**System Design Document**

**For**

**UAV Swarm**

Team Members:

* Samantha Balistreri
* Tyler Wise
* Joseph Moran
* Michael Fornito
* William Edwards
* Daniela Regueira

|  |  |
| --- | --- |
| Version/Author | Date |
| 1.0 Samantha, Tyler, Joseph, Michael, William, Daniela | 10/31/2020 |
| 2.0 Samantha, Tyler, Joseph, Michael, William, Daniela | 11/30/2020 |
|  |  |
|  |  |

Table of Contents

[**1. Overview**](#_kzljllt38qf5) **1**

[**2. Introduction**](#_zhewvj9dksxg) **2**

[2.1 Purpose and Scope](#_1fob9te) 2

[2.2 Project Executive Summary](#_3dhxg8isrpxv) 2

[2.2.1 System Overview](#_7r1spg8hmp1d) 2

[2.2.2 Design Constraints](#_w6hyohxyqol2) 2

[2.2.3 Future Contingencies](#_auxj9ocjr48d) 2

[2.3 Project References](#_deu6k1uwxvsf) 3

[2.4 Glossary](#_zeeanxtnkv9c) 3

[**3. System Architecture**](#_yknai7s92qsu) **3**

[3.1 System Software Architecture](#_i96ojzvj0i8a) 3

[3.2 Internal Communications Architecture](#_xaq7586rkmwa) 4

[**4. Human-Machine Interface**](#_k6xsvhohs7o9) **4**

[4.1 Inputs](#_bwagjs7gap7e) 4

[4.1.1 Input Parameters](#_33z6ahyk9xyy) 4

[4.2 Outputs](#_z67nvlo70fqo) 4

[4.2.1 User Input Feedback](#_7f63pprw4lph) 4

[4.2.2 Warning Messages](#_6icih3pl2wdy) 5

[4.2.3 Data Collection](#_z49jyi7vxtj) 5

[**5. Detailed Design**](#_up9pyqenk91v) **6**

[5.1 Software Detailed Design](#_xl6j57ibkuz9) 6

[5.2 Internal Communications Detailed Design](#_ju9f454oimlq) 8

[5.2.1 Flowchart for Asynchronous Drone Image Capture](#_xww2k8xa87r8) 8

[5.2.2 Flowchart for Asynchronous Drone Image Processing](#_srvkdr6wvx74) 9

[5.2.3 Flowchart for Asynchronous Drone Position Update Receipt](#_b0raxwp7taby) 10

[5.2.4 AirSim API Interface](#_rl946fa329cq) 11

[5.2.5 System Data Flow Diagram](#_i1lt7n1d4j9h) 12

[5.2.6 System UML Class Diagram](#_sz792s4vvs11) 13

[**6. External Interfaces**](#_do8lpsbmd8ew) **13**

[6.1 Interface Architecture](#_9bfbkeg6uz6s) 13

[6.2 Interface Detailed Design](#_cyymomd1vauq) 13

# 1. Overview

*The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.*

# 2. Introduction

## 2.1 Purpose and Scope

The purpose of this document is to show the architecture and design of the drone swarming simulation constructed with Unreal Engine and AirSim. The scope of this project is to develop a swarming algorithm that uses image recognition software to allow drones to swarm in a dynamic formation.

## 2.2 Project Executive Summary

The UAV Swarm project will be a useful program for development in air-space optimized surveillance research. The project has the ability to support ten drones that will simulate swarming behavior within Unreal Engine. The drones will use image processing to identify their location as well as the location of the other drones around them.

## 2.2.1 System Overview

The product to be produced will be a UAV Swarm Simulator that will allow a swarm of 10 drones, or less, fly in swarming formations. The drones will use image processing software and swarming algorithms to accomplish this goal. This product will be done strictly in a simulated environment and there are no physical components to this product.

## 2.2.2 Design Constraints

The system was designed to use the interface available with the Unreal Engine program. The trade-off with this approach is that the user requires training to use Unreal. This could cause conflicts in the program if the user does not follow the proper instructions and training. The project team assumes that users are capable of learning how to use the program.

## 2.2.3 Future Contingencies

A change that might arise that will affect the current design of the system is switching from an image based swarming algorithm to a GPS based algorithm. This would change the design as it would no longer need an image processing software and instead use GPS tracking.

## 2.3 Project References

* APAWSAN: Actor Positioning for Aerial Wireless Sensor and Actor Networks by Mustafa Ilhan Akbas and Damla Turgut, University Of Central Florida
* Vision Based Formation Control in Unreal Engine -Air Sim by Kaveh Fatian
* Actor Positioning Based on Molecular Geometry in Aerial Sensor Networks by Mustafa ˙Ilhan Akbas¸ Gurkan Solmaz and Damla Turgut, University of Central Florida

## 2.4 Glossary

* UAV - Unmanned Aerial Vehicle
* UAS - Unmanned Aircraft Systems
* UE4 - Unreal Engine 4
* MVS - Microsoft Visual Studio
* SBF - Swarm Behavior Framework
* AirSim - An open-source, cross platform simulator for drones, ground vehicles such as cars and various other objects, built on Epic Games’ Unreal Engine 4 as a platform for AI research.
* GPS - Global Positioning System

# 3. System Architecture

## 3.1 System Software Architecture

In this section, describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer-aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module.

