UCAVs Against Air Threats

Pre-feasibility Study for a Semi-Direct Control System

Graduation project report submitted in partial fulfillment of the requirements to obtain the degree of

Bachelor of Science
in
Creative Technology

Submitted by Fabian Benjamin Dijkstra

Under the guidance of

UNIVERSITY OF TWENTE.

Dr. ir. Jan Joris Roessingh
Drs. Gerald Poppinga

Dr. Alma Schaafstal
Dr. Hans Heerkens

Faculty of Electrical Engineering, Mathematics and Computer Science

University of Twente
Enschede, the Netherlands

20-06-2017

Abstract

The constraints of Unmanned Combat Aerial Vehicles (UCAVs) utilisation partially reside in the dependency on wireless communication protocols which may suffer from bandwidth limitation, latency and frailness. In addition, the geographical separation between operator and UCAV can cause the operators situational awareness to drastically decrease. These deficiencies restrict the effectiveness of UCAVs especially in engagements with adversaries. From this, the claim has evolved that UCAVs at present, are insufficiently survivable and effective in air-to-air combat situations. A potential solution overcoming the vulnerability of UCAVs in air-to-air combat scenarios is provided by Hans Heerkens and Frank Tempelman. The solution is being referred to as a Semi-Direct Control System (SDCS). The philosophy behind the SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with the SDCS directly, but instead sends commands with varying intervals. These commands provide waypoints or goals for the UCAV to achieve. On board "Intelligence" determines how to fulfil the command it has received from the operator. The SDCS is placed on the verge between direct joystick control and full autonomous control and aims at eliminating the drawbacks of these "classic" control options. This thesis handles the development of a demonstrational and developmental platform to act as a tool to perform research into the feasibility of the SDCS and to demonstrate its functioning. The prototype (MVP) developed in this thesis, is created from the open source flight simulator FlightGear, the programming language Python and by leveraging Ubuntu operating system functionality. Several functionalities require further development to successfully demonstrate the added value of the SDCS; such as the "intelligent" flight controller and the user interface. The current possibilities of commands that can be given to the UCAV are too limited to successfully perform UCAV related tasks. The MVP can function as a development platform for further research; The choice of open source software and the open and accessible architecture of FlightGear has strongly contributed to the MVP being future proof. In addition, the open source community may provide further functionality. Data link latency can be simulated with aid of the MVP which allows it to demonstrate the issues related with latency in direct joystick control of UCAVs and furthermore function as a human factors research platform. The MVP can be utilised as a research tool that can contribute to knowledge in the field of autonomy and human factors research in Remotely Piloted Aircraft Systems.

Keywords: Unmanned Combat Aerial Vehicles, UCAV, Semi-Direct Control System, SDCS Latency, Simulation, FlightGear, Python, Ground Control Station

Acknowledgments

First of all I would like to thank Hans Heerkens and Richard Bults for providing me with the opportunity to perform my research at the NLR. You have put in your efforts to realise my wish of performing the research that I wanted to perform.

I would furthermore like to thank Alma Schaafstal, again Hans Heerkens, Gerald Poppinga and Jan Joris Roessingh for supervising me during the past 4 and a half months. Your constant encouragement and guidance along with your expertise on diverse fields have allowed me to greatly improve the quality of my work on all aspects and have helped me to personally develop myself into a somewhat more confident young researcher.

I would also thank those who may have not had a direct impact on this thesis, but have greatly contributed to where I am now. Thank you, Erik Faber, for taking on a mentor role towards me during my studies and providing me with help on all aspects of my study and student life. Your door was always open if I had questions.

I also want to thank the Creative Technology Staff and Technician, Alfred de Vries, for being engaged and always willing to help and provide me with everything I needed to fulfil my study and even helping me beyond my studies. A warm thank you towards the Creative Technology community for providing me with feedback and advice during my studies and the writing of my thesis.

Furthermore I would like to thank my parents for their unconditional support and encouragement. My girlfriend, Kim, for taking me in your home and for providing me your unlimited support, encouragement and comfort. My roommates from 't Schijtpaleis for their understanding and support in times of my absence.

Without all of you I would not have been able to deliver this thesis.

The happiest and most successful people do not necessarily have the best of everything, but make the best of everything they have

Table of Contents

List of Figures

List of Tables

List of Acronyms

UAV	Unmanned Aerial Vehicles
TUAV	Tactical Unmanned Aerial Vehicles
UAS	Unmanned Aircraft System
RPAS	Remotely Piloted Aircraft System
UCAV	Unmanned Combat Aerial Vehicle
GCS	Ground Control Station
UT	University of Twente
NLR	Netherlands Aerospace Center
SDCS	Semi-Direct Control System
CMMS	Combat Manoeuvring Management System
MVP	Minimum Viable Product
IFF	Identify Friend or Foe
GEO	Geosynchronous Earth Orbit
MEO	Medium Earth Orbit
LEO	Low Earth Orbit
LOS	Line of Sight
ISRT	Intelligence Surveillance Recconaissance Targeting
BLOS	Beyond Line of Sight
GPS	Global Positioning System
ACL	Autonomous Control Level
OODA	Observe Orient Decide Act
HALE	High Altitude Long Endurance
TD	Technology Demonstrator
HMI	Human Machine Interface
COTS	Commercial Off the Shelf
ADP	Approximate Dynamic Programming
RC	Radio Controlled
SDK	Software Developers Kit
GCF	Computer Generated Forces
ATM	Air Traffic Management
ATC	Air Traffic Control
VR	Virtual Reality
BCI	Brain Computer Interface
EEG	Electroencephalogram
FDM	Flight Dynamics Model
GUI	Graphical User Interface
TCP	Tranmission Control Protocol

UPD	User Datagram Protocol
LAN	Local Area Network
FSM	Finite State Machine
PID	Proportional Integral Derivative
OOP	Object Oriented Programming
DS	Dynamic Scripting
GIL	Global Interpreter Lock
PFD	Primary Flight Display
PIO	Pilot Induced Oscillation

1 Introduction

1.1 Problem Introduction

1.1.1 Nomenclature

The nomenclature regarding unmanned aircraft are eminently diverse. Frequently occurring terms are Unmanned Aerial Vehicles (UAVs) or the trivial designation "Drone". Military instances refer to unmanned aircraft as Tactical UAVs (TUAVs) or Unmanned Aircraft System (UAS) and Remotely Piloted Aircraft System (RPAS):, where the latter denotes a broader aspect of unmanned systems and includes the accompanying important aspects of unmanned control: the Ground Control Station (GCS) in which the operator of the UAV resides, the datalink over which communication between the GCS and the UCAV takes place and other equipment. In this thesis, the term Unmanned Combat Aerial Vehicle (UCAV) will be adopted to designate the variety of remotely piloted, operated and/or monitored military combat aircraft.

1.1.2 Problem Background

At present, UAVs are widely used as alternative to manned aircraft in situations where they show the potential to outperform manned aircraft in terms of endurance, range and performance [1]. Unmanned aircraft can operate in extreme situations where there is high risk of loss of life of pilots [2] and where manoeuvres are required that excite G-forces that extend beyond the human incurring limits. More advantages of the absence of humans inside the aircraft reside in possible reduced weight, more streamlined designs of UAVs and extended operation times [3].

The constraints of UCAV and UAV utilisation partially reside in the dependency on wireless communication protocols for transceiving information between the operator and the UCAV. The efficiency and reliability of these protocols may be affected by bandwidth limitations and latency [4]. In addition, due to the fact that there is a large geographical separation between the operator and the UCAV, the operators of a broad range of UCAVs are necessitated to work with the occasionally limited information supply regarding the UCAVs environment, received from the sensors on board the UCAV in flight. Depending on the quality and complexity of the synthesised picture, the pilot's situational awareness may drastically decrease [5].

These deficiencies restrict the effectiveness of UCAVs in both long distance operation but especially in short distance operations and engagements with adversaries. In the latter case, swift responses are indispensable for successful interaction with the environment and adversary in close-combat situations. From this, the claim evolves that UCAVs at present, are insufficiently survivable and effective against manned figher aircraft and other UCAVs in air-to-air combat situations.

1.1.3 The Semi-Direct Control System

A potential solution overcoming the vulnerability of UCAVs in air-to-air combat scenarios is provided in the feasibility study by Hans Heerkens of the University of Twente (UT) and Frank Tempelman of the Netherlands Aerospace Center (NLR) [6]. The solution is being referred to as a Semi-Direct Control

System (SDCS).

The philosophy behind the SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with the SDCS directly, but instead sends commands with certain intervals (varying from 1 to a multitude of seconds). These commands do not control the UCAV directly, but provide waypoints or goals for the UCAV to achieve. For example: fly to a certain direction, fly to a certain position or follow an aircraft, approach an aircraft under a certain angle and so forth. In practice, this implies that the SDCS should possess a Combat Manoeuvring Management System (CMMS) that is equipped with a measure of "intelligence" that determines the most optimal, the most safe or the most effective way to fulfil the command it has received from the operator. The pilot determines the tactics to be used and can continuously modify the flight path and strategy based on the adversary's behaviour. In principle, the operator decides on the tactics and approach and the SDCS performs the actions.

The SDCS is placed on the verge between direct control and indirect control. Direct control, where the operator directly governs the control surfaces of the aircraft in real-time with a joystick is often not a viable option for controlling UCAVs in air to air combat scenarios due to latencies (more about this later). In the case of indirect control, the operator governs the UCAV with intervals of minutes or even hours on a two dimensional map without being able to let the UCAV perform combat manoeuvres. The SDCS aims at eliminating the drawbacks of these "classic" control options.

An UCAV equipped with the SDCS possesses the potential to increase the survivability of this UCAV in close quarter air-to-air combat situations in which a quick reaction time is essential. In a broader sense, the increased survivability of an UCAV contributes to the effectiveness of a broad range of air operations. UCAVs with the SDCS may provide better support to various other disciplines of military organisations.

1.2 Problem Statement

The NLR recognizes and stresses the importance of research in the SDCS and the first steps of conducting this feasibility study are being contemplated. As of today, no tangible results or prototypes of the SDCS exist and it is by far not certain whether the SDCS is a feasible concept. Research needs to be done in the feasibility of the SDCS and ultimately if the research results are promising, the NLR requires funding to initiate the development of the SDCS, as there are limited financial means available at this point in time. Within the UT and the NLR, the need has arisen to create a platform that serves two functions: Firstly, to effectively demonstrate the working principles and key functionality to (potential) stakeholders and financial backers of the development process and secondly, to serve as an environment in which these principles and functionalities can be further developed and tested. The problem statement can therefore be stated as follows:

The NLR does not possess a platform to demonstrate, develop and validate the core functionality of the SDCS, that can be set up and operated within their resource constraints.

1.3 Research Goal

Taking in consideration the problem statement, this thesis will focus on setting up a demonstrator and test platform or "Minimum Viable Product (MVP) of the SDCS".

The limited resources the NLR has available poses the requirements that the MVP should favourably be developed with open source software to allow growth and development from a wide variety of developers and users rather than depending on a company for development. Furthermore, the MVP should be portable (on a laptop) to ensure that demonstration of the SDCS can be conveyed any time. Ideally,

there should be no limitations regarding licensing and start-up costs.

In order to effectuate this implementation it is essential to map the working principles of the SDCS, in which form they need to be implemented and which elements the presentation should include. Furthermore, barriers and opportunities are explored that follow from the environment in which NLR has to operate.

1.4 Research Questions

The research question of this thesis is established through comparing the stated research goals with the problem statement. The research question reflects the problem statement and its answer should satisfy the research goal. The research question is formulated as follows:

What elements of the Semi-Direct Control System (SDCS) must be incorporated in a Minimum Viable Product (MVP) to demonstrate the functionality of the SDCS and to function as an effective development platform?

To answer the research question in a complete and structured manner it is advantageous to divide the main research question into sub research questions. Systematically answering sub research questions composes a solid answer to the research question. Furthermore it acts as a means to ensure that the complete set of relevant aspects regarding the design processes and options have been treated. This method enables precise conclusions and recommendations to be developed which can be subsequently be applied by the stakeholders as to maximise the usability of this thesis.

Sub research question 1 is formulated to acquire insight in the foundation and motivation of the SDCS and is formulated as follows:

What is the Semi-Direct Control System?

Sub research question 2 handles the aspects of the MVP that is being developed. The concept of a MVP should be made clear. Sub research question 2 is formulated as follows:

What is a Minimum Viable Product?

Sub research question 3 handles the motive for developing a MVP. It should provide insight into why this MVP is being developed and what the ought goals are of the MVP. Sub research question 3 is formulated as follows:

What is the goal of the MVP of the SDCS?

Sub research question 4 handles the mapping of functions of the SDCS that need to be demonstrated in the MVP in order to make the MVP effective. Sub research question 4 is formulated as follows:

What are the most important elements of the SDCS to be included in the MVP?

Sub research question 5 handles the implementation methods and realisation of the MVP. A multitude of developing environments can be utilised to create a MVP for the SDCS. The multitude of types of environments need to be explored and assessed. Sub research question 5 is formulated as follows:

What are suitable methods and tools for developing an MVP of the SDCS?

Sub research question 6 handles the realisation of the MVP and is formulated as follows:

How can this be translated into a MVP of the SDCS?

1.5 Research Context

The entire graduation project regarding this thesis will be performed during an internship period at the NLR in Amsterdam at the Aerospace Operations, Training and Simulation (AOTS) department under supervision of Dr. ir. Jan Joris Roessingh and Drs. Gerald Poppinga. Supervision from the University of Twente is provided by Dr. Alma Schaafstal and Dr. Hans Heerkens.

1.6 Research Scope

To ensure the manageability of this research, it is important to provide the scope of this thesis. It makes clear what the intention of the research is and importantly, what this thesis does and does not cover.

- This thesis focuses on developing a piece of software on a computer in which the stakeholders of the SDCS project can demonstrate and develop the functionalities of the SDCS and do further research into the feasibility of the project. This piece of software on the computer will form the MVP.
- This thesis will therefore not answer questions regarding the feasibility of the SDCS. At this point it is still too early to speculate on this. This research contributes to possibly answering these questions with help of the MVP that is being developed.
- The original focus of the SDCS lies in the application of this system in close quarter air to air combat. The SDCS may have spin-offs to different fields but the initial incentive for development of this system arises from close quarter air to air combat. In this thesis, there will therefore be references to close quarter combat, but in the remaining research the focuss will mainly lie on functionality of the SDCS and not yet on the application of these functionalities in combat settings.
- This thesis will also not handle the development of an UCAV or (inventarising) any of the techniques, technologies, hardware or software needed for the development of the SDCS or an UCAV that is equipped with the SDCS.
- The decisions that are made or the opinions that are expressed in this thesis origin mostly from the author, and may not necessarily reflect the point of view from the NLR or the UT.

1.7 Thesis Outline

This thesis proceeds with a chapter on the research methods and -techniques that are utilised in this thesis. Following, is a chapter on the theoretical framework which will cover the state of the art on a multitude of related subjects and will furthermore focus on answering sub research questions. Following this chapter are the chapters Ideation, Specification, Realisation and Evaluation. The names and structure of these chapters follow from a design method that serves as a design guideline in Creative Technology Bachelor theses. This design method will be further clarified in the chapter that covers the methods and techniques. The thesis concludes with a chapter covering conclusions and findings and a chapter on future work.

2 Analysis

This chapter aims at analysing the problem background and introduce a proper research approach to the problems ahead. The introduction chapter provided a brief overview of the problem and the posed solution to this problem that will be researched in this thesis, but this chapter will delve deep into the background of the SDCS and UCAVs in general to provide a solid basis and understanding of the topic and why the SDCS has come to exist. Furthermore, this will also serve as a justification of the research that is executed in this thesis. This chapter continues with a research on the state of the art of related topics of this research and concludes with a section on the applied research techniques and -methodology. The proceedings of this chapter will provide answer to sub research question 1 and sub research question 2.

2.1 Background of UCAVs and the SDCS

The application of UCAVs has shortly been treated in the Introduction of this thesis. A number of additional points can be summed up regarding the application of UCAVs. The absence of a pilot avoids incorporating load and systems needed for housing a living being inside the aircraft. The absence of a pilot furthermore introduces the potential of a lower empty weight and a larger amount of fuel replacing the weight of the pilot. In terms of aerodynamics, the absence of a pod for the human pilot may result in more streamlined UCAV designs. The absence of a pilot additionally introduces possibilities for operating at exceptionally high altitudes. In terms of endurance, fully or partially automated UAVs are not limited to human pilot fatigue. This poses the ability to extend operation times of UCAVs significantly. Furthermore, a supplemental argument for further developing UCAVs resides in the theory that (pilots of) manned aircraft incline towards being increasingly vulnerable to air threats as weapons- and flight technology develops. The utilisation of weapon systems that are more agile and less visible call for a smaller response time from the pilot, increasing both task complexity and the demand for the pilots competencies [7].

The constraints of UCAV utilisation have been briefly treated in the Introduction of this thesis and revolve around situational awareness, latencies and data link frailness. Moreover, cases may arise where the UCAV-pilot is not capable of sensing aerodynamic or mechanical haptic feedback such as vibrations of the airframe or acceleration, further decreasing situational awareness. A decreased ability to assess flight status and -envelope also leads to problems [8].

As mentioned, all these deficiencies restrict the effectiveness of UCAVs in both long distance operation along with short distance operations and engagements with adversaries. In air to air combat, quick responses are essential for mission success. From these problems, the claim has evolved that UCAVs, at present, are insufficiently survivable and effective against manned aircraft and other UCAVs in air-to-air combat situations [6].

Before considering potential solutions, it is important to justify that the occurrence of close quarter air-to-air combat must not, despite the shift towards longer distance engagements, be neglected.

2.1.1 Long Distance versus Close Quarter Air-to-air Combat

To date, military air combat places emphasis on long distance engagements [9]. The philosophy behind this remains unambiguous: defeating a target on long range, avoiding the possibility of the adversary becoming aware of the fact that you are present or even engaging said adversary. Long distance combat theory is connected to sophisticated fighters in possession of powerful and advanced radar, weapons control and -accuracy [10]. In instances, the long range fighting capabilities of manned aircraft may impede their short distance engagement effectiveness: As with all aircraft, the development process is largely a process of balancing needs. There is no perfect aircraft, only one that is perfect for a certain situation [11].

To continue revealing why close quarter engagements may occur despite the inclination towards long distance warfare, one can look at state of the art in weapons technology. Currently, the Identify Friend or Foe (IFF) technology that is responsible for identifying friendly forces among targets has not proven to be reliable enough for 100% reliable long range engagements [12]. Performance wise, long range missile technology has not fully matured and as of today, the mission statistics of various long range missiles have shown to be less promising than estimated, while costs for implementation are relatively large [13]. In addition, improving stealth technology enables one to approach adversaries, possibly provoking close quarter combat situations whilst remaining invisible when distances are larger [14]. Lastly, close combat scenarios are likely to occur if groups of aircraft engage targets or when rules of engagement are unfavourable [15]. The example of the Vietnam war emphasises on this argument. During the Vietnam war, the U.S. Air Force and Navy aircrews had prepared for long distance engagements throughout. Nonetheless, the technology was not well enough developed and the environment was unfavourable for long distance engagements for their strategy of choice leading to inevitable close quarter combat engagements [16].

To conclude it can be stated that in spite of the recent developments regarding long distance combat, there is a strong uncertainty regarding whether close quarter combat will no longer occur in the foreseeable future. Therefore it is a logical claim that UCAVs or manned aircraft should not be incapable of engaging in close quarter combat.

2.1.2 The Semi-Direct Control System

The philosophy behind the SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with a SDCS directly, but instead sends commands with certain intervals (varying from 1 to a multitude of secons) between commands. These commands do not control the UCAV directly, but provide waypoints or goals for the UCAV to achieve. For example: fly to a certain direction, fly to a certain position or follow an aircraft, approach an aircraft under a certain angle and so forth. In practice, this implies that the SDCS should possess a CMMS that is equipped with a measure of "intelligence" that determines the most optimal, the most safe or the most effective way to fulfil the command it has received from the operator. The SDCS constantly analyses positions and movements of both the UCAV with which it is equipped as well as adversary aircraft. It attempts to predict changes by utilising for example extrapolation techniques and using these to calculate and execute the necessary direction changes. The pilot determines the tactics to be used and can continuously modify the flight path and strategy based on the adversary's behaviour. In principle, the operator decides on the tactics and approach and the SDCS performs the actions. An impression of the working principle of the SDCS is provided in figure 2.1

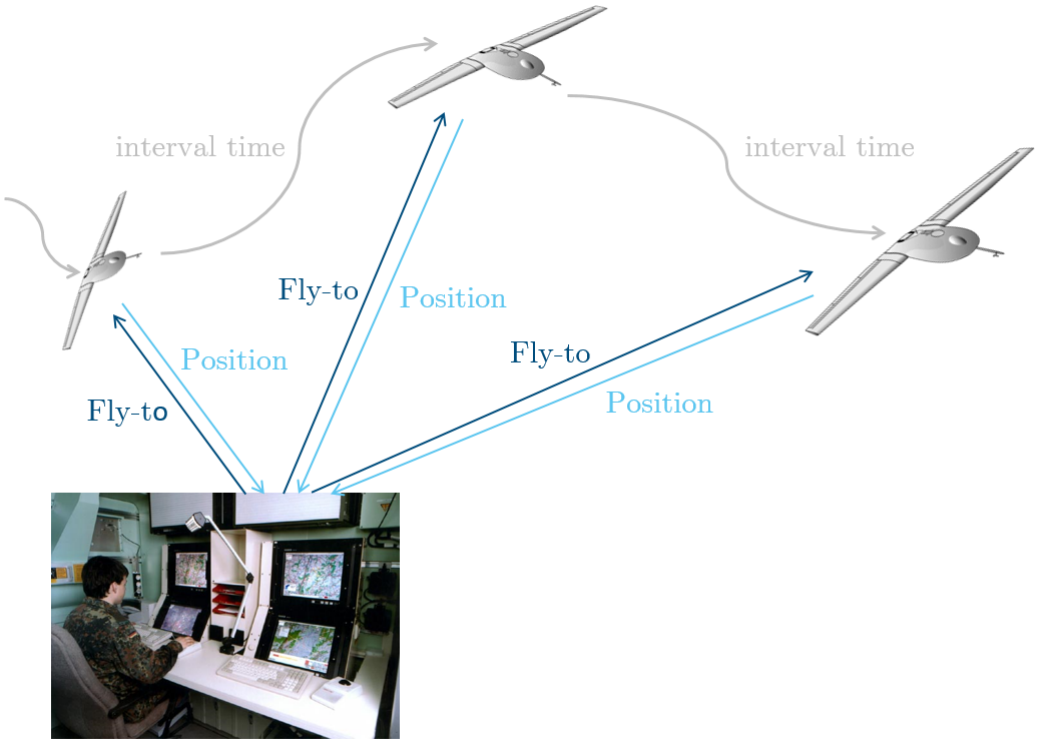


Figure 2.1: *Impression of the SDCS: The operator gives commands, in this case "Fly-to", to the UCAV. The CMMS establishes the route towards that waypoint*

The SDCS is placed on the verge between direct control systems and near-complete autonomous control systems. In the first, the operator directly adjusts the aircraft's control surfaces and equipment. In the latter, the UCAV performs control and operation tasks in solitude in which the role of the operator is decreased to a minimum. Drawbacks of the first control system in UCAVs reside in decreased situational awareness, latency problems and data link frailness. The exact amount of latency is difficult to determine and furthermore depending on the communication technique that is being utilised. Latencies in Geosynchronous Earth Orbit (GEO) satellite constellations require large distance to be covered by the signal resulting in a minimum of 240ms delays. Other constellations such as Medium Earth Orbit (MEO) satelitte constellations or Low Earth Orbit (LEO) satellite constellations give better results of delays between 60 to 140ms[17]. Line of Sight (LOS) operation may result in lower latencies but decreases the operational range of UCAVs. Latency in existing RPAS disable the operator to safely perform landing and take-off tasks, taking on delays of 2000ms (2 seconds) [18].

Drawbacks of (full) autonomous control reside in a too little developed state of the art in UCAV autonomy. Despite efforts in research in autonomy and artificial intelligence, a completely autonomous UCAV able to perform air-to-air combat manoeuvres in an unknown environment has not yet been realised.

The SDCS aims at eliminating the drawbacks of these two systems by an almost "best of both worlds"-approach. Due to on-board automation, there may be limited data exchange between pilot and UCAV without it being unable to perform combat manoeuvres. Furthermore, the SDCS overcomes the need of extensive automation capabilities on the short term. If some tasks prove to be excessively complex, the SDCS could enable sub-division of tasks, splitting a larger task into smaller sub-tasks that require less intelligence of the CMMS to solve, consequently requiring more control input from the UCAV operator. As research in fields as for example reinforcement learning and predictable controllers matures, the interval with which the operator has to send commands may decrease and the commands may become of a higher abstraction level (these are often tasks which require a large set of subtasks to be completed first. These kinds of tasks are very difficult to solve due to the large amount of variables and states).

With this, interesting possibilities for shifting workload towards the operator or towards the UCAV opens up. It can be stated that the SCDS is a control concept that has the ability to grow as knowledge matures.

Concluding, a UCAV equipped with SDCS possesses the potential to increase the survivability of this UCAV in close quarter air-to-air combat situations in which a quick reaction time is essential. In a broader sense, the increased survivability of a UCAV contributes to the effectiveness of a broad range of air operations. UCAVs with a SDCS may provide better support to various other disciplines of military organisations. In 21st century air operations, the ability to connect air, ground and maritime forces is regarded as being increasingly important [19]. Examples in which air assets can support decision making include ground forces, sea forces and other air forces: A UCAV possesses the ability to support ground troops with surveillance and providing an "eye in the sky", with increased independency, leaving more time for performing supplementary actions besides operating and maintaining the UCAV. Furthermore, a UCAV with SDCS may provide support for escorting or protecting manned or other unmanned aircraft. Additionally, UCAVs with a SDCS can alter warfare in a sense that human pilots can be separated from risky situations by means of placing the pilot outside the combat area. Extending, the SDCS may find spin-off applications in the civilian sector. Applications in that sector include operations in disaster relief or area's with limited infrastructure.

2.2 The SDCS in the Unmanned Paradigm

It has become clear what the essence of the SDCS is and what the motives of development are. What follows is research in existing literature and thinking in the field of unmanned control. This creates a reference frame in which the relative standing of the SDCS among existing control methods can be determined.

2.2.1 Nomenclature Designation

First, some definitions that are regularly utilised when discussing unmanned systems must be clarified. The three definitions that are recurring and need to be clearly defined within the scope of this research are autonomy, intelligence and automation. Clough [20] has noted that the terms autonomy and intelligence are used interchangeably throughout literature, although they are not necessarily linked. He discusses the distinction between the definitions autonomy and intelligence. Autonomy refers to the ability and freedom to initiate or modify actions [21] and intelligence refers to the capacity for understanding and the ability to exploit knowledge [22]. In this case, autonomous systems have adaptive capabilities that allow for response within a bound domain to situations that were not pre-determined. The adaptive capabilities demand a measure of self-dependence and the ability to learn (either through trial and error or through prediction), optionally by operator influence. Clough's discussion signifies that intelligent agents are not necessarily autonomous, and autonomous agents are not necessarily intelligent. A third relevant definition that needs clear defining is automation. Automation can, in this scope, be defined as performing tasks or sub-tasks with a decreased extent of human intervention [23]. In a broader sense, it can be stated that multiple automation sequences are required to enable equipment to work semi-autonomously or autonomously [24].

2.2.2 Automation in Unmanned Aircraft

Now that the definitions are clear, it is important to find out what they mean in pratice and what different values they can take. For automation, Hopcroft et al. [25] distinguish two separate continua in UCAVs. The first continuum regards decision making between variable strategic options and the latter involves the governing of the UCAVs control surfaces which in turn determine the values for the six degrees of freedom of the aircraft. This and the relative standing of a number of operational UAVs and UCAVs, along with the SDCS is denoted in figure 2.2.

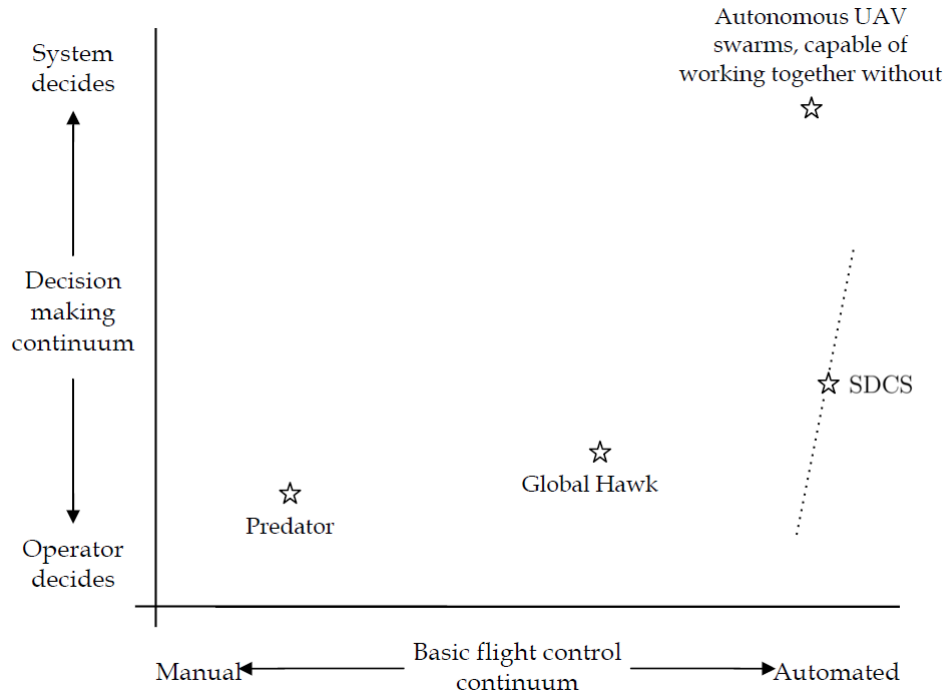


Figure 2.2: *The relative standing of operational UAVs and UCAVs along with the SDCS in the continua as described by Hopcroft et al.*

The Y-axis of figure 2.2 represents automation along the decision making continuum and the X-axis represents automation within the basic flight control continuum. Automation in the decision making continuum may be total, in which case the UCAV takes decisions and initiates actions without either consent or influence of the operator [25]. The relative standing of the SDCS along the dotted line is subject to further research and interpretation: However, the SDCS is allowed to decide how to execute a command that it was given although it should be possible to trace actions of the SDCS back to commands of the operator.

Hopcroft et al. also refer to two distinct categories of automation in the domain of decision making developed by Ruff et al. [26]. The two types are management by consent and management by exception. Management by consent is an management form in which the automated system suggests a multitude of options for actions, but holds the execution until the operator approves one of the actions. On the contrary, management by exception does not require response from the operator: the system will select actions and execute these without approval from the operator. The operator though may "except" certain actions by interrupting the UCAV. Having defined these definitions, it is important to point out the relative standing of the SDCS. In figure 2.2, the dotted line represents the levels of automation in the two distinct domains. On the subject of management type, the SDCS can be considered as a blended form of management by exception and management by consent: The working of the SDCS entails that the CMMS aboard the UCAV equipped with SDCS establishes its actions based on an instruction from the operator, where the CMMS fills in the details of the execution. However, in close combat air-to-air scenarios, it is vital that the operator can interrupt and redirect the UCAV if required. It is yet to be determined what the levels of automation in especially decision making should be. It complies with the basic philosophy of an SDCS that the amount of autonomy can grow within the SDCS as technology and research develops. What the exact balance needs to be is a subject for further research, but is included in figure 2.2 as the dotted line.

2.2.3 Manual versus Automated Control

To further navigate trough the unmanned paradigm this section will shed light on the state of affairs for the two control and governance methods. The two extremes of the spectrum of automation in UCAVs

are defined by the author as little to no automation (where control methods are direct, continuous and require little or no autonomous capabilities are present in the UCAVs CMMS) and on the other side, full automation (where a UCAVs CMMS posesses autonomous features and the role of the human operator is for controlling the UCAV is minimal).

Completely manually controlled UCAVs are widely equipped for acquiring Intelligence, Surveillance, Reconnaissance and Targeting data (ISRT). Examples include the RQ-11 Raven, a hand-thrown UAV operating in Line of Sight (LOS) or Beyond Line of Sight (BLOS). In this case LOS and BLOS denote wether the UAV the operator can visually observe and identify the UAV with his or her eyes. These UCAVs are often equipped with optical or infrared camera's which directly feed the video link to the operator. Increasing task complexity and variety of use demand for increased functionality of a UAV: especially the operation of UAV in the BLOS regime require for extended functionality such as location determination capabilities of the UCAV. It is exactly at this point that the verge between manual control and automated control constitutes. According to the author, a small amount of increase in functionality results in a shift towards autonomy. As an example: a UCAV equipped with Global Positioning System (GPS) functionality to enable a UCAV to automatically return to the point of take-off is virtually a sign of autonomous operation. Additional phenomena that encourage an incline towards autonomous operation of UCAVs include the risk that operator workload exceeds human capability thresholds [27], affecting mission safety - success. Additionally, if data-link connections are lost, interrupted or delayed, the UCAV could become unstable or even fail. Especially in (air-)warfare this is not an option. Furthermore, fully automated UCAVs require flight plans that are sometimes rather time consuming to establish in full detail [5]. The large amount of preparation needed for these plans can cause reduced allowance for real-time adaption of flight and mission plans, again not favourable in air to air combat where high speed decision making is inherent.

Apart from the fact that there is no state-of-the-art of fully autonomous UCAVs, fully automated systems seem to deliver similar constraints. While high workload poses problems, low workload seems equally troublesome: The operator may practically be excluded from the control loop and operator tasks in highly automated systems may solely include providing supervision an monitoring of system parameters and mission progress, perchance making minor alterations to parameters. These tasks, which require sustained attention from the operator, can be become repetitious over time and have been proven to lead to degraded task performance and vigilance based stress [28]. In addition, high levels of automation can prevent the operator from rapidly intervening in a (decision making) process [29]. Lastly, human operators should ultimately be responsible for UCAV actions, especially in operations in which friendly or civilian (asset-) locations or positions overlap with the "mission demographics" [4].

The literature tends to agree upon an amalgamate control protocol which resides between full automation and direct human control. Ideally, such a system should maximise on the strengths and minimise on the weaknesses of both the human and machine. Broadly stated, machines outperform humans at calculations and precision while humans are flexible and have natural problem solving, improvisation and decision making capabilities [30]. This and previous arguments seem to justify the research into the SDCS, which is an amalgamate control concept which aims to find the right balance between human-machine interaction.

2.2.4 Autonomy in Unmanned Aircraft

As mentioned in the previous section, chains of automation may be what forms autonomy in unmanned aircraft. To delve deeper into the literature and thinking in the field regarding autonomy in unmanned aircraft one can start looking at more theory by Clough. Clough has further developed the Autonomous Control Level (ACL) matrix in order to provide metrics for autonomy measurements in UCAVs [20]. For the sake of readability, the full matrix is placed in Appendix B. Its importance shows in differentiating and orientating in the unmanned paradigm, especially when proceeding to treat amalgamate UCAV control forms. The matrix defines 10 levels of autonomy with another added dimension, namely

the Observe-Orient-Decide-Act (OODA) loop, originally designed by John Boyd [31] who applied the concept to military mission planning. The matrix utilises the OODA loop to enable mapping of autonomy levels with certain capabilities of the UCAV. The higher autonomy levels describe multi-UCAV operations, possibly operating with the aid of subtasks to accomplish a larger goal. The author wishes to place the SDCS in autonomy level 4 of Clough's matrix, mainly due to the fact that the metric used in the act cycle constitutes "Self accomplishment of tactical plan as externally assigned" which describes the basic philosophy of the SDCS. More examples can be placed in Boyd his matrix to further develop a sense of the measure of autonomy in UCAVs.

2.2.5 Examples of Autonomy

A well-known example of an operational UCAV is the Northrop Grumman RQ-4 Global Hawk [32]. This UCAV is a High-Altitude, Long-Endurance (HALE) aircraft designed for ISRT operations on pre-programmed mission plans, which are complex and require validation of routes and plans based on target priorities [4]. Re-tasking during operation is not an option with the RQ-4 Global Hawk, degrading flexibility during unforeseen events [33]. The Global Hawk is automated to peform landing and take-off and to achieve correct altitude and airspeed when navigating towards a waypoint. It can be stated that the Global Hawk is a highly automated UCAV. However, placing the Global Hawk on Clough's matrix, it can be noted that the level of autonomy in the Global Hawk is rather low in it executing a pre-programmed mission plan without the capability to respond to changes. The General Atomics MQ-1 Predator and subsequently the MQ-9 Reaper are likewise suitable for ISRT operations but are able to carry a variety of armament. In addition, several Technology Demonstrators (TDs) are being developed. It should be noted that a TD generally serves as testbed for developed technology and often demonstrate a single aspect of UCAV technology, rather than operating as a stand-alone UCAV. The Barracuda developed by the Airbus Group [34] focused on coordinated operation between multiple UCAVs and adaptive mission scheduling. Efforts have been made to include the Barracuda in unregulated airspace [35]. The nEUROn project led by Dassault Aviation [36] presents a larger UCAV with a similar control concept with a ground base from which the UCAV is controlled, similar to the Barracuda.

2.2.6 Human-Machine Modus Operandi

For the foreseeable future, the human will likely remain in the loop when it comes to UCAV control, whether it be from the ground, from a naval vessel or from other air vehicles. Furthermore, factors regarding safety and policy should not be neglected and contribute to the importance of the human in the loop [15]. Especially when the extent of human in the loop is decreased, it must be accepted that the UCAV will no longer be under complete and continuous control [37]. This may be disadvantageous when unknown and unforeseen events take place and there is no means of taking back control. Attention should be paid to feedback that is received by the operator: feedback needs to come in certain forms in order to enable the operator to effectively do his or her job. Special attention must be paid to the type and amount of feedback. Along with this feedback arises the need of highly effective Human-Machine Interfaces (HMI). The operator must be aware of the current situation around the UCAV, understand what this implies and must, according to Sarter and Woods, be able to predict future possible outcomes [38]. The operator and the UCAV must behave in ways that complement each other. Automation can help to mitigate human error on high risk tasks, reduce workload and allows the operator to pay more attention to other tasks. In the best case, humans should have more time to perform tasks which humans do well, such as decision making rather than monitoring. Supervising roles may pose new challenges such as lack of vigilance and the danger of relying too much on an automated system that may or may not be fully capable to perform the task it was assigned [39]. Regarding human error, it is important to note that human errors can still occur in highly autonomous or highly automated systems rather in different stages of the mission, for example during mission planning, as could be the case in the Global Hawk example. Ramage accentuates that the success of responses depend on the amount of situational and contextual awareness. Depending on the scenario, some situations may ask for too much real time

decisions to be made for the operator to cope with, whereas in other situations, especially the expert judgement and action when there is a lack of information is needed [37]. The SDCS, in its current stage, allows for finding the right balance between human and automation, as also acknowledged in figure 2.2.

2.2.7 Future of Unmanned Autonomy in Air-to-Air Combat

Naturally, there are downsides when UCAV operation and development are considered. Although a number of advantages of the usage of UCAVs in comparison to manned aircraft are clear, UCAVs are not necessarily cheaper due to highly sophisticated technology and development costs [40]. Also in terms of task flexibility, the shortcomings are visible. Manned fighters may be designed as multi-role fighters (since a human being is capable of performing a incredible amount of different tasks) whereas UCAVs are to date developed for a less broad range of operations, mainly to give scope in research projects, making manned fighters possibly more cost effective [41]. This is likely to change overtime, but for now UCAVs seem to be unable to compete against their manned counterparts. Yet, the potential of highly autonomous UCAVs is recognised. The United States Airforce released their Unmanned Aerial System Flight Plan describing their plans for unmanned systems until 2047, in which they portray a positive attitude towards unmanned aerial operations [42]. One must furthermore not forget the policy and operational challenges that need to be faced. There is however a large amount of matured technology available and the policy support and operational needs have been established which makes it possible and additionally meaningful to develop UCAVs. The use of plug-and-play themed design and the use of Commercial Off the Shelf (COTS) technology and dual-use of technology may speed up this process.

2.2.8 Justification of Research into UCAVs

Although the literature on the development, testing and the potential of UCAVs is, as partially demonstrated in this thesis, quite extensive, there seems to be little literature to be found regarding UCAVs applied in close-quarter air to air combat. A reason for this is that close-quarter combat is so dynamic and complex. But it is important to look at the potential of UCAVs in close quarter combat nevertheless. Previously in this thesis, Boyds OODA loop was introduced, along with the claim that humans may perform decision making tasks better than computers, and that computers may be better in monitoring. This would mean that humans would outperform computers especially in the "Observe" and "Decide" phases in Boyd's loop, whereas computers would be faster in the "Act" phase, having a smaller reaction time and higher multitasking capabilities, also justifying the problems related to situational awareness in the context of man-machine interface. However, as Byrnes [43] noted, this observation may have been too shallow. If closer attention is paid to the different stages of Boyd's loop, it can be argued that machines may outperform humans also on the "Observe" and "Decide" phases. It is in the scope of this thesis not useful to speculate on sensor technology and computing power compared to the human brain in the "Observing"-phase, but there is an example available that falls within the "Decide" phase of combat which is rather striking and demonstrates the emergence of pilot autonomy in UCAVs. In 2008, McGrew et al. successfully implemented Approximate Dynamic Programming (ADP) for planning flight manoeuvre decisions in real-time [44]. What makes this research stand out is that their results are achieved by what in the discipline of machine learning is called "Unsupervised Learning". That is, to learn in a somewhat trial-and-error approach, how to behave in air combat situations without being explicitly programmed. In their research, the human involvement was limited to setting higher level goals, similar to the philosophy of the the SDCS. Another example comes forward in Bessemer's research which delves into the conversion of F-16's into a remotely piloted multi-role fighter capable of performing high-G manoeuvres [45].

How the human-machine modus operandi will take shape in the future of unmanned air-to-air combat remains topic of philosophy and discussion. The author has noted different viewpoints on this matter: Some sources state that there is a trend that tends to indicate that UCAVs end up as being wingmen of manned fighters and that the manned fighter will always remain playing the key role in unmanned warfare. Other agree and believe that UCAVs may be force multipliers manned fighters, so that com-

bined application of manned and unmanned individuals ends up in a team which capabilities are higher than the sum of that of the individual members.

The imposed alteration of the role of fighter pilots is in the authors point of view something that triggers emotion. From personal conversations with fighter pilots it has become clear that their love and pride for their profession is very prominent and valuable. Other fighters have expressed their acknowledgement towards this development as being imminent, regardless of the consequences for their position. Another difficult subject is the fact that military organisations may not be willing to invest in something that stands, as of today, far from reality [11].

To knowledge, allies and perhaps potential adversaries are mounting research and development programs for autonomous UCAVs. Thus, being largely passive or reactive to developing autonomous UCAVs imposes risk of losing future tactical advantage [43]. It is unclear in what way humans and technology will be interacting and it remains to be seen which options are viable, but the importance of this research is clear. The emergence of UCAVs is taking place and it is dangerous to ignore this subject. In the authors point of view, the conversation does not deal with competition between humans and machines, instead it concerns the nature of the cooperation between them. It is important that research is done to fill in blank spots in the literature about UCAVs in air-to-air combat and this thesis attempts to add knowledge to this field by developing the MVP that provides room for experimenting with different functionalities of the SDCS.

2.3 Related Work

Now that the background of UCAVs, UCAV control and the SDCS is thoroughly analysed and this research is justified, it is time to delve into the theory and practice more from the perspective of the subject that will actually be researched in this thesis, namely the MVP of the SDCS. A certain amount of aspects are closely related to the envisioned MVP of the SDCS. In the section that describes the scope of this research, it is stated that the MVP will constitute be the software running on a computer. There is a profound relation between this topic and an element of unmanned systems that thus far has not been extensively treated; The GCS. Other topics that one can think of that relate closely to developing a demonstrational and developmental environment for the SDCS are UCAV or manned aircraft (control) simulations, testing and training platforms for unmanned control and platforms for experimenting with operator performance. This section thus serves as a research into available options and the novelty of the posed research question. Related work may also serve as inspiration for the look and feel and final form of the MVP to be developed in this thesis.

2.3.1 Ground Control Stations of UAVs and UCAVs

The Ground Control Station is literally the human operator's portal to the UAV or UCAV. In its most generic form it is an booth or that can be located on an arbitrary location on the ground. However, there are examples in which the "Ground" Control Station is located aboard of another aircraft as "wingman". In the scenario where the manned aircraft finds itself heading for a risk infested environment or aerial setting, the pilot and operator may decide to divert course of the aircraft towards safer airspace and instead fly an unmanned aircraft in that direction from out the aircraft. Typical GCSs consist of hardware such as a computer and screens and the necessary hardware required for the communication with the UAV or UCAV. The GCS is a synthetic cockpit from which the operator controls the UAV or UCAV. Depending on the application and type of UAV, the GCS requires sophisticated HMIs; accurately representing the data coming from the UCAV towards the operator is a different branch of research well beyond the scope of this thesis. However, GCS in general seem to possess similar functionalities as well as a similar "look and feel".

MQ-9 Reaper GCS

The MQ-1 Reaper UCAV is handled in the previous section of this thesis, and its GCS is depicted below [46]:

Figure 2.3: *The Ground Control System of the MQ-9 Reaper UCAV*

The GCS is the largest part of the MQ-1 System and is a large container-like booth that can be carried in a C-130 transport plane to be deployed anywhere in the world. The GCS is equipped with operation hardware and software to accommodate two persons. The first person is the operator of the UCAV and the second operator is the sensor operator.

