4th year project
Automated Drone Air Traffic Control System

Functional Specification

Institute of Technology Carlow

Supervisor: Dr. Oisin Cawley
Author: James Hall
Submission Date: 27 November 2020

Table of Contents

1. Introduction

The purpose of this document is to highlight the functional and non-functional requirements of the Automated Drone Traffic Control System. This will be demonstrated through use cases and diagrams, as well as the use of the FURPS+ model. The document will also take

a look at the development tools that will be used to develop the application. Then we will look at similar products already on the market

2. Application Description

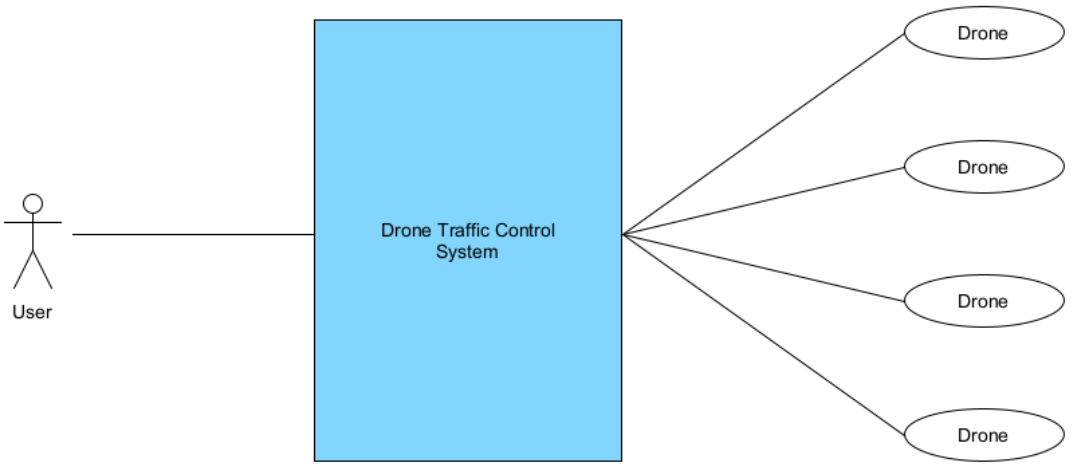
The finished application will be a system that is capable of automatically controlling a drone or fleet of drones. The system will be able to CRUD new drones and Locations. The Locations will be linked by a system of flight corridors for the drones to travel along. The system will be able to take instructions on a desired Location for individual drones to travel to and will automatically decide upon a flight path and instruct the drone to move to that Location. It will maintain real time contact with all drones on its network to ensure they arrive at their Locations safely, and will be able to relay this information to a human user in the form of an on screen map, showing the flight corridor network layout and the real time position of all drones.

3. Application Functionality

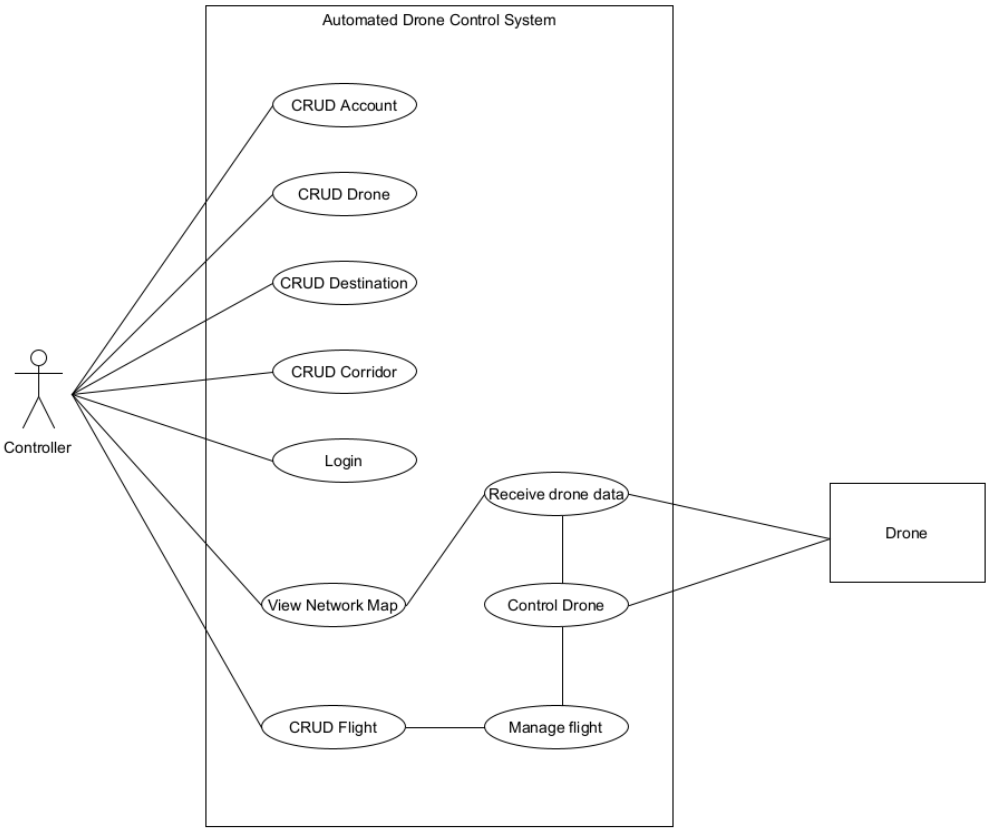
The basic functionality of the application is to be able to automatically control a fleet of drones.
This control will allow for the drones to be automatically moved from point to point in a network.
The user will be able to request a drone flight for a particular drone from a chosen point to a Location, where a location is a geo-point saved on the system, and the system will choose a path for the drone to take. A location is a point where a drone can take off and land from, or in other words, begin or end a journey.
The system will ensure that drones move safely throughout the network without colliding with each other or leaving the predetermined flight corridors.
The System will be able to receive data from drones in real time, such as Speed, Heading, Altitude, GPS coordinates and Battery life. This data will be used to inform the Controller and to allow the System to avoid collisions and guide the drones.
The user will be able to view the position in real time, of all the drones on the network.