All of the following software was used in development of this project:

* Python 3.6
* Unreal Engine 4
* AirSim

In regards to the project Python 3.6 was used in order to add functionality to the drones. Code written in python was used to control the flight of the UAVs, get images from the UAVs and handle the image processing. Unreal Engine 4 was used to let the user see a visual of the drones swarming and is all where the simulation is based out of. The users will also be able to use elements in unreal to control the settings of the swarm. AirSim acts as the bridge between UE4 and Python. AirSim gives drone objects to unreal and also supplies the API’s to python that allow it to control the drones.

## 3.2 Internal Communications Architecture

In this section, describe the overall communications within the system; for example, LANs, buses, etc. Include the communications architecture(s) being implemented, such as X.25*,* Token Ring, etc. Provide a diagram depicting the communications path(s) between the system and subsystem modules. If appropriate, use subsections to address each architecture being employed.

**Note:** The diagrams should map to the FRD context diagrams.

# 4. Human-Machine Interface

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. The interface will be the command terminal set to Python.

## 4.1 Inputs

The diagram below describes input parameters to ensure the user does not face any errors as stated in Section 3.2 and allows the user to freely customize their simulation execution.

### 4.1.1 Input Parameters

1. **Add Drone -** The user shall add up to 10 drones via the command terminal one at a time. This includes the drones positions (X,Y,Z) when creating a new drone.
2. **Remove Drone -** The user shall remove up to 9 drones via the command terminal one at a time
3. **Drone Distribution Size -** The user shall input their distribution size for how large the swarm should be for the simulation



## 4.2 Outputs

The diagram above describes output responses to ensure the user that all inputs are valid, any errors within the system, and to improve the software’s fault tolerance.

### 4.2.1 User Input Feedback

1. **Drone Added-** The terminal shall notify the user the drone has been added
2. **Drone Removed-** The terminal shall notify the user the drone has been removed
3. **Drone Display -** The terminal shall display all drones and their parameters (name, vehicle type, and location)
4. **Drone Distribution Size -** The terminal shall notify the user the distribution size is set

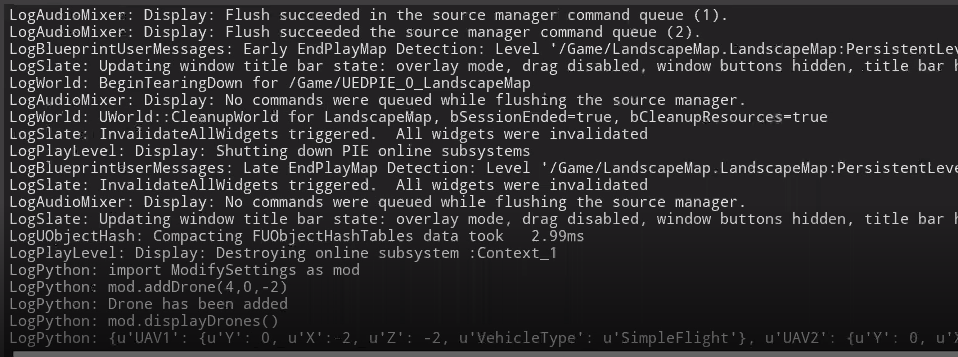
### 

### 4.2.2 Warning Messages

1. **Failed To Remove Drone -** The warning is displayed to the user if the user does not properly put the correct range of values as specified in Section 3.1 for the input interface design to remove the drone.
2. **Out of Range Distribution Size -** The warning is displayed to the user if the user does not properly put the correct range of values as specified in Section 3.1 for drone distribution.
3. **Out of Bounds -** The warning is displayed to the user if the user does not properly put the correct range of values as specified in Section 3.1
4. **Failed To Find Settings** - The warning is displayed if Unreal cannot find the settings.json
5. **Inputs can only be a number -** Based on the values discussed in Section 3.1, the inputs can only be integers or floats.

### 4.2.3 Data Collection

1. **Notification Terminal** - The terminal will give the user feedback when modifying the settings.json file in the Unreal Terminal
2. **Settings File -** The settings file contains all the information: drone type, the number of drones, names of drones, the sim mod, the distribution size, engine sound, log messages, and record UI visible.
3. The python script that runs the swarm has a dataframe that keeps track of all the drones positions so that it may output a csv file and scatter plots.