MUAV Factory's Portable GCS

The GCS developed by UAV Factory [47] is a portably designed GCS that is aimed to be a universal GCS designed to be operationable with a multitude of UAV systems. The system is designed with what UAV Factory calls a Modular Electronic Compartment. Different COTS hardware that is required for all kinds of functionality can be quickly installed in a plug and play manner. Because of this flexibility, the GCS can also be equipped for ground robots, such as bomb disposal robots. It can also be configured as a stand alone system that allows for the interfacing of sensor equipment for performing measurements.

Figure 2.4: *UAV Factory's Portable Ground Control Station*

Mission Planner GCS Software

Mission Planner developed by Michael Osborne [48] is a GCS with many functionalities for real time flight and after flight analysis.

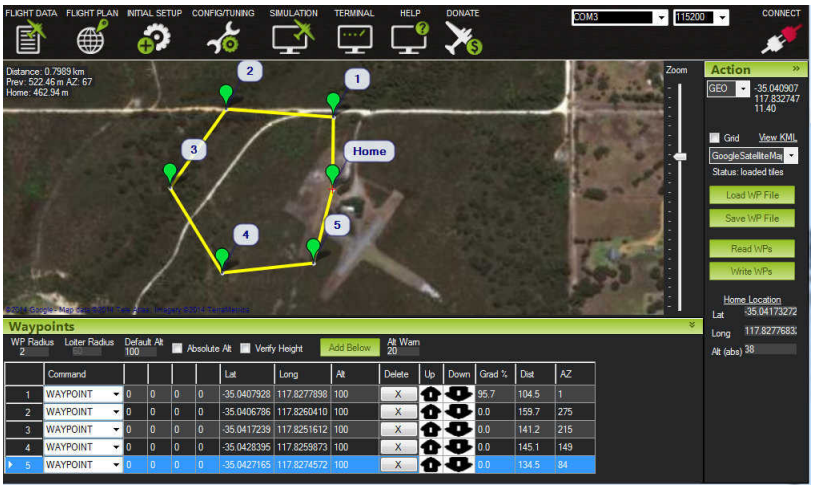


Figure 2.5: *Mission Planner GCS computer based software*

Mission Planner makes use of a clickable maps that can be imported from a variety of sources such as Google Maps or Bing Maps. It enables quick input of waypoints and easy mission planning. Different functionalities to be executed by the UAV can be selected from drop down menus and the user is able to download log files of flights to perform further analysis of mission statistics. It furthermore allows for interfacing with a flight simulator so that it becomes a multifunctional and realistic UAV control simulator.

MAVLink and QGroundControl GCS

MAVLink stands for Micro Air Vehicle Link, which is a communication protocol that enables communication between a broad range of small UAVs and several GCSs. The project was initiated by Lorenz Meier [49]. QGroundControl is a multi-platform software package that serves as a platform for UAV control and mission planning [50]. The software aims to provide a clear interface for a various range of users. The functionalities include mission planning, display of the flight map and its waypoints, video streaming and interfacing of UAV instruments and parameters. It furthermore supports control of multiple UAVs at once.

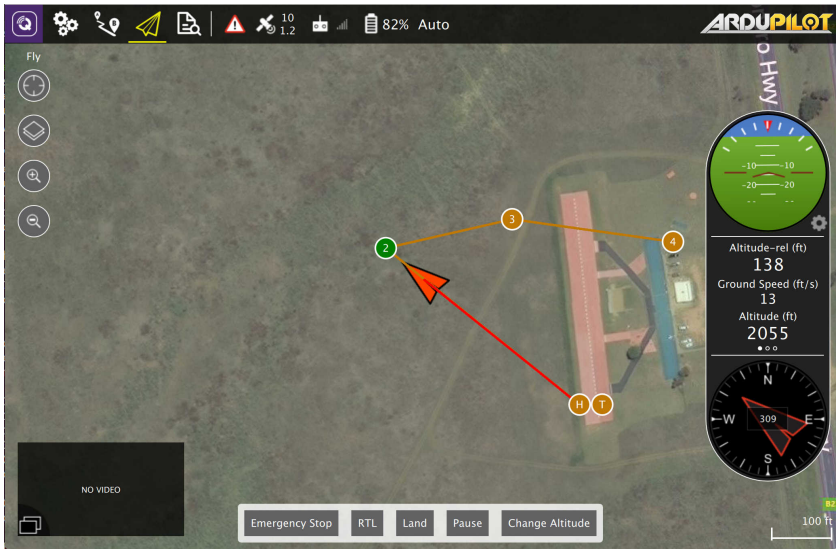


Figure 2.6: *QGroundControl GCS multi-platform interface*

MAVProxy GCS Software

MAVProxy is a ultra light-weight GCS application written in the programming language Python [51] It uses a simple command line interface and a 2D map for interfacing with UAVs. The usage of Python, a high-level programming language, makes that other developers can easily implement their own modules into the program. The program is designed to be networked with multiple computers, keeping in mind simplicity so that it can run with minimal hardware and software requirements.

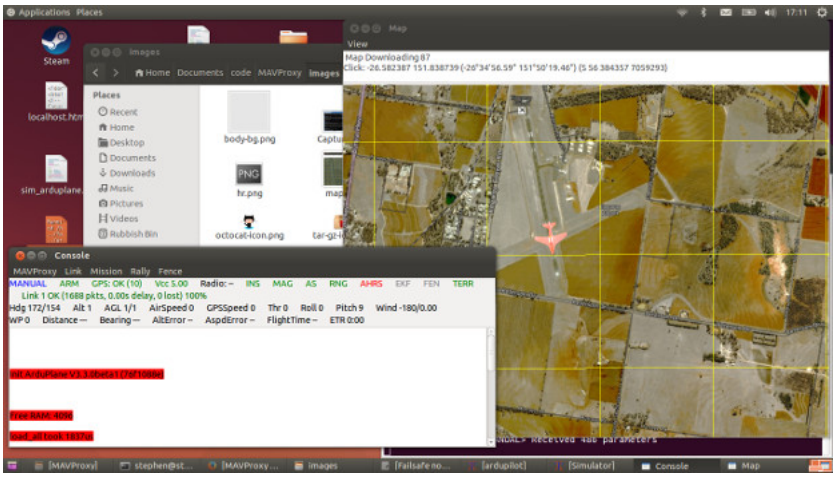


Figure 2.7: *MAVProxy's command line interface and 2D map*

UGCS Universal Ground Control Station Software

The UGCS possesses a extended 3D interface. The software supports an extensive amount of UAV systems from different UAV manufacturers. The software includes maps from different sources, allows for multiple layers of different maps with different information and also includes pre-programmed no fly zones around large airports across the world. It has professional possibilities such as transponder support and different input modes. Other functionalities include video recording, geo-tagging, flight playback and it supports multiple instances of the software to be connected to form a Ground Control Server for multiple operators.[52]

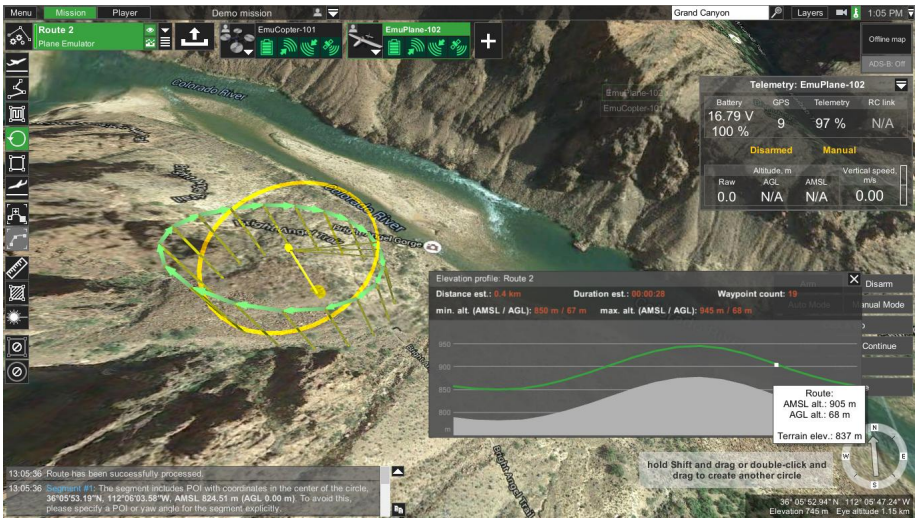


Figure 2.8: *UGCS 3D interface and waypoint based operation*

Other devices as GCS

GCS software furthermore has several versions available for tablets and phones. Advantages for running this software on a tablet or smartphone lie in portability and connectivity. These devices are easy to transport and smartphones may use cellular network for extended functionality in communication. A

drawback is the limited computing power available on these portable devices in contrary to a computer or laptop. The image below depicts the MAVPilot software capable of operating on Apple Iphones or Ipads [53]

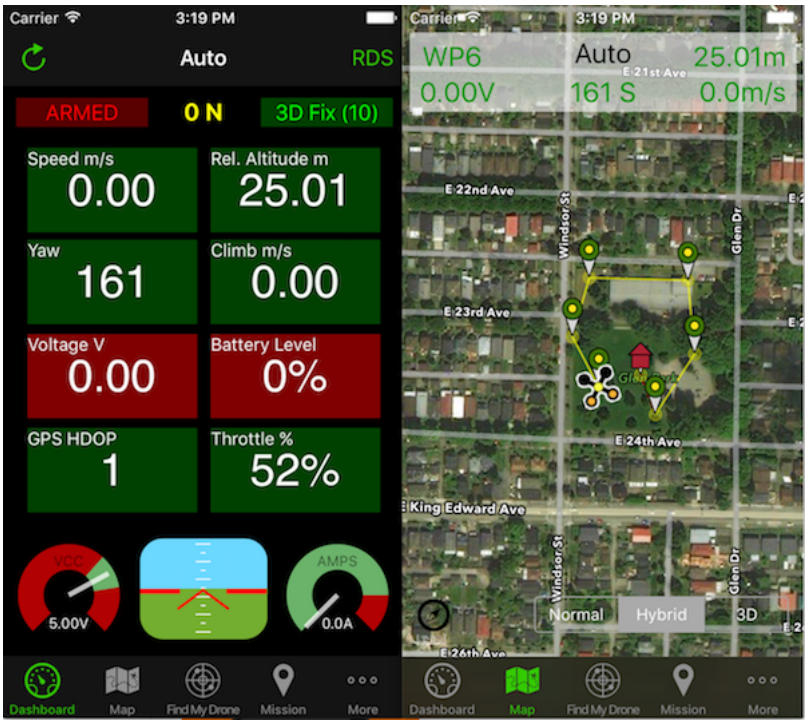


Figure 2.9: *Interface of the MAVPilot iOS application*

2.3.2 Simulations of UAV Control

Another related topic are simulation environments for UAV control. There is a multitude of software available for the simulation of small-scale (hobbyist or professional) UAV Control. The software aims at implementing models of the UAVs in combination with realistic flight dynamics and physics so that pilots can practice their manoeuvres or even complete missions in the software before performing them in real life where the risk of damaging often expensive equipment is larger.

Phoenix 5 R/C Simulation Software

Phoenix 5 R/C is computer simulation software that allows practising the operation of Radio Controlled (RC) UAVs [54]. The program is extensive in the way that it has incorporated a broad range of existing models of UAV manufacturers in its software. The models characteristics can be tweaked and altered and the user can even add own models to the software. This contributes hugely to the realism of the software so that flying the UAV will be as close to reality as possible to allow for accurate training. The same RC receiver as used in real life can be connected to the software so that the operator can fly with the same control receiver as he or she would normally do.

Figure 2.10: *Phoenix 5 R/C software displaying the RC receiver of the user in the left bottom of the screen, the movement of the joystick corresponds with that of the joystick on screen to clearly show the pilot what he or she is doing without having to look down on the receiver*

Aerosim RC Simulation Software

Aerosim RC is similar to the Phoenix 5 R/C software. It allows for the simulation of helicopers, fixed wing aircraft and multicopters, through the RC receiver. It also allows for simulating the on screen display that is found on some RC receivers [55]

Figure 2.11: *Aerosim RC interface*

2.3.3 UAV and UCAV Training and Testing Platforms

The NLR has a multitude of in-house technologies for the control, simulation, training and research around UAVs and UCAVs. These technologies provide background for this thesis and the internship has allowed for an in-depth view of these technologies. Unfortunately, not all information that was provided to the author can be disclosed in this thesis. The section below provides the information about these technologies that could be communicated externally.

MUST - Multi UA Supervision Testbed

MUST, which stands for Multi-UA Supervision Testbed, is developed by the NLR as a generic GCS research simulation facility for various unmanned systems. The simulator can be adjusted to different research demands and is used for research of human factors in UAV control. The purpose of this research is to improve human(-machine) performance in order to reduce the human error. The interface of MUST can be rapidly adjusted to suit various research demands. One of the research topics is human taskload. If the taskload for an operator increases, the operator becomes more vulnerable to errors. Furthermore, as mentioned, good situational awareness is key to operator performance and

the MUST allows for different HMIs to additionally experiment with this. Automation may help relief some tasks from the operator, but the challenge lies in finding a good balance in the level of automation in UAV or UCAV control. MUST can be configured into different levels of automation and therefore forms a suitable research platform on this subject. MUST is a platform that allows for two operators: one vehicle operator and the sensor operator. Furthermore, MUST can also be equipped with different kinds of input devices, for example: a SpacePilot mouse (3D mouse), touchscreens, game controllers or eye trackers. MUST allows for switching of tasks between operators to enable research on how this influences operator performance and efficiency. Some features of MUST include operation with different types of UAVs and UCAVs. MUST can be incorporated with both virtual and operational UAVs, such as the Pelican (NLR test UAV), NARSIM and ROVATTS (will be handled later) and X-Plane (Flight simulator). MUST allows for control of multiple UAVs at the same time and is STANAG 4856 compliant: STANAG 4586 is a NATO Standard Interface of the "Unmanned Control System and UAV interoperability" [56]. It defines a broad range of subjects such as information flows, communication protocols and data formatting. There is also a tablet version available of MUST, named MUACGS. Unfortunately, no screenshots can be taken from MUACGS or MUST.

Figure 2.12: *NLR's MUST allows room for two UAV operators, one flight operator and one sensor operator. Tasks can be switched between two operators, there is no seperation in functionality of the MUST*

ROVATTS - Remotely Operated Vehicle Adaptable Training/Tracking Systems

ROVATTS is a PC-based simulation environment that aims at providing a low-cost product that is highly customizable. It is utilised for simulation, research and development testing and evaluation, training and mission rehearsal operations and allows air, ground or sea based vehicle to be implemented. ROVATTS is able to run on COTS computers and laptops, displays and helmet mounted displays. At the NLR, ROVATTS is equipped in combination with MUST to simulate control of the MQ-9 Reaper UCAV. ROVATTS comes with a software Developers Kit (SDK) to provide developers to plug directly into ROVATTS "internal organs" to directly influence operations. It provides manual and autopilot operations and implements scripted air and ground vehicles to interact with. ROVATTS also makes use of Computer Generated Forces (CGF), realistic weather and high quality graphics for advanced mission training [57].

Figure 2.13: *ROVATTS MQ-9 Electro-Optical sensor view observing a simulated training camp in Nowzad Afghanistan*

NARSIM - NLR Air Traffic Control Research Simulator

NLRs ATC Reserach Simulator (NARSIM) combines simulations of different environments for area- and traffic control. The visual system of the NARSIM Tower facility consists of a 360 degree projection screen with a diameter of 11 meters and a height of 4.5 metres along with a large booth with multiple computers offering room for a large amount of (air traffic) operators. The NARSIM tower is utilised for development of HMIs in air traffic control, validation of Air Traffic Management (ATM) concepts and procedures and training in ATM. With the NARSIM Tower, NLR has provided training ranging from basic introduction to tower control to replacing (large parts of) the tower controller's on-the-job-training and large scale training. NARSIM can handle a large amount of persons operating at the same time and is built in a way that it is highly scalable. Its relevance in this thesis comes from the integration with MUST. Together with MUST, the NARSIM forms a flexible and modular UAV/ Air Traffic Control (ATC) validation environment that is designed to research and support the challenges of UAV control and operations. The facilities have flexible interface capabilities and are scalable and adaptable to meet specific validation goals for a wide variety of stakeholders, both civil and military.

Figure 2.14: *The renewed systems of the NARSIM, the tower consists of a 360 degrees horizontal and 40 degrees vertical projection space which allows synthetic view over different airports and other locations*

F4S - Fighter 4 Ship

Fighter 4-Ship (F4S) is a research simulation facility that can simulate tactical operations of up to four fighter jets. The F4S uses tube-frame pods and screens to allow for mobility and affordability. In the F4S, the NLR has CGFs called Smart Bandits. These artificial adversaries are flown by computer and incorporate complicated techniques to show realistic and unpredictable flight behaviour. The Fighter 4 Ship can be connected to various unmanned vehicle simulations and the CGF in Fighter 4 Ship can be used as a training measure for interacting with and engaging against UAVs and UCAVs. For this project, it might be interesting to use the same CGFs and apply them in the MVP to simulate well-trained adversaries to test the SDCS on.

Figure 2.15: *A tube-frame pod and the screen layout of the Fighter 4 Ship. The Fighter 4 Ship has four of these modules along with a mission planning center*

2.3.4 Methods for controlling UAVs and UCAVs

In the scope of this research, it is furthermore interesting to delve into different controller methods for UAVs and UCAVs. Aside from traditional joystick, game controller or 3D joysticks, there is a multitude of options for controlling UAVs or UCAVs. This section will shed light on a few methods.

Control through a smartphone or tablet

A wide variety of consumer "drones" make use of control through a smartphone or tablet. The internal acceleration sensors of smartphones and tablets can be coupled to the six degrees of freedom controllers of consumer drones so that the drone can be controlled by physically moving the smartphone or tablet. For example tilting the device forwards or backwards makes the drone fly towards the corresponding direction.

Control through physical motion

The research of more natural control methods for UAVs is widely performed. There are examples found where the human body is turned into the controller. For example, Sanna et al. applied a Microsoft Kinect to communicate with a Parrot AR multicopter through a computer to let the operator control the multicopter by using gestures and movement. Basal gestures such as lifting your arm can be connected to a take-off command for the multicopter [58]. A similar idea is presented in the paper by Chandarana et al. where a Leap-motion device is used to translate hand gestures to UAV control commands in order to simplify the user experience with controlling a UAV [59].

Control through speech

An application of control using speech is presented by Schouwenaar et al. This paper presents a UAV mission system that allows an operator to give a UAV commands in real time using speech. The interface developed enables communication between the operator and UAV in languages that are understood by both the operator and the UAV [60].

Control through vision

Several methods for controlling and interfacing in combination with eyesight methods have been explored. A software engineer at Microsoft used an open source platform for UAV drone control to come up with a proof of concept for tracking hand gestures using the Microsoft Hololens [61]. Although in this stage it needs a lot of work, the Hololens can form an interesting option for user interfacing, where control of a UAV may feel very natural and intuitive. An other interesting example is the research performed by Walter et al. They present an immersive GCS for the control of one or multiple UAVs. The GCS utilised Virtual Reality (VR) to form a synthesized picture of the space in which the UAVs are operating. Furthermore, the interface presents datastreams from the vehicles that are being governed. The research aims at improving operator's situational awareness. [62] Another example of using the eyesight in combination with UAV control is gaze. The research paper by Hansen et al. [63] describe the application of a device from the Eye Tribe Company to control a UAV by using motion of the eyeball itself. In this paper it is presented as an alternative for hand controlled options so that persons with a physical disability are able operate drones despite their disability. It is an interesting concept to control a UAV since one could literally look into the direction in which the operator wants the UAV to fly. It would be a more direct manner of controlling, bypassing the need for gesturing control commands.

Control through the brain

An interesting method of UAV control lies in the field of Brain Computer Interfaces (BCI). Researchers can capture and monitor the electronic signals running in the brain by means of taking an electroencephalogram (EEG). In short, this means that a user can control a UAV with only his or her thoughts. LaFleur et al. have described a method for controlling a UAV using a noninvasive method [64]. The research has been performed to aid those suffering from diseases or disabilities to freely explore the surroundings, but the concept of controlling UAVs with thought is interesting as it may offer a agile and intuitive control method.

2.4 Research Techniques and Methodology

This section handles the research methodology and techniques that are used trough out this thesis. The most prominent design method utilised in this thesis has been introduced in the paper 'A Design Process For Creative Technology' [65]. This method is specifically designed for projects inside the Creative Technology curriculum. A schematic overview of the design method is provided in Appendix A. In short, the design method consists of 4 phases: Ideation, Specification, Realisation and Evaluation. The phases are cyclic and each following phase has numerous methods for propagating findings and results upwards into an earlier phase in order to converge towards a product that satisfies the needs.

The design method commences with a design question, a problem or an idea. After the problem is identified, the Ideation phase commences which aims at creating elaborated ideas or solutions to the problem by executing a cyclic ideation process. Techniques including but not limited to interviewing, brainstorming and stakeholder analysing are utilised to obtain relevant information to converge towards a specific product idea.

The Specification phase commences after a product idea is realised. In this phase, the functional requirements of the envisioned product will be specified to set a clear frame for the realisation of the

project. A number of prototypes are utilised to shed light on a multitude of options for realising the requirements. The interplay between functionality and the result of the prototype is central in this stage and the change of one may require adaptation of the latter. The multitude of prototypes realise this relationship and multiple prototypes are discarded, improved or merged into newer prototypes. The result of this phase includes a clear functional specification of what the envisioned product should be able to accomplish.

The Realisation phase is the last cyclic phase of this design method and describes composition and validation of the product in terms of components.

Following this phase is the Evaluation phase. This phase entails testing the designed prototype against the set requirements created in the Ideation phase. The role of related work is to give a broader context of relative standing among existing work.

2.5 Research Techniques

2.5.1 Information Gathering

Information can be gathered from a multitude of sources which can be divided in primary and secondary sources of information. The distinction between primary and secondary sources is made as follows: Primary sources supply information from practice knowledge. Examples of primary information sources include interviews with experts in the field and performing case studies based on case files by clients. The secondary sources include literature sources from various sources including but not limited to journals, books, reports and other theses.

Primary Sources

There is a vast amount of information and expertise available within the NLR. As an intern based at the AOTS department there are various possibilities for consulting experts on their area of expertise. Topics on which the AOTS department houses knowledge include Human Factors in Aviation, Aviation Training, Operator Performance, Aviation Simulation and arising technological developments in aerospace. This information can be utilised in researching the state of the art and may provide guidance in decision making processes and implementation of generated ideas.

Secondary Sources

Academic literature forms the majority of secondary sources. Literature study is conducted to gain insight of the state of the art of the various subjects that are covered in this thesis. Using literature, sub research questions can be answered and furthermore, gaps in knowledge can be identified. Gaps in knowledge provide the opportunity to perform novel research for both the NLR and the academic community. Literature will be acquired from renowned academic platforms such as Scopus and Web of Science, access to which is provided by the UT. Interns of the NLR have access to the NLR database containing all non-classified articles, research projects and theses from all research conducted at the NLR. Other sources of information include government resources and news articles.

2.5.2 Stakeholder Analysis

This project relates to a multitude of stakeholders with own interest and influence on the project. To properly map involved parties and represent their wishes and influences the stakeholder analysis method defined by Sharp et al. [66]. In their work they provide a distinction between baseline stakeholders in the following four groups: Users, Developers, Legislators and Decision makers. Roles of stakeholders described by Sharp et al. include supplier roles, client roles and satellite roles. Identification and categorisation according to the model provided contributes to defining the stakeholders' interests, their

influence and which aspects of the design process they influence. Mitchel et al. [67] propose a classification technique to map stakeholders according to their power, their interest and the legitimacy of each stakeholder's relationship with the project. This classification may guide the communication process and -frequency towards the stakeholders and furthermore guide towards prioritising stakeholder claims and wishes.

2.5.3 Co-Design

This thesis is based on a multi-stakeholder project which does not have a well defined research scope. Each stakeholder might have individual requirements or wishes with varying priorities. This may cause the requirements to be unclear or inconsistent: The wishes of one stakeholder might interfere with the wishes of another. It is essential that close collaboration and communication, both formal and informal, with all active and prioritised stakeholders is maintained throughout the project to ensure the result of this project meets the prioritised needs of the stakeholders [68].

2.5.4 Kano Analysis

To elicit the requirements that are to be set regarding the MVP, the Kano Analysis method is utilised. Kano et al. [69] have developed a two-dimensional model to calculate the influence of the presence or absence of a product attribute on the quality of a product and stakeholder satisfaction. The quality can, according to Chen [70], denote that a product or service complies with the demands of stakeholders or that the stakeholders and customers are content with a product or service. The first step of a Kano analysis is to map the product needs of the costumers. This can be done with the aid of exploratory investigations with the stakeholders. It is important to delve for the "hidden" needs and problems, which involves interviews to gauge information. These requirements form the basis of the Kano questionnaire [71]. In this questionnaire, 1 pair of questions for each product feature is formulated. The first question concerns the reaction of the customer or stakeholder if the product would be equipped with that certain feature. It is denoted as the "functional form of the question". The second question concerns the reaction if the product does not possess that certain feature. This is denoted as the "dysfunctional form" of the question [71]. The interviewee can pick an answer out of 5 possible answered, ranging on a scale from positive to negative. The basic structure of an item in the the Kano survey including the 5 different possibilities for answering questions and the functional and dysfunctional form of the question are summarised in figure 2.16

Functional form of the question If a mobile phone has a display, how do you feel about that?	☐ I like it that way ☐ It must be that way ☐ I am neutral ☐ I can live with it that way ☐ I dislike it that way
Dysfunctional form of the question If a mobile phone does not have a display, how do you feel about that?	☐ I like it that way ☐ It must be that way ☐ I am neutral ☐ I can live with it that way ☐ I dislike it that way

Figure 2.16: *Kano's functional and dysfunctional questions with possible answers ranging from positive to negative*

Hauser and Clausing emphasize that throughout the modelling of the questions, the "voice of the costumer" should be noted. In practice, this means the question should describe a functionality or a

problem to be solved from the point of view of the costumer or stakeholder [72]. If a question attempts to elicit technical solutions or is formulated in such a way, the question may not be correctly understood by a customer. Hauser and Clausing rightfully state that the customer is merely interested in which of his problems will be solved, as opposed to how they will be solved. Furthermore, if the solution of the problem is already incorporated in the formulation of the question, a customer may expect it to be executed in that way, limiting possibilities and creativity in the implementation of this product feature [72]. There are furthermore different methods for carrying out the survey. Sometimes the most favourable option is via mail because of the low costs and high objectivity of the answers. However, the return rate may be low. Oral interviews are suitable, the standardised questionnaire reduces the influence the interviewer has on the interviewee, the return rate is high and when there is a lack of clarity or difficulties interpreting a question, there is immediate help available. When the survey is completed by the interviewee, the answers can be combined and analysed. This can be achieved by means of an evaluation table, denoted in figure 2.17.

	Product Requirement	Answer dysfunctional question				
		I like it that way	It must be that way	I am neutral	I can live with it	I dislike it that way
Answer functional question	I like it that way	Q	A	A	A	O
	It must be that way	R	I	I	I	M
	I am neutral	R	I	I	I	M
	I can live with it	R	I	I	I	M
	I dislike it that way	R	R	R	R	Q

Figure 2.17: *Evaluation table denoting the outcome of the question pair with either Q,A,O,R,I or M.*

The product features can be classified into categories which denote what impact they will have on customer satisfaction. The letters Q,A,O,R,I and M in figure 2.17 are defined as follows:

- Category M: Indicate the "Must-have" requirements. If these requirements are not fulfilled, the customer will be highly dissatisfied. Furthermore, since the customer takes these requirements for granted, fulfilment will not lead to an increase of satisfaction. There is only the case of dissatisfaction.
- Category O: Indicate One-dimensional requirements, the customer satisfaction is proportional to the level of fulfilment.
- Category A: Indicate that the feature is an attractive customer requirement. These requirements will have the highest influence on customer satisfaction. Fulfilling these requirements leads high satisfaction. On the contrary, the stakeholder may not feel less satisfied if this product feature is not included.
- Category I: Indicate that the customer is indifferent to this product feature, the presence or absence does not influence the quality or satisfaction. The customer may also not be willing to spend more on this feature.

- Category Q: Indicate a questionable result. Normally answers do not flal in this category. If present, a customer might have misinterpreted the question or that it was phrased incorrectly or the customer has filled in a wrong answer by accident.
- Category R: indicate that this feature is not only unwanted by the customer, but he or she demands the reverse.

Analyzing the results gives a clear impression of how different product features impact the satisfaction of customers or stakeholders.

2.5.5 MoSCoW Method

An agreement between stakeholders on the priority they place on different requirements can be assembled by means of the MoSCoW method. The priorization method by Clegg [73] can additionaly serve as menas of setting up the research scope and the form of the deliverables. MoSCoW is an acronym consisting of the first letter of each priorization level. In order of appearance: Must have, Should have, Could have, Won't have. The requirements labeled as "Must-have" are critical to the project completion. in the strictest form, if any of the requirements labeled as "Must-have" are not fulfilled, the project should be regarded as non-successful. Must can also be read as a stand-alone acronym for the Minimum Usable SubseT [74]. In other terms, these "Must-have" requirements make up a MVP [75]. Should have requirements are valuable additions, sometimes as important as "Must-have" requirements, to the project but may not be boundary conditions for success of the project. Requirements labelled as Could have are highly favourable but not necessary. They can improve the overall status of the project but are often disregarded if time and resources do not allow room for implementation. In some cases, could have requirements can be addressed in future work sections. The last category includes Won't have requirements, which are agreed upon to be left out of the project. They may be high risk, high cost-low payoff items or functionalities that are not feasible at the point of time. However, they may be included in future work. Including Won't have requirements in the requirements analysis further aids the creation of a research project scope and shields against introduction of these requirements in a later time stage in the project. An MVP is in the literature denoted as a product which contains solely the "Must-have" requirements and demonstrates core functionalities [75]. If time and resources allow, Should have and Could have requirements may be implemented in the MVP.

2.6 Conclusion

This chapter has provided the necessary information to answer sub research question 1: *What is the Semi-Direct Control System?* The SDCS was described and placed in a theoretical framework with the help of existing literature. This has further clarified the relative standing of the SDCS in the unmanned paradigm and has served as justification of this research. A section on state of the art and related work has demonstrated relevant topics and has served as an exploration of the design space as well as providing inspiration for the MVP to be developed in this thesis. The last section of this chapter has helped answering sub research question 2: *What is a Minimum Viable Product?* Existing models and theories on requirements elicitation have paved out a clear path to further research in the functionalities that the MVP should and should not possess.

3 Ideation

This chapter constitutes the first phase of the Creative Technology design process; the Ideation phase. This chapter commences with a section on the role and relative standing of the stakeholders of this project. Following that, the brainstorm processes are documented and elaborated. Requirements regarding the MVP of the SDCS are elicited as the first step of defining the wishes of the clients and confverging towards a product idea which includes and represents the prioritised wishes of the stakeholders. This chapter will furthermore provide answers to sub research questions 3,4 and 5.

3.1 Stakeholder Analysis

The stakeholders of this project are mapped to clarify which stakeholders influences this project to what extent and furthermore to determine what approach towards the various stakeholders is desired. Table 3.1 denotes the stakeholders and their involvement in the project. From this table, the Influence versus Interest grid can be deducted. This grid is represented in figure 3.1.

Stakeholder	Category	Role	Impact	Influence	Strategy
Jan Joris Roessingh	Developer & Client	External Supervisor	High	High	Frequent contact
Gerald Poppinga	Developer & Client	External Supervisor	High	High	Frequent contact
Hans Heerkens	Developer & Client	Internal Supervisor	High	High	Frequent contact
Alma Schaafstal	Decision Maker	Internal Supervisor	Medium	High	Frequent contact
Defense	Legislator	-	Low	Low	Gauge interests and knowledge
RPAS Operators	User	-	Low	Low	No involvement in process

Table 3.1: *Relevant stakeholders along with their category, role, interest in this project, their influence on this project and the approached strategy to be handed towards the stakeholder*

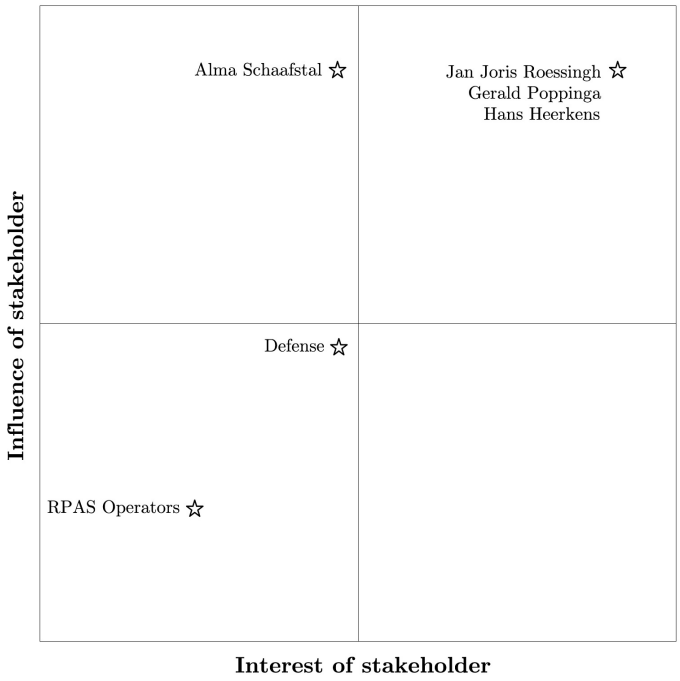


Figure 3.1: *The relative standing of stakeholders according their influence on the project and interest in the project*

Jan Joris Roessingh, Gerald Poppinga and Hans Heerkens are both supervisors and clients of this project. The result of this project will be transferred to the NLR. Their interest and influence in the project are equally high since they benefit from a successful completion of the project. Alma Schaafstal is the internal supervisor from the UT and has high influence on the project since she guards the graduation process and represents the University of Twente during this project. Her interest in the project focuses on a successful implementation of the Creative Technology Design Process and the scientific contribution of this project. Further clarification regarding the placement of the other stakeholders, Defense and UCAV operators, is needed: the current division seems to imply that this project is not designed with the end-users of the SDCS (most probably UCAV operators) in mind. This is not the case, since this project eventually requires to be further developed and consequently be utilised by for example the (Dutch) Air Force and possibly UCAV operators. However, it is important to retrospect on the fact that this project is a pre-feasibility study for the SDCS. In this case this means that upon completion of the MVP, research into the SDCS concept is not yet mature enough to actively involve the final customers or -stakeholders for this project.

From conversations with Jan Joris Roessingh, Gerald Poppinga and Hans Heerkens it can be deduced that the focus of their interest as of now does not primarily lie into how this concept should be sold to future customers or how the interaction with end users should be. Before the main stakeholders attempt to pitch and sell the project to future customers and users, it needs to be determined whether the SDCS is feasible and effective. Before having answered these questions, the stakeholders have shown no primary interest on letting in (a large amount) of potential users into the project, possibly making this project unwillingly more complex. However, initiating and maintaining contact with Defense or UCAV operators may be useful as a means of eliciting more information about the practice and state of the art on operation of UCAVs. These parties may even be approached for a mid-term evaluation or brainstorm session about the project. But if, how and when this will take place is not yet known.

3.2 Initial Brainstorms and Divergence Phase

In advance of delving into an approach for the realisation of the MVP, the design space has to be opened and multiple options for the realisation of the MVP ought to be considered. This section handles the early brainstorm sessions and an exploration of suitable software for developing the MVP.

Early meetings with the stakeholders on their vision of a MVP of the SDCS have been translated in a series of sketches that function as artists impressions of the brainstorm sessions, depicted and described in figure 3.2 to figure 3.4 below.

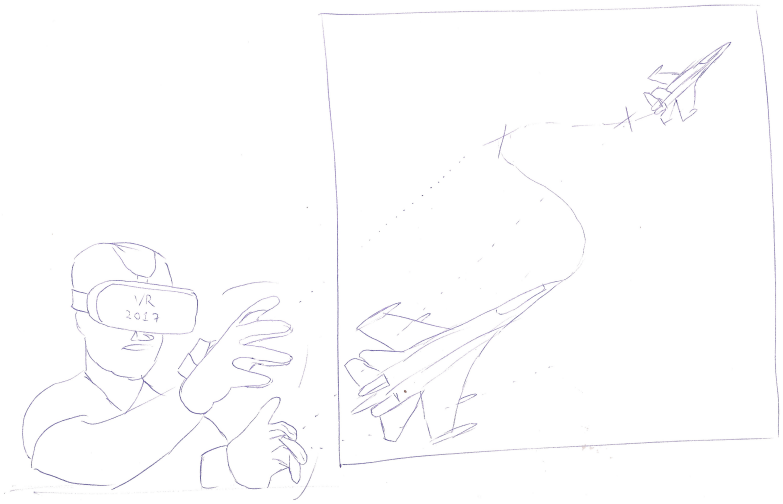


Figure 3.2: *Demonstrator of the SDCS by means of virtual reality and gesture tracking technologies*

Virtual Reality and gesture tracking provide a way of interfacing the SDCS control concept. This largely intuitive way of controlling a UCAV fits with the idea of passing high level commands to the UCAV. Positions and approach manoeuvres towards an adversary aircraft can be established by hand movements. A synthesised image of the UCAV in its battle space can be visualised through virtual reality, allowing for the possibility to physically look around the battle space.

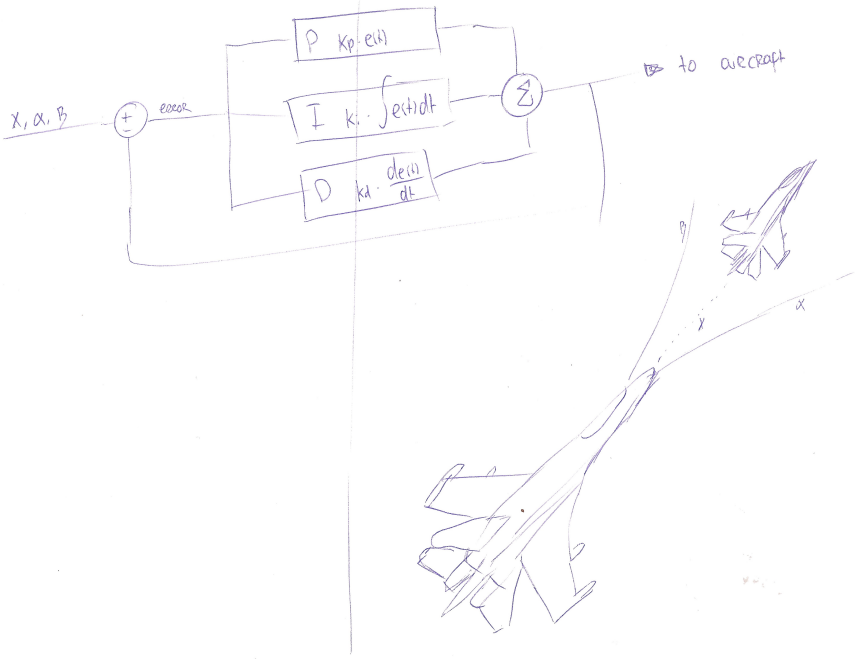


Figure 3.3: *Demonstrator of the SDCS by means of a controller*

At the heart of any system governing an aircraft autonomously lays one or a multitude of controllers that translate input data to desired output data and thus flight behaviour. In the case of the SDCS, this controller should be able to predict movements of the adversary aircraft. A controller can be realised by utilising PID controllers, a widely used type of controller. This controller could be equipped to maintain

a certain approach angle or loitering distance towards the adversary aircraft by constantly monitoring the current value and the desired value consequently attempting to minimise the difference between these two values by adjusting the trajectory or other flight parameters.

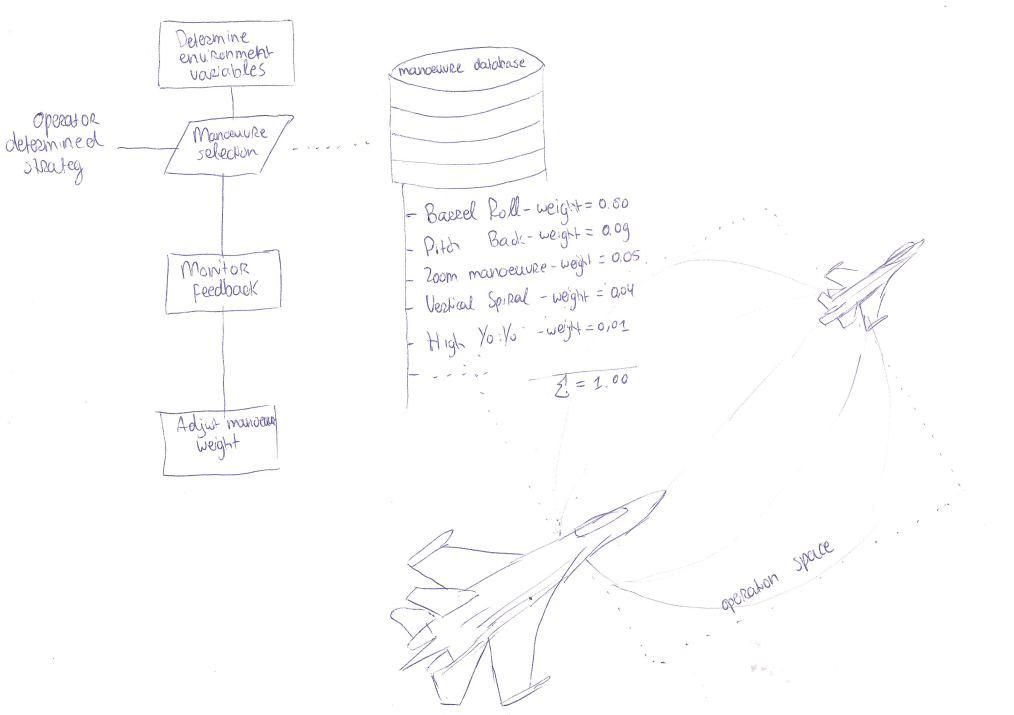


Figure 3.4: *Demonstrator of the SDCS by means of machine learning techniques*

To demonstrate and develop the autonomous operation of the SDCS, a machine learning technique can be applied that selects an air combat manoeuvre or a set of manoeuvres from a database, executes it and feeds back the outcome and effectiveness of this manoeuvre in a certain air combat scenario. If a certain manoeuvre has proven to be unsuccessful in a certain scenario, its weight is lowered in the database, making the probability of this manoeuvre re-occurring in this particular situation lower. This way, air combat behaviour by an autonomous agent can be dynamically trained.

3.3 Suitable Software

From the start of this project it was acknowledged that a visualisation of UCAV behaviour must be realised. As described in the problem statement, the MVP should ideally be developed through low cost and even open source software. This initiated a preliminary research in suitable software for the realisation of the MVP. The software packs are existing applications in the aerospace sector or programs that offer room for 3D simulation and development. The software packs that are evaluated are summed up below:

Stage

One of the two in house environments of the NLR is Stage. Stage is used in the NLR to work with CGFs for the simulation of combat scenarios. The F4S mentioned in the previous chapter operates with the Stage software. The Stage environment is not portable and working in Stage comes with relatively large technical barriers (for interns) making rapid prototyping challenging. Furthermore, the ease of implementation of different, externally developed modules, models or software is relatively due to the fact that Stage is off-liimhe software however accurately represents air combat scenario dynamics and the aerodynamics models are furthermore accurate. The F4S is aimed to be recognisable for aircraft pilots, which makes the ease of use for the pilot or operator high.

LWACS

LWACS, short for Light Weight Air Combat Simulator, is a simulation program originally developed by an employee of the NLR, to simulate a large amount of air combat scenarios. This program was developed for a machine learning technique that can train an artificial intelligence agent into selecting the right combat manoeuvre in each combat situation. There are no software licenses bounded to this software and it is fully programmed in Java and thus to a large extent adaptable by the user. Since its focus lies on housing the machine learning technique there are no accurate flight dynamics models included in this software. A primitive visualisation of the battlefield and the aircraft is however included.

X-Plane

X-Plane is a flight simulator that includes a broad range of aircraft models, from commercial aircraft to military aircraft. It furthermore possesses scenery models which are modelled after real life data of the earth's surface. X-plane comes in a desktop version on mac OS, Windows and Linux. The mobile version is available for Android, iOS and webOS. X-Plane uses a unique aerodynamic model which named the "blade element theory" [76]. In comparison; a number of different flight simulators utilise empirical flight data about various aircraft to calculate or determine the aerodynamic forces on the model in the simulation. Depending on the amount of data that is available for a certain aircraft, well performing models can be realised in software. X-Plane, on the contrary, breaks an aircraft model into separate parts and calculates the aerodynamic forces for each individual part (which consequently can be broken into smaller sub-parts). In a later stage, all the forces on the different sections are combined resulting in accurate representation of forces on the aircraft model which in turn leads to realistic flight behaviour. Users of X-Plane are encouraged to design their own aircraft, and design software is included with the program. This has created an active community of users who use the simulator for a variety of purposes. X-Plane has possibilities for the customisation and creation of aircraft models by its users and the software comes with a plugin interface to create new features. Because of the sophisticated flight dynamics model, X-Plane has an option for FAA certification to allow for training of pilots with aid of the software [77].

Microsoft Flight Simulator

Microsoft Flight Simulator consists of a series of flight simulators especially designed to work with Windows operating systems. The long history and popularity of this flight simulator have led to several add-on packages that allow for the tweaking and development of the flight simulator by third party agents. The most editable parts of the simulator are the scenery and the aircraft models. A user can customise cockpit layout, external models, sounds and scenery through simple programs such as notepad. The flight dynamics model is stored in the form of a large amount of parameters in one single file. All these individual parameters define the aircraft's flight behaviour and can easily be modified.

