thrust_coefficient_constant" is constant
RWT__total_capacity	integer	liter	[49]	Right wing fuel tank capacity
RWT__water_ sediments_probability	real	-	[0; 1]	Probability of existing water sediments in the right wing tank during initialization
Tire__roll_coefficient	real	m	[0.003]	For the calculation of the friction
Tire__wheel_radius	real	m	[0.075]	Wheel diameter with neglecting non equal size of nose and main wheels

TABLE A.4: Aircraft Agent external variables

Appendix B

Feature Overview

In this appendix an overview of all features is given. The table on the following pages are summarizing each feature and link the according chapter. Furthermore it shows also all possible and next features, that are not implemented yet. Nevertheless, it's recommended to use these basic features and basic architecture for a specific formulation of a scientific question like e.g. safety of takeoff procedure with small aircraft as presented during section 6.2. Afterwards some features can be picked for further implementation. Thus, the upcoming table is intended as an overview and reference for future work.

The rest of this page intentionally left blank

Feature	Type	Description	Done	Comment
Communication Sytem	Communication Architecture	Implement a communication system as a blackboard.	Yes	View section 5.5 for implemented communication architecture and how agents use this system
Weather access	Environment Architecture	Implement weather and gain access to data like wind, density etc.	Yes	Weather agent is chosen for accessing weather data and enable external variable input, see section 5.6.1
ISA Atmosphere	Environment Calculation	Add ISA Atmosphere variations and calculations for different environments	Yes	Weather agent offers calculation of density for any given height with input of temperature and pressure QNH
Airport information access	Environment Architecture	Add a convenient way for retrieving airport relevant information like waypoints, runway bearing etc.	Yes	Class for information is created, other possibilities are discussed in section 5.6.2
Weather vector layer	Environment Architecture	Add a weather vector layer, for time and location dependency in weather data	No	First ideas attempted and sketched in 5.6.1
Spawn and Kill Management	Architecture	Add a spawn and kill management for agents, especially spawn location	Yes	Solved with a combination of external variables, second initialization and an observer agent, see section 5.7
Adding landing aircraft	Agent/ Architecture	Adding landing aircraft for visualizing airport operations and as a bonus	Yes	Simplified landing aircraft with ATC handling takeoff and landing clearance, see section 5.7
Collision removal	Architecture	Remove aircraft and pilots when colliding	Yes	Actually implemented within the pilot, not fully working due too high tick size and low collision radius

Feature	Type	Description	Done	Comment
External Variables	Architecture / Agent	Offer the possibility to change the variables in csv. files, so other third partys are able to interact and change some data	Yes	See guideline in appendix A
Time handler	Architecture	Add a timehandler class for giving actions a time duration	Yes	Class implemented and callable for ATC and pilot
Smaller tick size	Architecture	Offer the possibility for variabel ticksize	No	Needs some refactoring, not available yet
Other Event Ideas	Event	Implement following events: High Crosswind, Animal on Runway, Cabin- Engine or Wing fire, Ice problems in engine or on wing, Canceled Take-Off Clearance by ATC	No	-
Airport flexibility	Architecture	Offer the possibility to change the provided GeoJson of an Airport and still running the simulation (based on a graph system and json information?)	No	-
State preflight check to take-off	Pilot Agent	Add "PreflightInspection", "StartingEngine", "TakeOffPreparationRequest", "Taxiing", "TakeOffPreparation", TakeOffHoldShortRequest", "TakeOffRequest", "TakeOff", "Climb" and "LeavingFrequencyRequest" state	Yes	States are structured added in tick method, see section 5.7
State Landing	Pilot Agent	Add "LandingRequest", "GoAround", "Landing" and "LeavingFrequency" state	Yes	States for landing are structured added in tick method, see section 5.7
Age and flight experience	Pilot Agent	Add age and flight experience and make some decisions dependend on these variables	Yes	Simplified added age and flight experience and made time needed and actions depending on it, see section 5.2

Feature	Type	Description	Done	Comment
Skipping actions	Pilot Agent	Possibility to skip control and repair action	Yes	For some engine related actions during e.g. "PreflightInspection" added
Abort takeoff mission	Pilot Agent	Kill me sequence when finding a mistake	Yes	Added for engine inspections or engine failure
Taxi along a path	Pilot Agent	Taxi along the path recieved from the ATC	Yes	Pilot is able to command and calculate heading and thrust. The thrust and thus speed handling works like a simple control system, see section 5.7
Collision avoidance during taxiing	Pilot Agent	Pilot should be able to see other aircraft and apply brakes when nearby	Yes	Pilot has a cone vision, collision system is documented in section 5.7
Takeoff and climb	Pilot Agent	The pilot is able to take off and control the aircraft speed and altitude	Yes	Rotate at a rotate speed and controls aircraft as a simple control system , see section 5.7
Aircraft action representation	Pilot Agent	Have the functions available for the pilot within the agent, so pilot based traits can influenced these results	Yes	Digital representation of aircraft passive function are always active in pilot (but without many influences so far), see section 5.2
Feeling and hearing	Pilot Agent	Being able to feel and hear things	Partly	Done for acceleration feeling (for easier taxiing) and stall speed acoustic so far
Engine failure procedure	Pilot Agent	Add engine failure procedure while taxiing or during takeoff	No	Very complicated and should be focused on one state like e.g. takeoff
Collision conflict handling	Pilot Agent	Add a handling for pilot, when e.g. two pilots moving onto each other or similar	No	-