# 5. Detailed Design

## 5.1 Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

* A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
* For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
* Data elements, record structures, and file structures associated with module input and output
* Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
* Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
* Report layout

## 5.2 Internal Communications Detailed Design

## 5.2.1 Flowchart for Asynchronous Drone Image Capture

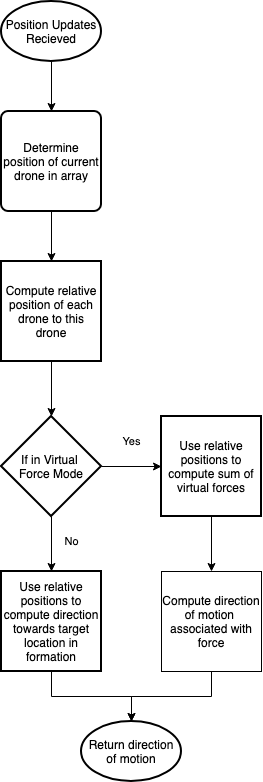
## 

## 5.2.2 Flowchart for Asynchronous Drone Image Processing



## 

## 5.2.3 Flowchart for Asynchronous Drone Position Update Receipt



## 5.2.4 AirSim API Interface

## 

## 

## 

## 

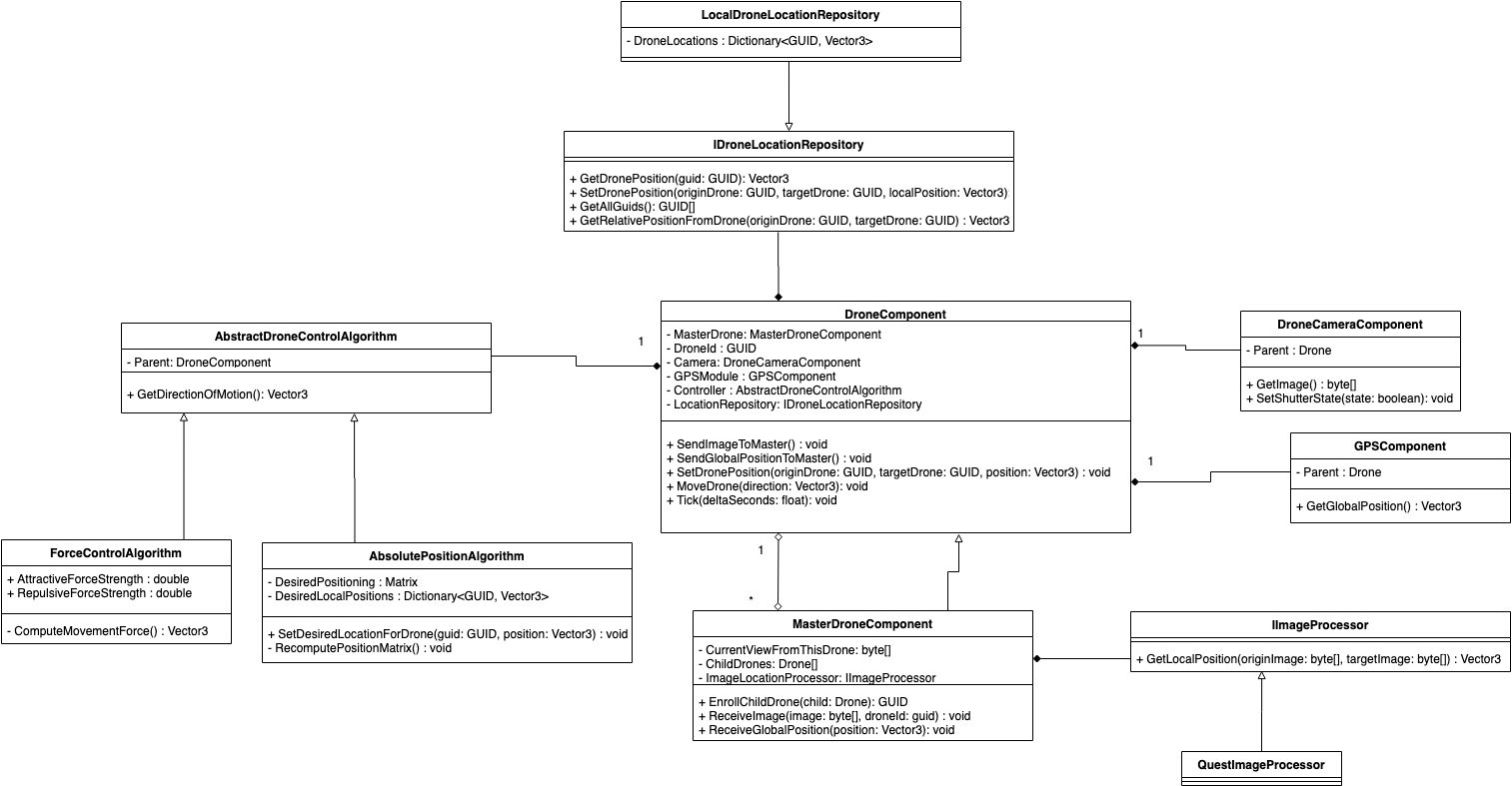
## 

## 

## 

## 5.2.5 System Data Flow Diagram

## 5.2.6 System UML Class Diagram



# 6. External Interfaces

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

## 6.1 Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

## 6.2 Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

* The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
* Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
* Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
* Graphical representation of the connectivity between systems, showing the direction of data flow
* Query and response descriptions

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.