Unity 3D

Unity is a cross-platform game engine for developing video games for PC, consoles, mobile devices and websites. Unity focuses on portability and currently has over 20 target platforms on which it can operate. The program is free for personal use, but comes with a license for commercial use. The program is suitable for rapid prototyping and developing simulations of all sort. The program is user friendly and game dynamics can be programmed with the aid of the C# scripts. The integrated graphics engine allow for realistic graphics. In the case of aeronautical applications it may offer room for realistic flight dynamics models, but these have to be programmed by the user, which can be exceptionally complex. In contrast, the user can tweak and modify all parameters within the program, making it suitable for building systems or simulations from scratch.

FlightGear

FlightGear Flight Simulator is a free, open source (under the GNU General Public License) multi-platform flight simulator developed originally by David Murr. There are versions available for multiple operating systems including Windows, MacOSX and Linux. Since it is an open source flight simulator, the source code is made available through a repository and programmers with knowledge of C and C++ programming language can develop and alter core functionalities of the flight simulator itself. The flight dynamics are, in a similar fashion to X-Plane, scripted in .xml files with a large amount of parameters denoting the flight behaviour. FlightGear can be fully transformed towards individual needs and this has led to the fact that FlightGear is widely used for academic and professional research [78] [79] [80]. FlightGear can for example be interfaced and controlled with the aid of MatLab and Simulink [81]. Consequently, the fact that FlightGear is freeware makes it suitable for researchers with a small budget. FlightGear lives from a large community that develops and maintain the software. There is a dedicated wiki available on the organisation and functions of FlightGear along with tutorials for programmers.

3.4 Requirements Eliciation

The artist's impressions in this section demonstrate the different points of view the stakeholders have had regarding what a MVP of the SDCS should be able to do and what it should look like. In several sessions following these preliminary brainstorm sessions it was decided that a new starting point for the MVP would become a platform or tool for the NLR to further develop envisioned functionalities of the SDCS in. The MVP will form a framework or a platform which allows for modules (that are later created in different stages of the feasibility study) to be "plugged" into the MVP. In practice this means, that if an interface with VR and motion gloves is desired, this can realised on top of the MVP. Similarly, an intelligent controller can use the output from the MVP and translate it to the desired output, which is then again interpreted by the MVP. The MVP will in this form offer a demonstrational and testing platform which allows for growth as research in the SDCS is continued after this thesis is completed.

This new product concept was agreed upon by all main stakeholders as being a suitable starting point of a pre-feasibility study. However, the wishes of the stakeholders were still widely diverse, which in turn necessitated the usage of different types of software. This resulted in a chicken-egg dilemma. It was decided that the starting point of this research should be determining how the different wishes of the stakeholders could be managed and mapped to consequently decide which wishes are to be included in the MVP. In other words, the stakeholders and the author need to agree on what "Must-have" requirements of the MVP are. The different interviews with the stakeholders have resulted in diverse preferences; Hans Heerkens was interested in the presentation of the concept and the simulation while Jan Joris was interested in the technical side and control theory of the CMMS of the SDCS. Asking each individual stakeholder what needed to be included in the MVP through interviews and conversations proved to be difficult; the results would have turned out to be diverse and there would be a need for levelling and balancing these wishes and requirements through thorough discussion. Instead, it is chosen upon to utilise the Kano method as described in chapter 2 of this thesis. Before describing how this method was applied it is important to state that this method will not be utilised for statistical purposes. Originally, the Kano method was developed to determine in which manner different product attributes influence customer satisfaction. To establish valid conclusions based on the outcome of this method, a large sample size must be obtained. In this case, no statistical analysis can be performed using the Kano method since the sample size would be the number of stakeholders of this project (as of now, that number is 3). Logically, no conclusions can be drawn from just the results of the Kano analysis and a discussion session about the outcomes will remain inevitable.

However, this method is different from regular structured or semi-structured interviews. In the latter, a stakeholder is asked about what is desirable to that certain stakeholder. From experience of the author, these questions may lead to the stakeholder mentioning a too large amount of product aspects as "must-have"-requirements and furthermore treating every aspect as being equally important. For the

MVP, there must be an agreed upon set of product requirements and therefore, concessions must be made. The Kano method may be useful in this situation since it does not only ask what a stakeholder thinks should be included in this MVP, but also attempts to elicit the reaction of the stakeholder in the hypothetical situation where that certain aspect is left out of the MVP. In this way, the Kano method may provide a means of mapping stakeholder's interest and determining where the overlap between the stakeholders' interests lies.

The first stage of the Kano method requires a questionnaire to be set up which includes all relevant product features for a MVP of the SDCS. To ensure that all wishes of the three main stakeholders are included, several sessions with Jan Joris Roessingh, Gerald Poppinga and Hans Heerkens have been organised. These sessions have led to a survey with 26 different product features, these are all the features the stakeholders have mentioned to be relevant to the MVP of the SDCS. Once the survey has been sent, the questions can no longer be adapted. Therefore, "remarks"-sections have been included along every product feature and at the end of the survey so the interviewee may provide their thoughts and comments on the question itself, or the subject that this question covers. The full survey is included in appendix C. The survey has been filled in by the three main stakeholders. The surveys returned by the 3 stakeholders are included in appendix D through F. The results of the survey help to categorise this certain product feature into the categories as they are defined by the theory in chapter 2 of this thesis. The categories are: "must" (M), "attractive" (A) "proportional" (O), "reverse" (R) and "questionable" (Q). The answers of the stakeholders in the surveys are analysed and categorised accordingly. The result of this process is summarised in table 3.2

#	Description	A	O	M	I	R	Q	Total
1	Aerodynamics	-	-	1	1	1	-	3
2	Aircraft parameters	1	-	-	2	-	-	3
3	Aircraft equipment	2	-	-		-	-	3
4	Aircraft armament	1	-	1	1	-	-	3
5	Damage models	1	-	-	1	1	-	3
6	Multiple friendly UCAV control	1	-	-	1	1	-	3
7	Multiple adversary UCAV control	1	-	-	1	1	-	3
8	External friendly flight controller	1	-	-	2	-	-	3
9	External adversary flight controller	1	-	-	1	1	-	3
10	Variable abstraction input commands	1	-	2	-	-	-	3
11	Command frequency variance	1	-	2	-	-	-	3
12	Latency on data link	-	-	3	-	-	-	3
13	Disruptions on data link	2	-	-	1	-	-	3
14	Flight parameters display	-	-	3	-	-	-	3
15	Filtering flight information	2	-	1	-	-	-	3
16	Set terrain of battlefield	1	-	-	1	1	-	3
17	Weather and time of battlefield	1	-	-	2	-	-	3
18	Semi-stationary adversaries	1	-	-	2	-	-	3
19	Interaction with adversary	1	-	-	2	-	-	3
20	High quality graphics	-	-	1	2	-	-	3
21	Different points of view	2	-	1	-	-	-	3
22	AR or VR techniques	1	-	1	1	-	-	3
23	Input control devices	1	-	1	1	-	-	3
24	System requirements	2	-	-	-	1	-	3
25	No licensed software	1	-	1	1	-	-	3
26	Expandable by other developers	1	-	2	-	-	-	3

Table 3.2: *Answers of the Kano surveys. The numbers denote how many of the respondents answers filled in the corresponding category per product feature.*

The results show that there is both overlap and discrepancy between the wishes of the stakeholders, as

expected. In chapter 2 it has been stated that a MVP should fulfil all the "Must" requirements that have been set. Therefore, the first stage is to identify the product features that have been marked as "Must" by at least one stakeholder. The focus was - in this stage - on the product features with the most mutual importance for all stakeholders. 1 set of product features contains the requirements that have been marked as a "Must" product feature by all stakeholders, or was marked as a "Must" product feature by at least one stakeholder and marked as an "attractive" product requirement by the other stakeholders. These are the requirements that all stakeholders value either a lot, or think it must be included at least. The second set contains the product features that have been marked by at least one stakeholder as being a "must" product feature, but has also been marked as "indifferent" or even where one stakeholder wishes the reverse of this requirement.

The first set containing must and/or attractive only are the product features with numbered 10, 11, 12, 14, 15, 21 and 26. The second set contains the requirements numbered 1, 4 20, 22, 23 and 25. Based on these results, the product features were translated into requirements and ordered with aid of the MoSCoW method. The following draft prioritisation in table 3.3 is the result of this process.

Category	**#**	**Description**
Must	10	Allow the input commands for the UCAV(s) in to be variable in nature or abstraction level
	11	Allow to vary the frequency with which the user can provide input to the UCAV(s)
	12	Allow to vary the amount of latency in the datalink between the operator and the UCAV(s)
	14	Allow to display flight parameters of the UCAV(s)
	15	Allow to filter what information is presented to the user
	21	Allow to select different points of view
	26	Allow for editing and extension by other developers
Should	1	Incorporate realistic flight dynamics, aerodynamics and physics
	4	Allow the user to set the armament of the UCAV(s) and other aircraft
	20	High quality graphics, a detailed environment and detailed models
	22	Allow visualisation through techniques as Augmented Reality or Virtual Reality
	23	Allow for different types of input devices
	24	Allows to run on different computers without demanding high system requirements
	25	Allow for operation and development through unlicensed software
Could	2	Allow to set the flight parameters of the UCAV(s)
	3	Allow to set the equipment of the UCAV(s)
	5	Include damage models influencing flight performance
	6	Allow for control of multiple friendly UCAVs
	7	Allow for control of multiple adversary UCAVs
	8	Allow for an external flight controller for friendly UCAVs
	9	Allow for an external flight controller for adversary UCAVs
	13	Allow to simulate disruptions or errors on the datalink between UCAV(s)
	16	Allow to set the terrain of the battlefield
	17	Allow to set the weather type and time of day
	18	Allow to add one or more (semi-)stationary adversaries
	19	Allow to adjust the parameters that govern interaction of adversary forces

Table 3.3: *Prioritisation of the requirements according to the MoSCoW method.*

The drafted suggestion of requirements in table 3.3 was presented to the stakeholders of the NLR. At the end of the meeting it was agreed upon that the 7 requirements that are posed in the "Must"-category of the MoSCoW method were the most important requirements to focus on during the development of the MVP during this pre-feasibility study. It should be noted that product feature number 1 (flight dynamics and aerodynamics) and number 24 (system requirements) have contrasting results in the different surveys.

Requirement numbered 24, was marked as "reverse" by one stakeholder. In his comment section he stated that as long as the program runs on a Windows based computer it is enough. The other stakeholders have marked it as "attractive". The focus will lie on developing a Windows based application,

but the important part is that the software should also be able to run on systems that do not have sophisticated graphic cards or hardware.

Product feature 1 was marked by one stakeholder as a "Must" while another has marked this as "Reverse"; these two are opposite indications. This discrepancy was resolved in the meeting with the stakeholders and it turned out that the survey question corresponding to this product feature was interpreted differently by different stakeholders.

It is important to note that this may have happened with more product features in the survey, leading to distorted results. However, the Kano method was solely used to form a draft list of requirements. In chapter 2 it was mentioned that discussion of results was inevitable and therfore this list has been presented to the stakeholders of the NLR. The "Must-have" in the list denoted in table 3.3 are agreed upon by the stakeholders of the NLR. It is not said that the other requirements are completely neglected, but they are given a lower priority level during the development of the MVP.

3.5 Further Brainstorm Sessions and Concepts

What should further be noticed is that the established "Must-have" requirements are related to the functionality of the MVP. These requirements do not influence the form and shape or "look and feel" of the MVP to be produced. In other words, one could realise requirement 12 (the requirement about latency) in both X-Plane and Unity. In the problem statement it was mentioned that the MVP serves two goals, to demonstrate the principles of the SDCS and to further develop functionalities of the SDCS. A pure demonstrational platform may have a different look and feel than a platform meant for solely performing tests or developing functionality accurately. It is therefore useful to differentiate between what a demonstrational platform and a test platform entail and consequently decide on a more specified direction to work towards in this thesis. To realise this, three different product concepts were made and presented to the stakeholders. The concepts aim to give shape to the implementation of requirements by combining three items: The established product requirements, the possible software packs and the related work section in chapter 2 of this thesis. Each concept represents the identity of the different shapes of the MVP. One resembles a pure demonstrational platform, one a pure test platform and the other a blended form. The concepts act as mood boards to convey a message about the possible "look and feel" of the different products along with possible users and possible realisation possibilities. The resulting product concepts are presented below. The mood boards as presented to the stakeholders of the NLR are included in appendix G to I

Concept 1 – "Demonstrator"

Description

The "demonstrator" features high quality graphics and a clear and attractive interface. The user should be able to quickly grasp the interaction with the program. The programs focus does not lay on accurate flight dynamics models or realistic flight data, but does allow for virtual reality or augmented reality control methods to evoke an increased engaging and immersive user interaction.

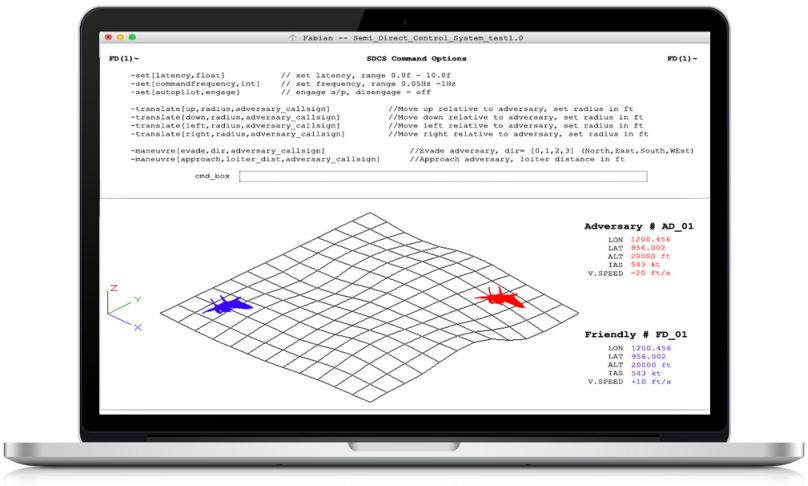
Potential User

Users of the demonstrator may include a potential project associate or potential investor. These users may be interested in the project but need to be convinced of the SDCS control concept in a fashion similar to an elevator pitch. The user may not have a substantial amount of time to look at the demonstrator and thus the demonstrator must be successful in demonstrating the control concept along with making an impression that stands out and lasts.

Realisation

The realisation of the demonstrator can be performed in for example Unity 3D. There may be no need for realistic flight dynamics, making the use of flight simulators perhaps to sophisticated. The user should not be bothered with the complexity of controlling an aircraft while interacting with the demonstrator. The interface of this concept is inspired by that of NLR's MUST and the Aerosim RC interface. The user interface should be user friendly and sophisticated. Unity furthermore has excellent capabilities for virtual reality support and support of for example Leap-motion gesture controllers. The simulation in Unity allows for rapid prototyping and quick adaptation of simulation aspects without having to perform calculations or coding to maintain realistic flight behaviour.

Concept 2 - "Test Platform"



Description

The “test platform” puts less emphasis on looks and interface. There is a larger amount of numbers and data presented on screen which makes it more difficult in use except for those who thoroughly know the program. The “test platform” harnesses accurate flight dynamics and is suitable for recreating air combat situations as close to reality as possible. The “test platform” may be less attractive for pure demonstrational purposes due to its complexity.

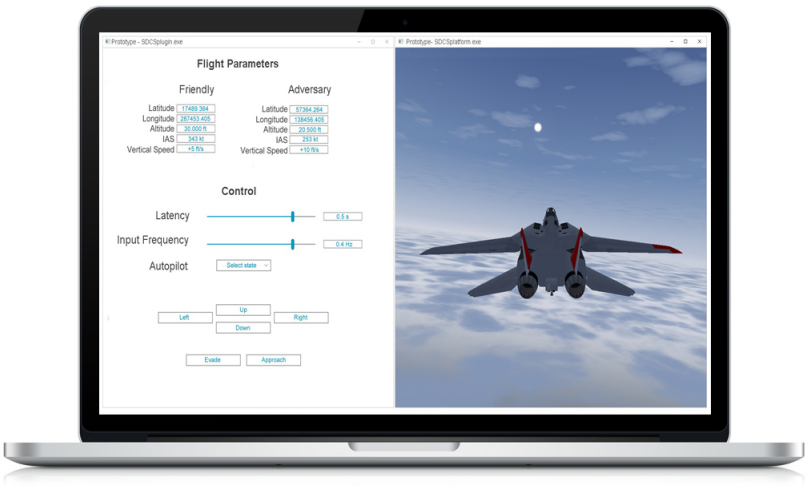
Potential User

A potential user for the “test platform” is a researcher or scientist (possibly from the NLR) who is solely interested in the mechanics and working principles of the system. He knows the ins and outs of the system and does not necessarily care for an attractive look and feel, as long as it produces the data needed for his research.

Realisation

The inspiration for the test platform comes from the LWACS software described in the previous section along with the MAVproxy software described in chapter 2. Both LWACS and MAVproxy utilise a grid-like representation of the battlefield and the aircraft. In the realisation, a flight dynamics model of a flight simulation can be used to perform the necessary calculations for the flight data. The command line interface represented in this mock-up is furthermore inherited from MAVproxy: It uses the python command line interface in which the commands to be performed need to be typed. In a similar fashion, a researcher may be familiar with all the possible commands that the “test platform” supports.

Concept 3 - ”Blended Form”



Description

The "blended form" houses a balanced mixture of quality graphics, a clear interface and realistic flight dynamics. The interface is not per se focussed on attractiveness, but it aims at representing information in a logical way and to enable a rather quick grasp of the programs functionalities and that of the control concept. The graphics and flight dynamics aim to provide a close to reality representation of the UCAV in its environment.

Potential User

Potential users include UCAV operators or even F-16 fighters from the Dutch Air Force. These pilots have thorough knowledge of flight principles and dynamics. The pilot is possibly highly interested in seeing similar behaviour in the demonstrator as he would encounter during the operation of his own aircraft. However, the system must be relatively easy to understand. The pilot might need a short briefing of around half an hour to be able to effectively work with the system. The increased graphics enable him to see clearly what effects his actions have on screen.

Realisation

At first, the realisation can be performed in a flight simulator to acquire realistic flight dynamics and high graphics. The mock-up is inspired by the MAVPilot iOS application described in chapter 2 of this thesis. The screen is split in two, where one half provides a simple interface, a bit more evolved than a command interface. The other half of the screen represents the battle space in which the UCAV resides. Alternatively, the interface or representation of information may be done in a manner similar to that of Aerosim or that of NLR's Fighter 4 Ship to provide a pilot with a more familiar look and feel.

3.6 Presentation of Concepts

The three concepts above were presented to the stakeholders at the NLR. It became clear that there was more interest in a blended form rather than a pure demonstrational or pure test platform as presented. One stakeholder has expressed that the blended form had his interest, but he also wanted to be able to switch towards a view of the scenario as was depicted in the "test platform"-concept, where there is no chase view of the aircraft, but an "overview" of the combat situation from a point that is far away. Another remark was that the only visual differences between the "blended form" and "demonstrator

mock-ups was the interface, but after explaining that the flight dynamics differ heavily between these concepts it became more clear that there was an incline towards the blended form mock-up over a pure "test platform" or pure "demonstrational" platform.

Although this choice only slightly converges towards a more elaborated product idea, it gives a clue as to which direction is going to be taken. An important moment in this meeting is that the stakeholders have expressed their preference towards using the FlightGear software. This choice was based on the fact of a large part of the "Should-have" requirements. Furthermore, in the eyes of the stakeholders FlightGear promises to be a long term solution to the problem at hand and allows the NLR to implement growth and further research since a flight simulator comes with a substantial amount of relevant aerospace functionality built-in that would otherwise, for example when using Unity 3D, have to be built in. FlightGear satisfies a couple of "Should"- requirements from table 3.2. One "Should-have" requirement was to incorporate realistic flight dynamics, aerodynamics and physics. FlightGear comes with realistic Flight Dynamics Models (FDMs) built in. Another "Should-have" requirement was that the user should be able to set the armament of the UCAV(s) and other aircraft. FlightGear includes (fully functioning) weapons and the user can set the armament type of various military aircraft that are included in FlightGear. High quality graphics, a detailed environment and detailed models are also included in FlightGear while at the same time also providing options to run on computers with less powerful hardware by turning off various rendering and image options. Therefore, FlightGear is able to run on an average consumer based laptops. VR and AR techniques along with input devices other than mouse, keyboard and joystick are not in a developed phase in FlightGear. Since FlightGear is an open-source and fully editable flight simulator, it could be tweaked to comply with usage of VR, Augmented Reality (AR) and different input devices. This is agreed to be a subject for later research. What made FlightGear furthermore a favourable option is the fact that it is completely open-source and can be operated without any license, making it very suitable to use without posing restrictions on development, usage and budget.

This choice was a key moment in the Ideation phase as it allows to focus on how the "Must-have" requirements should be realised in the chosen software. As mentioned earlier, the "Must-have" requirements can be fulfilled in almost every software pack, but now that the choice is made for an environment and tool to develop this in, the Specification phase can be initiated.

3.7 Conclusion

This chapter has provided the necessary information to answer sub research question 3: *What is the goal of a MVP of the SDCS?*. The goal Mock-ups have been set up and evaluated with the stakeholders. This has led to a more elaborated idea of the MVP and has furthermore specified more between the demonstrator and test platform balance. Sub research question 4 was formulated as follows: *What are the most important elements of the SDCS to be included in the MVP?* This question has been answered by setting up and prioritising the requirements with the aid of the Kano method and the MoSCoW model. Sub research question 5 was formulated as follows: What are suitable methods and tools for developing an MVP of the SDCS? This research question has been answered by considering multiple options for the realisation of the MVP and eventually, one software pack was selected as the most favourable option to develop the MVP in. The result of this section will serve as input for the next phase in the Creative Technology design process. This thesis will continue with specifying how the "Must-have" requirements should be realised in FlightGear.

4 Specification

This chapter constitutes the second phase of the Creative Technology Design Process: the Specification phase. The Specification phase aims to give shape to the requirements and functionality as specified in the Ideation phase. A number of mock-ups are utilised to shed light on a multitude of options for realising the requirements. The outcomes of the Specification phase will serve as design artefact for the Realization phase of this thesis. Along with the following phase, the Realisation phase, the Specification phase will attempt to answer sub research question 6.

4.1 System Architecture

The system architecture of the MVP evolves from the 'blended form' mock-up in the Ideation phase. That mock-up is depicted again in figure 4.1.

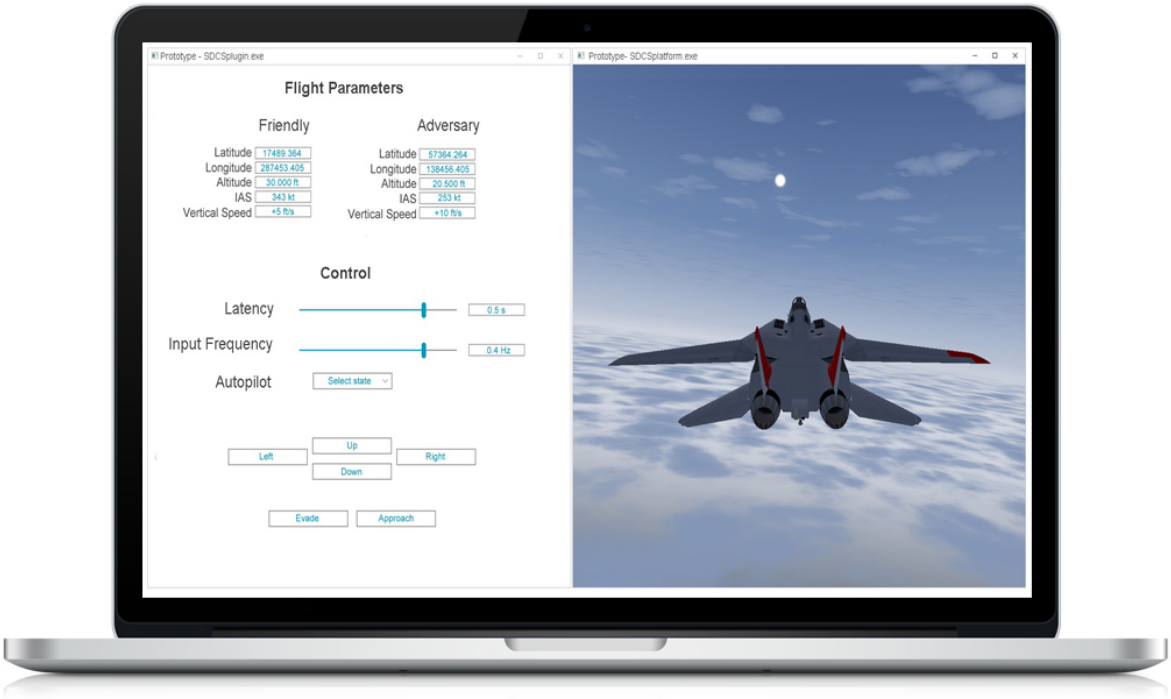


Figure 4.1: *'Blended-form'-mockup from the Ideation phase.*

This mock-up suggests that the MVP should consist out of a small interface in which the user will provide his or her input. The right part of the screen is the visualisation of his or her actions in FlightGear. This leads to the specification of the system architecture as depicted below in figure 4.2.

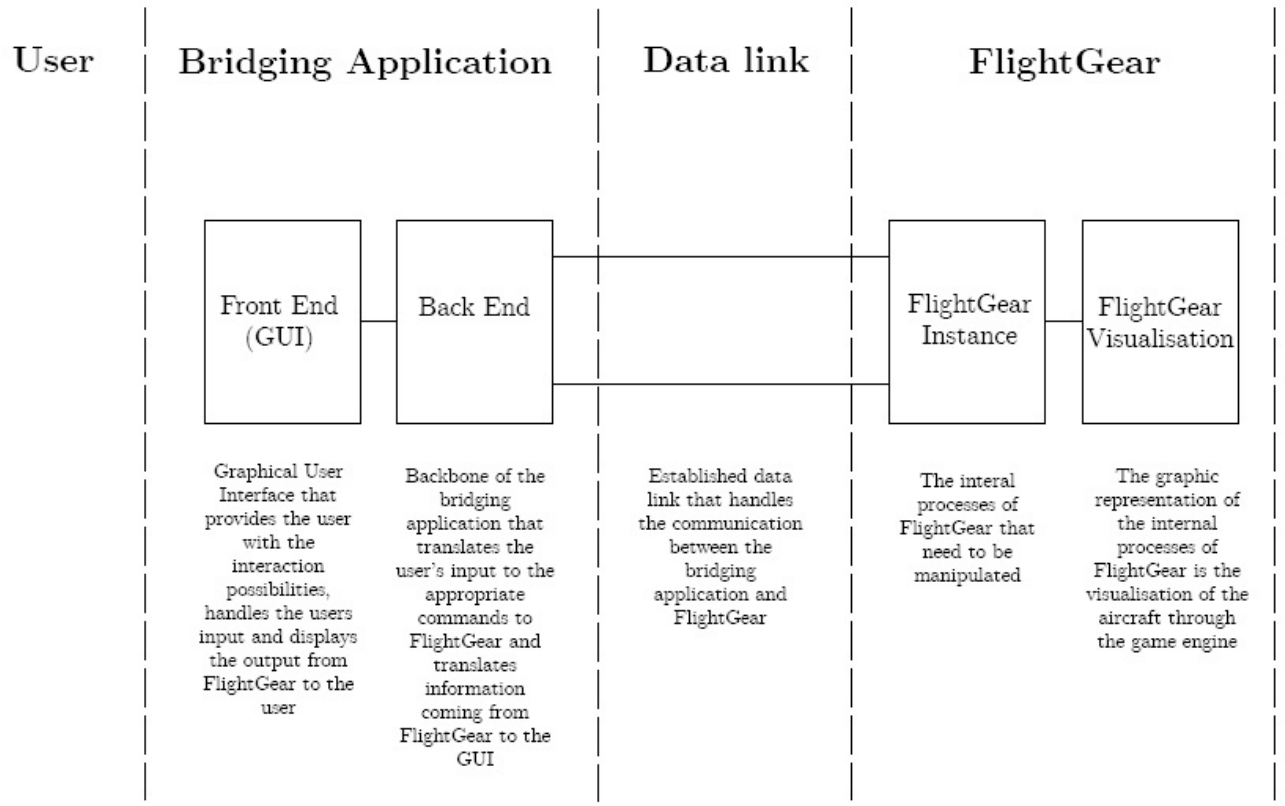


Figure 4.2: *Overview of the Bridging Application/FlightGear architecture of the MVP of the SDCS*

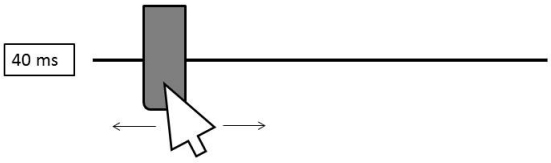
The different subcomponents of the system architecture and their relation are described below. The user is the person interacting with the MVP. He or she will mainly encounter the subcomponents 1 and 5 of figure 4.2. Subcomponent 1 is the user interface, which represents the left side of the screen in figure 4.1. The user interface contains the buttons and data fields with which the user directly interacts. Subcomponent 5 is the visualisation of the flight behaviour in FlightGear, and represents the right side of the screen of figure 4.2. In between there are 3 subcomponents that need to be realised to establish the MVP. Element one of this architecture is labelled as "Bridging Application". It is called such since it forms the bridge between the user and FlightGear. The subcomponents of this element are denoted by the numbers 1 and 2 in figure 4.1 and denote the interface and the application that translates the input of the user into commands that can be directly fed into FlightGear. This subcomponent is furthermore responsible for translating the data that is coming from FlightGear to the user interface so that the user can comprehend and interpret this data in a natural manner. The next subcomponent is the data link, denoted by number 3 in figure 4.1. This data link handles the communication between the bridging application and FlightGear. The element "FlightGear" represents two subcomponents, the internal processes of FlightGear and the visualisation based on these internal processes.

4.2 Specification of "Must-have" requirements

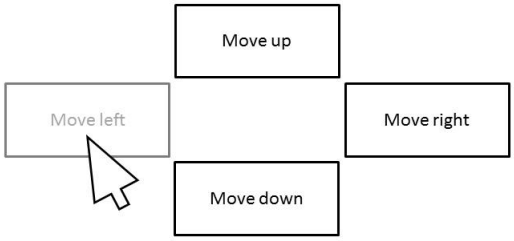
This section handles the translation of the requirements into intended functionality. The "Must-have" requirements are visualised through the use of PowerPoint mock-up storyboards of functionality and user interaction. These are presented to the stakeholders in a session to demonstrate how the functionalities could be realised. This session furthermore allows for input and feedback from the stakeholders, once confirmation of the specification is given, these specifications will be realised in the chapter "Realisation" following this chapter.

4.2.1 Requirement 1: Variable Latency

As a recurring theme regarding UCAV control, the induced latency on the data link caused by the geographical separation of the GCS and the UCAV needs to be simulated in the MVP. The storyboard depicted in figure 4.3 denotes the storyboard depicting how the user can set the latency in the MVP.



Use the slider to set the amount of latency that is induced in the control line



Select a new command for the UCAV to be performed

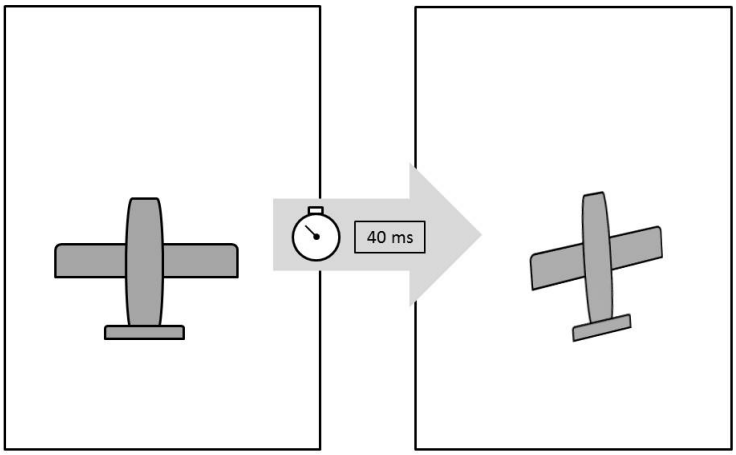
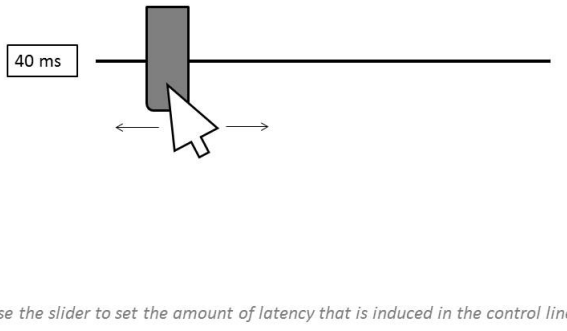
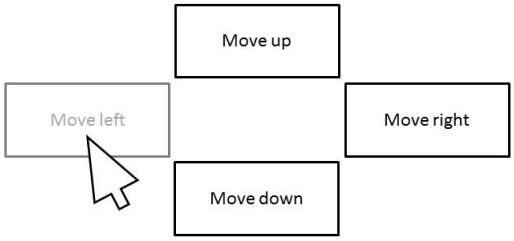


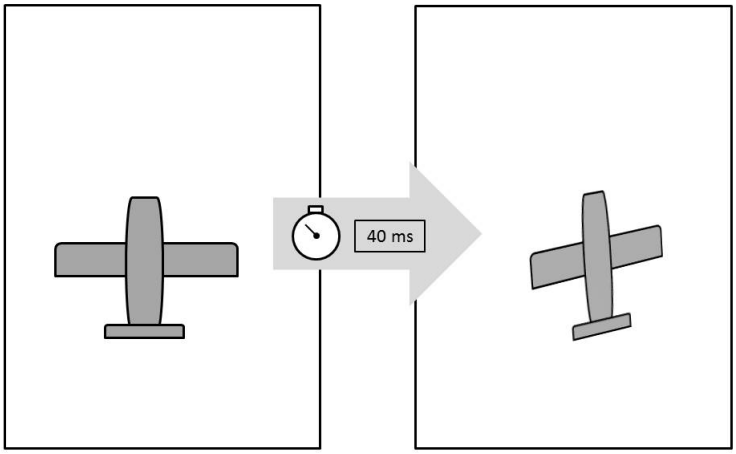
Figure 4.3: *Storyboard for requirement: 'variable amount of latency in the data link between the operator and the UCAV(s)'*

The stakeholders noted that in this situation, the latency is solely realized in one direction. The commands end up at the UCAV with the desired latency, but the visualization and the parameters that are fed back to the user should also be delayed, since it also takes 40 milliseconds to get data from the UCAV to the operator. This was added to the storyboard and resulted in the final storyboard depicted in figure 4.4

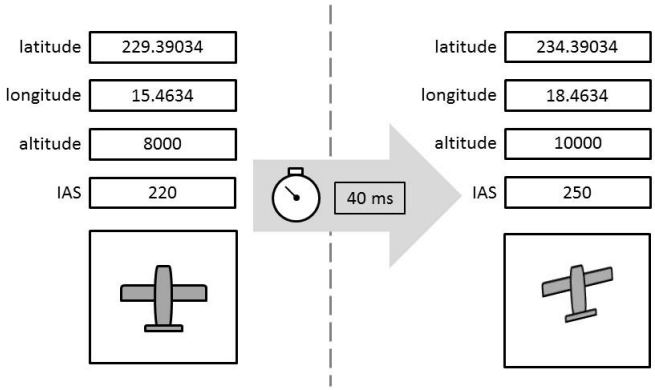




Select a new command for the UCAV to be performed



The selected action is performed by the UCAV on screen after the selected latency

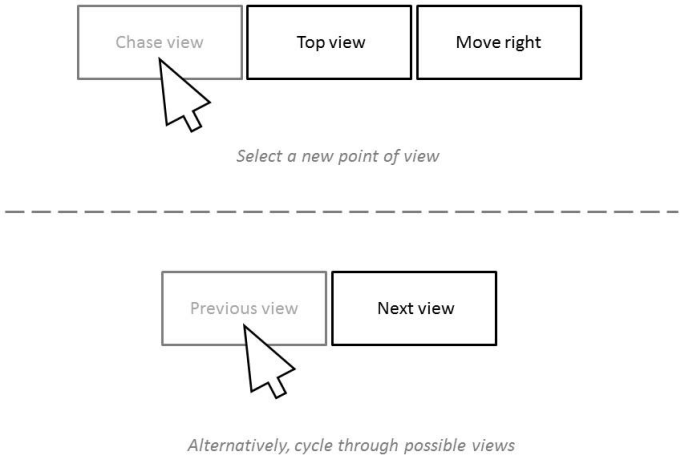


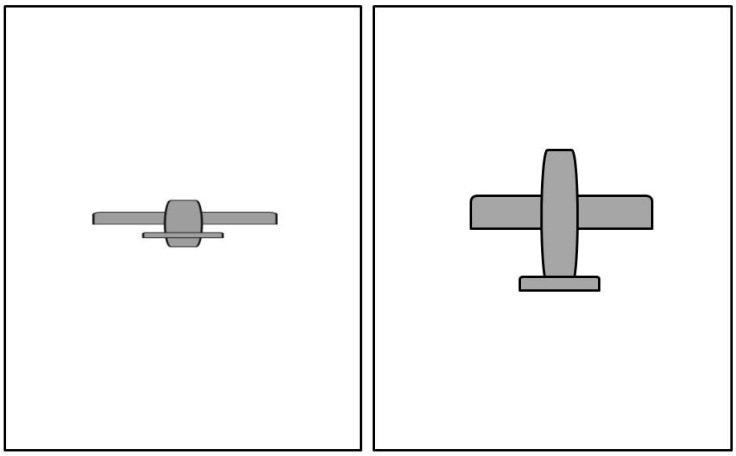
All parameters and the visualisation are presented to the user (refreshed) after the set delay

Figure 4.4: *Final storyboard for requirement: 'variable amount of latency in the data link between the operator and the UCAV(s)'*

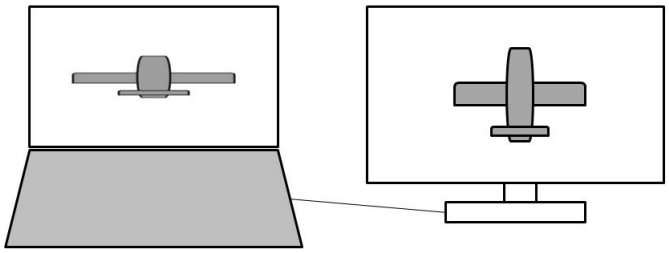
4.2.2 Requirement 2: Different Points of View

There should be different points of view available to look at the combat situation. The storyboard depicted in figure 4.5 denotes the storyboard depicting in what way the user can select different points of view in the MVP.





Viewpoint is changed accordingly on the main screen



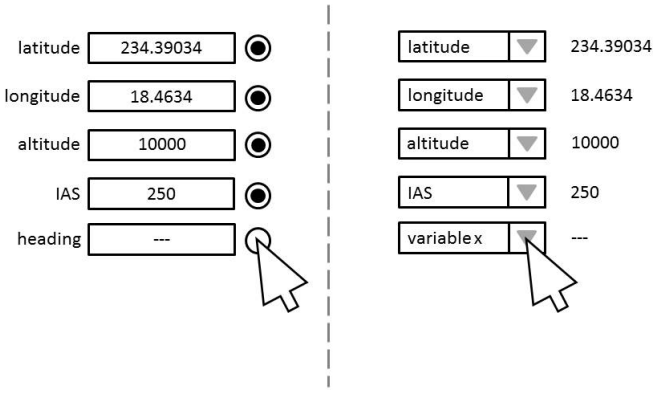
Alternatively, another view can be opened on another screen. Multiple screens for every view is also a possibility

Figure 4.5: *Storyboard for requirement: 'Allow to select different points of view'.*

The stakeholders noted that there was no consensus as to what exact views are desirable as of now. The cycling through views option will suffice for the MVP. There is a preference for an option to easily add views and disable pre-programmed views while working with the MVP.

4.2.3 Requirement 3+4: Selecting and Displaying Parameters

These two requirements are bundled together in one storyboard because the functionalities have overlap. The storyboard depicted in figure 4.6 denotes how flight parameters are displayed and how the user can select what information is displayed.



The user can select what information or which flight parameters are displayed with for example drop down lists out of which the parameters can be selected or radio buttons to enable/disable logging of parameters

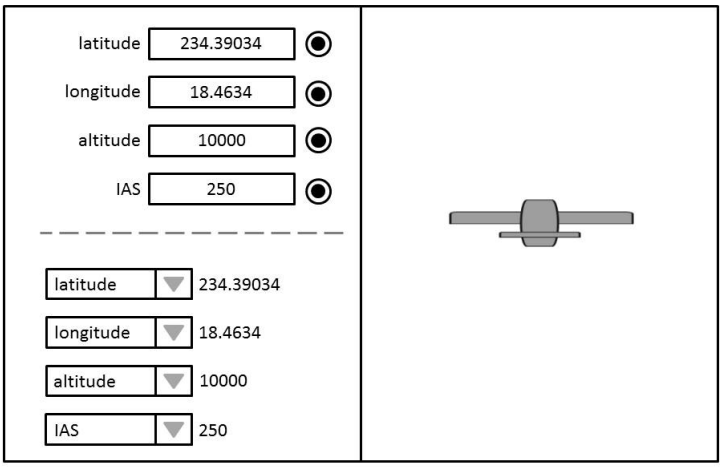
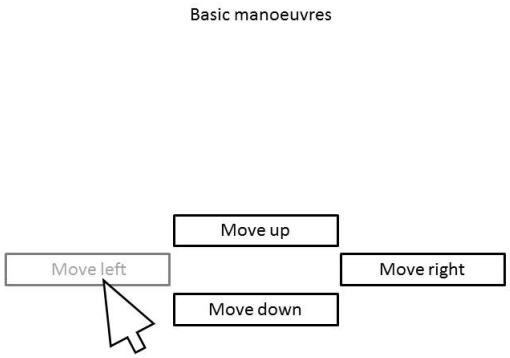


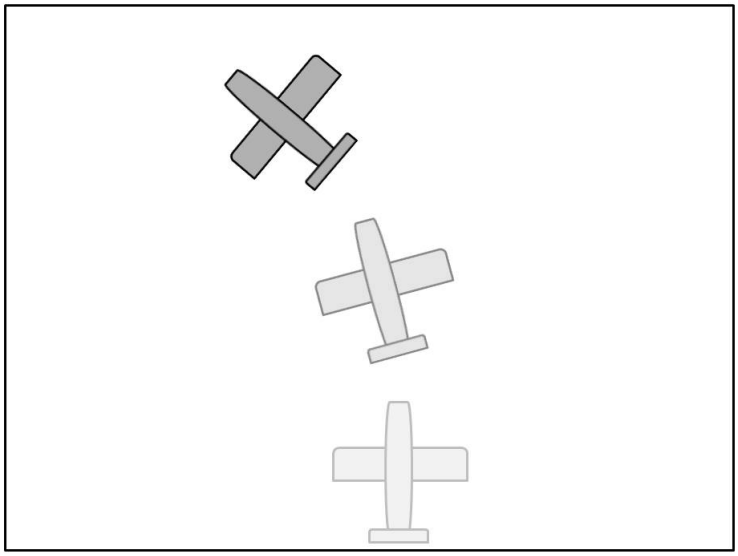
Figure 4.6: *Storyboard for requirements: 'Allow to display flight parameters of the UCAV(s) and allow to filter what information is presented to the user'.*

The stakeholders mentioned that the representation of the information and flight parameters may be more meaningful if they can be translated into a representation that is more intuitive. However to demonstrate that all parameters can be logged was said to be sufficient for the MVP.