Feature	Type	Description	Done	Comment
Realistic Instrument Screening	Pilot Agent	Implement a realistic instrument screening generator during taxiing and takeoff	No	-
Autonomous ground operation	Pilot Agent	Steering on the airport with exploring on its own without ATC defining way points, thinking about graph based system	No	-
Pilot human character traits	Pilot Agent	The pilot should have some characteristics of a human being and these should be implemented for various behavior of the flight operations	No	-
Aircraft control during taxi and takeoff	Pilot Agent	Implement a intelligent way to control and manoeuvre the aircraft during taxi and takeoff	No	Try approaches with e.g. neural networks
Cognitive Control	Pilot Agent	Based on belief-desire-intention model or similar approaches the agent should be able to detect from a set of actions, which needs to be the next action to achieve his goals	No	see section 5.2 for a first introduction
Aerodynamic system	Aircraft Agent	Update of drag and lift (and other aerodynamic values) and thus vertical and and horizontal acceleration and speed	Yes	Done for two dimensional movement, see section 5.3
Brake system	Aircraft Agent	Possibility of applying brakes	Yes	Simple solution with maximum brake force and percentual input, see section 5.3.2
Engine system	Aircraft Agent	Engine system with e.g. engine oil and its temperature, mixture control etc., also calculation of RPM	Yes	Simplified system representation, see chapter 5.3.2 for all functionality

Feature	Type	Description	Done	Comment
Propeller system	Aircraft Agent	Calculation of thrust	Yes	implemented as in chapter 4.1.4
Tire system	Aircraft Agent	Calculation of friction, flat tire possibility and effect on brake and friction force	Partly	Only calculation of friction added so far, see section 5.3.3
Fuel system	Aircraft Agent	Add fuel consumption and fuel tank capacity	Yes	-
Instrument panel	Aircraft Agent	Add a instrument panel for all action that allow the pilot to read values from the aircraft	Yes	Done for some real instrument in simplified form and with naming scheme as shown in figure 5.4
Control input panel	Aircraft Agent	Add a control input panel for all actions that allow the pilot to execute inputs like e.g. thrust, pitch, turn engine switch on	Yes	Done for various inputs and with naming scheme as shown in figure 5.4
Engine failure causes	Aircraft Agent	Add conditions and events for increasing engine failure probability like engine oil leakage and thus minimum oil, water sediments in fuel tanks or oil pump condition,	Yes	These root causes are implemented and with combination of pilot not repairing or overlooking these lead to increase of engine failure probability, see section 5.7
Engine failure	Aircraft Event	Add engine failure to possible events while engine is running	Yes	Add a probability per tick, depending on condition of engine and preflight checks
Tire inflation failure	Aircraft Event	Add flat tire as failure possibility (Brake power loss, friction increase ..)	No	-
Engine power dependency	Aircraft Calculation	Add engine power dependency on e.g. mixture control, weather, carburetor heat, low engine oil etc.	No	Implemented only with mixture control so far

Feature	Type	Description	Done	Comment
Flaps system	Aircraft Agent	Add flaps system and its influence on aerodynamics	No	-
Realistic altitude measurment	Aircraft Calculation	Display the real measurement of altitude with a altimeter and a reference pressure	No	-
Realistic speed measurement	Aircraft Calculation	Display the indicated airspeed derived from dynamic pressure	No	-
3-dimensional flight physics	Aircraft Agent	Add realistic three dimensional aircraft physics with the possibility to roll and yaw the aircraft	No	-
State communication and listening	ATC Agent	Adding the states communication and listening stated as both are the main action of th ATC	Yes	Similar to pilot, but only having these two states and with being only passive-active (see chapter 5.4)
Determine Runway Heading	ATC Calculation	The ATC should be able to calculate the runway heading and decide to which point of the runway the pilot has to go for the takeoff	Yes	Done with simple wind directions calculations
Retrieve and pass taxipoints	ATC Agent	Retrieving the waypoints and communicating it to the pilot when requested	Yes	-
Takeoff and landing clearance	ATC Agent	Derive whether and takeoff or landing clearance can be given	Yes	Check for other landing or starting aircraft before giving clearance, see section 5.7

Feature	Type	Description	Done	Comment
ATC human character traits	ATC Agent	Add character traits that influence the decisions of the ATC	No	-
Active communication	ATC Agent	Add active requests and handling of airport operations	No	-

TABLE B.1: Feature Overview

Appendix C

CD Content Overview

The following overview describes the content of the enclosed CD with the same order:

1. Folder “01_cessna_digital_twin”: Contains the MARS simulation program, as it can be found at https://github.com/Erikx3/cessna_digital_twin_multi_agent
    - Folder “config_jupyter_notebook_example”: Include config files for simulation and analyzing with jupyter notebook afterwards
    - Folder “config_kepler_notebook_example”: Include config files for simulation and analyzing with kepler afterwards
    - Folder “src”: MARS soruce code
    - Folder “src-gen”: Generated code with config files
    - Jupyter Notebook Script “Data Visualization.ipynb”: For visualization after simulation
    - Excel File “Variable_Overview.xlsx”: Overview of all variables in the config files
2. Folder “02_Figures”: Contains all figures of the thesis
3. Folder “03_Literature”: Contains all papers and thesis used in this thesis
4. Video Cessna_Digital_Twin_Simulation_Animation.mp4: Video of the simulation animation with kepler
5. Digital Bachelor Thesis PDF “Cessna_Digital_Twin.pdf”
6. Additional Abstract PDF “Cessna_Digital_Twin_Abstract.pdf”

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Suer

Vorname: Erik

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Concept Development of a Cessna 150 Digital Twin using Multi-Agent-Systems

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Hamburg	06.03.2020	
Ort	Datum	Unterschrift im Original