4.2.4 Requirement 5: Variable Input Commands

This requirement handles the different commands that the operator should be able to issue in the MVP. The difference in abstraction level or nature is in the storyboards translated to different sets of manoeuvres. The first set of manoeuvres constitutes basic manoeuvres such as banking right and left and pitching up and down. These manoeuvres are not dependent on the position of the adversary UCAV. The second set of manoeuvres is depending on a control loop which constantly monitors if the desired value has been reached. The third set of manoeuvres includes movements which are depending on the position and movement of the adversary aircraft. The corresponding storyboards are depicted in figure 4.7

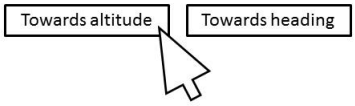




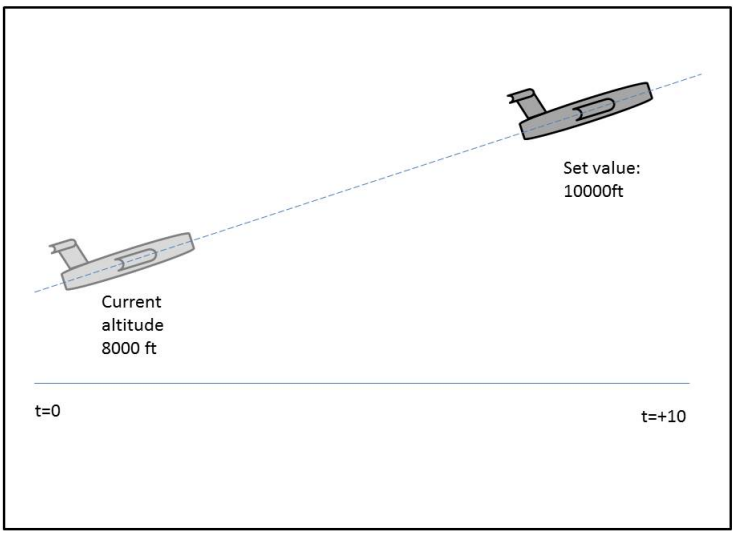
Direction based manoeuvres

User sets the appropriate values towards which the UCAV should manoeuvre, then confirms this by clicking the "towards" button

Set value: 10000 ft Set value: 278 deg



Manoeuvres based on either autopilot or controller inputs and outputs, a control loop is needed to perform these manoeuvres



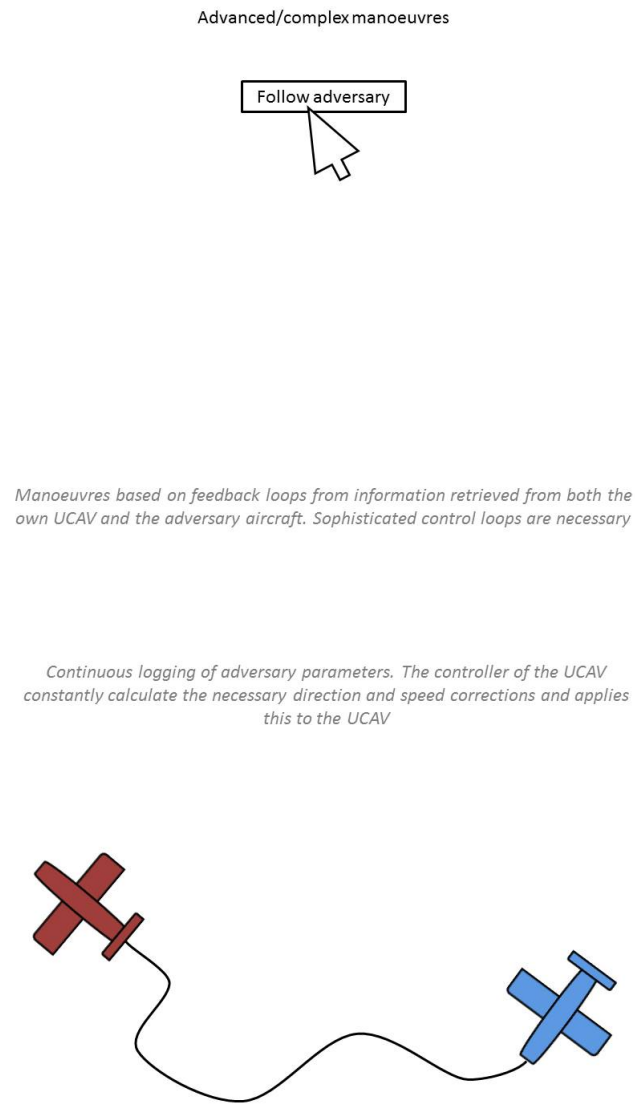


Figure 4.7: *Storyboard for requirement: 'Allow the input commands for the UCAV(s) in to be variable in nature or abstraction level'.*

The stakeholders noted that for the more complex manoeuvres, something like a control loop is needed and that this is a topic for research that they are highly interested in. For now, basic flight manoeuvres will suffice.

4.2.5 Requirement 6: Variable Input Command Frequency

During the meeting it was noted that the realisation of this requirement is difficult within this design space. Originally, the frequency with which the operator provides commands to the UCAV is dependent on the amount of autonomy the UCAV possesses. Therefore, the frequency with which the operator provides input is a matter of asking what the UCAV will do in the time that it does not receive any commands. In non-combat situations, it is desirable that the UCAV maintains its current flight patterns and does not require any operator influence to remain airborne. In combat situations, the UCAV has to explicitly be told what to do and as infrequently as possible undertake actions by itself (for example only in emergency situations). The most self-evident solution would be for the UCAV to fly straight ahead, that is, maintain the current heading and pitch. In the case of the MVP it is desirable to ensure that the UCAV will maintain a steady level flight throughout the idling periods, possibly with the aid of an uncomplicated controller. The UCAV should not fly into obstacles or fall from the sky during idle operation.

4.3 Conclusion

It should be noted that the last requirement: 'Allow for editing and extension by other developers' is not yet treated in this chapter. This requirement is a matter of program organisation and design choices that will be made in the next chapter of this thesis. The specification of the remaining requirements has been treated in this section by the means of storyboards. The next phase in this thesis, the Realisation phase, will utilise the outcomes of this chapter as leading design artefacts.

5 Realisation

This chapter forms the third phase of the Creative Technology Design Process, the Realisation phase. This chapter describes the implementation of the MVP. The realisation phase builds on the outcome of the Specification phase, attempting to translate the storyboards from this chapter into a working product. This chapter commences with a section on the realisation of the system architecture set up in figure 4.2 and will move on into the realisation of the established "Must-have" requirements.

5.1 System Architecture

5.1.1 FlightGear

As mentioned in Chapter 2 of this thesis, FlightGear is an open-source Flight Simulator. The flight simulator has been chosen for the MVP since it is largely editable by the user. What makes FlightGear useful for the development of the MVP is that the internal functions are housed in and governed by what is called the "Property Tree" of FlightGear. The Property Tree can be seen as the skeleton or central nervous system of FlightGear. The Property Tree is an overview and placeholder of all state variables that govern all behaviour in FlighGear through an intuitive tree-like hierarchy. This Property Tree allows for FlightGear's behaviour to be easily modified and read at run time. In other terms: one is able to "peek" under the hood of a FlightGear instance that is running on one's computer and directly access and change the internals of FlightGear.

The Property Tree keeps track of all the current values and states of all subsystems in FlightGear such as the weather system or the FDM utilised in FlightGear. The internal variables are presented in a hierarchical system using paths to categorise and group variables. The Property Tree can furthermore act as an interface between different subsystems of a single FlightGear instance and in addition, the outside world. Data of one sub-system can be exposed in the Property Tree, where it can be read and or modified by either other internal FlightGear subsystems or by an external application; in the case of the MVP, the Bridging Application will fulfil the role of external application.

For example, given the hypothetical situation where one desires to know the current deflection of the elevator of the aircraft in FlightGear. One first requires to establish the path to that variable in the Property Tree. In this case the path of the variable "elevator deflection" in FlightGear is denoted by:

```
/controls/flight/elevator
```

Once the path to the variable is known, one can verify the state of that variable by requesting a return with the pseudo code below:

```
#Read value from FlightGear
get-item fg[/controls/flight/elevator]
#Deflection is 0.1 downwards
>> -0.1
```

The data type and range of each variable are not denoted in the return sequence. FlightGear includes a Property Tree Browser inside an active session of FlightGear. The property Tree browser is accessible in-game by navigating to *Debug* > *Browse Internal Properties* and following the path to the desired variable. The property Tree browser is depicted in figure 5.1

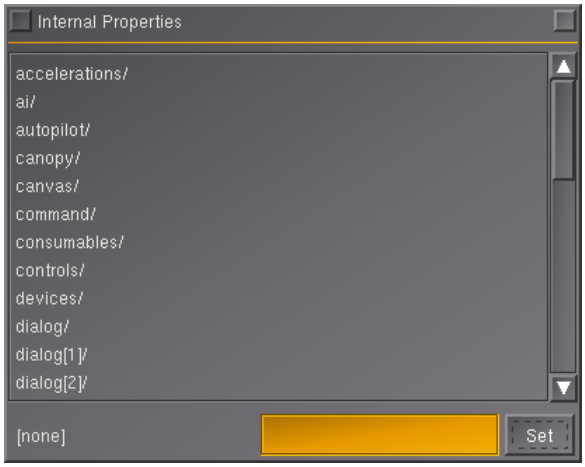


Figure 5.1: *FlightGear's internal Property Tree browser*

FlightGear's Property Tree browser allows for browsing the Property Tree in a categorized manner. The Property Tree displays current values, states and datatypes of each variable. The variable */controls/flight/elevator* has the data type 'double' and ranges from -1 to 1 (0 being origin). To continue, one can manipulate the value of the deflection of the elevator of the aircraft by modifying the value of the parameter in the Property Tree of FlightGear in runtime as follows:

```
set-item fg[/controls/flight/elevator] = 0.5
# Write value to FlightGear
```

This will move the elevator of the aircraft in the visualisation and the aircraft will commence to pitch in FlightGear. Consequently, one can see the value being updated in the Property Tree browser in the active FlightGear session. Controlling FlightGear is thus not per se a matter of complex coding, but merely knowing the path to the data one wants to read or modify. To demonstrate that literally all variables can be accessed, more paths to variables are depicted below:

```
Aircraft control surfaces variables...

/controls/flight/rudder
/controls/flight/rudder-trim
/controls/flight/flaps
/controls/flight/slats
/controls/flight/spoilers
/controls/flight/speedbrake
/controls/flight/wing-sweep

Orientation variables...

/orientation/roll-deg
/orientation/pitch-deg
/orientation/heading-deg

Engine controls...

/controls/engines/eninge[%d]/throttle
/controls/engines/eninge[%d]/starter
/controls/engines/eninge[%d]/fuel-pump
/controls/engines/eninge[%d]/afterburner
```

The Bridging Application will thus not alter the internal source code of FlightGear, rather it returns and manipulates values inside the Property Tree to control the simulation.

5.1.2 Data Link

As described in figure 4.2, a data link is required to communicate between FlightGear (or rather FlightGears Property Tree) and the bridging application. To realise this, the internal command server

of FlightGear can be utilised. The so called "telnet" server provides an entrance into the running FlightGear process that can be utilised to return and manipulate state variables and thus furthermore to run commands. This allows for external scripting and the creation of external operator GUIs.

```
telnet=(socket, bi, 60, 127.0.0.1, 5555, tcp)
# Instance of a connection to FlightGear in two (bi) directions, a frequency of 60 HZ
    and a TCP connection routed to port 5555 of the computer the FlightGear session is
    running on.
```

The data link will provide the communication between FlightGear and the Bridging application with the aid of a Transmission Control Protocol (TCP). With regard to the OSI model, TCP is placed in the Transport layer and Telnet is placed in the Application layer. TCP is a connection oriented service utilising a 'Three-way handshake' to establish a full-duplex connection between client and server. TCP provides a reliable connection in comparison with User Datagram Protocol (UDP), which is a lightweight transport protocol that provides unreliable transport between client and server [82]. This means that the client simply sends everything he needs to send towards the server, without telling the server that data is being sent and without checking if the data did actually arrive at its destination. This makes that UDP has no methods for dealing with packets that went missing, packets that arrived out of order and traffic control methods. TCP on the other hand has methods for acknowledging that a packet has arrived, asking for a re-transmission if a packet is lost and bit error checking [82]. These functionalities make that TCP is a reliable protocol for communication between FlightGear and the Bridging Application.

The code block responsible for translating user input to commands that FlightGear understands is represented by the "FlightGear" class. The "Telnet" and "FlightGear" class are written by GitHub user mrfranz [83] and reference to his work is mentioned in the code of the MVP.

5.1.3 Bridging Application

The final element of the system architecture is the Bridging Application. It holds the classes that manage the sending and receiving of commands through the Telnet connection with FlightGear and translating the commands to data that FlightGear comprehends. The bridging application furthermore houses the GUI with which the user will interact, realised with use of the Tkinter GUI- module of Python. The user interface is depicted in figure 5.2.

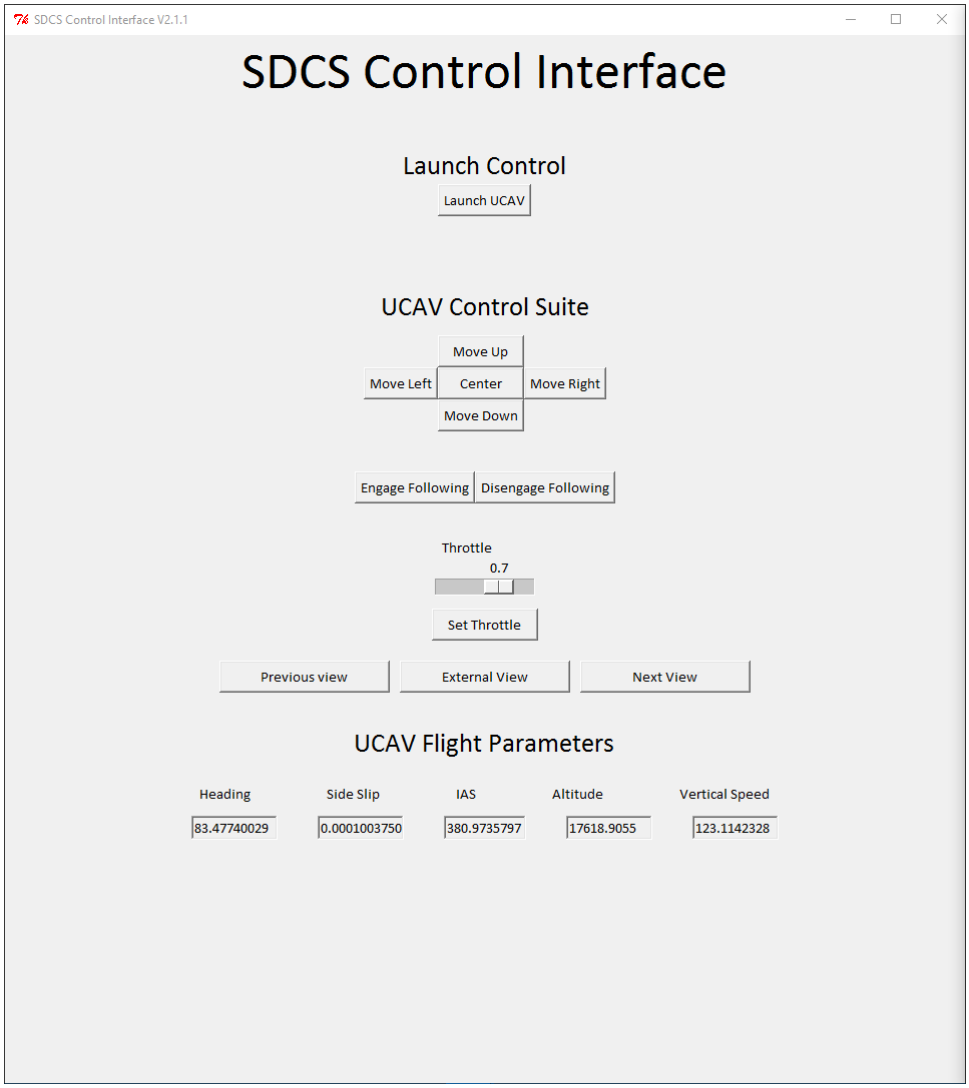


Figure 5.2: *Graphical User Interface of the MVP*

5.1.4 Aircraft

A custom test aircraft is realised for the MVP. This test aircraft is a modification of the F-14B Tomcat fighter jet that is included in FlightGear. As is mentioned, one can adapt all parameters of aircraft inside FlightGear. Therefore, to demonstrate this in the MVP, the existing F-14B is modified into specific test model. To achieve this, a new livery is created which houses the NLR logo and a non-transparent cockpit to suggest that no pilots are inside this aircraft. Furthermore, to demonstrate how the flight dynamics of these aircraft can be altered to match that of a UCAV, the "*f-14b-yasim.xml*" file of the aircraft can be adjusted. Yasim is one of the FDM types of FlightGear that this particular aircraft utilises. All aerodynamic and structural properties are denoted in this .xml file. As an example, the weight of the aircraft can be decreased with 200 kilograms to simulate the missing weight of the two fighter pilots and their gear (Note that this weight decrease has absolutely no scientific reasoning, it is merely an example how to edit the weight parameter) the weight of the aircraft is a matter of opening the .xml file and editing the right parameter:

```
<?xml version="1.0"?>
<!-- Grumman F-14B tomcat (formerly F-14A+)-->
<!-- References : -->
<!-- F-14 Tomcat by Denis Jenkins, Aerofax -->
<!-- Torsten Anft website on the F-14: -->
<airplane mass="41780">
```

To the following:

```
<?xml version="1.0"?>
<!-- Grumman F-14B tomcat (formerly F-14A+)-->
```

```
<!-- References : -->
<!-- F-14 Tomcat by Denis Jenkins, Aerofax -->
<!-- Torsten Anft website on the F-14: -->
<airplane mass="41380">
```

The FDM automatically takes the new weight to perform all aerodynamic approximations. Thus, changing the weight automatically reflects in an altered flight handling and -envelope of the aircraft in FlightGear. A number of screenshots of the NLR UCAV test aircraft are depicted in figure 5.3

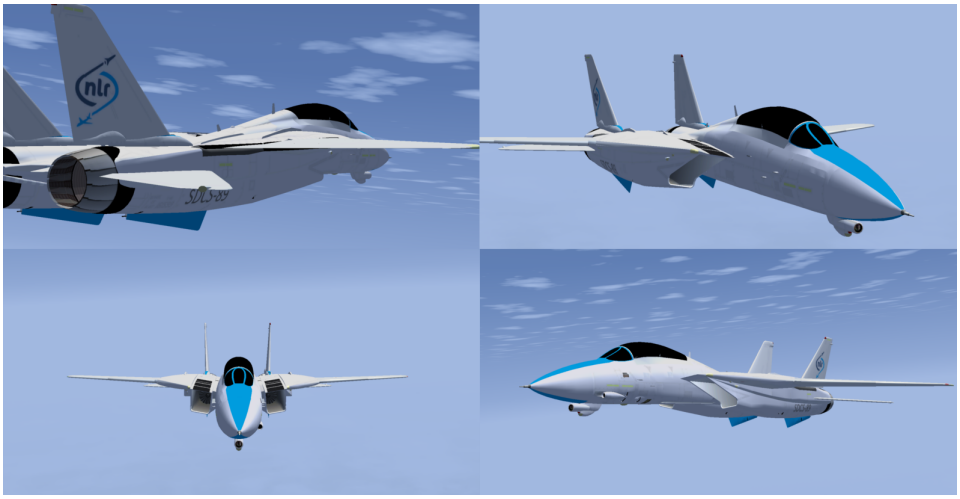


Figure 5.3: *NLR UCAV test aircraft designed for the MVP*

5.1.5 Interface and FlightGear

With a session of FlightGear and the Bridging Application running, the layout of the screen looks as depicted in figure 5.4

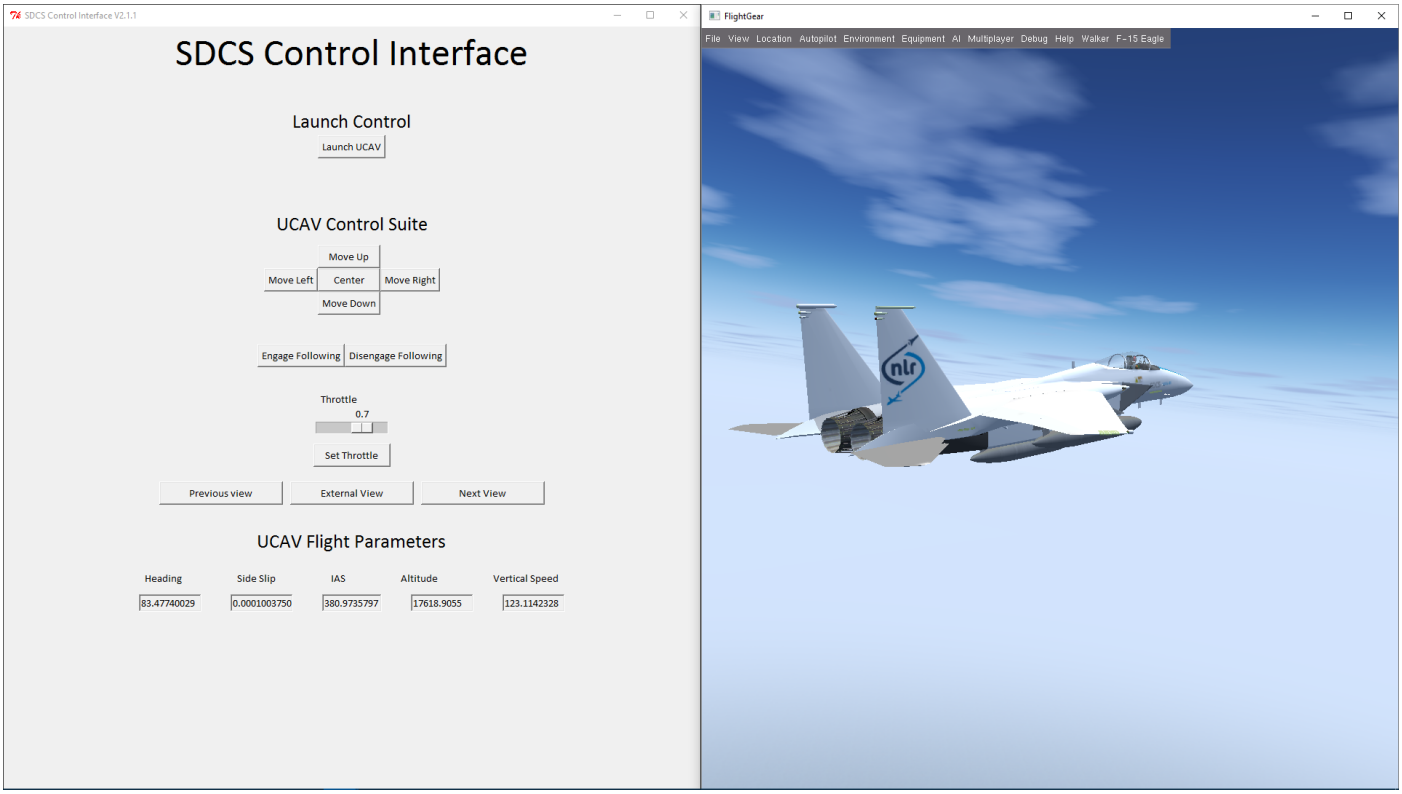


Figure 5.4: *FlightGear and the Bridging Application running*

5.1.6 Multiplayer

One aspect that has not yet been mentioned whilst being of key importance for the MVP is the possibility to fly and interact with multiple aircraft. To realise this, one can set up a Local Area Network (LAN) of multiple computers and a FlightGear server. This FlightGear server handles as a "virtual private world" where various FlightGear instances can interact by plugging in to this server. The server used for this MVP is similar to the online servers that are well-known and used by the large FlightGear community. The difference lies in the fact that this server will (for now) not be operated offline and is thus only accessible through the LAN. To setup a LAN, the following organizational chart in figure 5.5 has to be realised:

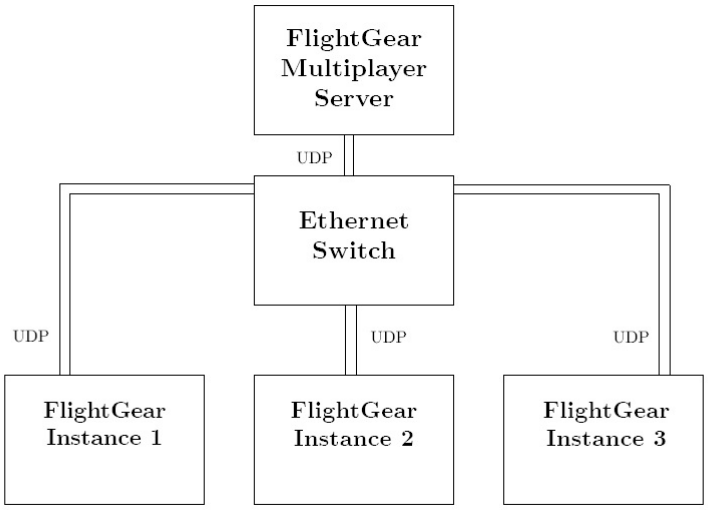


Figure 5.5: *FlightGear server connected to the FlightGear instances running on the computers via a network switch*

5.1.6.1 FlightGear Server

The server is set up using a small pc running the Ubuntu 16.04 LTS operating system. The server can be run and compiled on a UNIX based operating system. The server can be run as a background process. Booting of the server can be done with a terminal command into the specified folder in which the server is compiled. This is depicted in figure 5.6

```
flightgear@flightgear-server:~$ cd /home/flightgear/build-fgms
flightgear@flightgear-server:~/build-fgms$ fgms
Could not read config file '/usr/etc/fgms.conf' => using defaults
Could not read config file '/home/flightgear/fgms.conf' => using defaults
processing fgms.conf
# using logfile fgms.log
28.05.2017 13:41:40 # This is fgnlr2017 # this is only on LAN()
28.05.2017 13:41:40 # FlightGear Multiplayer Server v0.12.0 started
28.05.2017 13:41:40 # using protocol version v1.1 (LazyRelay enabled)
28.05.2017 13:41:40 # listening to port 5000
28.05.2017 13:41:40 # telnet port 5001
28.05.2017 13:41:40 # admin port 5002
28.05.2017 13:41:40 # using logfile fgms.log
28.05.2017 13:41:40 # listening on 192.168.1.50
28.05.2017 13:41:40 # tracking is disabled.
28.05.2017 13:41:40 # I have 0 relays
28.05.2017 13:41:40 # I have 0 crossfeeds
28.05.2017 13:41:40 # I have 2 blacklisted IPs
28.05.2017 13:41:40 # Files: exit=[/tmp/fgms_exit] stat=[/tmp/fgms_stat]
28.05.2017 13:41:40 Main server started!
28.05.2017 13:41:40 # Admin port disabled, please set user and password

-----------------------------------------------
FlightGear Multiplayer Server CLI
-----------------------------------------------

fgnlr2017 # this is only on LAN>
```

Figure 5.6: *Booting the FlightGear server via the Ubuntu command line*

5.1.6.2 Ethernet Switch

Since the network that is created is a local network, a networking switch can be used to connect the FlightGear instances to the server. Running a LAN enables users to set up and operate this network with the absence of (wireless) internet access. The usage of a switch requires the user to manually set the IP-addresses and the subnet masks of the computers connected to the network, as opposed to a router. The switch used during the prototyping phase was a switch with 5 ports, which means 4 computers and thus 4 instances of FlightGear can be running in the same virtual world at the same time. In theory, a larger switch could house an increased number (depicted by FlightGear PC [n] in figure 5.5.) of computers and FlightGear instances on the network, depending on how much load the server can handle.

5.1.6.3 FlightGear Instances

The user has to activate multiplayer mode inside FlightGear with the server adress, ports and callsign specified, as depicted below:

```
--multiplay = out, 10, 192.168.1.50, 5000
--multiplay = in,10,,5001
--callsign=sdcs
# Multiplayer session defining the output port as 5000 (this should be the same as the
    listening port of the server in figure \ref{fig:serverinit}) and the server's IP
    address. The in port is defined as output port +1, thus 5001.
```

Running two computers with these lines in the setup of FlightGear, results in FlightGear connecting through the server. Once two computers are running an active session of FlightGear, a connection to the server is automatically established.To list active users inside the network at any time, the user can navigate to *Multiplayer* > *Pilot List* inside an active FlightGear session. One can also perform the command line command "*show users*" in the server terminal.

5.2 Realisation of "Must-have" requirements

This section handles the software realisation of all "Must-have" requirements, as established in the Ideation chapter of this thesis, in the MVP. The storyboards created in the Specification chapter of this thesis will form the design artefacts for the realisation of the "Must"-requirements of the MVP.

5.2.1 Requirement 1: Allow the input commands for the UCAV(s) in to be variable in nature or abstraction level

As was established in the Specification session, a number of basic manoeuvres will be implemented in the MVP. The basic manoeuvres can be realised by pre-programming a sequence that directly sets the control surfaces of the aircraft in FlightGear. For example; if the aircraft should pitch up, one can manipulate the elevator to deflect for a certain amount of time and consequently return it to its origin. The aircraft will commence to pitch up for the time that is set before the elevator is returned to 0 degrees deflection. The pseudo code for such a manoeuvre is depicted below:

```
def pitch_up:
    # Set the elevator
  fg['/controls/flight/elevator'] = -0.2
  # Hold it in this position for 2 sec.
  time.sleep(2.0)
  # Release elevator
  fg['/controls/flight/elevator'] = 0.0
```

The basic manoeuvres for pitching up, pitching down, banking left, banking right, levelling out and more complex manoeuvres, taking off from the ground and following an aircraft are implemented in the MVP. This class furthermore contains an implementation of a Finite State Machine (FSM). A FSM is a computational model based on a hypothetical "machine" which consists of one or more states [84].

FSMs are commonly equipped to organise and represent a flow of executions and states of an agent. The machine can only be in exactly one state at a time and the machine transitions from one state to another state in order to perform different actions. In case of the MVP, the manoeuvres are the states: the aircraft can only perform one action at a time. The implementation of the "pitch up"- manoeuvre in the FSM can be realised with the following pseudo code:

```
def pitch_up:
   def initialize:
    # Initialize manoeuvre bool
  pitch_up_boolean = False

def pitch_up:
    # method to change boolean value
  pitch_up_boolean = True

def finite_state_machine:
  while True:
    while pitch_up boolean == True:
        # If manoeuvre boolean is true, perform manoeuvre
      fg['/controls/flight/elevator'] = -0.2
      time.sleep(2.0)
      fg['/controls/flight/elevator'] = 0.0
      # and exit loop
      pitch_up_boolean = False

# Initialization and starting of thread
manoeuvre_thread = threading.Thread(target=finite_state_machine)
manoeuvre_thread.start()
```

In the MVP, every manoeuvre is a state the FSM can be in. An elaboration on this structure is required: The FSM is programmed as an infinite loop (denoted in the psuedocode as *def finite_state_machine*) that commences when starting the separate thread. Once the thread starts, the infinite loop commences. As long as no action is undertaken by the user, the corresponding boolean will remain *False*. However, as soon as the user clicks the "Move up" button of the GUI, the button binding will call the *pitch_up method*, changing the boolean to *True*. In the next iteration of the loop of the FSM, it will encounter the boolean of the if-statement for the manoeuvre to be *True*. The if-statement will be entered and the manoeuvre is performed. The *pitch_up_ boolean* is set to *False* at the end of the manoeuvre. The if-statement will be exited and the loop will be continued. In the MVP, every method that describes a manoeuvre is a state.

The main reason for utilising a FSM in FlightGear is to prevent the Tkinter GUI application from freezing while it waits from a call-back. Directly calling the method that performs the manoeuvre from the Tkinter GUI is possible, however the call-back that the event scheduler of the Tkinter application requires, needs to be returned within a 4 second time frame. Short manoeuvres can be realised by directly binding the method call to the Tkinter GUI, but longer manoeuvres (such as the take-off manoeuvre) requires more time to finish. This will delay the call-back to the Tkinter module, causing it to freeze. The FSM allows the GUI to work with "flagging" instead of waiting for a function to be completed; it "delegates" the workload to another method by simply changing only the Boolean and then returning the call-back.

The discovery of a functioning autopilot in FlightGear provided inspiration of the realisation of the more difficult manoeuvres that require a control loop of some sort. Upon further investigation of the internal workings of the autopilot of the aircraft it turned out that the autopilot systems are based on an universal type of controller; a Proportional-Integral- Derivative (PID) controller. A PID controller includes a feedback mechanism and is widely used in different fields such as thermostats, governing of industrial equipment and also auto-pilot systems for aircraft [85]. A PID controller works by monitoring a current value and a desired value and attempting to minimize the difference between these two values. The difference between the desired value and the current value is the 'error value'. The PID controller

attempts to apply a correction on the output signal in order to minimize the error value by means of proportional, integral and derivative terms. A block diagram of a PID controller is depicted in figure 5.7 [86]

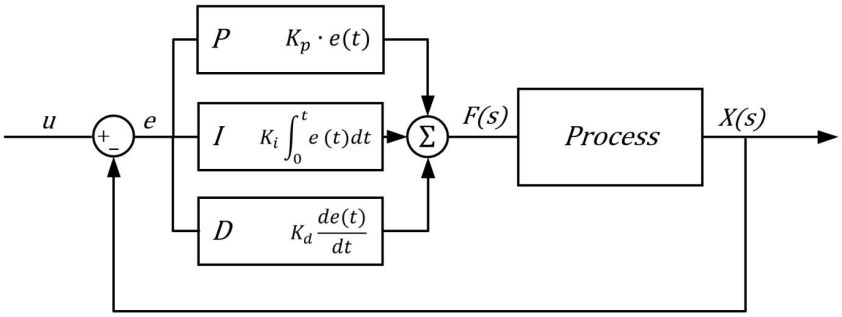


Figure 5.7: *Block diagram of a PID controller*

Implementations of PID and PI controllers are performed throughout the autopilot script of the F-14B aircraft. As an example, the file *f-14-AFCS.xml* of the F-14B describes the functions of the autopilot. As an illustration, the vertical speed mode of the autopilot of the F-14B is depicted below:

```
<!--Vertical Speed Hold-->
<pid-controller>
    <name>VerticalSpeed</name>
    <debug>false</debug>
    <enable>
        <prop>/autopilot/locks/altitude</prop>
        <value>vertical-speed-hold</value>
    </enable>
    <input>
        <prop>/velocities/vertical-speed fps</prop>
    </input>
    <reference>
        <prop>/autopilot/settings/vertical-speed-fpm</prop>
        <scale>0.01667</scale>
    </reference>
    <output>
        <prop>/controls/flight/elevator-trim</prop>
    </output>
    <config>
        <Kp>
            <prop>sim/model/f-14b/systems/afcs/vs-pid-pgain</prop>
        </Kp>
        <beta>0.1</beta>
        <Ti>10.0</Ti>
        <Td>0.00001</Td>
        <u_min>-0.15</u_min>
        <u_max>0.15</u_max>
    </config>
</pid-controller>
```

Depending on the quality and complexity of the work by the author of the model, various autopilot functions are available. The direction based manoeuvres as described in the Specification phase can be realised using various autopilot functions. The basic manoeuvres can be altered in the following way:

```
def pitch_up:
  # Set the value for the pitch angle
  fg['/autopilot/settings/target-pitch-deg'] = 10
  # Set the autopilot in pitch hold mode
  fg['/autopilot/locks/altitude'] = 'pitch-hold'
  # Exit the manoeuvre
  pitch_up_boolean = False
```

By setting the appropriate value for the pitch angle for the autopilot and after that engaging the autopilot in pitch-mode, the aircraft will commence to pitch with 10 degrees and will maintain this pitch angle until the user says otherwise. Consequently, the same can be down for the *pitch_down* method, adapting the pitch angle to -10 instead of 10 will result in a pitch down of the aircraft with a 10 degree negative angle (without speed hold).

A more difficult manoeuvre like following an aircraft can be established by making smart use of the autopilot. Theoretically, all modes of the autopilot (in this case heading mode, vertical mode and speed mode) can be equipped to follow a certain aircraft. One can "copy" the heading, altitude and current speed of the adversary and "paste" it inside the autopilot of the UCAV. If this is done continuously, the autopilot forces the UCAV to maintain the identical heading, altitude and speed as the adversary, which results in the UCAV following and mimicking the adversary's behaviour and manoeuvres. FlightGear houses a "*target-tracking*"-function that enables the passing of adversary values inside the UCAVs autopilot. The Property Tree browser can be used to determine the working of this function.

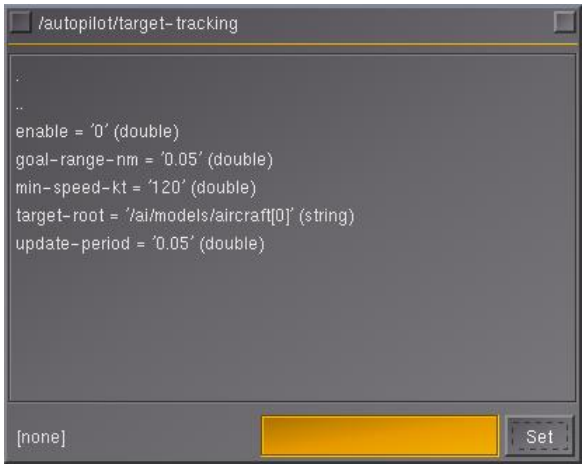


Figure 5.8: *Property Tree browser at the target tracking functionality*

As can be seen from figure 5.10, the target tracking function can be enabled and disabled, the distance between the aircraft (similar to loitering distance in circling UAVs) can be set, the minimum speed the following aircraft should maintain (this should be set to above stall speed of that certain aircraft), the target location in the property tree and the update period.

By enabling the target tracking function, the parameters heading, altitude and speed are routed towards the autopilot of the UCAV. The target-root should be set to the aircraft that the user desires to follow: in figure 5.8 it is set to an AI aircraft, however, in a multiplayer session the path of the target-root should be set to */ai/models/multiplayer* or with multiple multiplayer players in one session, the denomination of that player. In the case that there are 5 players inside the server, the first player is denoted as */ai/models/multiplayer*, the second as */ai/models/multiplayer[1]* and so forth until the last player 'n' (*ai/models/multiplayer[n]*). To determine which player is denoted by which index in the brackets, the Property Tree browser can be utilised to browse to the path of that multiplayer (*ai/models/multiplayer[n]*) and look up the callsign of the pilot. To enable the target tracking function, the following pseudo code can be used:

```
def engage_following:
  while self.engage_following_boolean:
      # Enable target tracking with specific target and set autopilot accordingly
    fg['/autopilot/target-tracking/enable'] = '1'
    fg['/autopilot/target-tracking/target-root'] = 'ai/models/multiplayer'
    fg['/autopilot/locks/altitude'] = 'altitude hold'
    fg['/autopilot/locks/heading'] = 'dg-heading-hold'
   # exit manoeuvre
    engage_following_boolean = False
```

This enables the target-tracking function and selects the proper aircraft to follow. The values for heading, altitude and speed are continuously routed to the autopilot of the UCAV. By setting the autopilot in *'altitude-hold'* and *'dg-heading-hold'*, the autopilot will try to reach the altitude of the adversary and maintains the same heading as the adversary, effectively following the adversary at the given distance as defined as *'goal-range-nm'*, which is the distance the UCAV should keep with respect to the aircraft that it is following in nautical miles.

To recap, the autopilot functionality (consisting of PID-controllers) has been utilised to fulfil basic manoeuvres (up, down, left and right) and a more difficult "following-function". These manoeuvres are coupled to the user interface so that the user can select an action and the Python script will manipulate FlightGears autopilot settings to achieve the manoeuvre.

5.2.2 Requirement 2: Allow to vary the frequency with which the user can provide input to the UCAV(s)

In the Specification phase it was noted that the realization of this requirement is challenging within this design space. Originally, the frequency with which the operator provides commands to the UCAV is dependent on the amount of autonomy the UCAV possesses. As mentioned in Chapter 2; if the UCAV is able to operate more autonomously, the frequency with which the operator provides commands could decrease. If a task proves to be difficult, it can be split up in sub-tasks to be solved by the SDCS, in turn possibly increasing the frequency with which the operator gives commands. Therefore, the frequency with which the operator provides input is a matter of asking what the UCAV will do in the time that it does not receive any commands rather than determining what the right frequency of giving commands will be. This question cannot be answered with the current level of knowledge about the SDCS and is open for further research. According to the literature described in Chapter 2, two options are possible in a scenario where the UCAV receives no direct commands: management by consent or management by exception. In the MVP, the UCAV will neither initiate actions nor wait for commands. In the MVP, the UCAV will maintain the manoeuvre until the user directs the UCAV to do something else or stop performing the manoeuvre. For example, if the user gives the command to bank right, the UCAV will bank right and maintain this bank angle until the user gives another command. If no command is given, the FSM will not reset the autopilot modes and thus the manoeuvre will be maintained. However, the UCAV should have an "idle state" as was described in the Specification phase as being a state of steady, level flight. The user can manually set the UCAV to its idle state by pushing the corresponding button on the GUI. The auto-pilot of the F-14B will be equipped to realise this. The pseudo code for maintaining steady, level flight in idle state is depicted below:

```
def finite_state_ machine:
  While True:
      # Steady level flight manoeuvre
    while idle_boolean == True:
      fg['/autopilot/settings/vertical-speed-fpm']=0
      fg['/autopilot/locks/altitude'] = 'vertical-speed-hold'
      fg['/autopilot/locks/heading']= 'wing-leveler'
    # Pitch up manoeuvre
    while pitch_up boolean == True:
          fg['/autopilot/settings/target-pitch-deg'] = 10
          fg['/autopilot/locks/altitude'] = 'pitch-hold'
          pitch_up_boolean = False
    while pitch_down_boolean = True
      # more code and manoeuvres
```

This piece of code executes the following 4 commands: It sets the desired value for vertical speed of the autopilot to 0. A vertical speed of 0 denotes a level flight. Continuing, it sets the autopilot in *'vertical speed hold mode'*. The autopilot attempts to maintain the value for the vertical speed as defined in the command above, namely 0. Furthermore, the heading mode is set to *'wing-leveler'* mode, which ensures the wings remain levelled. The aircraft will maintain a vertical speed of 0 feet per minute with its wings levelled, thus a steady, level flight. It is to be noted that in FlightGear, the autopilot can hold

speed, altitude and heading at the same time as opposed to some other autopilot systems. After that, the Boolean for the idle state is set to *False* to exit this loop. The FSM will continue with its loop. It is to be noted that as of now, the FSM factually is in no state rather than an idle state. This change has resulted from feedback from Hans Heerkens stating that it would be useful if the UCAV maintained a manoeuvre until commanded otherwise rather than returning to idle state after a manoeuvre. An idle state may in the future be implemented as follows:

```
def finite_state_ machine:
  While True:
    While idle == True
      Desired idle action
    while manoeuvre==True:
      perform manoeuvre()
    while other_manoeuvre == True:
      perform other_manoeuvre()
    while another_manoeuvre = True
      perform another_manoeuvre()
    # At the end of the loop, automatically call idle function
    idle_boolean = True
```

The example shows that the *idle_boolean* is set to *True* at the end of each loop of the FSM. This will result in the FSM transitioning towards its idle state at the end of each loop unconditionally. Resulting in the fact that upon completion of a command and when no command is given, the FSM will transition to idle state (what that idle state may be is left to the implementation).

To recap, the variable frequency of giving commands is translated to building an idle state or "origin" for the UCAV to fall back to when no commands are given. Currently, the aircraft will attempt to maintain the manoeuvre that was last commanded. The idle state is steady level flight.

5.2.3 Requirement 3: Allow to vary the amount of latency in the data link between the operator and the UCAV(s)

As mentioned in Chapter 2 and in the Specification phase, the simulation of the latency on the data link should be bidirectional; the latency appears when the operator sends a command to the UCAV and the latency should also appear in the presentation of the flight parameters and visualization of the aircraft towards the operator. Before treating the realization of the latencies, the latency inside the local network should be determined. A ping from one of the computers to the other computer in the network results in the image in figure 5.9:

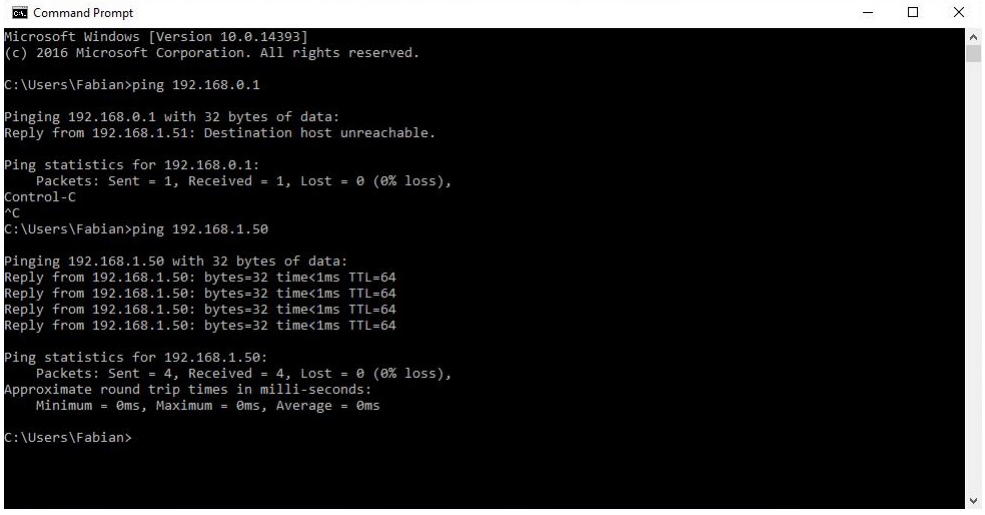


Figure 5.9: *Ping to the other computer on the LAN*

It can be deducted that the reaction time in the network is less than 1 millisecond. This way it can be ensured that the influence of network latency is so small, that it will not have impact or addition to

simulated latencies.

Before delving into an approach to realise the simulation of latencies in the MVP, a clear designation of latencies should be performed. Latency from this point will be referred to as the time in milliseconds that is required for a round trip; meaning from the operator towards the UCAV and from the UCAV back to the operator on a full-duplex data link. This is coherent with reality and what was specified in the Specification phase. A Round Trip Time of 250ms denotes the following; the operator sends a command that is received by the UCAV only after the latency of half the round trip time (125ms) and the operator receives all information coming from the UCAV 125ms later than the UCAV actually send it. In the MVP, it will take 250ms before the operator sees the impact of his or her commands.

To realize the latency, two instances of FlightGear running on different computers are configured in a master/slave fashion where one computer (the master) sends its controls towards the slave and the slave routes its FDM data towards the master, both utilising the UDP protocol on the same destination and receiving port, as depicted in figure 5.10.

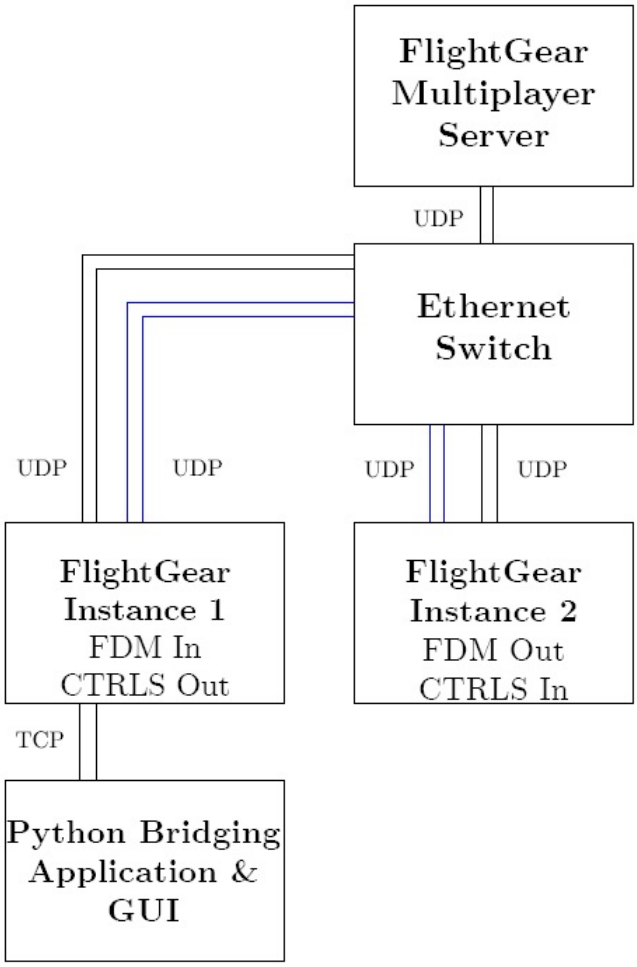


Figure 5.10: *Two FlightGear instances coupled in a master slave fashion through UDP (blue lines) on port 5500*

In this configuration, one FlightGear instance, the slave instance, possesses no FDM but solely routes the control input of a joystick or the Bridging Application (or both) towards the master instance over UDP. The separate UDP stream between the two instances is denoted by the blue lines in figure 5.10 The master utilises this information to locally perform all FDM calculations. The FDM information that the master has calculated is returned to the slave instance where it is utilised to control the aircraft on the slave instance. Now, two aircraft are visualised on both FlightGear instances, where the

master instance receives the controls from the slave instance and the slave instance receives the FDM information from the master instance. When no delay is applied, the two separate aircraft rendered on both FlightGear instances will move in sync. However, if a delay of for example 125 ms, is deliberately introduced on both separate UDP streams between the instances, the instances will receive the necessary information only after that delay. In chronological order: The joystick or control data from the Bridging Application processed by the slave instance is send towards the master instance over the delayed UDP stream, arriving at the master only after 125ms instead of directly. The master instance calculates the FDM information and sends it back to the slave instance over another delayed UDP stream, which causes the FDM information to arrive at the slave only after 125ms.

During usage, the operator sees two visualisations: One visualisation (that of the master) which shows what one would see if he or she was actually present at the position where the UCAV resides. This means that only the delay of the controls is visual (one way delay). However, the operator is actually located in his or her GCS. Thus he or she should see the delayed visualisation of his or her actions. This visualization is present on the slave instance since it receives the FDM information with a delay in respect to the master instance. The visualisation of the slave lags that of the master instance with the set delay. This way, by looking at the visualisation of the slave instance, the operator sees a RTT delay that is realised in a fashion that is coherent with reality.

To clarify, the Master instance henceforth referred to as "Theatre View", there it displays the situation as it is taking place. The Slave instance is henceforth referred to as the "Ground Control Station" since it displays the situation as the operator in his or her GCS will see. Summarized in figure 5.11:

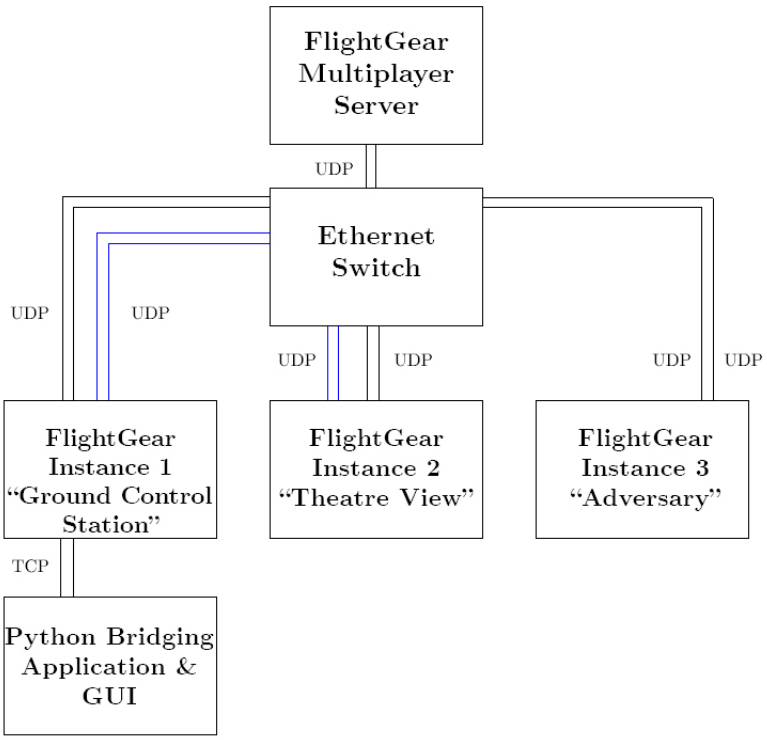


Figure 5.11: *Two FlightGear instances coupled in a master slave fashion through UDP (blue lines) on port 5500*

Inducing delay on the UDP is performed by leveraging the functionality of the Netem package on Unix based operating systems [87]. Netem provides Network emulation for testing protocols and allows for the emulation of variable delay, loss, duplication and re-ordering of data packets that are handled with both UDP and TCP. The FlightGear options for the master and slave are defined as follows:

```
1 Master:
```

```
--native-fdm = socket,out,60,ip_address_of_slave,5500,udp
--native-ctrls = socket,in,60,,5500,udp
--fdm = automatic selection

Slave:

--native-fdm = socket,in,60,,5500,udp
--native-ctrls = socket,out,60, ip_address_of_master,5500,udp
--fdm = external
```

The commands for setting a delay on all outgoing traffic from both the master and slave machines can be realised with a queuing discipline command and consequently applying a filter to only affect the traffic outbound on port 5500. The following commands can be inserted in the command line of Unix based operating systems:

```
sudo tc qdisc add dev eth0 root handle 1: prio priomap 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sudo tc qdisc add dev eth0 parent 1:2 handle 20: netem delay 3000ms
sudo tc filter add dev eth0 parent 1:0 protocol ip u32 match ip dport 5500 0xffff
    flowid 1:2
```

The user can set separate delays for each data stream and Netem furthermore includes options for applying delays with different distributions rather than a fixed delay. In reality, the delay will not be fixed, but may vary with a certain deviation based on a normal distribution. Netem possesses the ability to apply a delay with variation:

```
sudo tc qdisc change dev eth0 root netem delay 100ms 20ms distribution normal
```

The distribution tables are compiled on the operating system itself. It is also possible to configure a distribution based on own experimental data [87]. For the MVP, a batch script is written that allows the user to select the latency on a UNIX machine. This batch script is included in appendix J

To recap, the latency present on the data link can be simulated by placing to FlightGear instances into a master-slave configuration and deliberately introducing a delay between the data links of these two instances. This is achieved with the aid of Ubuntu functionality. The uplink and downlink are separate datastreams and can furthermore be individually delayed. This is a simulation of latency which is true to real life UAV systems.

5.2.4 Requirements 4+5: Allow to display flight parameters of the UCAV(s) and allow to filter what information is presented to the user

In the Specification stage it was specified that the user could check boxes or select variables from a drop down list for monitoring. This way, the user can filter what information he or she desires to be displayed on the screen. Later on, it was decided that such functionality was not necessary and that the main interest lies in the method for retrieving the parameters from FlightGear rather than which parameters should be visible. 5 important variables to monitor for an operator are: Magnetic Heading, Slide Slip Angle, Indicated Airspeed, Altitude and Vertical Speed. The MVP will constantly monitor these parameters. The following block of pseudo code demonstrates how the logging of one variable is performed in the MVP.

```
class ParameterLogging:

side_slip_logging = DoubleVar()

def parameter_logging:
  While True:
    fg['/orientation/side-slip-deg']= side_slip_logging

parameter_logging_thread = Thread(target=parameter_logging)
parameter_logging_thread.start()
```

This code block runs a separate thread on which the infinite loop of the method *parameter_logging* is being performed. The use of the data type *DoubleVar()* needs elaboration: *DoubleVar()* is a Python class variable designed for use with the Tkinter GUI module. The Tkinter toolkit utilises a feature called *'tracing'* to update GUI widgets when the associated variable is modified. The value of the DoubleVar() can be set by utilising the corresponding *.get()* method. This enables the MVP to automatically update the values of the variables on screen without the need of tracing or call back functions.

To recap, this section shed light on how the flight parameters can be presented on the Graphical User Interface so the user can determine the current state of flight parameters of the aircraft.

5.2.5 Requirement 6: Allow to select different points of view

In the Specification phase it was not decided upon which views are desirable. The MVP will demonstrate different options for switching, selecting and changing views and view angles. FlightGear comes with built-in views and commands for switching views. From the Bridging Application, one can easily cycle through views that are available in FlightGear with the following psuedocode:

```
def next_view:
  fg.set('/command/view/next', 'true')
```

Alternatively, separate buttons dedicated to a single point of view can be realised by writing FlightGear's property for "*current-view*" to the corresponding ID number of the view coded in the aircrafts*–set.xml* file.

```
def outside_view:
  fg.['/command/current-view'] =  4
```

The MVP allows for cycling through views with a "*previous view*" and a "*next view*" button. The user is able to activate and deactivate different views inside Flightgear, including or excluding them to the list of possible views the user can cycle through. In an active FlightGear session this can be realised by browsing to *View > View Options* and ticking or un-ticking view options in the pop-up menu, denoted in figure 5.12

Figure 5.12: *Pop-up menu for enabling and disabling views*

Furthermore, FlightGear allows more viewing options under the view menu. Different views can be added and modified by adapting the aircrafts *-set.xml* file. In the MVP, two extra views are added. The first view is called "*SDCS Custom View - Overview*" and provides a heavily zoomed out camera perspective to overlook a large area. This is useful for orientation and localisation of potential adversaries when interacting with the MVP. This is realised by defining a new view option in the F-14B's *–set.xml* file. The xml code is provided below:

```
</view>
<view n="101">
    <name>SDCS Custom View - Overview</name>
    <type>lookat</type>
    <internal type="bool">false</internal>
    <config>
          <from-model type="bool">false</from-model>
          <from-model-idx type="int">0</from-model-idx>
          <eye-lat-deg-path>/position/latitude-deg</eye-lat-deg-path>
          <eye-lon-deg-path>/position/longitude-deg</eye-lon-deg-path>
          <eye-alt-ft-path>/position/altitude-ft</eye-alt-ft-path>
          <eye-heading-deg-path>/orientation/heading-deg</eye-heading-deg-path>

          <at-model type="bool">true</at-model>
          <at-model-idx type="int">0</at-model-idx>

          <ground-level-nearplane-m type="double">0.5f</ground-level-nearplane-m>

          <x-offset-m type="double">0</x-offset-m>
          <y-offset-m type="double">500</y-offset-m>
          <z-offset-m type="double">-100</z-offset-m>
        </config>
</view>
```

The view is defined by translation of the camera's perspective with respect to the origin of the aircraft. The <*type*> statement defines whether the viewpoint is located inside the aircraft (for pilot and co-pilot views) or outside views (for chase views). The <*offset*> items describe how far the camera is placed from the origin. There are a multitude of options available for scripting views in FlightGear. The script above translates to a "far-chase" cam as depicted in figure 5.13 and figure 5.14.

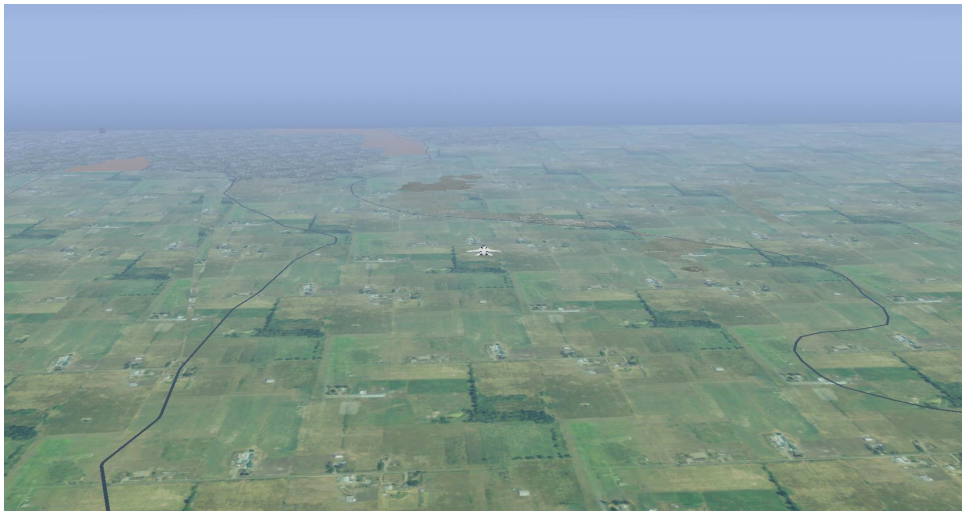


Figure 5.13: *SDCS Custom View – Overview camera setting provides an overview of the direct surroundings of the UCAV with the UCAV centered*

Figure 5.14: *SDCS Custom View – Overview camera setting provides an overview of the direct surroundings of the UCAV, set to directly above the aircraft*

The second view created for the MVP is called "*SDCS Custom View - POV*" and provides a camera perspective modelled to emulate the screens a camera aboard an UCAV generates. When working with UAVs, there is no option for bird's-eye viewing since there is geographical separation between the operator and the UAV. Therefore, the operator has to depend on images that are captured by a camera mounted on the UAV. To simulate this view, a new view option is defined by placing the game-camera near the camera mounted on the F-14B. This mounted camera is depicted in figure 5.15:

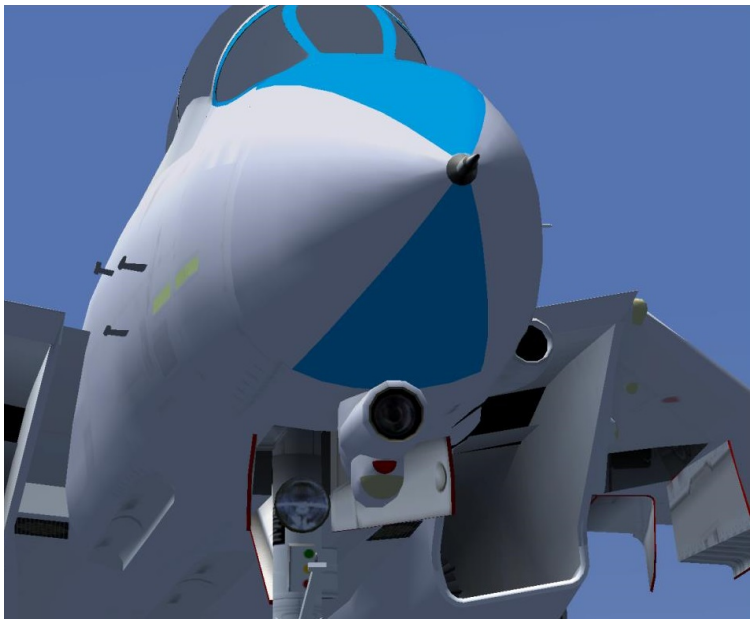


Figure 5.15: *Camera mounted on the fuselage of the F-14B*

This view is more realistic in the scope of UAV or UCAV operation. The camera is not mounted on a gimbal and does therefore move stationary with the fuselage. Any tilting of the aircraft will result in the tilting of the image. This is realized by defining a new view option in the F-14B's *–set.xml* file. The xml code is provided below:

```
view n="102">
<name>SDCS Custom View – POV</name>
  <type>lookfrom</type>
    <internal type="bool">false</internal>
    <config>
        <from-model type="bool">true</from-model>
        <from-model-idx type="int">0</from-model-idx>

```

```
        <x-offset-m type="double">0</x-offset-m>
        <y-offset-m type="double">-1.5</y-offset-m>
        <z-offset-m type="double">-7.8</z-offset-m>
    </config>
</view>
```

The result of this view is depicted in figure 5.16

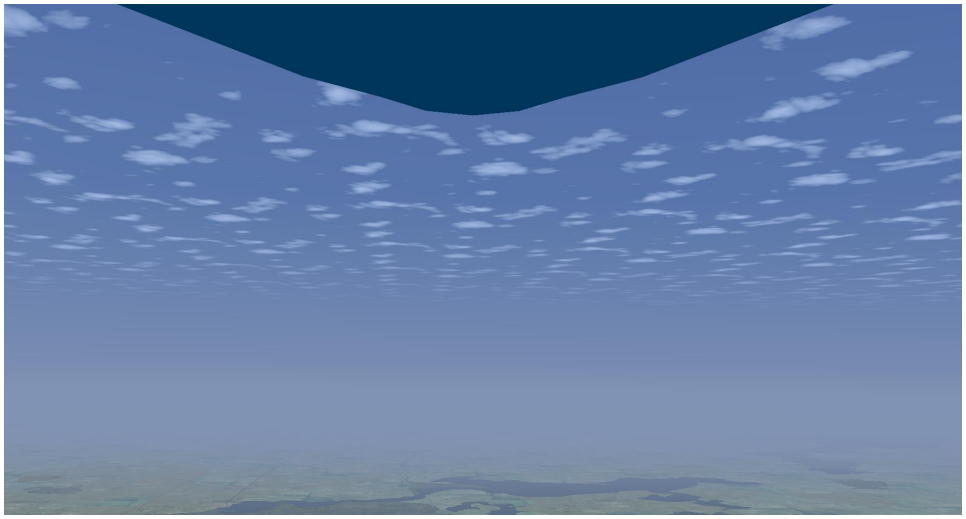


Figure 5.16: *SDCS Custom View – POV camera setting provides a view from the fuselage camera of the F-14B. The blue triangle in the top of the image is the fuselage*

After declaring these views in the F14B –set.xml file, the configured views will be added to the list of FlightGear views and can be enabled and disabled via *View > View Options* and ticking or un-ticking view options in the pop-up menu that appears, as depicted in figure 5.17

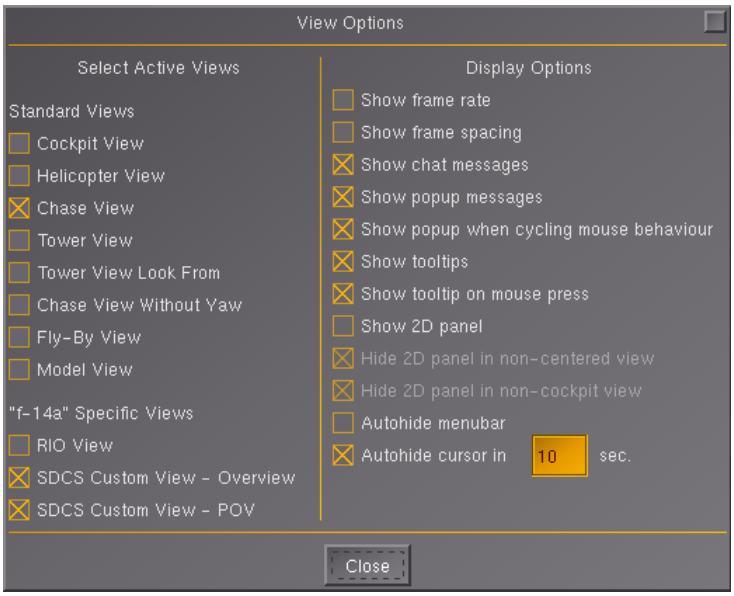


Figure 5.17: *The view options pop-up, now including the custom views*

The user can enable these views and consequently cycle through them in the MVP. In addition to these views, the MVP also includes an overview of (all) aircraft present in the airspace. In the GUI of the MVP the option is denoted as “External View” and upon clicking, it opens a browser which integrates a 2D overview with an overlay of Google Maps. This is depicted in figure 5.18

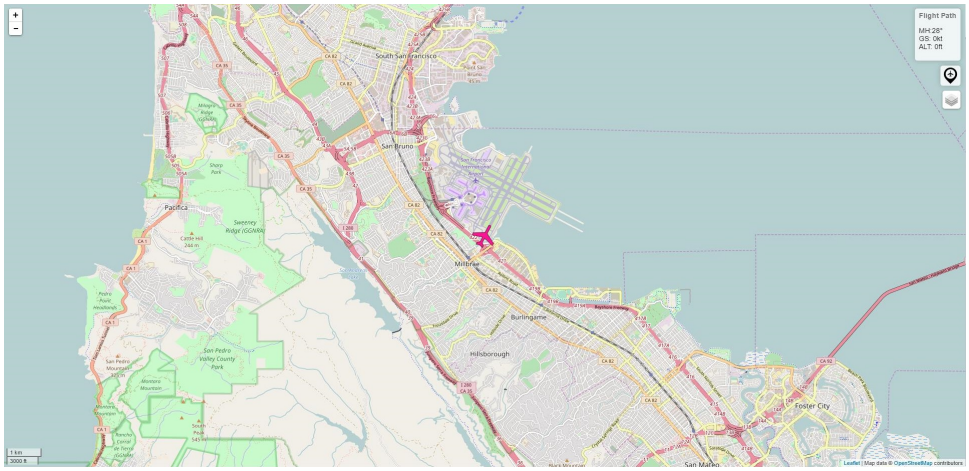


Figure 5.18: *Google Earth overview, the aircraft is depicted by the pink aircraft icon (currently located on the beginning of a runway)*

To recap, this section has shed light on how different camera views have been defined and implemented in the MVP. The user has a broad range of camera aspects and maps to choose from during the interaction with the MVP. Furthermore, a view has been defined that may be useful in air to air combat and one that is relevant in the light of UAV operation.

5.2.6 Requirement 7: Allow for editing and extension by other developers

This requirement was not mentioned during the Specification phase since the fulfilment of the requirement as it is severely dependent on the realisation of the MVP. Two major factors contribute for the MVP to be editable and extendable by other developers. One factor is the fact that FlightGear is chosen for the realisation of the MVP. FlightGear is released under the General Public License, as mentioned in the Ideation chapter. For the MVP this means that the flight simulator is editable for research usage by the NLR [88]. If alterations are made to the source code by the NLR and then commercially used, the modified source code must be made public. The source code is made available which allows for deep modification of FlightGear's working. The property tree in its turn, allows for easy access to the running instance of FlightGear. This combination makes FlightGear a highly editable program with a wide range of built-in functionalities that can be exploited and used for further development of this MVP, also in later stages. The second major factor that contributes to this requirement is the fact that the Bridging Application is fully written in the programming language Python. Python is a high-level programming language, which means that it stands relatively far away from the details of how a computer interprets commands. In practice this means that the commands may include more natural language like elements and that the user has to be less worried about the translation of code into commands that the computer can understand. A large difference between for example Python and C++ is that Python works with indentation of lines of code rather than brackets and curly brackets in C++ which allows for a clean programming syntax that is less complicated to read. For example, defining a method containing a nested while-loop can be programmed in both Python and C++. Below the two implementations are done, and it can be noted that the implementation in Python looks cleaner due to the absence of accolades, brackets and operator signs. Firstly, in C++:

```
void new_function(){
  while (boolean == TRUE){
    while(boolean_2 == FALSE){
      foo();
    }
  }
}
//Nested while-loop using C++ syntax
```

And in Python:

```
Def new_function:
  while boolean:
    while not boolean_2:
foo()

# Nested while-loop using Python syntax
```

It can be deducted that the Python example contains no brackets or operator signs, instead of using the double equal sign (==) which usually denotes a comparison to either TRUE or FALSE, Python implements this clearer by making use of natural language elements: *while not boolean_2*, where as in C++ it is implemented as: while *boolean_2 == FALSE*. Python is furthermore a multi-paradigm programming language which allows for Object Oriented Programming (OOP) [89]; the bridging application is written in an OOP style, denoting all functions as classes and calling methods on instances of these classes (objects). This method should be understandable for both experienced and beginning programmers. Python houses a great amount of standard libraries for different types of applications and next to that, collections of modules and API's are available.

Furthermore, Python interpreters are available for a large number of operating systems, such as Windows, MacOS and UNIX based operating systems. A possible downside of the usage of Python lies in the fact that it is an interpreted language [90]. This means that, in contrary to compiled languages, that Python may be performing slower. If that will pose a problem in the MVP is a subject of research during the development of the MVP.

5.3 Overview

A complete overview of the system with the Bridging Application, FlightGear instances, switch, server and all transport layer protocols between nodes is depicted in figure 5.19.

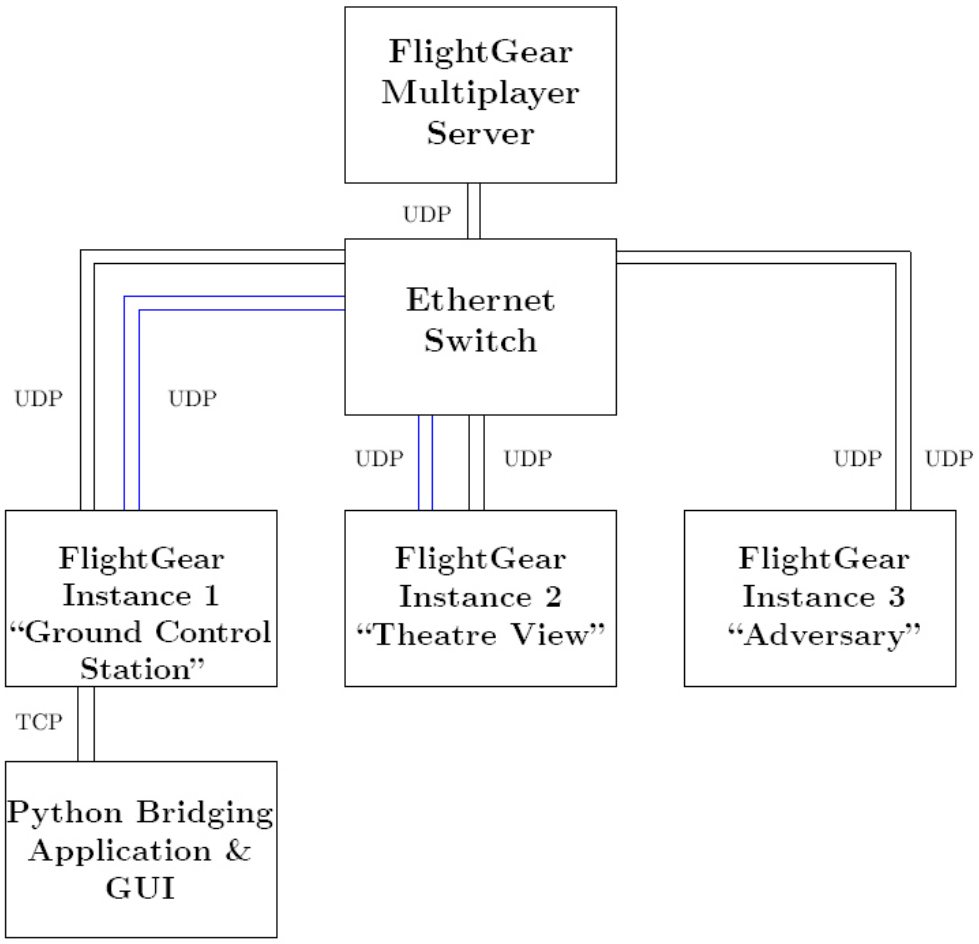


Figure 5.19: *Overview of the MVP with the Bridging Application, FlightGear instances, switch, server and all transport layer protocols between nodes*

5.4 Conclusion

The MVP is realised according to the specifications along with further capabilities to enable multiple pilots to utilise the MVP. The sub research question *How can this be translated into a MVP?* can be answered; The Ideation and Specification phases have led to a design of the MVP and in this chapter, the MVP is realised according to the requirements with the aid of the open source flight simulator "FlightGear", the programming language Python and UNIX based operating systems. This complete setup will be subject to testing and evaluation in the next chapter: Evaluation.

6 Evaluation

This chapter forms the final phase of the Creative Technology Design Process, the Evaluation phase. This chapter describes the evaluation of the realisation of the MVP and aims to elicit experiences and observations from the interaction with the MVP that help answering the main research question. This chapter commences with a description of the user test and the results of the user test. Furthermore it will evaluate to what extent the "Must-have" requirements are fulfilled. In addition, the "Should-have" requirements are also evaluated.

6.1 User Testing

The user test performed in this phase serves two purposes: the evaluation of the fulfilment of the requirements set for the MVP and the evaluation of the user interaction of the MVP. The main advantage of this user test is that, next to problems that are known to the designer, also problems that the designer is not aware of may reveal them self. Inviting users and allowing them interact with the MVP and observing their possible struggles gives an indication of problems that need to be solved, next to any problems that have already arisen during other phases of the project. Feedback from the user tests along with ideas generated during every phase of the project serve as a base for a set of well-formulated and relevant Recommendations at the end of this thesis.

6.1.1 User Test Setup

The Evaluation is done in the form of a structured interview. It is deliberately decided not to record the sessions in audio or video, as from experience of the author it may occur that the interviewee may hold back his or her feedback. An interview that is not recorded but merely documented allows interviewees to feel less "watched" and feel inclined to provide more feedback. The manner of assigning tasks to the user has impact on the outcome of the interview, therefore during this user test, the instructions on how to realise a task that are given to the interviewee are kept to a minimum. This is to realise that the interviewee finds a way to realise a certain action by him or herself, which contributes to determining the usability of the product. The user test setup document that has been designed and utilised during this phase, is included in appendix K.

The complete setup of the MVP is set up in a quiet room at the NLR, free from any distractions. Images of the setup are included in figure 6.1 through 6.4.

Figure 6.1: *Complete MVP test setup*

Figure 6.2: *Three main screens of the MVP: From left to right: Delayed UCAV view, map overview and the adversary view*

Figure 6.3: *Right side view of the MVP*

Figure 6.4: *Left side view of the MVP*

6.1.2 User Test Main Stakeholders

The user test is performed with the stakeholders of the NLR, Jan Joris Roessingh and Gerald Poppinga. They were subjected to the user test set up in appendix K in one session in which the MVP was presented.

The choice of the project stakeholders to be the only interviewees during the evaluation has pros and cons. An evaluation with the stakeholders is necessary to evaluate to what extend their wishes are met. Performing this user test on the stakeholders provides an interactive way to evaluate the MVP. However, the stakeholders are highly likely to have a biased standpoint towards the MVP and they furthermore possess a substantial amount of inside-information about the MVP. This may give distorted results regarding the research question and the outcomes of this user test. Since the stakeholders are the customers of this project, they will be included in the user test. The documented results of the user test interview are included in appendix L

6.1.3 Flight Test Tjebbe Haringa

In addition to the user test with the stakeholders, a flight test has been conducted with Tjebbe Haringa. Tjebbe is F-16 pilot and has been chief test pilot of the Royal Netherlands Airforce. His expertise in piloting and air combat make him a valuable test subject. In addition, Tjebbe has no further knowledge of the SDCS and will be less biased than the stakeholders regarding the MVP. Tjebbe did not perform the complete user test along with the qualitative interview, but has performed the same tasks as in the user test. Namely flying with joystick with 0, 250, and 1000 ms delay respectively and flying with the SDCS Interface with a latency of 250ms. During Tjebbe's test, a map of the airspace was placed on the middle screen of the test setup so that it was easier to find eachother during flight. Tjebbe mentioned that during the joystick flying with delay used a different approach to controlling the aircraft. Instead of using a "closed loop" method where every action is countered, he anticipated the result of the aircraft and corrected with small "pulses" of joystick movement. Tjebbe was successful in following the adversary aircraft with joystick control under a latency of 250 ms, at this point, the stakeholders (who have no highly developed skill in flying jet aircraft) had difficulties in even flying the aircraft. Tjebbe mentioned to anticipate more on the behaviour of the aircraft, thinking one move in advance. Tjebbe was able to follow the adversary aircraft under latencies of 250ms. A delay of 500ms made following difficult. He mentioned that the latency was not so much a problem in following another aircraft, but for what he called "high-gain" tasks, such as air to air combat, the latency of 250ms was already too much. He acknowledged that there should be a different form of control than joystick control when there is latency present on the data link. Tjebbe was therefore introduced with the SDCS and was presented the SDCS control interface in which he flew with a 250ms delay. Tjebbe acknowledged that this form of control helps mitigating the increased workload of actually controlling the aircraft under latency altough, with the current state, performing the tasks was not less difficult due to the limited control options. Tjebbe would like to see an option where you can set a heading and consequently, the aircraft will initiate a bank towards that heading. However, the need for a control system such as the SDCS was illustrated.

6.2 Results and Discussion

6.2.1 Fulfilment of "Must-have" requirements

When collecting and comparing the results of the user test it can be concluded that the "Must-have" requirements have been fulfilled. The user test provided an opportunity to test the functionalities during real usage of the prototype.

"Must-have" requirement 1: Allow to vary the amount of latency on the data link

The latency is realised correctly. There is a separate uplink and downlink which can be delayed separately. The controls are sent over one link and the results (visualisation) receives its data over the other data link. This situation is true to the real-life situation of UAV control. The latency can take any value from 0 to infinite seconds and statistical models of latency can be incorportated.

"Must-have" requirement 2: Allow the user to select different points of view

The view options have proven to be useful, especially the 2D maps inside FlightGear. Another useful view that is more realistic for UAV operation is the "SDCS Custom View- POV" in combination with the Head Up Display in FlightGear.

"Must-have" requirement 3 + 4: Allow to display the flight parameters on screen and allow to select what information is displayed on screen

The parameters are printed on the GUI of the SDCS Control Interface. However, it would be more meaningful to present the operator with a Primary Flight Display or the basic flight instruments instead of just numbers since they are not meaningful or intuitive.

"Must-have" requirement 5: Allow for commands to be different in nature and abstraction level

The most simple commands (move up, down, left and right) have been realised along with the base of a "follow-function", which is of a different abstraction level. These manoeuvres are incorporated, however there is a strong need for more degrees of freedom in controlling the aircraft. More manoeuvres and a more sophisticated flight controller are needed.

"Must-have" requirement 6: Allow to vary the frequency with which the user provides input

The frequency with which user provides input was a requirement that was dependent on what the UCAV needed to do when it is not receiving commands. In the Specification phase it was established that the UCAV should be able to maintain steady level flight. In the MVP this is correctly realised by allowing for steady level flight.

"Must-have" requirement 7: Allow for editing and extension by other developers

FlightGear is a completely open flight simulator that has all run time state variables accessible. This allows for easy development since most data required for functionality is provided. Furthermore, the open source community behind FlightGear may develop new functionality that can be utilised for the MVP. FlightGear is open source, allowing to access source code for manipulation. Python is a high level programming level that is relatively straightforward to grasp by both experienced aswell as beginning programmers.

From these results it becomes clear that some requirements may need to be implemented in a different form. The Recommendations section of this thesis will handle these improvements.

6.2.2 Fulfilment of "Should-have" requirements

These requirements were originally not treated in the Realisation phase however, they have been utilised as guidance for selecting the software packs during the Ideation phase. A short evaluation of these requirements in the MVP is handled here.

"Should-have" requirement 1: The MVP should incorporate realistic flight dynamics, aerodynamics and physics

In FlightGear, the aircraft behaviour is modelled after look-up tables that contain empirical data (for example, air tunnel test data and manufacturer information) which are used in the calculations for the motion for the aircraft. This results in realistic flight behaviour depending on the quality of the available data. The F-15 had sufficiently good performance to make a realistic impression on non-pilots.

"Should-have" requirement 2: The MVP should furthermore allow the user to set the armament of

the UCAV(s) and other aircraft

FlightGear allows the possibility to add working armament to aircraft. For example, the F-14B can be equipped with fully functioning AIM-9 and AIM-120 air to air missiles that can be launched via a procedure and have lock-on capabilities with the aid of built-in radar in the F-14B. The launching of a missile can be scripted in a similar manner as the manoeuvres.

"Should-have" requirement 3: The MVP should possess high quality graphics, a detailed environment and detailed models

FlightGear encompasses high quality rendering with 1920x1200 resolution with a frame rate higher than 60 frames per second, depending on the used hardware. Different qualities and complexities for rendering environment, models and weather can be adjusted to match the graphical computational power of the computer.

"Should-have" requirement 4: The MVP should allow visualisation through techniques as Augmented Reality or Virtual Reality

As of now, no solid integration of VR techniques is present in FlightGear. There is however a dedicated wiki page available with the current standing of the development of this subject [91]. It is to be expected that the community behind FlightGear will develop this functionality to keep up with other flight simulators.

"Should-have" requirement 5: The MVP should allow for different types of input devices

As of now, the mouse, keyboard and joystick work. The MVP has demonstrated that there is excellent capability for external scripts to be used as input device for FlightGear. Rumours of integration of Leap Motion are not yet to be found, however. The MVP should be able to run on different computers without demanding high system requirements; As mentioned, the rendering settings of FlightGear can be adjusted to be operationable on a variety computers. With low quality renderings settings, FlightGear can run on a consumer based mid/high range laptop (with an i5 processor and 8GB of RAM as benchmark) along with the Python script without problems.

"Should-have" requirement 6: The MVP should allow for operation and development through unlicensed software

This is completely realised by using an open source flight simulator (FlightGear) and open source programming language Python and the free operating system Ubuntu 16.04 LTS on all computers and on the computer that runs the server. The hardware was provided by the NLR and was thus already in their possession. As of now, no expenses had to be made for the development of the MVP.

6.2.3 Demonstration

During the user testing of the MVP, the most notable observation to be made is that the interaction with the UCAV itself is rather self-explanatory, but the largest limiting factor for successful user testing was the fact that as of now is the difficulty of flying together. During the user tests it appeared to be excessively difficult to locate each other in the battlefield and maintain a formation flight when controlling the aircraft with the SDCS Control Interface. The tasks given to the interviewees as described in the test setup (following the adversary aircraft) was so difficult, that the attention was shifted towards attempting to follow an aircraft so that a significant amount of attention was drawn away from the functionality. This decreases the ability of effectively conveying the message of the SDCS with the MVP. Due to the limited amount of manoeuvres, the effectiveness of the SDCS in accomplishing the same task (following the adversary aircraft) was limited. Furthermore, no workload was taken away

from the operator, but only added. There was not enough autonomy on board the UCAV to succesfully perform UCAV related tasks with the MVP. This was acknowledged by Tjebbe, as he mentioned that the possibilites of manoeuvring with the SDCS Control interface was too limited to fulfil tasks. The interaction with the prototype should be natural and obvious for the user, so that the user can focus more on the core functionalities of the SDCS rather than attempting to interact with the MVP. Points of improvement regarding this are made in the Recommendations section.

However, as a side finding, the MVP proved to be rather suitable for demonstrating the core problem associated with UCAV operation; Direct control with a latency on the data link. The fact that joystick control could be delayed was a side effect of the implementation (originally, only attempts were made to induce a delay on the data that was send by the bridging application). Since the joystick utilises the same transport layer and control section, flying with a joystick delayed was now also an option. The delayed joystick operation proved to cause PIO and the SDCS basic interface helped in a less difficult way of controlling the UCAV. Tjebbe also acknowledged that latencies of even 250ms are already problematic for high-gain tasks such as air to air combat. It can therefore be concluded that the MVP can demonstrate the problem that the SDCS attempts to address and solve and also can convince a user that there should be a system such as the SDCS when there is a delay present on the data link. The MVP can not yet demonstrate how the SDCS can prove to be of added value.

6.2.4 Development

The fact that FlightGear possesses an open structure contributes to the possibility to further develop the MVP. The basis for different calculations one may need for example are accessible in FlightGear. Every variable that may be needed for a future functionality can, with a high probability, be found in the property tree of FlightGear. This will be elaborated more in the Recommendations section. In addition, the fact that FlightGear is open source may further contribute to development. FlightGear has an active user community who are dedicated in maintaining and improving FlightGear. It may be the case that certain functionality that is needed for the MVP is already realised by an enthousiast. Since this is an open source community, all source code will be publicly available.

7 Conclusion

The results from the different phases of the Creative Technology Design process, the outcomes of the user test and the answers to the sub research question can be utilised to answer the main research question. In addition the relevance for the NLR and the academic relevance are described. To commence, the sub research questions and their consequtive answers will be stated once more:

What is the SDCS?

The SDCS is further elaborated and is placed in a broader context with the aid of scientific literature in order to evaluate its relative standing in the unmanned paradigm. Important definitions have been defined and treated and the SDCS was placed in a theoretical framework of theory in the unmanned aircraft paradigm.

What is a Minimum Viable Product?

A definition for an MVP was established; an MVP has to fulfil all requirements denoted as "Must-have", which were established in an exceeding chapter of this thesis.

What is the goal of the MVP of the SDCS?

A differentiation between a purely demonstrational platform, a test platform and a blended form has been made to elicit the needs of the stakeholders. These results have been used to further shape the final form of the MVP.

What are the most important elements of the SDCS to be included in the MVP?

A Kano analysis has been performed to elicit the highest priority aspects of both the SDCS and the MVP in order to prioritise what is being treated during the limited timeframe of this thesis. The requirements following from this process have been prioritised in "Must-have" and "Should-have" requirements.

What are suitable methods and tools for developing an MVP of the SDCS?

To answer this sub question, several software packs have been treated and in combination with Related Work section, the answer to this sub question has provided with different tools for the envisioned MVP.

How can this be translated into a MVP of the SDCS?

A software pack has been chosen and the MVP has been developed. This resulted in a low-cost (no expense) open source based MVP consisting utilising FlightGear, Python and Ubuntu. The main research question was formulated in the first chapter of this thesis:

"What elements of the Semi-Direct Control System (SDCS) must be incorporated in a Minimum Viable Product (MVP) to demonstrate the functionality of the SDCS and to function as an effective develop-

ment platform?"

The answers to the sub questions and the results of the Evaluation section allows answering of the main research question. The MVP at its current state has not proven to be fully able to demonstrate effectively what the SDCS should be capable of and thus also be able to convince future stakeholders of the necessity.

The MVP however successfully illustrates the problem that the SDCS attempts to tackle; latency on data links severely impact effective direct control of UCAVs and UAVs in general and limits their effectiveness in these situations. This can be demonstrated by inducing delay on the joystick control. The NLR can leverage this functionality in order to illustrate their motives for the research in the SDCS. The problem of latency in UCAV control can be explained through speech, but being able to allow a person to experience the troubles of flying with delay is more powerful. To effectively demonstrate the functionality of the SDCS the MVP should include more sophisticated versions of elements of the SDCS than it does now. The current possibilities of commands that can be given to the UCAV are too limited to interact with another aircraft. This was visible during the user test, where finding each other and maintaining formation proved to be so difficult, that it was hard to have full focus on engaging in air to air manoeuvres. The most important elements of the SDCS that need to be improved are the manoeuvres and functionality of the CMMS. This claim will be thoroughly elaborated in the Recommendations chapter.

The MVP can function as a development platform for further research. The choice of open source software and the open and accessible architecture of FlightGear has strongly contributed to the MVP being future proof; since it is not yet clear what research is going to be performed on the SDCS in the future, the design choices have been made such that the MVP allows for the addition of external scripts and in addition allows access to all flight variables and parameters that may be needed to realise new functionality of the SDCS. A key test variable, latency, can be simulated so that the MVP can also function as a human factors research platform. This claim will be elaborated in the Recommendations section. Furthermore, since FlightGear is an open source flight simulator, a substantial community lives online that actively contributes to the development of (functions of) FlightGear. This way, the "power of the masses" may contribute to the speed of development.

Thus, the MVP needs further development to fully mature and gain the possibility of demonstrating the concepts of the SDCS. The following chapter, Recommendations, will elaborate on methods for achieving this and furthermore provides the NLR with options to develop the MVP and perform research with it.

The academic relevance lies in the application of open-source, low cost software for developing UAV or UCAV test beds suitable for HIL testing and verification of control concepts. This is especially beneficial to researcher with a smaller budget or limited resources. Furthermore, this tool provides opportunity for assessing the SDCS control concept in the future and helps determining what autonomy should be present for effective application of UCAVs in air to air combat.

8 Recommendations

This chapter handles the recommendations have followed during the research process. It furthermore includes suggestions for the approach of these improvements. These recommendations are part of the deliverables of this thesis.

8.1 Recommendations regarding the improvement and extension of the MVP

8.1.1 Leverage FlightGear functionality with the Property Tree

The first recommendation that can be made is a personal recommendation from the author. During this thesis, FlightGear has repetitively proven to be a powerful tool for realising MVP and SDCS functionality. Leveraging functionality and the structure of FlightGear can greatly contribute to the realisation of desired functions. It could furthermore drastically shorten the time required to develop functionalities. For example, in the Specification phase it was determined that solely the basic manoeuvres (pitching up, pitching down, banking left and banking right) were to become part of the MVP and that the sophisticated manoeuvres required an integrated control loop. After thorough inspection of FlightGear, it was discovered that FlightGear's autopilot system utilises the control loops required for the more difficult manoeuvres (PID controllers). This enabled the realisation of more sophisticated manoeuvres including maintaining altitude and maintaining heading. This in turn extended the basic manoeuvres by enabling to maintain a certain manoeuvre without difficult control structures but by leveraging FlightGear autopilot functionality. It has furthermore lead to an implementation of the "follow function" which has proven to be of critical importance in demonstrating SDCS principles. It strongly contributed to realising more functionality than was initially anticipated to be possible within the research- and time scope.

In short, it is important for a future developer to keep the following steps in mind during the development of functionalities: Firstly, visualise the functionality that is required; secondly, draw out how this should be realised; and lastly, summarise what information, parameters or other data is needed for the realisation. At this point it is wise to utilise the Property Tree browser to browse the available functions and state variables (the target tracking function discovered in this thesis was found by browsing in the property tree in curiosity). Any function or variable one might need for their function is likely to reside inside FlightGear as is. For example, one can look up the compression of the tires and the roll speed of the separate wheels of the landing gear of an aircraft. Furthermore the values that the flight instruments show (including malfunctions and distorted values due to for example frost on the pitot tube) can be accessed. The fuel level of all separate tanks of the aircraft are readily accessible in the units imperial gallons, US gallons, kilograms, pounds and m3. Therefore, everything that would normally be performed from within the cockpit by the pilot can be externally scripted. That includes landing and taking off to all necessary steps to localise an adversary with the on-board radar and firing a missile towards that adversary, all with the press of 1 button that automatically manipulates the state variables in the correct sequence.

8.1.2 Enhance Controller Functionality

The second recommendation handles the control structures inside FlightGear, focussing on the controllers of the autopilot function. These controllers are of the type PID as mentioned in the Realisation phase. In the MVP, the autopilot is being manipulated and utilised as a simple yet effective FMS, thus it is to be recommended to make use of the autopilot functionality in this way. What can be noticed that during the manoeuvres is that once the UCAV reached the desired state (for example steady flight) it tended to "wobble". The author's impression is that this wobbling behaviour is caused by oscillation in the output of the PID controller due to bad controller tuning. The mathematical notation of a PID controller is depicted in equation 8.1

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t) \tag{8.1}$$

In equation 8.1, the Proportional, Integral and Derivative terms are coloured Blue, Green and red respectively. The factors in bold (K_p, K_i *and* K_d) denote the Proportional, Integral and Derivative gain coefficients respectively. These constants can be any arbitrary value and have influence on the output of the PID controller, since the output of the controller is a weighted sum of the P, I and D terms, with K_p, K_i and K_d denoting the weights. Without delving in too much detail, the proportional gain coefficient has influence on the magnitude of the change in output of the PID controller. As the gain of for example the proportional term is increased, the controller will be more sensitive and will respond stronger to an error signal. If the gain is set too low, the controller will react slowly to a large input error and will thus form a less sensitive controller. A too large gain on the other hand may cause overshoot, as seems to be the case in the F-14B. To demonstrate, a couple of PID controller simulations in Excel can be run with different values of K_p, K_i and K_d. For these simulations, K_i, K_d and the time step will be held constant. The output *u(t)* of the PID controller with K_p=10, K_i=0.8 K_d = 0.7 and a time step of 0.025 is denoted in figure 8.1:

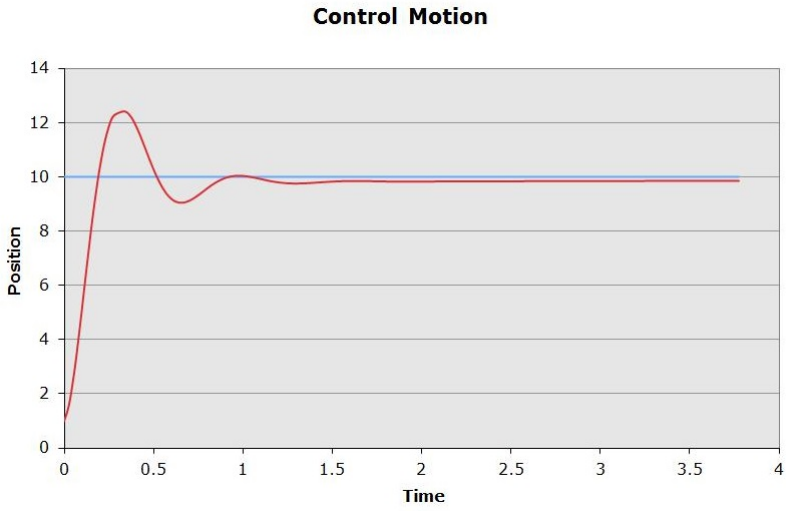


Figure 8.1: *Output of PID controller u(t) with K_p =10, K_i=0.8 K_d = 0.7 and a time step of 0.025*

The red line shows an oscillation similar to the one observed in the F-14B, the aircraft would wobble up and down twice before reaching "steady state". Another simulation is run with a smaller value for K_d, namely 2. The result is depicted in figure 8.2

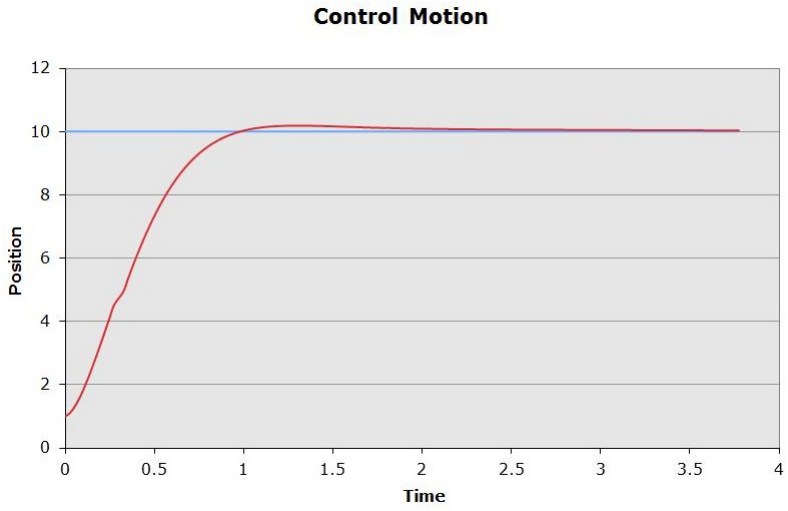


Figure 8.2: *Output of PID controller u(t) with K_p =2, K_i=0.8 K_d = 0.7 and a time step of 0.025*

As can be seen through comparison of figures 8.1 and 8.2, lowering K_d with a factor 5 mitigates the oscillation. However, as can also be observed, the initial curves vary in steepness. This means that with a lower setting for Kd, the aircraft will react more slowly to an issued command. As can be seen, the times it takes to reach steady state remains the same, however with the settings of figure 8.1, the aircraft will manoeuvre fiercer and faster (resulting in higher G turns and -manoeuvres.). As can furthermore be seen, the tuning of these variables has a significant influence on the flight behaviour of the aircraft on autopilot (or on the FMS if it is constructed from PID controllers). The tuning of these variables is a complex but well-documented topic in the field of Control Systems Engineering. The autopilot functionalities can be strongly leveraged during further development of the MVP.

For example, the partially working "follow"-function of the MVP has a controller for the distance that is kept between the UCAV and the aircraft that it is following. Said controller is responsible for managing the throttle of the UCAV. If the distance between adversary and UCAV falls below the threshold value, the controller will sense an error signal and act accordingly by lowering the output that is coupled to the throttle, decreasing throttle until the desired following distance is established. In an analogous way, it increases throttle if the distance exceeds the desired threshold value. Overshoot can be observed if the UCAV demonstrates behaviour in which the throttle is forced back and forward in a periodic motion. If this controller is tuned well, the "following-function", an important function in the MVP, could be realised.

In addition, PID controllers can play a strong role in the realisation of more sophisticated manoeuvres. For example, a manoeuvre could be the approach of an aircraft under a certain aspect angle. One PID controller could be responsible for maintaining the following distance or decreasing the following distance (determines on whether one would want to follow or approach) and one PID controller could maintain the aspect angle by monitoring the UCAVs current heading and heading of the adversary aircraft. In addition, an "aileron roll" manoeuvre could be realised by introducing roll motion and determine the point the aircraft should return to steady level flight. Here, the PID controller can aid the rolling aircraft back to steady level flight. A looping can be realised by setting the desired value of angle of attack to a certain high value so that the aircraft starts pitching. The PID controller can maintain the pitch and at the point that the aircraft has completed a complete 360 degree turn, the PID controller can aid the aircraft back in steady level flight. It should be noted that for such a manoeuvre, the desired value (of for example the elevator deflection) should be changed throughout the manoeuvre for the PID controller to perform the correct manipulation of the elevator. One can imagine that for more difficult manoeuvres, more difficult control structures and sequences have to be written.

To continue elaborating how the correct values for the gain coefficients can be found, one can determine the global desired behaviour of the aircraft (fast response and some overshoot or a slow response with

gradual change) and attempt to establish the correct gain coefficient settings related to this behaviour. The plots in figure 8.1 and 8.2 preview the predicted flight behaviour of the aircraft.

Following this, methods such as the heuristic Ziegler-Nichols method for tuning PID controllers can be performed. As an alternative, a Monte Carlo optimisation can be run by performing sweeps of all three gain coefficients at the same time within a certain range. This will produce a large amount (depending on how large the step sizes between constants are) of different curves. One can pick a curve that resembles the desired behaviour and set the values for the gain coefficients of the autopilot accordingly to alter the UCAV's flight behaviour.

One should repeatedly test the desired coefficient settings with the aircraft selected. It may occur that the output of the controller exceeds the flight envelope of the aircraft. However, developing a UCAV model inside FlightGear (more about this will be handled later in this chapter), allows a valuable tool to experiment with possible manoeuvres or flight behaviour that Unmanned aircraft enable due to the absence of humans and a stronger and lighter construction, as mentioned in Chapter 2.

If desired controller behaviour is determined that is outside of the flight envelope, one could furthermore adjust the scale of coupling between controller output and the control surfaces; limiting (or damping) the output of the controller to the control surfaces by a certain fraction may maintain the desired flight behaviour but avoid exceeding the flight envelope.

8.1.3 Designing a UCAV test model inside FlightGear

During the Realisation phase, an effort has been made to create a test UCAV aircraft. However, changing the livery to an NLR livery and decreasing the weight by 400 pounds is far from realistically approximating an UCAV. However, FlightGear allows users to create their own aircraft. When research into the SDCS is in a developed stage, one could consider modelling a UCAV inside FlightGear. The model of the UCAV can be designed with the free 3D-modelling software Blender [92]. This allows for the design of a completely custom Flight Dynamics Model to effectively simulate UCAV flight behaviour as discussed in Chapter 2. One could recycle certain systems of existing aircraft since these aircraft fall under the same open source license as FlightGear. For example, the armament system of the F-14B can be recycled since it is fully functioning. Through the property tree, the firing commands and radar functions can be externally scripted so that missiles can for example be fired by a single click on the GUI. It would make sense to list what kind of sensors and systems are needed aboard a real life operating UCAV. After listing these functions, one can determine what systems readily exist in the models distributed with FlightGear. If a realistic model is constructed that enables for higher speeds, higher load limits and higher manoeuvrability and can be combined with more aggressive control sequences (PID controller behaviour) to determine how far the limits of this aircraft can be stretched in combat situations. An accurate representation with decent assumptions should allow researchers to draw conclusions on whether the higher load limits and higher manoeuvrability that is claimed in literature actually is of added value in air to air combat. Additionally, a custom aircraft can be fully adjusted to the research needs and/or restrictions.

8.1.4 Machine Learning in Air to Air Combat

Once sophisticated and dynamic manoeuvres have been implemented, a recommendation can be made regarding the "intelligence" aboard the UCAV. Chapter 2 mentions that a UCAV equipped with the SDCS should be able to determine the most efficient, the fastest or the safest execution of the issued command. To acquire that, the intelligence of the UCAV needs to be extended. One way to realise this is through machine learning techniques. A possibly suitable technique that is already successfully implemented by the NLR is the reinforcement learning technique Dynamic Scripting (DS) [93]. It aims at teaching behaviour by means of unsupervised learning. DS utilises a database of weighted actions (or manoeuvres) to be taken and selects certain actions from this database to be performed. After

performing these actions, feedback is given regarding if the action was successful or not. In case of a success, the weight of that certain manoeuvre is increased so that in the next iteration, this manoeuvre has a higher probability to be selected again. In the application of air to air combat, this can be translated as follows: the database of actions represent different combat manoeuvres (high yo-yo, low yo-yo, speed-brake reversal, spiral dive and so forth). Secondly, different criteria that determine success and failure have to be determined. For example the following base case in which the adversary aircraft is flying behind the UCAV. This means the UCAV is in a less preferable situation since the adversary can simply engage the UCAV from behind. The desired setting would be the other way around; the UCAV is located behind the adversary. In this case, the latter situation is denoted as success and the first as failure. Now, different manoeuvres can be selected from the database and be performed. After performing this manoeuvre it can be evaluated if there is success (the UCAV ended up behind the adversary) or failure (the UCAV is still located in front of the adversary). The manoeuvres weight in the database is adjusted and depending on whether there was success or not, this manoeuvre will probably or will probably not be selected the next time the UCAV finds itself in this identical situation (base case). This way, the UCAV equipped with SDCS can be trained to perform manoeuvres depending on the situation and rules of engagement through machine learning techniques. NLR possesses expertise in application of DS in the modelling of CGF [94] and even in the simulation of air combat scenarios [95]. These same CGFs could be used to train the UCAV equipped with SDCS in combat scenario's. CGF technology is mature and can deliver unpredictable behaviour.

8.1.5 Alternative GUI for the MVP

As of now, the MVP utilises Pyhton's Tkinter package for the GUI. During the development it could be noted that enabling Tkinter to function alongside the Bridging application was troublesome. It was mentioned in the Realisation phase that multithreading can be performed in Python. However, the Tkinter library inherently takes the main thread of the Python program for the interface and needs constant access to the thread. This was the main reason for implementing a Finite State Machine for the manoeuvres. Furthermore, passing variables between other threads and the main thread seemed troublesome. Pythons Global Interpreter Lock (GIL) only allows one thread to access a variable at any time and thus in the future, the use of locks on variables [96] will be inevitable. In addition to this, the MVP requires a more natural and more effective user interface. Different ideas have been realised during short feedback sessions of the requirements. The first option would be to integrate the GUI (and possibly also the Bridging Application) into FlightGear. To integrate the Bridging Application into FlightGear, one should become familiar with Nasal scripting [97] or Canvas [98]. Nasal scripting is used to create internal functions and dialog menus in FlightGear and Canvas is an image with OpenGL textures that can be drawn and manipulated in game. For example, in order to replace the manoeuvres that the MVP possesses now (left, right, up and down) an interactive overlay inside FlightGear could be realised as follows:

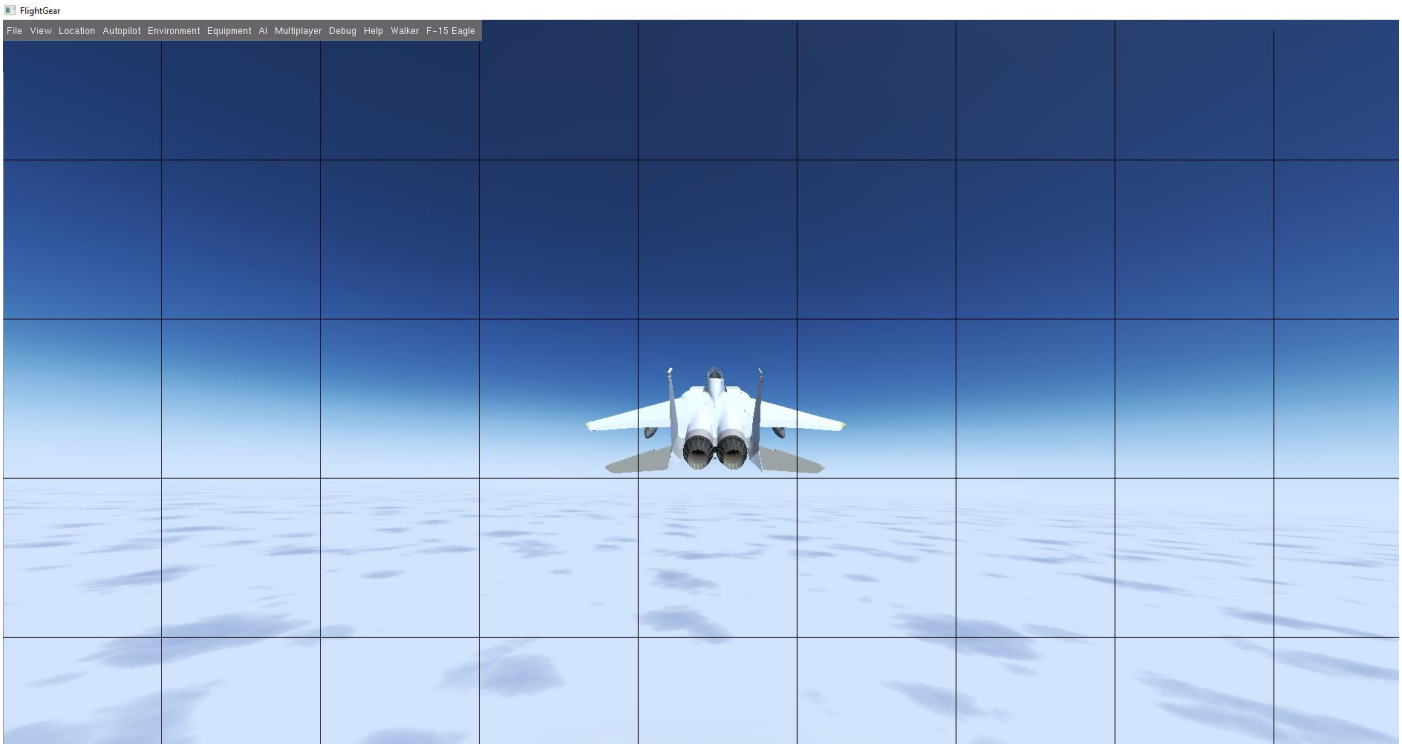


Figure 8.3: *Grid overlay GUI structure in FlightGear*

Assume that the screen is parted into a grid (which may or may not be visible to the user). The user can now click on a tile to mark where he or she desires the UCAV to fly to. This can be achieved through either a transparent overlay within FlightGear or on top of the running FlightGear instance. If for example the user clicks the middle tile (where the UCAV is situated in the image), the UCAV will maintain steady level flight. If the user clicks on the first tile to the left of the centre, the UCAV will bank left. If the user clicks on the second tile to the left of the centre, the UCAV will bank sharper and so forth. This way, the user is provided with more degrees of freedom for controlling the UCAV in comparison to the current MVP. In addition, upon clicking a tile in the grid, the user could be presented with a drop-down menu with options for manoeuvres.

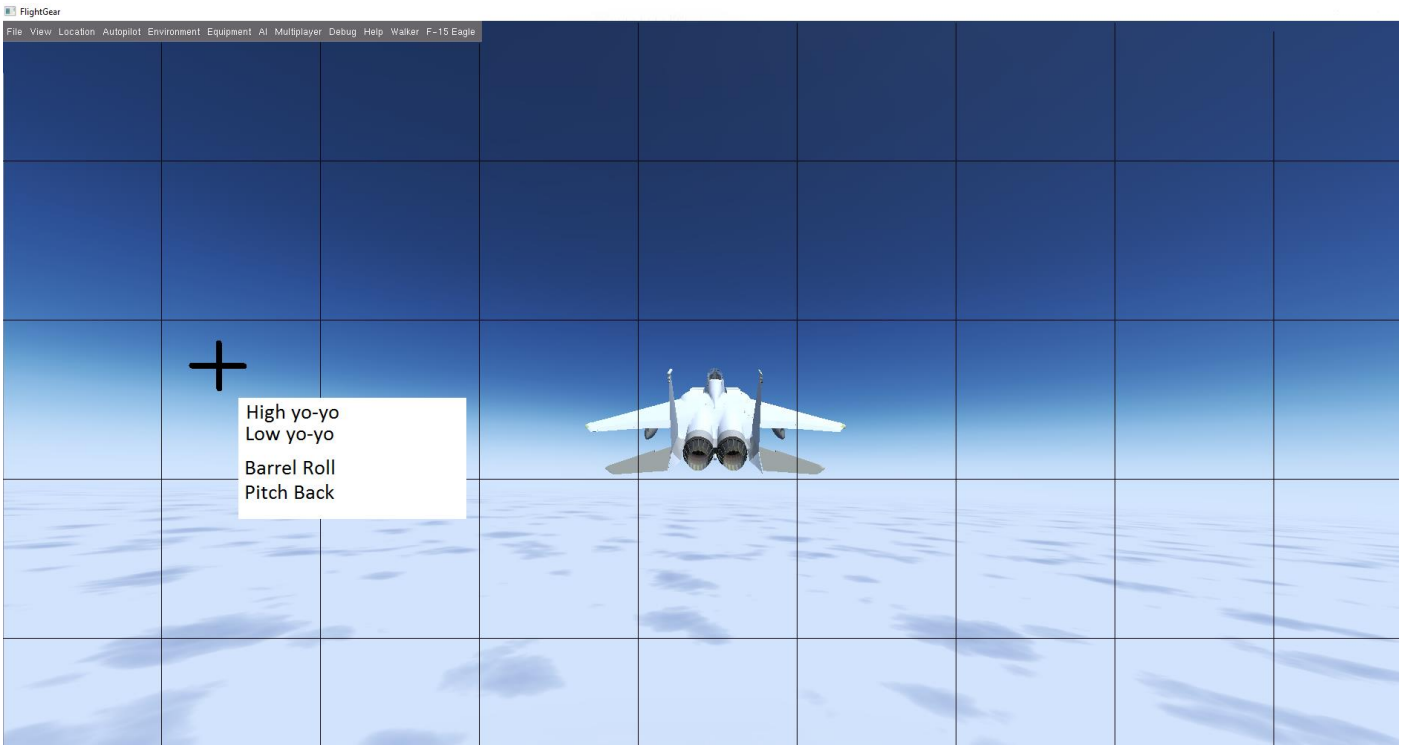


Figure 8.4: *Grid overlay GUI structure with drop down command menu in FlightGear*

Upon clicking on a tile and selecting a manoeuvre, the UCAV could fly towards that tile by means of the selected manoeuvre. This manner of interfacing requires the controllers to be rather sophisticated, but might form a more intuitive and natural form of interfacing, especially when the incorporation of VR and motion or gesture techniques (as described in Chapter 2 and the Ideation chapter) are treated where the operator could literally point towards the desired direction.

8.1.6 Primary Flight Display or Head-Up Display for Flight Parameters

During the user test it was observed that the current method of logging parameters was not ideal for the operator. The display of the 6 basic flight instruments depicted in figure 8.5 as opposed to solely the numbers in the GUI is more intuitive and more informative.

Figure 8.5: *The 6 basic flight instruments*

The FlightGear community has made several efforts in realising a 2D flight instruments panel as depicted in 8.5. FGPanel is originally designed as a light weight standalone panel that does not require sophisticated hardware to be operated [99]. The FGPanel software has been discontinued, however the source code is available through a repository and therefore it might be possible to reconstruct the program. The panel was originally only designed for the Cessna 172P.

Another open source project that, from impression, has been discontinued is FG Glass Cockpit [100]. The source code is still available form source forge [101]. The early version included a Primary Flight Display (PFD) for the Boeing 777, depicted in figure 8.6:

Figure 8.6: *PFD of the FlightGear Glass Cockpit coupled to the avionics of a Boeing 777*

In addition, a third open source Glass Cockpit Project can be mentioned; OpenCG (Open Glass Cockpit). OpenGC is developed as a simulator independent software pack able to operate on various operating systems [102]. The PFD provided by OpenCG is depicted in figure 8.7:

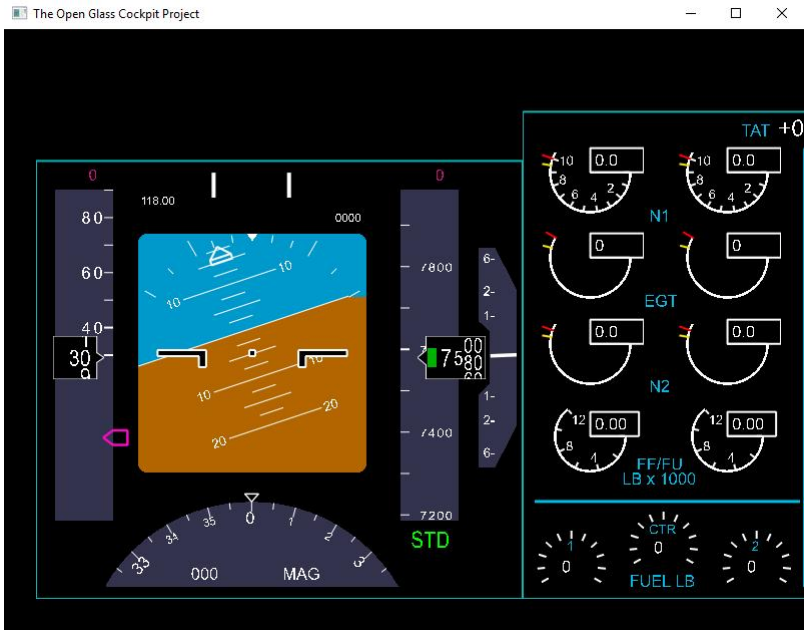


Figure 8.7: *PFD of the OpenCG sofware pack*

Another option would be to utilise a built-in functionality of FlightGear, the Head Up Display (HUD). As with a multitude of functionalities within FlightGear, the HUD is scripted in .xml files and various layouts for the HUD can be designed and implemented [103]. Figure 8.8 depicts the HUD of FlightGear.

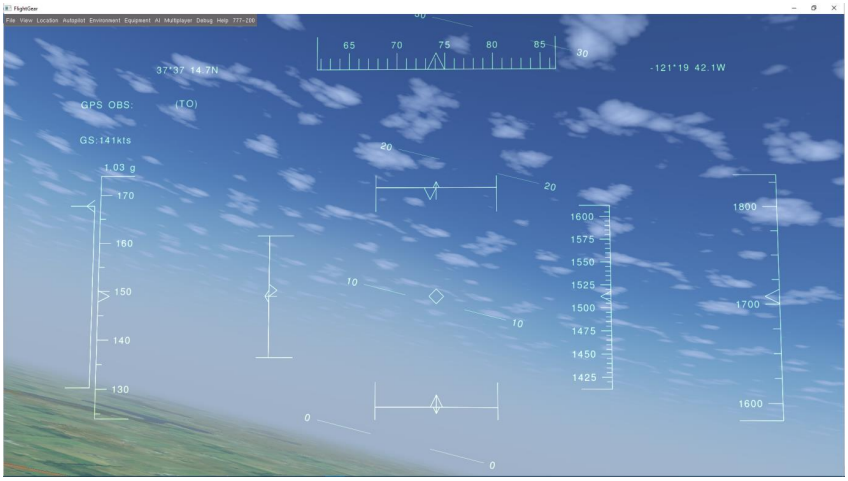


Figure 8.8: *FlightGear HUD as alternative for a PFD*

To elaborate, if the camera view “SDCS Custom View - POV” that is realised in the MVP is combined with the HUD, one can create an image that is far more representative for UCAV or UAV control in general (much like ROVATTS, chapter 2). Most operators are relying on video feed from the UCAV in combination with the flight data either represented in a PFD or HUD. Thus figure 8.8 actually represents a substantially more realistic picture of UAV operation than any other camera view angle in combination with the GUI of the Bridging Application that have been utilised in the MVP so far. To mimick the control of a UCAV equipped with a gimbal mounted camera, the operator can use the mouse or joystick to move the camera view around as like moving the camera position to ”scan” the surroundings.

Another option that can be realised is to define a new camera view from within the cockpit, aimed at the instruments of the aircraft. This second camera view can be projected on a second screen by the same Master/Slave configuration as was applied for the Realisation of the latency requirement.

8.1.7 Communication Protocol Bridging Application and FlightGear

Currently, as stated in the Realisation section, the communication between the Bridging Application and FlightGear is handled by the Telnet server with TCP. It was furthermore mentioned that TCP utilises a three-way handshake to establish a secure connection between client and server. This is schematically depicted in Figure 8.9

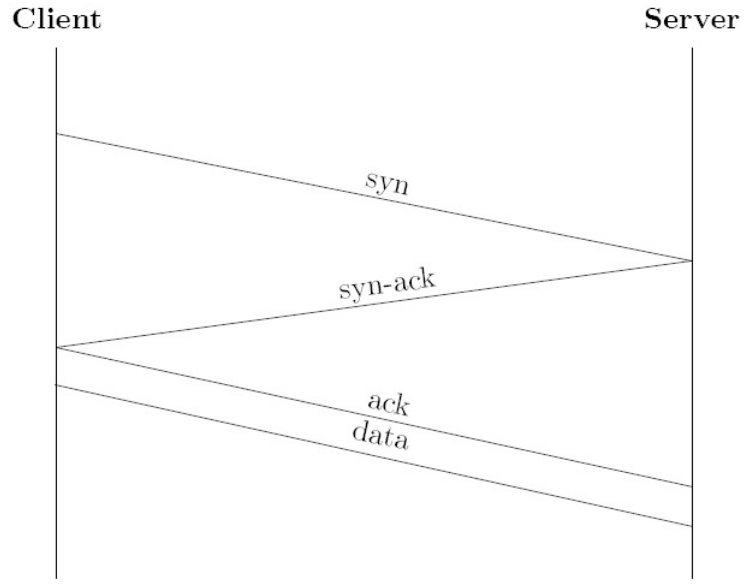


Figure 8.9: *TCP three-way handshake to establish a connection between client and server*

As can be deducted from figure 8.9, the transmission of 1 TCP data packet in total requires three messages back and forth. When inducing delay during the simulation, all three packets will be delayed, causing the 1-way delay to be three times as large as is set by the queuing discipline. For example, if the 1-way delay is set to 250ms, it will actually take 750ms for the packet to correctly arrive at the server. UDP does not suffer from this increase in delay since it is a stateless protocol. UDP rather sends away the data without establishing the connection with the client first and each packet can be delayed with the desired delay. This is exactly the reason that the joystick can be delayed properly: It directly sends its data to the master instance over UDP. With short delay times, for example 50-100ms 1-way delay, this effect is barely noticeable. However, when applying larger delays, it can be clearly seen that the total delay is far larger. It therefore makes sense to investigate the option to deviate from communication over the telnet server (which only supports TCP) to a data link between the Bridging Application and FlightGear that utilises UDP instead. If it is chosen upon to integrate the Bridging Application inside FlightGear, this may no longer be an issue.

8.1.8 Delaying of Adversary

As of now, the adversary that is seen through the GCS (MVP), is not yet delayed. It should be the case that the adversary is only visible after the one way delay towards the operator since the operator can not directly see the adversary, but is relying on the information gathered by the UCAV to determine where the adversary is located. This is the case because the data about the adversary is not send to the slave over the same delayed UDP line as the FDM information, but directly deliverd by the server over the switch. However, the server communicates with the instances over UDP so the connection could be delayed in a similar fashion as the Master-slave configuration. However, only the server output to the slave instance should be delayed, so perhaps the datastreams from the server towards the instances should be separated (send through different ports) and the right port should be delayed to target only the outgoing stream towards the slave instance.

8.2 Recommendations regarding research projects with aid of the MVP

In addition to the functional improvements for the MVP, some recommendations for research projects with aid of the MVP can be designed.

8.2.1 Determine latency threshold for the occurrence of Pilot Induced Oscillation in Direct Joystick Control

As a side effect of inducing the latency it was discovered that the regular joystick input can also be delayed since the joystick data is transferred over UDP. This offers the possibility to perform research into finding the threshold at which operation of an unmanned aircraft by means of direct joystick becomes too difficult due to induced latency on the data link. A test setup would consist of the MVP and a defined task for a pilot or operator. A task could be a final approach towards a runway or an aerial refuel. It is important to make this a task that is related to UCAV operation. One could then let the pilot perform this task with different latency settings to determine at what amount of latency Pilot Induced Oscillation (PIO) or control problems in general will occur. A benchmark needs to be created in which it is determined what is denominated as a PIO or a task failure. Say that if for example this research finds that at 500ms round trip delay, direct joystick control becomes impossible (due to PIO or other problems) this can be used as benchmark to validate different types of communication methods between GCS and UCAV. If for example is established that 500ms round trip delay time is the threshold for joystick control, one can compare this to the amount of latencies that is typically present in different communications protocols (satellite connection, sky wave propagation connection or other types of electromagnetic waveform communication channels). If there proves to be no communication method that can be operated with a delay smaller than 500ms, this can form a strong selling point for the SDCS. Through research it can then be stated that with probability, the necessity of another control method (in this case the SDCS) is evident.

Another research project could include pilots flying at certain (to them unknown latencies) and letting the pilot estimate how much latency is on the link. This research can provide information in how pilots experience latency and further research how they take this into their decision making processes.

8.3 Recommendations and thoughts regarding the SDCS

During the development of the MVP, some insights were gained regarding the SDCS itself. In the scope of this thesis it was stated that this thesis will not answer questions regarding the feasibility of the SDCS or any questions regarding the development or inventarising any of the techniques, technologies, hardware or software required for the development of the SDCS. However, ideas that have arisen might be useful for a future developer. They are not based on scientific reasoning per se and are speculations and personal experiences of the author, rather than research that is covered within the scope of this thesis. Therefore, they are not included in this thesis as a section, but it has been decided to place these in appendix M. This appendix is written on a more personal note than the body of this thesis.

References

[1] R. Austin, *Unmanned aircraft systems: UAVS design, development and deployment*, vol. 54. John Wiley & Sons, 2011.

[2] B. Tice, "Unmanned aerial vehicles – the force multiplier of the 1990s," *Airpower Journal*, vol. 5, no. 1, pp. 41–54, 1991.

[3] D. Bookstaber, "Unmanned combat aerial vehicles: What men do in aircraft and why machines can do it better," *Air and Space Power Journal*, pp. 4–5, 6 2000.

[4] M. Mouloua *et al.*, eds., *Ergonomics of UAV/UCAV mission success: Considerations for data link, control, and display issues*, vol. 45 of *2*, (Los Angeles, CA), Proceedings of the Human Factors and Ergonomics Society Annual Meeting, SAGE Publications Sage CA, 2001.

[5] K. Williams, "A summary of unmanned aircraft accident/incident data: Human factors implications," Tech. Rep. 2, Civil Aerospace Medical Institute, Washington, DC 20591, 12 2004.

[6] W. Feichtinger and B. Hensellek, *Armed Forces for 2020 and beyond*, ch. 2, pp. 73–86. Vienna, Austria: Republik Österreich / Bundesministerium für Landesverteidigung und Sport, 3 2016. The Chapter "Unmanned Aircraft Systems against Air Threats: The Semi-Direct Control System" written by Hans Heerkens and Frank Tempelman introduce the subject of the Semi-Direct Control System. This thesis has evolved from this article and refers to concepts and statement in this article throughout.

[7] E. Svensson *et al.*, "Information complexity-mental workload and performance in combat aircraft," *Ergonomics*, vol. 40, no. 3, pp. 362–380, 1997.

[8] J. L. Drury *et al.*, "A decomposition of uav-related situation awareness," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 88–94, ACM, 2006.

[9] G. Kromhout, "Exercise report: Frisian flag," *Air Forces Monthly*, vol. 328, 6 2015.

[10] W. Kross, *Military reform: the high-tech debate in tactical air forces*. National Defense University Press, 1985.

[11] H. Heerkens, "Lectures on aerospace industry." The lecture material that is provided on the course "Aerospace Industry" by Hans Heerkens as part of the minor "Aerospace Management and Operations" taught at the University of Twente in Enschede, the Netherlands, 2 2015.

[12] E. Riccioni, "Description of our failing defense acquisition system." Retrieved at 27-02-2017 from http://www.pogo.org/m/dp/dp-fa22-Riccioni-03082005.pdf, 3 2005.

[13] P. Higby, *Promise and Reality: Beyond Visual Range (BVR) Air-To-Air Combat*. PhD thesis, Virginia Military Institute, Maxwell, 2005.

[14] J. Greenert, ed., *Payloads over Platforms: Charting a New Course*, vol. 138 of *7*, United States Naval Institute, 7 2012.

[15] R. Frampton, "Mission applications and concepts of operations," *RTO Educational Notes: Applications, Concepts and Technologies for Future Tactical UAVs*, pp. 30–31, 11 2004.

[16] J. Stillion, "Trends in air-to-air combat: implications for future air superiority," tech. rep., Center for Strategic and Budgetary Assessments, Washington DC, 2015.

[17] S. Dougherty, "An examination of latency and degradation issues in unmanned combat aerial vehicle environments," *Air Force Institute of Technology*, pp. 12–13, 3 2002.

[18] R. Blackhurst, "The air force men who fly drones in afghanistan by remote control," *The Telegraph*, 9 2002. Accessed at 21-06-2017 Retrieved from:`http://www.telegraph.co.uk/news/uknews/defence/9552547/The-air-force-men-who-fly-drones-in-Afghanistan-by-remote-control.html`.

[19] R. F. Laird, "A 21st-century concept of air and military operations," *Defense Horizon*, vol. 66, 2009.

[20] B. Clough, "Metrics, schmetrics! how the heck do you determine a uav's autonomy anyway?," in *AIAA's 1st Technical Conference and Workshop on Unmanned Aeropsace Vehicles*, (Portsmouth, VA), Air Force Research Lab, 5 2012.

[21] "Autonomous, def. 3." Retrieved from `http://www.dictionary.com/browse/autonomous?s=t` at 03-13-2017.

[22] "Intelligence, def. 1." Retrieved from `http://www.dictionary.com/browse/intelligence?s=t` at 03-13-2017.

[23] "Automation, def. 1." Retrieved from `http://www.dictionary.com/browse/automation?s=t` at 03-13-2017.

[24] K. Fox, "Lectures on automation versus autonomy." The lecture material that is provided by Kyle Fox who operates as Senior Systems Engineering Manager on Combat Systems and Platform Integration at Raytheon Massachusetts, USA, 12 2014.

[25] R. Hopcroft *et al.*, "Unmanned aerial vehicles for maritime patrol: Human factors issues," tech. rep., Defence Science and Technology Organisation, Port Melbourne, Australia, 5 2006.

[26] H. Ruff *et al.*, "Human interaction with levels of automation and decision-aid fidelity in the supervisory control of multiple simulated unmanned aerial vehicles," *Presence*, vol. 11, no. 2, pp. 335–351, 2002.

[27] M. Mustapha *et al.*, "Workload, situation awareness, and teaming issues for uav/ucav operations," *Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting*, pp. 162–165, 11 2001.

[28] P. A. Hancock, "A dynamic model of stress and sustained attention," *Human factors*, vol. 31, no. 5, pp. 519–537, 1989.

[29] J. S. McCarley and C. D. Wickens, *Human factors implications of UAVs in the national airspace*. University of Illinois at Urbana-Champaign, Aviation Human Factors Division, 2005.

[30] S. F. Scallen and P. A. Hancock, "Implementing adaptive function allocation," *The International Journal of Aviation Psychology*, vol. 11, no. 2, pp. 197–221, 2001.

[31] J. Boyd, "Essence of winning and losing." Unpublished briefing by John Boyd, slides Edited by Chet Richards and Chuck Spinney. Produced and designed by Ginger Richards. Retrieved from: `http://pogoarchives.org/m/dni/john_boyd_compendium/essence_of_winning_losing.pdf` at 03-14-2017, 8 2010.

[32] M. J. Sullivan, "Gao-13-294sp defense acquisitions: Assessments of selected weapon programs," tech. rep., United States Government Accountability Office, Washington, DC, 3 2013. Retrieved from `http://www.gao.gov/assets/660/653379.pdf` on 03-15-2017.

[33] "Rq-4a global hawk uav," 2002. Retrieved from http://www.dote.osd.mil/pub/reports/fy2002/pdf/af/2002rq-4aglobalhawkuav.pdf on 03-15-2017.

[34] "Eads press release 27 july 2009," 7 2009. Retrieved from: http://www.airbusgroup.com/int/en/news-media/press-releases/20120723_cassidian_barracuda.html on 03-20-2017.

[35] M. Bartenbach, ed., *BARRACUDA – TCAS demonstration with an UAV*, (Toulouse, France), 22nd Annual European Chapter Symposium, The Society of Flight Test Engineers, 6 2011.

[36] "Dassault website on neuron." Retrieved from: http://www.dassault-aviation.com/en/defense/neuron/introduction/ at 03-20-2017, 7 2003.

[37] J. K. Ramage, "Critical uav application issues and system trade-off factors," *RTO Educational Notes: Applications, Concepts and Technologies for Future Tactical UAVs*, 6 2003.

[38] N. Sarter and D. D.Woods, "How in the world did we ever get into that mode? mode error and awareness in supervisory control," *Human Factors*, no. 37, pp. 5–19, 1995.

[39] J. B. F. van Erp, "Controlling unmanned vehicles: The human factors solution," in *Advances in vehicle systems concepts and integration: RTO Meeting Proceedings*, vol. 44, DTIC Document, 2000.

[40] "The uk approach to unmanned aircraft systems." Development, Concepts and Doctrine Centre, UK Ministry of Defence, 3 2011. Retrieved from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/33711/20110505JDN_211_UAS_v2U.pdf on 03-20-2017.

[41] P. C. Nolin, "Unmanned aerial vehicles: opportunities and challenges for the alliance," *NATO Parliamentary Assembly*, 2012.

[42] "United states air force unmanned aircraft systems flight plan 2009-2047." United States Air Force, Washington DC, 5 2009. Retrieved from: https://fas.org/irp/program/collect/uas_2009.pdf on 03-20-2017.

[43] M. W. Byrnes, "Nightfall, machine autonomy in air-to-air combat," *Air & Space Power Journal*, pp. 48–75, 5 2014.

[44] J. S. McGrew *et al.*, "Air-combat strategy using approximate dynamic programming," *Journal of guidance, control, and dynamics*, vol. 33, no. 5, pp. 1641–1654, 2010.

[45] W. G. Bessemer, "Transitioning to unmanned combat aerial vehicles," Master's thesis, Naval Postgraduate School, Monterey, California, 9 2006.

[46] Image of the MQ-9 GCS Accessed on 14-04-2017. Retrieved from: https://upload.wikimedia.org/wikipedia/commons/6/6a/MQ-1_Predator_controls_2007-08-07.jpg.

[47] UAV Factory Portable GCS Datasheet. Accessed at 14-04-2017 Retrieved from: http://www.nexutech.com/Portable_GCS_Datasheet_v2.3.pdf.

[48] Mission Planner Software. Accessed at 14-04-2017 Retrieved from: http://ardupilot.org/planner/docs/mission-planner-overview.html.

[49] MAVLink Github page. Accessed at 14-04-2017 Retrieved from: https://github.com/mavlink/mavlink/commit/a087528b8146ddad17e9f39c1dd0c1353e5991d5.

[50] QGroundControl Website. Accessed at 14-04-2017 Retrieved from: http://qgroundcontrol.com/.

[51] MAVProxy Github page. Accessed at 14-04-2017 Retrieved from: https://github.com/ArduPilot/MAVProxy.

[52] UGCS Website. Accessed at 14-04-2017 Retrieved from: https://www.ugcs.com/.

[53] MAVPilot release article. Accessed at 15-04-2017 Retrieved from: http://diydrones.com/profiles/blogs/mav-pilot-1-4-for-iphone-released.

[54] Phoenix 5 R/C website. Accessed at 15-04-2017 Retrieved from: http://www.phoenix-sim.com/.

[55] Aerosim RC website. Accessed at 15-04-2017 Retrieved from: http://www.aerosimrc.com/en/home.htm.

[56] A. Cihangir, "Stanag 4586 edition 3," tech. rep., NATO Standardization Agency, Brussels, Belgium, 11 2012.

[57] ROVATTS brochure from SDS International. Accessed at 15-04-2017 Retrieved from: http://www.atdlink.com/brochures/ROVATTS_16v1.pdf.

[58] A. Sanna *et al.*, "A kinect-based natural interface for quadrotor control," *Entertainment Computing*, vol. 4, p. 179–186, 8 2013.

[59] M. Chandarana *et al.*, "A natural interaction interface for uavs using intuitive gesture recognition," *Advances in Human Factors in Robots and Unmanned Systems*, vol. 499, 7 2016.

[60] T. Schouwenaar *et al.*, "Linear programming and language processing for human/unmanned-aerial-vehicle team missions," *Journal of guidance, control and dynamics*, vol. 29, 4 2006.

[61] A. Dachis, "Bitcraze makes drone flying feel like telekinesis on the hololens." Hololens Website, 1 2016. Accessed at 16-04-2017 Retrieved from: https://hololens.reality.news/news/bitcraze-makes-drone-flying-feel-like-telekinesis-hololens-0172414/.

[62] B. Walter *et al.*, "Virtual reality control of unmanned vehicles," *AIAA 3rd "Unmanned Unlimited" Technical Conference*, 9 2004.

[63] J. P. Hansen *et al.*, "The use of gaze to control drones," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 27–34, ACM, 2014.

[64] K. LaFleur *et al.*, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface," *Journal of neural engineering*, vol. 10, no. 4, 2013.

[65] A. Mader and W. Eggink, "A design process for creative technology," in *Proc. 16th international conference on engineering and product design education 'Design education & human technology relations'*, (Enschede, the Netherlands), Sept. 2014.

[66] H. Sharp *et al.*, "Stakeholder identification in the requirements engineering process," in *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*, pp. 387–391, 1999.

[67] R. K. Mitchell *et al.*, "Toward a theory of stakeholder identification and salience: Defining the principle of who and what really counts," *Academy of management review*, vol. 22, no. 4, pp. 853–886, 1997.

[68] P. Bradwell and S. Marr, "Making the most of collaboration: An international survey of public service co-design," Tech. Rep. 23, Demos, London, England, 2008.

[69] N. Kano *et al.*, "Attractive quality and must-be quality," *Hinshitsu: the Journal of the Japanese Society for Quality Control*, pp. 39–48, 5 1984.

[70] Y. H. Chen and C. T. Su, "A kano-ckm model for customer knowledge discovery," *Total Quality Management & Business Excellence*, vol. 17, no. 5, pp. 589–608, 2006.

[71] K. Matzler and H. H. Hinterhuber, "How to make development projects more successful by integrating kano's model of customer satisfaction into quality function deployment," *Technovation*, vol. 18, no. 1, pp. 25–38, 1998.

[72] J. R. Hauser and D. Clausing, "The house of quality," *Harvard Business Review*, pp. 63–73, 5 1988.

[73] D. Clegg and R. Barker, *Case Method Fast-Track: A RAD Approach*. Boston, MA, USA: Addison-Wesley Longman Publishing Co. Inc., 1994.

[74] B. Davis, *Agile Practices for Waterfall Projects: Shifting Processes for Competitive Advantage*. Florida, USA: J. Ross Publishing, 10 2012.

[75] B. Wernham, *Agile Project Management for Government*. London, England: Maitland and Strong, 1 ed., 2012.

[76] "The mathematical blade element theory in x-plane." X-Plane Website. Accessed at 26-05-2017 Retrieved from: `http://www.x-plane.com/desktop/how-x-plane-works/`.

[77] "Faa certification of x-plane." X-Plane Website. Accessed at 26-05-2017 Retrieved from: `http://www.x-plane.com/pro/certified/`.

[78] Z. H. Qi *et al.*, "Airborne imu simulation based on simulink and flightgear," *Journal of Chinese Inertial Technology*, vol. 16, no. 4, pp. 400–403, 2008.

[79] H. Huang *et al.*, "Real-time visual flight simulation system based on flightgear simulator," *Journal of System Simulation*, vol. 19, no. 19, pp. 780–785, 2007.

[80] S. Kurnaz *et al.*, "Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1229–1234, 2010.

[81] I. F. Nusyirwan, "Engineering flight simulator using matlab, python and flightgear," *SimTecT, Melbourne, Australia*, 2011.

[82] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach: International Edition*. Pearson Education Limited, 2013.

[83] "Github user mrfranz has written the telnet class and the flightgear class that represent the communication between flightger and the bridging application. this reference provides a link to the github page." Github. Accessed at 26-05-2017 Retrieved from: `https://sourceforge.net/p/flightgear/flightgear/ci/1d15a9272b962f32c07f22f9ee56d014fffa2930/log/?path=/scripts/python/FlightGear.py`.

[84] T. Koshy, *Discrete Mathematics With Applications*. Academic Press, 2004.

[85] J. van Amerongen, *Dynamical Systems for Creative Technology*, vol. 4 of *10*. Enschede, the Netherlands: Controllab Products B.V., 2010.

[86] "Block scheme of a pid controller." Wikipedia. Accessed at 20-06-2017 Retrieved from: `https://www.dewesoft.com/assets/img/pro/uploads/PIDfront.PNG`.

[87] "Unix netem kernel package." Linux Foundation. Accessed at 20-06-2017 Retrieved from: `https://wiki.linuxfoundation.org/networking/netem`.

[88] "General public license." GNU Software. Accessed at 21-06-2017 Retrieved from: `https://www.gnu.org/licenses/quick-guide-gplv3.html,moreontheGPL`.

[89] "Classes and instances in python." The Python Foundation. Accessed at 21-06-2017 Retrieved from: https://docs.python.org/2/tutorial/classes.html.

[90] "Python executive summary." The Python Foundation. Accessed at 21-06-2017 Retrieved from: https://www.python.org/doc/essays/blurb/.

[91] "Current standing of vr implementation in flightgear." FlightGear wiki. Accessed at 21-06-2017 Retrieved from: http://wiki.flightgear.org/OculusImplementation.

[92] "Blender 3d modelling software." Blender foundation. Accessed at 21-06-2017 Retrieved from: https://www.blender.org/.

[93] P. Spronck *et al.*, "Adaptive game ai with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.

[94] A. Toubman *et al.*, "Modeling cgf behavior with machine learning techniques: Requirements and future directions," in *Proceedings of the 2015 Interservice/Industry Training, Simulation, and Education Conference*, pp. 2637–2647, 2015.

[95] A. Toubman *et al.*, "Dynamic scripting with team coordination in air combat simulation," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 440–449, Springer, 2014.

[96] "Locking of variables in python." The Python foundation. Accessed at 21-06-2017 Retrieved from: https://docs.python.org/2/library/threading.html.

[97] "Nasal scripting." FlightGear Wiki. Accessed at 21-06-2017 Retrieved from: http://wiki.flightgear.org/What_is_Nasal.

[98] "Canvas." FlightGear Wiki. Accessed at 21-06-2017 Retrieved from: wiki.flightgear.org/Canvas.

[99] "Fgpanel." FlightGear Wiki. Accessed at 21-06-2017 Retrieved from:http://wiki.flightgear.org/FGPanel.

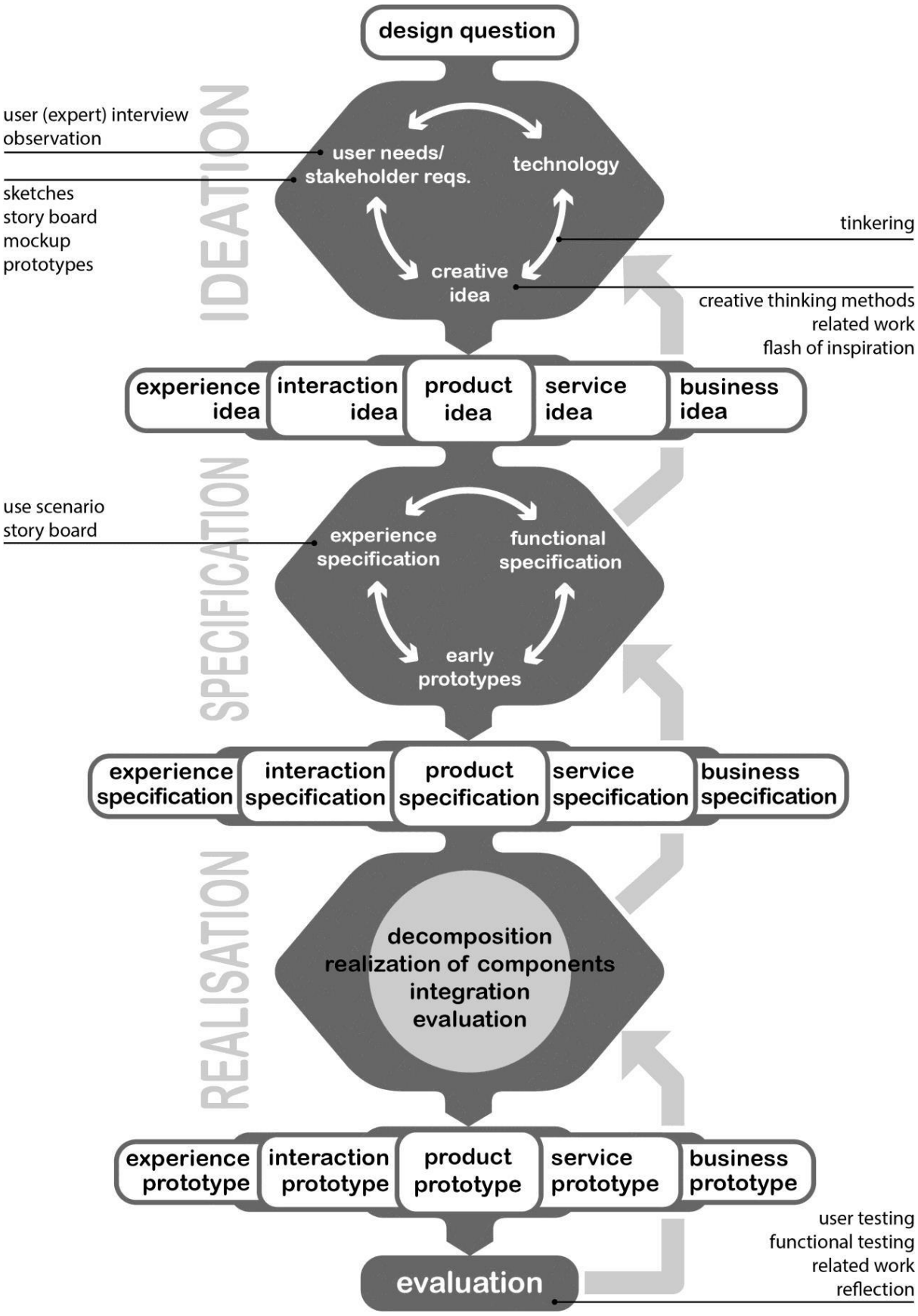
[100] "Flightgear glass cockpit." FlightGear Wiki. Accessed at 21-06-2017 Retrieved from:http://wiki.flightgear.org/FlightGear_Glass_Cockpit.

[101] "Flightgear glass cockpit source code." SourceForge. Accessed at 21-06-2017 Retrieved from:https://sourceforge.net/projects/fggc/?source=typ_direct.

[102] "Open glass cockpit project." SourceForge. Accessed at 21-06-2017 Retrieved from:http://opengc.sourceforge.net/.

[103] "Flightgear hud." FlightGear Wiki. Accessed at 21-06-2017 Retrieved from:http://wiki.flightgear.org/Head-up_display.

A Creative Technology Design Process


design question
IDEATION
user (expert) interview
observation
sketches
story board
mockup
prototypes
user needs/
stakeholder reqs.
technology
creative
idea
tinkering
creative thinking methods
related work
flash of inspiration
experience
idea
interaction
idea
product
idea
service
idea
business
idea
SPECIFICATION
use scenario
story board
experience
specification
functional
specification
early
prototypes
experience
specification
interaction
specification
product
specification
service
specification
business
specification
REALISATION
decomposition
realization of components
integration
evaluation
experience
prototype
interaction
prototype
product
prototype
service
prototype
business
prototype
user testing
functional testing
related work
reflection
evaluation

B The Autonomous Control Level (ACL)-Matrix

The ACL Matrix is introduced and described by Bruce T. Clough [20]

Level	Level Descriptor	Observe	Orient	Decide	Act
		Perception/Situational Awareness	Analysis/Coordination	Decision Making	Capability
10	Fully Autonomous	Cognizant of all within Battlespace	Coordinates as necessary	Capable of total indepenance	Requires little guidance to do job
		Battlespace inference - Intent of self and others	Strategic group goals assigned	Distributed tactical group planning	Group accomplishment of strategic goal with
9	Battlespace	(allies and foes).		Individual determination of tactical goal	no supervisory assistance
	Swarm	Complex/Intense environment - on-board tracking	Enemy strategy inferred	Individual task planning/execution	
	Cognizance			Choose tactical targets	
8		Proximity inference - Intent of self and others	Strategic group goals assigned	Coordinated tactical group planning	Group accomplishment of strategic goal with
	Battlespace	(allies and foes)		Individual task planning/execution	minimal supervisory assistance
	Cognizance	Reduced dependance upon off-board data	Enemy tactics inferred	Choose targets of opportunity	(example: go SCUD hunting)
			ATR		
7		Short track awareness - History and predictive battlesp	Tactical group goals assigned	Individual task planning/execution to meet goals	Group accomplishment of tactical goal with
	Battlespace	data in limited range, timeframe, and numbers	Enemy trajectory estimated		minimal supervisory assistance
	Knowledge	Limited inference supplemented by off-board data			
6	Real Time		Tactical group goals assigned	Coordinated trajectory planning and execution to meet	Group accomplishment of tactical goal with
	Multi-Vehicle	Ranged awareness - on-board sensing for long range,	Enemy location sensed/estimated	goals - group optimization	minimal supervisory assistance
	Cooperation	supplemented by off-board data			
					Possible close air space separation (1-100 yds)
5	Real Time	Sensed awareness - Local sensors to detect others,	Tactical group plan assigned	On-board trajectory replanning - optimizes for	Group accomplishment of tactical plan as externally
	Multi-Vehicle	Fused with off-board data	RT Health Diagnosis; Ability to compensate for most	current and predictive conditions	assigned
	Coordination		failures and flight conditions; Ability to predict onset of	Collision avoidance	Air collision avoidance
			failures (e.g. Prognostic Health Mgmt)		Possible close air space separation (1-100 yds) for
			Group diagnosis and resource management		AAR, formation in non-threat conditions
4		Deliberate awareness - allies communicate data	Tactical plan assigned	On-board trajectory replanning - event driven	Self accomplishment of tactical plan as externally
	Fault/Event		Assigned Rules of Engagement	Self resource management	assigned
	Adaptive		RT Health Diagnosis; Ability to compensate for most	Deconfliction	
	Vehicle		failures and flight conditions - inner loop changes		
			reflected in outer loop performance		Medium vehicle airspace separation (100's of yds)
3	Robust Response	Health/status history & models	Tactical plan assigned	Evaluate status vs required mission capabilities	Self accomplishment of tactical plan as externally
	to Real Time		RT Health Diag (What is the extent of the problems?)	Abort/RTB if insufficient	assigned
	Faults/Events		Ability to compensate for most control failures and		
			flight conditions (i.e. adaptive inner-loop control)		
2		Health/status sensors	RT Health diagnosis (Do I have problems?)	Execute preprogrammed or uploaded plans	Self accomplishment of tactical plan as externally
	Changeable		Off-board replan (as required)	in response to mission and health conditions	assigned
	Mission				
1	Execute	Preloaded mission data			
	Preplanned	Flight Control and Navigation Sensing	Pre/Post Flight BIT	Preprogrammed mission and abort plans	Wide airspace separation requirements (miles)
	Mission		Report status		
0	Remotely	Flight Control (attitude, rates) sensing	Telemetered data	N/A	Control by remote pilot
	Piloted	Nose camera	Remote pilot commands		
	Vehicle				

Table 4: Final ACL Chart

C Kano Requirements Survey

SDCS Minimum Viable Product Requirements Elicitation Survey

This survey attemtps to elicitate requirements for an Minimum Viable Product for the Semi-Direct Control System. This section briefly deals with the background and motivation for this survey and the thesis that this survey will be included in.

The constraints of UCAV utilization partially reside in the UCAVs dependency on wireless communication protocols for transceiving information between the pilot and the UCAV. The efficiency and reliability of the indicated protocols are affected by bandwidth limitations and latency. In addition, due to the fact that the UCAV is operated beyond visual range (BVR), the pilots of a broad range of UCAVs are necessitated to work withthe (occasionally limited) information supply regarding the UCAVs environment, received from the sensors on board the UCAV in flight. Depending on the quality and complexity of the synthesised picture, the pilot's situational awareness may drastically decrease. Moreover, cases may arise where the UCAV-pilot is not capable of sensing aerodynamic or mechanical haptic feedback such as vibration of the plane or acceleration, further decreasing situational awareness and the ability to assess flight performance and –envelope. mThese deficiencies restrict the effectiveness of UCAVs in both long distance operation along with short distance operations and engagements with adversaries. In the latter case, swift responses are indispensable for successful interaction with the envionment and adversary in combat situations. This engenders the claim that UCAVs are insufficiently survivable and effective against MCAVsand other UCAVs in within visual range (WVR) air-to-air combat situations.

A potential solution overcoming the vulnerability of UCAVs in WVR air-to-air combat scenarios is provided in the feasibility study by Hans Heerkens of the University of Twente (UT) and Frank Tempelman of the Netherlands Aerospace Center (NLR). The solution is being referred to as a Semi-Direct Control System (SDCS).

The philosophy behind a SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with a SDCS inreal-time, but locates a position inside the airspace where the UCAV should fly to, a predetermined time and geographical distanceapart from the current location. The frequency with which the pilot presents commands regarding the position and operation of the UCAV may vary.The commands may also incorporate operational information regarding approach possibilities, target acquisition and target engagement. In practice, this implies that the SDCS should possess a Combat Manoeuvring Management System (CMMS) that is equipped with a measure of "intelligence" that is being utilised in the decisionmaking processes the UCAV with a SDCS will encounter. The CMMS determines the optimal way to fulfil the requirements of the command it has received regarding designated parameters such as position, velocity, weapons- and systems configuration. The SDCS constantly analyses positions and movementsof both the UCAV with which it is equipped as well as adversary aircraft. It attempts to predict changes by using extrapolation techniques and using these to calculate and execute the necessary direction changes and also learning from its actions and decisions. The pilot determines the tactics tobe used and can continuously modify the flight path and strategy based on the adversary's behaviour.

The SDCS is placed on the verge between direct control systems and near-complete autonomous control systems. In the first, the operator directly adjusts the aircraft's control surfaces and equipment. In the latter, the UCAV performs control and operation tasks in complete solitude in which the role and influence of the operator is reduced to a minimum. Drawbacks of the first control system in UCAVs reside in latency problemsand data link frailness. Drawbacks of the latter reside in a too little developed state of the art in UCAV autonomy. Despite efforts in research in autonomy and artificial intelligence, a completely autonomous UCAV able to perform WVR air-to-air combat manoeuvres in an unknown environment is not yet been realised. The SDCS aims at eliminating the drawbacks of these two systems: Due to on-board automation, there may be limited data exchange between pilot and UCAV without it being unable to perform combat manoeuvres. Furthermore, the SDCS overcomes the need of extensive automation capabilities. Asresearch in fields as mahcine learning and predictable controllers matures, the interval with which the operator has to send commands may decrease and the commands may become of a higher abstraction level. Concluding, an UCAV equipped with SDCS possesses the potential to increase the survivability of this UCAV in WVR air-to-air combat situations in which a quick reaction time is essential.

To date, the NLR does not possess a platform to demonstrate, visualise and develop the core functionality of a SDCS that canbe set up and operated within their resource (budget) constraints. The thesis that I am handling regards the development of thisplatform to fulfil a higher goal: To perform research in the feasibility and efficiency of the SDCS. The functionality of the SDCS is mapped, however there is a need for a clear setof requirements regarding a Minimum Viable Product (A MVP is a stripped down version of a product with just enough features to gather validated learning about the intended product and its continued development. Gathering insights from an MVP is less expensive than developing a product with more features,which increase costs and risk if the product fails, for example, due to incorrect assumptions) of the platform that demonstrates these functionalities and also offers room to develop these and other functionalities. This survey aims to elicit the requirements for the functionality of the MVP, makinga distinction between whether a certain function should be included in this version of the MVP, or that it should be added in alater stage, but that the MVP must be able to later adoptthis functionality.

In this survey, statements regarding functionalities are presented that may be relevant to include in a Minimum Viable Product. For each statement of functionality, two questions are posed. Your answers to these two questions will help me prioritising functionalities of the Minimum Viable Product. Each question can be answered with a scale ranging from 1 to 5. Please fill in your selected answer using the dropdown menus next to each question. The scaling works as follows:

1 = I definitely want this to be the case in the MVP
2 = I would like it if this is the case in the MVP, however it is not essential
3 = I am neutral on this topic, it does not matter whether this is the case in the MVP or not
4 = I would not like it if this is the case in the MVP, but I can live with it if it were
5 = I definitely do not want this to be the case in the MVP

Please keep in mind that all factors may be relevant for the SDCS project, but please rate the functionalities according to if you think they should be included in a Minimum Viable Product of the platform.If you have ideas or remarks regarding a certain statement, you can mention this in the "remarks"-section next to the statement.If you have further ideas or reccomendations regarding the requirements in general or a single requirement, you can also mention this in the remarks section. Thank you for contributing to our research.

Basic Information

Name: _____
Date: _____
Professional background: _____
Institution: _____

UCAV and Model Specification

This section handles the specification of the UCAVs and other models inside the MVP, what functionality they should have and what that should look like

1	If the MVP incorporates realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not incorporate realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): select	
2	If the MVP allows the user to set the flight performance parameters of the UCAV(s) or other aircraft (such as range, endurance, operational ceiling, maximum and minimum velocity, maximum g -force, fuel quantity) , how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not allow the user to set the flight performance parameters of the UCAV(s) or other aircraft, how would you feel about that?	Please select your answer (1-5): select	
3	If the MVP allows the user to set the equipment of the UCAV(s) or other aircraft in the MVP (such as different types of sensors, infrared, camera and radar), how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not allow the user to set the equipment of the UCAV(s) or other aircraft in the MVP, how would you feel about that?	Please select your answer (1-5): select	
4	If the MVP allows the user to set the armament (type of weapon, amount) of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not allow the user to set the armament of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): select	
5	If the MVP includes damage models (that simulate damage to UCAV(s) or other aircraft and influences their flight performance), how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not include damage models, how would you feel about that?	Please select your answer (1-5): select	

UCAV and Model Governance

This section handles the behaviour and the control options of the UCAVs and other models inside the MVP

6	If the MVP allows for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not allow for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): select	
7	If the MVP allows for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): select	Remarks:
	If the MVP does not allow for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): select	

8	If the MVP allows for an **external** (for example from a different computer or other computer program) flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow for an **external** flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) select	
9	If the MVP allows for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) select	
10	If the MVP allows the input commands for the UCAV(s) to be variable in nature or abstraction level (implementing different command possibilities, such as "go to this position" (which is based on an action) or "follow this plane", which is based on the action of another UCAV or airplane or "attack this plane" , which is a mixture of complex tactics along with intelligence of the UCAV), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow the input commands for the UCAV(s) to be variable in nature or abstraction level , how would you feel about that?	Please select your answer (1-5) select	

UCAV and Model Control Link

This section handles possibilities regarding the control link that is established between the user and the UCAV in the MVP

11	If the MVP allows to vary the frequency with which the user can provide input to the UCAV(s) (for example from 0.5 seconds to 10 seconds), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow to vary the frequency with which the user can provide input to the UCAV(s), how would you feel about that?	Please select your answer (1-5) select	
12	If the MVP allows to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) select	
13	If the MVP allows the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) select	

UCAV Information Quantity and Quality

This section handles possibilities regarding the information quantity and quality

14	If the MVP displays data about flight parameters of the UCAV(s) or other aircraft (such as fuel quantity, armament status, damage status, position, altitude and speed) on screen, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not display data about flight parameters of the UCAV(s) or other aircraft on screen, how would you feel about that?	Please select your answer (1-5) select	
15	If the MVP allows to choose and filter what information (regarding the UCAV(s) and their environment) is presented to the user (for example to emulate the possible bandwidth limitations between operator and UCAV, for example by disabling video feed, or only displaying coordinates of the friendly and adversary forces), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow to choose and filter which information (regarding the UCAV(s) and their environment) is presented to the user, how would you feel about that?	Please select your answer (1-5) select	

Virtual Engagement Environment

This section handles the virtual enviroment of the MVP in which the UCAV(s) operate and interact

16	If the MVP allows the user to set the terrain of the battlefield (for example flat land, mountains or ocean), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow the user to set the terrain of the battlefield, how would you feel about that?	Please select your answer (1-5) select	
17	If the MVP allows the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) select	
18	If the MVP allows to add more than one (semi -) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow to add more than one (semi -) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) select	

19	If the MVP allows to adjust the parameters that govern the interaction of the above mentioned adversary forces (such as accuracy and rate of fire) and that would influence the effectivity against the UCAV(s), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow to adjust the parameters that govern the interaction of the above mentioned adversary forces , how would you feel about that?	Please select your answer (1-5) select	

Visualisation

This section handles visual aspects of the Minimum Viable Product such as the overall look, feel and interfacing of the MVP

20	If the MVP has high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not have high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5) select	
21	If the MVP allows to select different points of view, like cockpit view, chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow to select different points of view, like cockpit view , chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5) select	
22	If the MVP allows visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the Hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5) select	
23	If the MVP allows control through different types of input devices (such as joystick, mouse, hand gestures, eye tracking or speech), how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP does not allow control through different types of input, how would you feel about that?	Please select your answer (1-5) select	

Compatibility

This section handles other requirements regarding hardware, software and development

24	If the MVP is able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP is not able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5) select	
25	If the MVP can operate and be developed without licensed software (through open source software) , how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP can not operate or be developed without licensed software (through open source software), how would you feel about that?	Please select your answer (1-5) select	
26	If the MVP is to a large extend editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5) select	Remarks:
	If the MVP is not editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5) select	

You have reached the end of this survey. Thank you for your time and cooperation. Please make sure you have selected valid answers on every statement (1 through 26). If you have further remarks, please formulate them here.

Remarks:

D Kano Requirements Survey - Jan Joris Roessingh

SDCS Minimum Viable Product Requirements Elicitation Survey

This survey attemtps to elicitate requirements for an Minimum Viable Product for the Semi-Direct Control System. This section briefly deals with the background and motivation for this survey and the thesis that this survey will be included in

The constraints of UCAV utilization partially reside in the UCAVs dependency on wireless communication protocols for transceiving information between the pilot and the UCAV. The efficiency and reliability of the indicated protocols are affected by bandwidth limitations and latency. In addition, due to the fact that the UCAV is operated beyond visual range (BVR), the pilots of a broad range of UCAVs are necessitated to work with the (occasionally limited) information supply regarding the UCAVs environment, received from the sensors on board the UCAV in flight. Depending on the quality and complexity of the synthesised picture, the pilot's situational awareness may drastically decrease. Moreover, cases may arise where the UCAV-pilot is not capable of sensing aerodynamic or mechanical haptic feedback such as vibration of the plane oracceleration, further decreasing situational awareness and the ability to assess flight performance and– envelope. mThese deficiencies restrict the effectiveness of UCAVs in both long distance operation along with short distance operations and engagements with adversaries. In the latter case, swift responses are indispensable for successful interaction with the environment and adversary in combat situations. This engenders the claim that UCAVs are insufficiently survivable and effective against MCAVs and other UCAVs in within visual range (WVR) air-to-air combat situations.

A potential solution overcoming the vulnerability of UCAVs in WVR air-to-air combat scenarios is provided in the feasibility study by Hans Heerkens of the University of Twente (UT) and Frank Tempelman of the Netherlands Aerospace Center (NLR). The solution is being referred to as a Semi-Direct Control System (SDCS).

The philosophy behind a SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with a SDCS inreal-time, but locates a position inside the airspace where the UCAV should fly to, a predetermined time and geographical distance apart from the current location. The frequency with which the pilot presents commands regarding the position and operation of the UCAV may vary. The commands may also incorporate operational information regarding approach possibilities, target acquisition and target engagement. In practice, this implies that the SDCS should possess a Combat Manoeuvring Management System (CMMS) that is equipped with a measure of "intelligence" that is being utilised in the decision making processes the UCAV with a SDCS will encounter. The CMMS determines the optimal way to fulfil the requirements of the command it has received regarding designated parameters such as position, velocity, weapons- and systems configuration. The SDCS constantly analyses positions and movements of both the UCAV with which it is equipped as well as adversary aircraft. It attempts to predict changes by using extrapolation techniques and using these to calculate and execute the necessary direction changes and also learning from its actions and decisions. The pilot determines the tactics to be used and can continuously modify the flight path and strategy based on the adversary's behaviour.

The SDCS is placed on the verge between direct control systems and near-complete autonomous control systems. In the first, the operator directly adjusts the aircraft's control surfaces and equipment. In the latter, the UCAV performs control and operation tasks in complete solitude in which the role and influence of the operator is reduced to a minimum. Drawbacks of the first control system in UCAVs reside in latency problems and data link frailness. Drawbacks of the latter reside in a too little developed state of the art in UCAV autonomy. Despite efforts in research in autonomy and artificial intelligence, a completely autonomous UCAV able to perform WVR air-to-air combat manoeuvres in an unknown environment is not yet been realised. The SDCS aims at eliminating the drawbacks of these two systems: Due to onboard automation, there may be limited data exchange between pilot and UCAV without it being unable to perform combat manoeuvres. Furthermore, the SDCS overcomes the need of extensive automation capabilities. As research in fields as mahcine learning and predictable controllers matures, the interval with which the operator has to send commands may decrease and the commands may become of a higher abstraction level. Concluding, an UCAV equipped with SDCS possesses the potential to increase the survivability of this UCAV in WVR air-to-air combat situations in which a quick reaction time is essential.

To date, the NLR does not possess a platform to demonstrate, visualise and develop the core functionality of a SDCS that canbe set up and operated within their resource (budget) constraints. The thesis that I am handling regards the development of this platform to fulfil a higher goal: To perform research in the feasibility and efficiency of the SDCS. Thefunctionality of the SDCS is mapped, however there is a need for a clear set of requirements regarding a Minimum Viable

In this survey, statements regarding functionalities are presented that may be relevant to include in a Minimum Viable Product. For each statement of functionality, two questions are posed. Your answers to these two questions will help me prioritising functionalities of the Minimum Viable Product. Each question can be answered with a scale ranging from 1 to 5. Please fill in your selected answer using the drop-down menus next to each question. The scaling works as follows:

1 = I definitely want this to be the case in the MVP
2 = I would like it if this is the case in the MVP, however it is not essential
3 = I am neutral on this topic, it does not matter whether this is the case in the MVP or not
4 = I would not like it if this is the case in the MVP, but I can live with it if it were
5 = I definitely do not want this to be the case in the MVP

Please keep in mind that all factors may be relevant for the SDCS project, but please rate the functionalities according to if you think they should be included in a Minimum Viable Product of the platform. If you have ideas or remarks regarding a certain statement, you can mention this in the "remarks"-section next to the statement If you have further ideas or reccomendations regarding the requirements in general or a single requirement, you can also mention this in the remarks

Basic Information

Name: Roessingh
Date: 27032017
Professional background: Scientist
Institution: NLR

UCAV and Model Specification

This section handles the specification of the UCAVs and other models inside the MVP, what functionality they should have and what that should looklike

1	If the MVP incorporates realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): 5	Remarks:	
	If the MVP does not incorporate realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): 1		
2	If the MVP allows the user to set the flight performance parameters of the UCAV(s) or other aircraft (such as range, endurance, operational ceiling, maximum and minimum velocity, maximum g-force, fuel quantity), how would you feel about that?	Please select your answer (1-5): 1	Remarks: or other aircraft? We onderzoeken toch besturing van UCAVs?	
	If the MVP does not allow the user to set the flight performance parameters of the UCAV(s) or other aircraft, how would you feel about that?	Please select your answer (1-5): 4		
3	If the MVP allows the user to set the equipment of the UCAV(s) or other aircraft in the MVP (such as different types of sensors, infrared, camera and radar), how would you feel about that?	Please select your answer (1-5): 2	Remarks: id	
	If the MVP does not allow the user to set the equipment of the UCAV(s) or other aircraft in the MVP, how would you feel about that?	Please select your answer (1-5): 4		
4	If the MVP allows the user to set the armament (type of weapon, amount) of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): 2	Remarks: id	
	If the MVP does not allow the user to set the armament of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): 3		
5	If the MVP includes damage models (that simulate damage to UCAV(s) or other aircraft and influences their flight performance), how would you feel about that?	Please select your answer (1-5): 5	Remarks: id	
	If the MVP does not include damage models, how would you feel about that?	Please select your answer (1-5): 1		

UCAV and Model Governance

This section handles the behaviour and the control options of the UCAVs and other models inside the MVP

6	If the MVP allows for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): 5	Remarks:	
	If the MVP does not allow for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): 1		
7	If the MVP allows for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): 5	Remarks: begrijp de vraag niet. We onderzoeken/demonstreren toch geen adversary UCAV?	
	If the MVP does not allow for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): 1		
	If the MVP allows for an **external** (for example from a different computer or other computer program) flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft,	Please select your answer (1-5): 3	Remarks:	

	how would you feel about that?		
8	If the MVP does not allow for an **external** flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 3	
9	If the MVP allows for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 5	Remarks:
	If the MVP does not allow for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 1	
10	If the MVP allows the input commands for the UCAV(s) to be variable in nature or abstraction level (implementing different command possibilities, such as "go to this position" (which is based on an action) or "follow this plane", which is based on the action of another UCAV or airplane or "attack this plane" , which is a mixture of complex tactics along with intelligence of the UCAV), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow the input commands for the UCAV(s) to be variable in nature or abstraction level , how would you feel about that?	Please select your answer (1-5) 4	

UCAV and Model Control Link

This section handles possibilities regarding the control link that is established between the user and the UCAV in the MVP

11	If the MVP allows to vary the frequency with which the user can provide input to the UCAV(s) (for example from 0.5 seconds to 10 seconds), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow to vary the frequency with which the user can provide input to the UCAV(s), how would you feel about that?	Please select your answer (1-5) 4	
12	If the MVP allows to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) 1	Remarks:
	If the MVP does not allow to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) 5	
13	If the MVP allows the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) 4	

UCAV Information Quantity and Quality

This section handles possibilities regarding the information quantity and quality

14	If the MVP displays data about flight parameters of the UCAV(s) or other aircraft (such as fuel quantity, armament status, damage status, position, altitude and speed) on screen, how would you feel about that?	Please select your answer (1-5) 1	Remarks: at least basic flight parameters
	If the MVP does not display data about flight parameters of the UCAV(s) or other aircraft on screen, how would you feel about that?	Please select your answer (1-5) 5	
15	If the MVP allows to choose and filter what information (regarding the UCAV(s) and their environment) is presented to the user (for example to emulate the possible bandwidth limitations between operator and UCAV, for example by disabling video feed, or only displaying coordinates of the friendly and adversary forces), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow to choose and filter which information (regarding the UCAV(s) and their environment) is presented to the user, how would you feel about that?	Please select your answer (1-5) 4	

Virtual Engagement Environment

This section handles the virtual enviroment of the MVP in which the UCAV(s) operate and interact

16	If the MVP allows the user to set the terrain of the battlefield (for example flat land, mountains or ocean), how would you feel about that?	Please select your answer (1-5) 3	Remarks:
	If the MVP does not allow the user to set the terrain of the battlefield, how would you feel about that?	Please select your answer (1-5) 3	
17	If the MVP allows the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) 3	Remarks:
	If the MVP does not allow the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) 3	
18	If the MVP allows to add more than one (semi -) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) 3	Remarks:
	If the MVP does not allow to add more than one (semi -) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) 3	
19	If the MVP allows to adjust the parameters that govern the interaction of the above mentioned adversary forces (such as accuracy and rate of fire) and that would influence the effectivity against the UCAV(s), how would you feel about that?	Please select your answer (1-5) 3	Remarks:
	If the MVP does not allow to adjust the parameters that govern the interaction of the above mentioned adversary forces , how would you feel about that?	Please select your answer (1-5) 3	

Visualisation

This section handles visual aspects of the Minimum Viable Product such as the overall look, feel and interfacing of the MVP

20	If the MVP has high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5): 3	Remarks:
	If the MVP does not have high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5): 3	
21	If the MVP allows to select different points of view, like cockpit view, chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP does not allow to select different points of view, like cockpit view , chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5): 4	
22	If the MVP allows visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5): 3	Remarks:
	If the MVP does not allow visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the Hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5): 3	
23	If the MVP allows control through different types of input devices (such as joystick, mouse, hand gestures, eye tracking or speech), how would you feel about that?	Please select your answer (1-5): 3	Remarks:
	If the MVP does not allow control through different types of input, how would you feel about that?	Please select your answer (1-5): 3	

Compatibility

This section handles other requirements regarding hardware, software and development

24	If the MVP is able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP is not able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5): 4	
25	If the MVP can operate and be developed without licensed software (through open source software) , how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP can not operate or be developed without licensed software (through open source software) , how would you feel about that?	Please select your answer (1-5): 4	
26	If the MVP is to a large extend editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5): 1	Remarks:
	If the MVP is not editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5): 5	

You have reached the end of this survey. Thank you for your time and cooperation. Please make sure you have selected valid answers on every statement (1 through 26). If you have further remarks, please formulate them here.

Remarks:

E Kano Requirements Survey - Gerald Poppinga

SDCS Minimum Viable Product Requirements Elicitation Survey

This survey attemtps to elicitate requirements for an Minimum Viable Product for the Semi-Direct Control System. This section briefly deals with the background and motivation for this survey and the thesis that this survey will be included in

The constraints of UCAV utilization partially reside in the UCAVs dependency on wireless communication protocols for transceiving information between the pilot and the UCAV. The efficiency and reliability of the indicated protocols are affected by bandwidth limitations and latency. In addition, due to the fact that the UCAV is operated beyond visual range (BVR), the pilots of a broad range of UCAVs are necessitated to work with the (occasionally limited) information supply regarding the UCAVs environment, received from the sensors on board the UCAV in flight. Depending on the quality and complexity of the synthesised picture, the pilot's situational awareness may drastically decrease. Moreover, cases may arise where the UCAVpilot is not capable of sensing aerodynamic or mechanical haptic feedback such as vibration of the plane or acceleration, further decreasing situational awareness and the ability to assess flight performance and –envelope. mThese deficiencies restrict the effectiveness of UCAVs in both long distance operation along with short distance operations and engagements with adversaries. In the latter case, swift responses are indispensable for successful interaction with the environment and adversary in combat situations. This engenders the claim that UCAVs are insufficiently survivable and effective against MCAVs and other UCAVs in within visual range (WVR) air-to-air combat situations.

A potential solution overcoming the vulnerability of UCAVs in WVR air-to-air combat scenarios is provided in the feasibility study by Hans Heerkens of the University of Twente (UT) and Frank Tempelman of the Netherlands Aerospace Center (NLR). The solution is being referred to as a Semi-Direct Control System (SDCS).

The philosophy behind a SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with a SDCS inreal-time, but locates a position inside the airspace where the UCAV should fly to, a predetermined time and geographical distanceapart from the current location. The frequency with which the pilot presents commands regarding the position and operation of the UCAV may vary. The commands may also incorporate operational information regarding approach possibilities, target acquisition and target engagement. In practice, this implies that the SDCS should possess a Combat Manoeuvring Management System (CMMS) that is equipped with a measure of "intelligence" that is being utilised in the decision making processes the UCAV with a SDCS will encounter. The CMMS determines the optimal way to fulfil the requirements of the command it hasreceived regarding designated parameters such as position, velocity, weapons- and systems configuration. The SDCS constantly analyses positions and movements of both the UCAV with which it is equipped as well as adversary aircraft. It attempts to predict changes by using extrapolation techniques and using these to calculate and execute the necessary direction changes and also learning from its actions and decisions. The pilot determines the tactics to be used and can continuously modify the flight path and strategy based on the adversary's behaviour.

The SDCS is placed on the verge between direct control systems and near-complete autonomous control systems. In the first, the operator directly adjusts the aircraft's control surfaces and equipment. In the latter, the UCAV performs control and operation tasks in complete solitude in which the role and influence of the operator is reduced to a minimum. Drawbacks of the first control system in UCAVs reside in latency problems and data linkfrailness. Drawbacks of the latter reside in a too little developed state of the art in UCAV autonomy. Despite efforts in research in autonomy and artificial intelligence, a completely autonomous UCAV able to perform WVR air-to-air combat manoeuvres in an unknown environment is not yet been realised. The SDCS aims at eliminating the drawbacks of these two systems: Due to on-board automation, there may be limited data exchange between pilot and UCAV without it being unable to perform combat manoeuvres. Furthermore, the SDCS overcomes the need of extensive automation capabilities. As research in fields as mahcine learning and predictable controllers matures, the interval with which the operator has to send commands may decrease and the commands may become of a higher abstraction level. Concluding, an UCAV equipped with SDCS possesses the potential to increase the survivability of this UCAVin WVR air-to-air combat situations in which a quick reaction time is essential.

To date, the NLR does not possess a platform to demonstrate, visualise and develop the core functionality of a SDCS that canbe set up and operated within their resource (budget) constraints. The thesis that I am handling regards the development of thisplatform to fulfil a higher goal: To perform research in the feasibility and efficiency of the SDCS. The functionality of the SDCS is mapped, however there is a need for a clear set of requirements regarding a Minimum Viable Product (A MVP is a stripped down version of a product with just enough features to gather validated learning about the intended product and its continued development. Gathering insights from an MVP is less expensive than developing a product with more features, which increase costs and risk if the product fails, for example, due to incorrect assumptions) of the platform that demonstrates thesefunctionalities and also offers room to develop these and other functionalities. This survey aims to elicit the requirements for the functionality of the MVP, making a distinction between whether a certain function should be included in this version of the MVP, or that it should be added in a later stage, butthat the MVP must be able to later adopt this functionality.

In this survey, statements regarding functionalities are presented that may be relevant to include in a Minimum Viable Product. For each statement of functionality, two questions are posed. Your answers to these two questions will help me prioritising functionalities of the Minimum Viable Product. Each question can be answered with a scale ranging from 1 to 5. Please fill in your selected answer using the dropdown menus next to each question. The scaling works as follows:

1 = I definitely want this to be the case in the MVP
2 = I would like it if this is the case in the MVP, however it is not essential
3 = I am neutral on this topic, it does not matter whether this is the case in the MVP or not
4 = I would not like it if this is the case in the MVP, but I can live with it if it were
5 = I definitely do not want this to be the case in the MVP

Please keep in mind that all factors may be relevant for the SDCS project, but please rate the functionalities according to if you think they should be included in a Minimum Viable Product of the platform.If you have ideas or remarks regarding a certain statement, you can mention this in the "remarks"-section next to the statement If you have further ideas or reccomendations regarding the requirements in general or a single requirement, you can also mention this in the remarks section. Thank you for contributing to our research.

Basic Information

Name: **Gerald Popping**
Date: **29-03-2017**
Professional background: **Sr. R&D manager**
Institution: **Netherlands Aerospace Center NLR**

UCAV and Model Specification

This section handles the specification of the UCAVs and other models inside the MVP, what functionality they should have and what that should look like

1	If the MVP incorporates realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): 1	Remarks:	
	If the MVP does not incorporate realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): 5		
2	If the MVP allows the user to set the flight performance parameters of the UCAV(s) or other aircraft (such as range, endurance, operational ceiling, maximum and minimum velocity, maximum g -force, fuel quantity) , how would you feel about that?	Please select your answer (1-5): 2	Remarks:	
	If the MVP does not allow the user to set the flight performance parameters of the UCAV(s) or other aircraft, how would you feel about that?	Please select your answer (1-5): 4		
3	If the MVP allows the user to set the equipment of the UCAV(s) or other aircraft in the MVP (such as different types of sensors, infrared, camera and radar), how would you feel about that?	Please select your answer (1-5): 2	Remarks:	
	If the MVP does not allow the user to set the equipment of the UCAV(s) or other aircraft in the MVP, how would you feel about that?	Please select your answer (1-5): 4		
4	If the MVP allows the user to set the armament (type of weapon, amount) of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): 1	Remarks:	
	If the MVP does not allow the user to set the armament of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): 5		
5	If the MVP includes damage models (that simulate damage to UCAV(s) or other aircraft and influences their flight performance), how would you feel about that?	Please select your answer (1-5): 2	Remarks:	
	If the MVP does not include damage models, how would you feel about that?	Please select your answer (1-5): 2		

UCAV and Model Governance

This section handles the behaviour and the control options of the UCAVs and other models inside the MVP

6	If the MVP allows for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): 2	Remarks:	
	If the MVP does not allow for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): 4		
7	If the MVP allows for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): 2	Remarks:	
	If the MVP does not allow for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): 4		
	If the MVP allows for an **external** (for example from a different computer or other computer program) flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would	Please select your answer (1-5): 2	Remarks:	

	you feel about that?		
8	If the MVP does not allow for an **external** flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 3	
9	If the MVP allows for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 4	
10	If the MVP allows the input commands for the UCAV(s) to be variable in nature or abstraction level (implementing different command possibilities, such as "go to this position" (which is based on an action) or "follow this plane", which is based on the action of another UCAV or airplane or "attack this plane" , which is a mixture of complex tactics along with intelligence of the UCAV), how would you feel about that?	Please select your answer (1-5) 1	Remarks:
	If the MVP does not allow the input commands for the UCAV(s) to be variable in nature or abstraction level , how would you feel about that?	Please select your answer (1-5) 5	

UCAV and Model Control Link

This section handles possibilities regarding the control link that is established between the user and the UCAV in the MVP

11	If the MVP allows to vary the frequency with which the user can provide input to the UCAV(s) (for example from 0.5 seconds to 10 seconds), how would you feel about that?	Please select your answer (1-5) 1	Remarks:
	If the MVP does not allow to vary the frequency with which the user can provide input to the UCAV(s), how would you feel about that?	Please select your answer (1-5) 5	
12	If the MVP allows to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) 1	Remarks:
	If the MVP does not allow to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) 5	
13	If the MVP allows the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) 4	

UCAV Information Quantity and Quality

This section handles possibilities regarding the information quantity and quality

14	If the MVP displays data about flight parameters of the UCAV(s) or other aircraft (such as fuel quantity, armament status, damage status, position, altitude and speed) on screen, how would you feel about that?	Please select your answer (1-5) 1	Remarks:
	If the MVP does not display data about flight parameters of the UCAV(s) or other aircraft on screen, how would you feel about that?	Please select your answer (1-5) 5	
15	If the MVP allows to choose and filter what information (regarding the UCAV(s) and their environment) is presented to the user (for example to emulate the possible bandwidth limitations between operator and UCAV, for example by disabling video feed, or only displaying coordinates of the friendly and adversary forces), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow to choose and filter which information (regarding the UCAV(s) and their environment) is presented to the user, how would you feel about that?	Please select your answer (1-5) 4	

Virtual Engagement Environment

This section handles the virtual enviroment of the MVP in which the UCAV(s) operate and interact

16	If the MVP allows the user to set the terrain of the battlefield (for example flat land, mountains or ocean), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow the user to set the terrain of the battlefield, how would you feel about that?	Please select your answer (1-5) 3	
17	If the MVP allows the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) 3	
18	If the MVP allows to add more than one (semi -) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow to add more than one (semi -) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) 3	
19	If the MVP allows to adjust the parameters that govern the interaction of the above mentioned adversary forces (such as accuracy and rate of fire) and that would influence the effectivity against the UCAV(s), how would you feel about that?	Please select your answer (1-5) 2	Remarks:
	If the MVP does not allow to adjust the parameters that govern the interaction of the above mentioned adversary forces , how would you feel about that?	Please select your answer (1-5) 3	

Visualisation

This section handles visual aspects of the Minimum Viable Product such as the overall look, feel and interfacing of the MVP

20	If the MVP has high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5): 1	Remarks:
	If the MVP does not have high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5): 5	
21	If the MVP allows to select different points of view, like cockpit view, chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP does not allow to select different points of view, like cockpit view , chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5): 4	
22	If the MVP allows visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP does not allow visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the Hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5): 4	
23	If the MVP allows control through different types of input devices (such as joystick, mouse, hand gestures, eye tracking or speech), how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP does not allow control through different types of input, how would you feel about that?	Please select your answer (1-5): 4	

Compatibility

This section handles other requirements regarding hardware, software and development

24	If the MVP is able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5): 2	Remarks:
	If the MVP is not able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5): 4	
25	If the MVP can operate and be developed without licensed software (through open source software) , how would you feel about that?	Please select your answer (1-5): 1	Remarks:
	If the MVP can not operate or be developed without licensed software (through open source software) , how would you feel about that?	Please select your answer (1-5): 5	
26	If the MVP is to a large extend editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5): 1	Remarks:
	If the MVP is not editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5): 5	

You have reached the end of this survey. Thank you for your time and cooperation. Please make sure you have selected valid answers on every statement (1 through 26). If you have further remarks, please formulate them here.

Remarks:

F Kano Requirements Survey - Hans Heerken

SDCS Minimum Viable Product Requirements Elicitation Survey

This survey attempts to elicitate requirements for an Minimum Viable Product for the Semi-Direct Control System. This section briefly deals with the background and motivation for this survey and the thesis that this survey will be included in.

The constraints of UCAV utilization partially reside in the UCAVs dependency on wireless communication protocols for transceiving information between the pilot and the UCAV. The efficiency and reliability of the indicated protocols are affected by bandwidth limitations and latency. In addition, due to the fact that the UCAV is operated beyond visual range (BVR), the pilots of a broad range of UCAVs are necessitated to work with the (occasionally limited) information supply regarding the UCAVs environment, received from the sensors on board the UCAV in flight. Depending on the quality and complexity of the synthesised picture, the pilot's situational awareness may drastically decrease. Moreover, cases may arise where the UCAV-pilot is not capable of sensing aerodynamic or mechanical haptic feedback such as vibration of the plane or acceleration, further decreasing situational awareness and the ability to assess flight performance and –envelope. mThese deficiencies restrict the effectiveness of UCAVs in both long distance operation along with short distance operations and engagements with adversaries. In the latter case, swift responses are indispensable for successful interaction with the environment and adversary in combat situations. This engenders the claim that UCAVs are insufficiently survivable and effective against MCAVs and other UCAVs in within visual range (WVR) air-to-air combat situations.

A potential solution overcoming the vulnerability of UCAVs in WVR air-to-air combat scenarios is provided in the feasibility study by Hans Heerkens of the University of Twente (UT) and Frank Tempelman of the Netherlands Aerospace Center (NLR). The solution is being referred to as a Semi-Direct Control System (SDCS).

The philosophy behind a SDCS entails a scenario in which the UCAV operator does not control the UCAV equipped with a SDCS in real-time, but locates a position inside the airspace where the UCAV should fly to, a predetermined time and geographical distance apart from the current location. The frequency with which the pilot presents commands regarding the position and operation of the UCAV may vary. The commands may also incorporate operational information regarding approach possibilities, target acquisition and target engagement. In practice, this implies that the SDCS should possess a Combat Manoeuvring Management System (CMMS) that is equipped with a measure of "intelligence" that is being utilised in the decision making processes the UCAV with a SDCS will encounter. The CMMS determines the optimal way to fulfil the requirements of the command it has received regarding designated parameters such as position, velocity, weapons- and systems configuration. The SDCS constantly analyses positions and movements of both the UCAV with which it is equipped as well as adversary aircraft. It attempts to predict changes by using extrapolation techniques and using these to calculate and execute the necessary direction changes and also learning from its actions and decisions. The pilot determines the tactics to be used and can continuously modify the flight path and strategy based on the adversary's behaviour.

The SDCS is placed on the verge between direct control systems and near-complete autonomous control systems. In the first, the operator directly adjusts the aircraft's control surfaces and equipment. In the latter, the UCAV performs control and operation tasks in complete solitude in which the role and influence of the operator is reduced to a minimum. Drawbacks of the first control system in UCAVs reside in latency problems and data link frailness. Drawbacks of the latter reside in a too little developed state of the art in UCAV autonomy. Despite efforts in research in autonomy and artificial intelligence, a completely autonomous UCAV able to perform WVR air-to-air combat manoeuvres in an unknown environment is not yet been realised. The SDCS aims at eliminating the drawbacks of these two systems: Due to on-board automation, there may be limited data exchange between pilot and UCAV without it being unable to perform combat manoeuvres. Furthermore, the SDCS overcomes the need of extensive automation capabilities. As research in fields as mahcine learning and predictable controllers matures, the interval with which the operator has to send commands may decrease and the commands may become of a higher abstraction level. Concluding, an UCAV equipped with SDCS possesses the potential to increase the survivability of this UCAV in WVR air-to-air combat situations in which a quick reaction time is essential.

To date, the NLR does not possess a platform to demonstrate, visualise and develop the core functionality of a SDCS that can be set up and operated within their resource (budget) constraints. The thesis that I am handling regards the development of this platform to fulfil a higher goal: To perform research in the feasibility and efficiency of the SDCS. The functionality of the SDCS is mapped, however there is a need for a clear set of requirements regarding a Minimum Viable Product (A MVP is a stripped down version of a product with just enough features to gather validated learning about the intended product and its continued development. Gathering insights from an MVP is less expensive than developing a product with more features, which increase costs and risk if the product fails, for example, due to incorrect assumptions) of the platform that demonstrates these functionalities and also offers room to develop these and other functionalities. This survey aims to elicit the requirements for the functionality of the MVP, making a distinction between whether a certain function should be included in this version of the MVP, or that it should be added in a later stage, but that the MVP must be able to later adopt this functionality.

In this survey, statements regarding functionalities are presented that may be relevant to include in a Minimum Viable Product. For each statement of functionality, two questions are posed. Your answers to these two questions will help me prioritising functionalities of the Minimum Viable Product. Each question can be answered with a scale ranging from 1 to 5. Please fill in your selected answer using the drop-down menus next to each question. The scaling works as follows:

1 = I definitely want this to be the case in the MVP
2 = I would like it if this is the case in the MVP, however it is not essential
3 = I am neutral on this topic, it does not matter whether this is the case in the MVP or not
4 = I would not like it if this is the case in the MVP, but I can live with it if it were
5 = I definitely do not want this to be the case in the MVP

Please keep in mind that all factors may be relevant for the SDCS project, but please rate the functionalities according to **I** you think they should be included in a Minimum Viable Product of the platform. If you have ideas or remarks regarding a certain statement, you can mention this in the "remarks"-section next

Basic Information

Name: **Hans Heerkens**
Date: **########**
Professional background: **Assistent professor**
Institution: **UT**

UCAV and Model Specification

This section handles the specification of the UCAVs and other models inside the MVP, what functionality they should have and what that should looklike

1	If the MVP incorporates realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): 1	Rem For assessing the utility of tyhe SDCS we have to show realistic combat behavior.
	If the MVP does not incorporate realistic flight dynamics such as aerodynamics and (collision) physics, how would you feel about that?	Please select your answer (1-5): 4	
2	If the MVP allows the user to set the flight performance parameters of the UCAV(s) or other aircraft (such as range, endurance, operational ceiling, maximum and minimum velocity, maximum g -force, fuel quantity) , how would you feel about that?	Please select your answer (1-5): 3	Remarks: If a few representative aircraft would be included in the demonstrator that would be sufficient
	If the MVP does not allow the user to set the flight performance parameters of the UCAV(s) or other aircraft, how would you feel about that?	Please select your answer (1-5): 3	
3	If the MVP allows the user to set the equipment of the UCAV(s) or other aircraft in the MVP (such as different types of sensors, infrared, camera and radar), how would you feel about that?	Please select your answer (1-5): 3	Remarks: Idem
	If the MVP does not allow the user to set the equipment of the UCAV(s) or other aircraft in the MVP, how would you feel about that?	Please select your answer (1-5): 3	
4	If the MVP allows the user to set the armament (type of weapon, amount) of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): 3	Remarks: Idem
	If the MVP does not allow the user to set the armament of the UCAVs or other aircraft, how would you feel about that?	Please select your answer (1-5): 3	
5	If the MVP includes damage models (that simulate damage to UCAV(s) or other aircraft and influences their flight performance), how would you feel about that?	Please select your answer (1-5): 3	Remarks: Basic damage model would be sufficient; as ling as the hit rate is realistic, the damage is not so important.
	If the MVP does not include damage models, how would you feel about that?	Please select your answer (1-5): 3	

UCAV and Model Governance

This section handles the behaviour and the control options of the UCAVs and other models inside the MVP

6	If the MVP allows for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): 1	Remarks:Testing the coordination advantages of 2 UCAv is very important, to see whether this noffsets possible limitations of UCA compared to manned aircraft.
	If the MVP does not allow for the control and governance of multiple **friendly** UCAVs in parallel, how would you feel about that?	Please select your answer (1-5): 4	
7	If the MVP allows for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): 4	Remarks: Not important, as long as the intelligence level of the enemy UCAvS IS REALISTIC
	If the MVP does not allow for the control and governance of multiple **adversary** UCAVs or other aircraft in parallel, how would you feel about that?	Please select your answer (1-5): 3	
	If the MVP allows for an **external** (for example from a different computer or other computer program) flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5): 4	Remarks: Not so important for what we want to demonstrate

8	If the MVP does not allow for an **external** flight controller to govern the behaviour of **friendly** UCAV(s) depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 2	
9	If the MVP allows for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 4	Remarks: Nice to have, but not important at this stage
	If the MVP does not allow for an **external** flight controller to govern the behaviour of **adversary** UCAV(s) or other aircraft, depending on the environment and other UCAV(s) or aircraft, how would you feel about that?	Please select your answer (1-5) 2	
10	If the MVP allows the input commands for the UCAV(s) to be variable in nature or abstraction level (implementing different command possibilities, such as "go to this position" (which is based on an action) or "follow this plane", which is based on the action of another UCAV or airplane or "attack this plane" , which is a mixture of complex tactics along with intelligence of the UCAV), how would you feel about that?	Please select your answer (1-5) 1	Remarks: Vital for showing the feasibility of SDCS. Not all variations need to be present, but the potential needs to be there.
	If the MVP does not allow the input commands for the UCAV(s) to be variable in nature or abstraction level , how would you feel about that?	Please select your answer (1-5) 5	

UCAV and Model Control Link

This section handles possibilities regarding the control link that is established between the user and the UCAV in the MVP

11	If the MVP allows to vary the frequency with which the user can provide input to the UCAV(s) (for example from 0.5 seconds to 10 seconds), how would you feel about that?	Please select your answer (1-5) 1	Remarks:Idem
	If the MVP does not allow to vary the frequency with which the user can provide input to the UCAV(s), how would you feel about that?	Please select your answer (1-5) 5	
12	If the MVP allows to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) 1	Remarks:Idem
	If the MVP does not allow to vary the amount of latency in the data link between the user and the UCAV(s), how would you feel about that?	Please select your answer (1-5) 5	
13	If the MVP allows the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) 3	Remarks: Not needed for the demonstrator, it is a practical issue that can be solved later, and with other demonstrators.
	If the MVP does not allow the user to simulate disruptions or errors on the datalink between the UCAV(s) or other aircraft and the user, how wouldyou feel about that?	Please select your answer (1-5) 3	

UCAV Information Quantity and Quality

This section handles possibilities regarding the information quantity and quality

14	If the MVP displays data about flight parameters of the UCAV(s) or other aircraft (such as fuel quantity, armament status, damage status, position, altitude and speed) on screen, how would you feel about that?	Please select your answer (1-5) 1	Remarks: Essential attribute of SDCS
	If the MVP does not display data about flight parameters of the UCAV(s) or other aircraft on screen, how would you feel about that?	Please select your answer (1-5) 5	
15	If the MVP allows to choose and filter what information (regarding the UCAV(s) and their environment) is presented to the user (for example to emulate the possible bandwidth limitations between operator and UCAV, for example by disabling video feed, or only displaying coordinates of the friendly and adversary forces), how would you feel about that?	Please select your answer (1-5) 1	Remarks: Idem
	If the MVP does not allow to choose and filter which information (regarding the UCAV(s) and their environment) is presented to the user, how would you feel about that?	Please select your answer (1-5) 5	

Virtual Engagement Environment

This section handles the virtual enviroment of the MVP in which the UCAV(s) operate and interact

16	If the MVP allows the user to set the terrain of the battlefield (for example flat land, mountains or ocean), how would you feel about that?	Please select your answer (1-5) 5	Remarks: Doe snot make any difference for assessing SDCS
	If the MVP does not allow the user to set the terrain of the battlefield, how would you feel about that?	Please select your answer (1-5) 1	
17	If the MVP allows the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) 3	Remarks: May be of egronomic value
	If the MVP does not allow the user to set the weather type and time of day in the battlefield, how would you feel about that?	Please select your answer (1-5) 2	
18	If the MVP allows to add more than one (semi-) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) 4	Remarks: Not relevant for air2air
	If the MVP does not allow to add more than one (semi-) stationary adversaries (like road vehicles or boats, bunkers or radarposts), how would you feel about that?	Please select your answer (1-5) 2	
19	If the MVP allows to adjust the parameters that govern the interaction of the above mentioned adversary forces (such as accuracy and rate of fire) and that would influence the effectivity against the UCAV(s), how would you feel	Please select your answer (1-5) 3	Remarks: Fixed values are sufficient for assessing SDCS
	If the MVP does not allow to adjust the parameters that govern the interaction of the above mentioned adversary forces , how would you feel about that?	Please select your answer (1-5) 2	

Visualisation

This section handles visual aspects of the Minimum Viable Product such as the overall look, feel and interfacing of the MVP

20	If the MVP has high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5): 3	Remarks: Adds immersion, but representing interface accurately is sufficient
	If the MVP does not have high quality graphics, a detailed environment and detailed models, how would you feel about that?	Please select your answer (1-5): 2	
21	If the MVP allows to select different points of view, like cockpit view, chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5): 1	Remarks: Vital for SA and controlling multiple UCAVs
	If the MVP does not allow to select different points of view, like cockpit view , chase camera or 2D top view, how would you feel about that?	Please select your answer (1-5): 5	
22	If the MVP allows visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5): 1	Remarks: Vital for assessing potential of SDCS
	If the MVP does not allow visualization through, or incorporation of techniques such as Augmented Reality or Virtual Reality (by using for example the Hololens or virtual reality glasses), how would you feel about that?	Please select your answer (1-5): 5	
23	If the MVP allows control through different types of input devices (such as joystick, mouse, hand gestures, eye tracking or speech), how would you feel about that?	Please select your answer (1-5): 1	Remarks: See above
	If the MVP does not allow control through different types of input, how would you feel about that?	Please select your answer (1-5): 5	

Compatibility

This section handles other requirements regarding hardware, software and development

24	If the MVP is able to run on different computers without demanding high system requirements , how would you feel about that?	Please select your answer (1-5): 4	Remarks: As long as it can run on Windows-based computer it is OK
	If the MVP is not able to run on different computers without demanding high system requirements, how would you feel about that?	Please select your answer (1-5): 1	
25	If the MVP can operate and be developed without licensed software (through open source software) , how would you feel about that?	Please select your answer (1-5): 3	Remarks: As long as licence is not very expensive it is no problem
	If the MVP can not operate or be developed without licensed software (through open source software), how would you feel about that?	Please select your answer (1-5): 2	
26	If the MVP is to a large extend editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5): 2	Remarks: We do not know how SDCS will evolve, so development flexibility ios paramount
	If the MVP is not editable and expandable by other developers, how would you feel about that?	Please select your answer (1-5): 3	

You have reached the end of this survey. Thank you for your time and cooperation. Please make sure you have selected valid answers on every statement (1 through 26). If you have further remarks, please formulate them here.

Remarks:

G Demonstrator Mockup

Demonstrator

Potential investor or potential associate: someone who might be interested in the project but needs to be convinced of the project in a fashion similar to an elevator pitch. The person may not have a lot of time to look at the demo and it must therefore be succesful in displaying the concept and also make an impression that lasts.

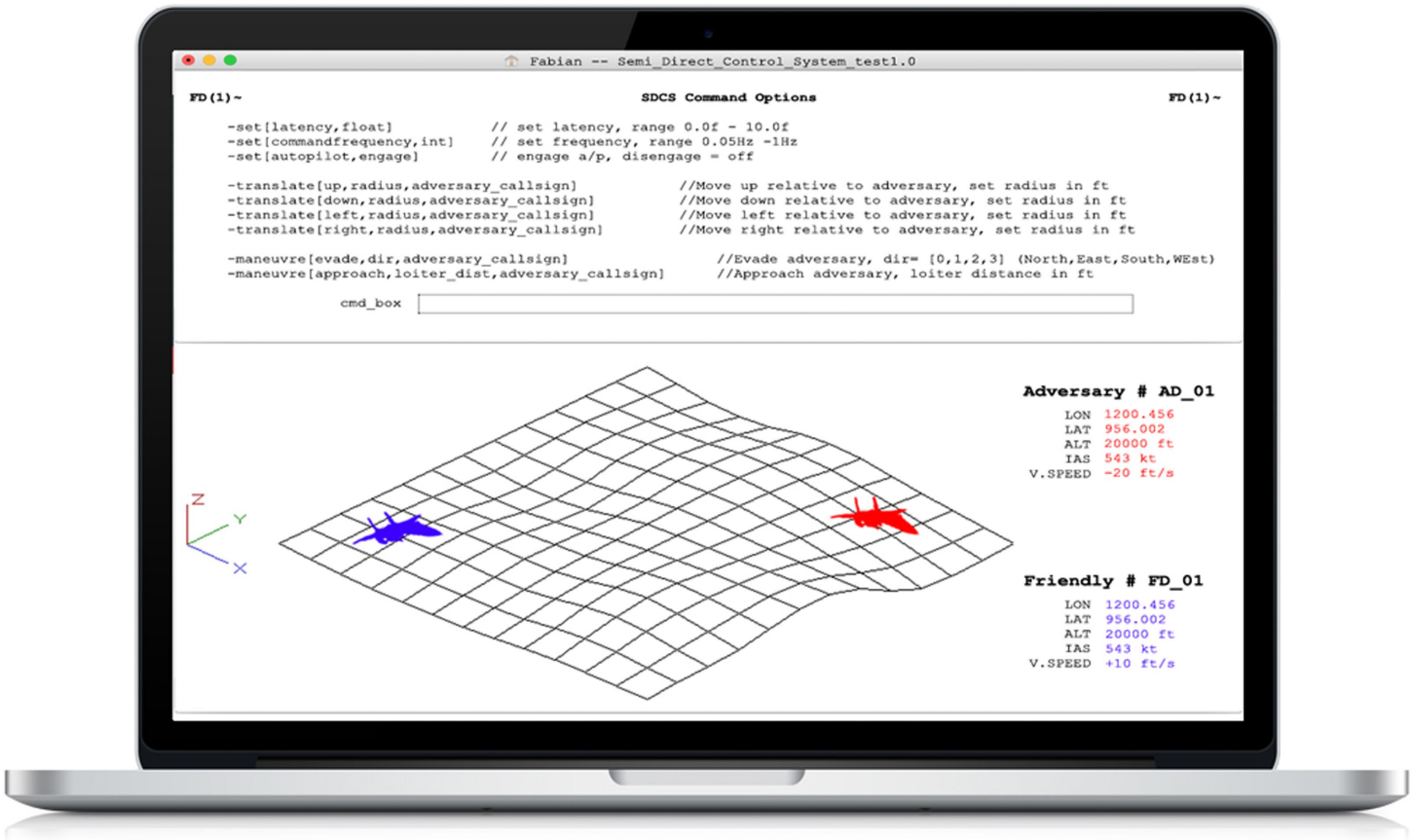
High quality graphics. Clear and attractive interface. Very easy to grasp, not per se accurate flight dynamics, suitable for Virtual Reality and other control methods to evoke a more engaged user experience.

H Test Platform Mockup

Test Platform

Researcher who is solely interested in the mechanics and working principles of the system. He knows the ins and outs of the system and does not necessarily care for an attractive look and feel. For him it is more important that the models and the mechanics are as close to reality as possible to acquire accurate test results.

Less emphasis on design and interface. Larger amount of numbers and data presented on sreen, more difficult in use except for those who thoroughly know the program. Very accurate flight dynamics, possibly less attractive for demonstrational purposes.

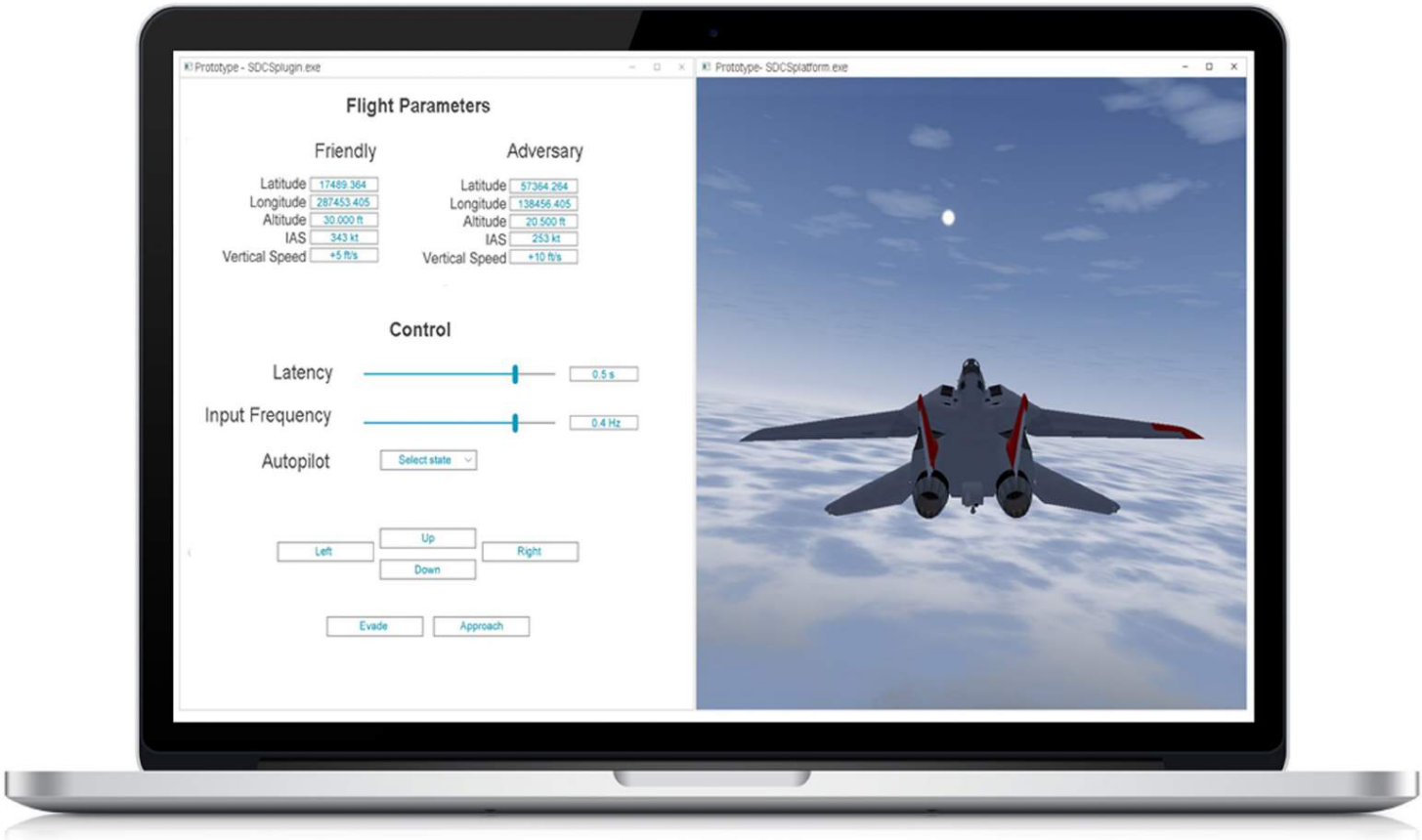


I Blended Form Mockup

Demo-Platform

RPAS operator or even F-16 fighter from the Dutch Airforce. The pilot has knowledge of flight principles and dynamics. He is interested in seeing similar behaviour as he would encounter during the operation of his own aircraft. However the system must be relatively easy to understand. He might need a short briefing of around half an hour to be able to effectively work with the system. The increased graphics enable him to see clearly what effects his actions have on screen.

Balance between graphics and realistic flight dynamics. Interface is not per se focussed on attractiveness, but remains simple to enable quick grasp of concept and control.



J Latency Batch Scirpt

```
#!/bin/bash
modprobe sch_prio
modprobe sch_netem

# run as root, "sudo bash" and then filename of this script "bash latencychange.sh"

tc -s qdisc
tc qdisc del dev enp0s25 root

# enp0s25 is the denominator for the network card. This differs per computer, to find out yours, type "sudo ifconfig" to find out the name the computer gave your network card terminal responsible for the ethernet connection and replace enp0s25 with that.

tc qdisc add dev enp0s25 root handle 1: prio priomap 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
echo "Please enter the desired latency as 'xxxms'"
read latencyms
tc qdisc add dev enp0s25 parent  1:2 handle 20: netem delay \$latencyms
tc filter add dev enp0s25 parent 1:0 protocol ip u32 match ip dport 5500 0xffff flowid 1:2
tc -s qdisc
```

K MVP Evaluation User Test

User test – MVP of the SDCS

The following text describes the flow of the user test and acts as a storyboard of the user test to be conducted.

Thank you for participating in my user-test. For now I will not yet explain what subject exactly I am doing research in, but instead start out with a task. Please imagine that you are the operator of an unmanned aircraft flying in an air to air combat situation.

Test 1: Flying with the joystick and latency

For the first part of the test, I would like you to fly an F-15C with direct joystick control. At first there will be no latency so you can get a feel for how the aircraft behaves and responds to your joystick input. In the next flights, I will induce latency on the joystick and the aircraft, meaning it will take a while before you see the joystick input work on the aircraft that you are controlling. You will fly with a latency of 100, 250 and 1000 milliseconds Round Trip Time (RTT) and perform some interactive manoeuvres with me while I fly as adversary pilot. Your task is to follow me and react to the actions my aircraft undertakes to maintain following me.

Questions:

What are your experiences during the three short flights?

After answering this question, the user is presented with the following text:

The constraints of Unmanned Combat Aerieal Vehicle (UCAVs) utilization partially reside in the UCAVs dependency on wireless communication protocols for transceiving information between the pilot and the UCAV. The efficiency and reliability of the indicated protocols are affected by bandwidth limitations and foremost by latency (as you have experienced during your previous flights). In addition, operators of a broad range of UCAVs are necessitated to work with the (occasionally limited) information supply regarding the UCAVs environment, received from the sensors on board the UCAV in flight. Depending on the quality and complexity of the synthesised picture, the pilot's situational awareness may drastically decrease. Moreover, cases may arise where the UCAV-pilot is not capable of sensing aerodynamic or mechanical haptic feedback such as vibration of the plane or acceleration, further decreasing situational awareness and the ability to assess flight performance and –envelope.

These deficiencies restrict the effectiveness of UCAVs in both long distance operation along with short distance operations and engagements with adversaries. In the latter case, swift responses are indispensable for successful interaction with the environment and adversary in combat situations. This engenders the claim that UCAVs are insufficiently survivable and effective against manned aircraft and other UCAVs in air-to-air combat situations.

A potential solution overcoming the vulnerability of UCAVs in air-to-air combat scenarios is provided in the feasibility study by Hans Heerkens of the University of Twente (UT) and Frank Tempelman of

the Netherlands Aerospace Center (NLR). The solution is being referred to as a Semi-Direct Control System (SDCS).

The philosophy behind the Semi-Direct Control System entails a scenario in which the UCAV operator does not control the UCAV equipped with the Semi-Direct Control System in real-time with direct (joystick) control, but locates a position inside the airspace where the UCAV should fly to, a predetermined time and geographical distance apart from the current location. The frequency with which the pilot presents commands regarding the position and operation of the UCAV may vary. The commands may also incorporate operational information regarding approach possibilities, target acquisition and target engagement. In practice, this implies that the Semi-Direct Control System should possess a Combat Manoeuvring Management System (CMMS) that is equipped with a measure of "intelligence" that is being utilised in the decision making processes the UCAV with a Semi-Direct Control System will encounter. The CMMS determines the optimal way to fulfil the requirements of the command it has received regarding designated parameters such as position, velocity, weapons- and systems configuration. The Semi-Direct Control System constantly analyses positions and movements of both the UCAV with which it is equipped as well as adversary aircraft. It attempts to predict changes by using extrapolation techniques and using these to calculate and execute the necessary direction changes and also learning from its actions and decisions. The pilot determines the tactics to be used and can continuously modify the flight path and strategy based on the adversary's behaviour.

The Semi-Direct Control System is placed on the verge between direct control systems and near-complete autonomous control systems. In the first, the operator directly adjusts the aircraft's control surfaces and equipment. In the latter, the UCAV performs control and operation tasks in complete solitude in which the role and influence of the operator is reduced to a minimum. Drawbacks of the first control system in UCAVs reside in latency problems and data link frailness. Drawbacks of the latter reside in a too little developed state of the art in UCAV autonomy.

The Semi-Direct Control System aims at eliminating the drawbacks of these two systems: Due to on-board automation, there may be limited data exchange and communication between pilot and UCAV without the UCAV being unable to perform combat manoeuvres. Furthermore, the Semi-Direct Control System overcomes the need of extensive automation capabilities. As research in fields as machine learning and predictable controllers matures, the interval with which the operator has to send commands may decrease and the commands may become of a higher abstraction level.

Concluding, an UCAV equipped with Semi-Direct Control System possesses the potential to increase the survivability of this UCAV in air-to-air combat situations in which a quick reaction time is essential.

To date, the NLR does not possess a platform to **demonstrate**, **visualise** and **develop** the core functionality of the Semi-Direct Control System. The thesis that I am handling regards the development of such a platform to fulfil a higher goal: To perform research in the feasibility and efficiency of the Semi-Direct Control System. You will be interacting with the prototype that I have developed during my internship.

After reading, the user will be presented with the following questions:

You just read about UCAV control and the problem of latency, did the three flights (with latency) you have performed before reading this text successfully demonstrate why we want to develop the SDCS for air to air combat rather than using direct control with a joystick?

Did this interaction convince you of (our) perceived necessity of the SDCS for air to air combat in contrast to flying with a joystick?

Test 2: Flying with the SDCS

For this part I would like you to fly with the SDCS test aircraft and the control interface that I have made during my internship. We are going to perform a first flight where I will guide you through the features of the control interface. The latency will be set to 250 milliseconds Round Trip Time (RTT)

What are your experiences during this flight?

Test 3: Flying with the SDCS

For the next part of the task I would like you to fly with the SDCS test aircraft and try to follow my aircraft. After that, I will fly behind you and I would like you to try to get away from my aircraft with the commands on the user interface.

I have set up a set of requirements for this prototype. They are defined as follows:

- Allow to vary the amount of latency on the data link
- Allow the user to select different points of view
- Allow to display the flight parameters on screen and allow to select what information is displayed on screen
- Allow for commands to be different in nature and abstraction level
- Allow to vary the frequency with which the user provides input

Regarding how these requirements were translated in this prototype, how would you rate them? Did they have added value or should they be implemented differently?

What is your overall impression of the prototype?

What is your opinion on the control interface and the giving of commands to the UCAV in the prototype?

What is or what are the largest downfalls or shortcomings of the prototype?

Does the prototype reflect that what I let you read about the SDCS earlier on in this user test? Does this prototype reflect what the SDCS should (be able to) do?

What should, in your opinion, be added to the prototype to enhance functionality and to demonstrate how the SDCS works?

Did it become clear from your interaction with the prototype what the added value of the Semi Direct control system is?

Testing and Developmental platform (only for defined stakeholders)

The last requirement is about the expansion and implementation of future research and is formulated as follows:

- Allow for extension and developing by other developers

In your opinion, does the prototype have enough space for expansion for your future plans regarding the Semi-Direct Control System?

Did the brainstorm process lead to a natural solution that houses potential for further development?

Does the prototype meet your expectations?

L MVP Evaluation User Test

User test Stakeholders

Test 1: Flying with the joystick and latency

Both Jan Joris Roessingh and Gerald Poppinga have flown with delay. Gerald has flown with 100ms, 250ms and 1000ms delay and Jan Joris has flown with arbitrary, for him unknown delays. This was performed to determine if he was able to tell if there was a delay and make an estimate of how much that delay was.

What are your experiences during the three short flights?

After some steady flying and a couple of faster and more agile manoeuvres, Gerald mentioned that you have to force yourself to anticipate every movement and pre-calculate accordingly. He found that the delays of 100ms and 250ms were doable (with correction of overshoot), however 1000ms appeared to be too much to correctly handle the aircraft. Especially when performing actions such as following an aircraft and speculates that this would have even more impact during air to air combat scenarios. Gerald noted that the sensitivity of the joystick in lateral direction was rather high, which needed some adaptation. He mentioned that it is very difficult to determine the result of your joystick input. Both Jan Joris and Gerald noted that at a certain point, Pilot Induced Oscillations (PIO) will occur. Especially with 1 second delay and repetitive motions (trying to get out of a PIO-situation) it was difficult to determine which joystick input corresponded to what behaviour of the aircraft. Jan Joris noted that he tended to estimate the latency to be shorter than it actually was, causing PIO during a couple of manoeuvres. Gerald and Jan Joris presented the example of personal conversation with a Global Hawk operator, who had claimed that delays of 6 seconds are not uncommon. During landing and take-off, the controls have to be handed over to a local operator since the large latencies have caused multiple crashes especially during landing. A comment was made that it is a good idea to set the latency back to zero between the subsequent tests at 100ms, 250ms and 1000ms in order to regain a feeling for the original handling of the aircraft.

It proved to be very difficult to fly with the UCAV and an adversary at the same time. Attempts have been made to fly together and stay close enough together to perform a follow manoeuvre and some take-over manoeuvres, but it proved to be too difficult to effectively fly together to perform complete tasks.

After answering this question, the user is presented with the introductory text for the SDCS and its background.

Jan Joris and Gerald are stakeholders of this project and thus are familiar with the SDCS, its background and possible applications.

The stakeholders were presented with the following questions:

You just read about UCAV control and the problem of latency, did the three flights (with latency) you have performed before reading this text successfully demonstrate why we want to develop the SDCS for air to air combat rather than using direct control with a joystick?

The MVP will only be able to effectively demonstrate how difficult flying with direct control under latency if it is possible to really perform UCAV-related tasks. The manoeuvres and simple tasks that were performed during this user test however do effectively demonstrate how difficult it is to directly control an UCAV with latency. It is easy to induce PIO and it is difficult to recover from PIO with latency.

Did this interaction convince you of (our) perceived necessity of the SDCS for air to air combat in contrast to flying with a joystick?

The perceived necessity is already clear for the stakeholders. However, to make the demonstration of the necessity more evident, the user tests should really include a UCAV-based task. A good example would be a working following function that enables the pilots to come closer to each other and actually perform combat manoeuvres rather than being too busy with trying to find each other in the skies before being able to interact. Tasks suitable for such a test would be UCAV air refuelling or following another aircraft. For now, the follow function realised in the MVP was not able to offer enough assistance to easily follow the adversary aircraft, limiting the effectiveness of the test flight.

Test 2: Flying with the SDCS

Jan Joris and Gerald have flown with the user interface, earlier on in the project.

What are your experiences during this flight?

The launch function takes off the UCAV after which it will remain steady level flight, everything looks as specified in an earlier stage of the project (Specification phase). Jan Joris notes that the flying is rather different than flying with joystick; there is less freedom and you are more dependent on the commands that you have. Jan Joris states that the main problem for now is the fact that different PID controllers need to be implemented for the different degrees of freedom of the UCAV. For the final product, a more sophisticated flight controller is needed. However, Gerald and Jan Joris mentioned that the MVP does make them realise the complexity of the development of a FMS for the SDCS. In addition, for air to air combat, even more sophisticated controllers are necessary.

Test 3: Flying with the SDCS

For the next part of the task I would like you to fly with the SDCS test aircraft and try to follow my aircraft. After that, I will fly behind you and I would like you to try to get away from my aircraft with the commands on the user interface.

I have set up a set of requirements for this prototype. They are defined as follows:

- Allow to vary the amount of latency on the data link
- Allow the user to select different points of view
- Allow to display the flight parameters on screen and allow to select what information is displayed on screen
- Allow for commands to be different in nature and abstraction level
- Allow to vary the frequency with which the user provides input

Regarding how these requirements were translated in this prototype, how would you rate them? Did they have added value or should they be implemented differently?

Latency is realised as it should, there is a clear differentiation between uplink latency and downlink latency as is desired and which furthermore reflects the real-life situation. It is useful that the delay can be set to any value that is desirable, for example 250ms may be common, but Jan Joris have mentioned that they are also interested in displaying more extreme delays of up to 10 seconds.

The different points of view are good, it is not yet known which views are actually needed, but FlightGear comes with enough options to produce custom views, as demonstrated by the MVP. Gerald mentioned that it would be useful to make use of different monitors and is curious if that would also work without a locally calculated FDM.

The displaying of the flight parameters is performed as specified in the Specification phase. FlightGear also provides the Property Tree which allows access to all variables and allows modifying these. Gerald and Jan Joris both noticed that it would be more meaningful if the flight parameters are not presented as numbers as is now. The numbers are not meaningful and not useful during flight. An improvement would be a visualisation of the 6 basic flight instruments in the form of a Primary Flight Display (PFD).

The commands for now have been well realised, a positive thing is that the base of a "follow function" is already established. Although it does not function completely, it is a rather important manoeuvre. Also, as of now, the basic manoeuvres perform as they should but the possibilities for manoeuvring are limited.

The frequency with which user provides input was a requirement that was dependent on what the UCAV needed to do when it is not receiving commands. In the Specification phase it was established that the UCAV should be able to maintain steady level flight. In the MVP this is correctly realised.

What is your overall impression of the prototype?

Jan Joris mentioned that the MVP is a good first step in the research process for the SDCS. There are a number of different ideas that can be generated and implemented as continuation of this research. It can serve as a tool to further develop the SDCS.

What is your opinion on the control interface and the giving of commands to the UCAV in the prototype?

Gerald mentioned there could be a better differentiation between which side of the screen is for logging information and which parts are for providing commands. Also, Gerald mentioned that in the future it might be an idea to implement the GUI inside the FlightGear environment so that it can sit together in one screen with FlightGear or in a transparent overlay fashion.

What is or what are the largest downfalls or shortcomings of the prototype?

Jan Joris mentioned that it is not per se a shortcoming, but the fact that the "follow function" is not completely functional; it is difficult to effectively demonstrate the SDCS in a typical UCAV-task. Flying around is possible now, but it would be more interesting to have a use case in which the operator has to perform an UCAV-related task.

For now, that is difficult to realise with the functionality the MVP possesses. In that aspect, the MVP is not yet mature enough.

Using FlightGear, it is easy to simulate functions like following an aircraft because you have access to all flight parameters of all aircraft. In real life situations, one would be dependent on radar and sensors to collect all data of adversaries and other aircraft. Eventually, it is a subject of research as to what sensors or radar techniques are needed for these functionalities. Consequently, these need to be implemented in the MVP.

Does the prototype reflect that what I let you read about the SDCS earlier on in this user test? Does this prototype reflect what the SDCS should (be able to) do?

The simulation of an UCAV over a delayed data link is convincing. The manoeuvres need more work.

What should, in your opinion, be added to the prototype to enhance functionality and to demonstrate how the SDCS works?

Jan Joris and Gerald mentioned once more that the follow function needs to be functional

Did it become clear from your interaction with the prototype what the added value of the Semi Direct control system is?

The MVP is not yet mature enough to fully convey the message of the SDCS in different UCAV-related tasks is.

Testing and Developmental platform (only for defined stakeholders)

The last requirement is about the expansion and implementation of future research and is formulated as follows:

- Allow for extension and developing by other developers

In your opinion, does the prototype have enough space for expansion for your future plans regarding the Semi-Direct Control System?

Jan Joris and Gerald mentioned that it is difficult to estimate how well future implementations can be done. However, the organization of FlightGear gives access to necessary variables. A separate session should be planned to look at how desired functionalities that Gerald and Jan Joris propose, can be implemented in the MVP.

Did the brainstorm process lead to a natural solution that houses potential for further development?

Gerald mentioned that what he is really positive about is the fact that everything made so far is indeed realised with open source software. Gerald is convinced that the work that is delivered by the enthusiastic community will in the future maintain to deliver functionality and improvements that have to potential to save a lot of work for the NLR (power of the masses) in the case that the development of the MVP is continued. For example, it may happen that someone designs an UCAV (Predator or the like) that is equipped with functioning armament (functioning armament is already

an included function in FlightGear) that may be very suitable for adaptation in the MVP without putting in a large amount of work.

Does the prototype meet your expectations?

Jan Joris and Gerald mentioned that good progress is made on the MVP and that it becomes clear what the next steps of development should be. They are furthermore curious in further research into the SDCS and think the MVP allows a platform to develop the SDCS in.

Gerald mentioned that interaction with the MVP also has a fun-factor. This may have a positive influence on the attractiveness and the overall impression of the MVP.

M Ideas SDCS development

The creation of a MVP of the SDCS inevitably sparked ideas about the development of the SDCS. This section will handle these thoughts that can be taken into consideration during the development of the SDCS. This section will refer to theory that has been covered in the Analysis chapter of this thesis and will maintain an informal writing style accompanied by personal (author's) viewpoints.

Firstly, I think the most important part to realise first is the CMMS. In the recommendations, I have talked about the leveraging of autopilot functionality to accomplish manoeuvres. The autopilot functionality is a very handy tool but to realise all the functionality of the SDCS. A manoeuvre like cutting a corner to approach an adversary may be possible by means of these controllers. It is also mandatory that the controller (which may partially consist of autopilot elements) also has the ability to predict actions of the adversary. I have some ideas about how this should be realised; one could extrapolate on figures for heading, speed and altitude to calculate an estimated trajectory that the adversary will follow and display this on the screen. For the rest of the project: It is difficult to determine how this should be realised when we do not yet know what sensors will be aboard the UCAV. Another choice that has to be made is whether to continue using Python, which is an external program, or embed the CMMS inside FlightGear and provide commands to this module inside FlightGear. Embedding the CMMS inside FlightGear for now seems to be the most logical option. It will then become for example a menu inside FlightGear, much like the Autopilot now is a menu inside FlightGear and you can then send single commands towards FlightGear from Python, instead of having to run Python scripts that send a lot of commands to FlightGear. However, if one decides to place it as a module inside FlightGear, the only option (for as far as I know) is to do this with the aid of Nasal scripting. I do not know if Nasal scripting is an efficient scripting language or if it even possesses the ability to harness such complex functionality such as an CMMS.

The next item is closely related to the above paragraph. The "intelligence" aboard the UCAV can be realised with respect to Boyd's OODA loop. Special attention should be paid to which stage of Boyd's loop actually requires intelligence. The human may be very well at rational and creative thinking, but a computer may think faster. I recommend to keep Boyd's OODA loop as a frame of reference when introducing intelligence. You can very easily categorise the different kinds of intelligence needed into the different stages. One can also figure out where intelligence aboard the UCAV is needed by analysing in which phases of Boyd's loop human errors frequently occur, or which phases are related to a high operator workload. It is in addition to that also very important to think about which stages you would even want machine intelligence. It can be reasoned that a computer can make decisions faster than a human and it therefore may make sense to add intelligence to the "Decide" phase of the OODA loop. But it is important to realise that the human must remain in control at all times.

This brings us to the subject of management types. In the analysis chapter I have defined two types of management: management by consent and management by exception. I have furthermore stated that the SDCS, as proposed by Hans and Frank, can be seen as a mixed form of these two management forms. This may seem logical, however I think it is important to make a choice between where and when these management types should be used. From observations in the user testing (and test flying) it could be seen that the workload of the pilot was not decreased by using the SDCS Control Interface with the commands that the MVP possesses as of now. I think the presence of these two management forms will

make things even more complicated. As an example: In the section "Alternative GUI for the MVP" in the Recommendations I described an alternative interface with a drop down menu for commands. This however was only an example to make the interface more intuitive than it would be when you can only click on stationary buttons. If indeed management by consent was equipped for the SDCS, the operator is more busy with controlling the actions of the UCAV rather than determining strategy. It gets worse when the two management types are combined: the UCAV proposes and action and the operator gives consent. I think this will lead to the operator having to scroll down a list of possible options (this amount may be really large due to the dynamics of air to air combat) and choosing one option. This may cost so much time that it is not an effective way of controlling. This will probably also be heavily dependent on the amount and type of information that the operator gets from the UCAV.

The second most difficult part of developing the SDCS is creating an effective interface. The amount of data that is fed back to the operator is heavily depending on the amount of intelligence on board and vice versa. If the operator cannot oversee everything, a larger amount of intelligence may be needed. If however the operator has the option to be more aware of the surroundings and state of the aircraft, the intelligence aboard the UCAV might be decreased. I recommend to first look at the options for information retrieval before speculating about interfacing and interfacing methods. I know that in most cases a HUD and a video feed can be send over the data link to the operator. But systems with lower bandwidth may not even be able to feed video back to the operator. In this case, the operator may only have its primary flight instruments to rely on in combination with a 2D geographical map. First it needs to be established what the maximum amount of information is that can be send over the data link by making reasonable assumptions using data of existing RPAS systems). For example, what is the maximum video quality, is there a possibility to stream 360 degree video feed (this would make the use of Virtual Reality very attractive). And what more information can be send? Is it enough to develop a fully synthesised picture of the battlefield with great detail? Another problem that needs more intention with relation to interfacing is the way the operator provides commands. In the examples mentioned so far (gesture control or clicking on screen) we have to be aware that we are talking about a two dimensional screen or interface while the battlefield is 3 dimensional. This is a big problem since this does not allow us to see depth and potentially disable us to give sensible commands; a barrel roll can be performed, but over what distance? How do we properly select a distance between our UCAV and the adversary without the need of different 2D viewpoints from different angles?