4. Use Cases

4.1 Context Diagram



4.2 Use Case Diagram



4.3 Use Cases

Add Drone

Use Case	**Add Drone**
Actors	User, System, Drone
Pre Conditions	The User is logged in and wishes to add a new drone to the system. The Drone is powered on.
Success Scenario	1. This use case begins when the User clicks the “Add Drone” button on the main screen. 2. The System will scan for any available Drones and check they are not already added to the System. 3. The system will display a list of available Drone(s) to the User and the User will choose the Drone they wish to add to the system. 4. The System will prompt the User to name the Drone. 5. The System will inform the User that the Drone has been added successfully and then return the User to the main screen.

Post Conditions	The new Drone has been added to the System.
Alternative	3a. The Drone is not visible to the System. 3a.1 TheUser will power the Drone off and on and then “refresh” the available Drone list.

Update Drone

Use Case	**Update Drone**
Actors	User, System, Drone
Pre Conditions	The user is logged in and wishes to update information for a drone. The Drone is powered on.
Success Scenario	1. The use case begins when the User clicks the “Update Drone” button on the main screen. 2. The System displays a list of Drones to the user that are registered with the System and are currently being picked up by the System and are not currently in flight. 3. The user chooses the Drone they wish to update the information for and the System prompts the User to rename the Drone. 4. The System then informs the User that the drone information has been updated and returns the User to the main screen.
Post Conditions	The drone information is updated on the system.
Alternative	3a. The Drone is not visible to the System. 3a.1 TheUser will power the Drone off and on and then “refresh” the available Drone list.

Delete Drone

Use Case	**Delete Drone**
Actors	User, System
Pre Conditions	The user is logged in and wishes to remove a drone from the system.
Success Scenario	1. The use case begins when the User clicks the “Delete Drone” button on the main screen. 2. The System displays a list of Drones to the user that are registered with the System and are not currently in flight or part of any flight plan. 3. The user chooses the Drone they wish to delete from the System and the System asks the User are they sure they want to delete the drone.

	4. The System deletes the Drone. 5. The System then informs the User that the Drone has been deleted and returns the User to the main screen.
Post Conditions	The drone is removed from the system.
Alternative	

Add Location

Use Case	**Add Location**
Actors	User, System
Pre Conditions	The User is logged in and wishes to add a new Location to the System.
Success Scenario	1. This use case begins when the User is on the main screen and wishes to add a new Location to the System. 2. The user selects “Add Location” and is prompted to enter the GPS coordinates of the new Location and give it a name. 3. The User submits the information and the System confirms to the User that the new Location has been added.
Post Conditions	A new Location has been added to the System.
Alternative	3a. The GPS coordinates are not correctly formatted. 3a.1 The System prompts the User to re enter the coordinates.

Update Location

Use Case	**Update Location**
Actors	User, System
Pre Conditions	The User is logged in and wishes to update the information on one of the the currently existing Locations on the System
Success Scenario	1. This use case begins when the User is on the main screen and wishes to update Information for a Location on the System. 2. The user selects “Update Location” and is presented with a list of current Locations. 3. The user selects the required Location and is presented with textboxes displaying the current information.

	4. The user may edit the information in the text boxes. 5. The User then submits the updated information and the System confirms to the User that the Location has been updated.
Post Conditions	The Location's information is updated on the System.
Alternative	5a. The GPS coordinates are not correctly formatted. 5a.1 The System prompts the User to re enter the coordinates.

Delete Location

Use Case	**Delete Location**
Actors	User, System
Pre Conditions	The User is logged in and wishes to remove a Location from the System.
Success Scenario	1. This use case begins when the user is on the main screen and wishes to remove a Location from the System. 2. The User selects "Delete Location" as is presented with a list of current Locations that are not part of any planned or active flight plan. 3. The User selects the Location they wish to delete and the System prompts them with an "are you sure?" message. 4. The User agrees and the System deletes the Location and any corridors connecting to that Location. 5. The System then informs the User that the deletion has been successful.
Post Conditions	The Location is removed from the System.
Alternative	3a. The User chooses the wrong Location for deletion. 3a.1 At the "are you sure?" prompt, the user selects "no" and is returned to the Location selection menu.

Add Corridor

Use Case	**Add Corridor**
Actors	User, System
Pre Conditions	The user is logged in and wishes to add a new corridor between two Locations.

Success Scenario	1. This use case begins when the User is on the main screen and wishes to add a new flight corridor between two Locations. 2. The User clicks the “Add Location” button and is prompted by the System to enter in the two Locations to be connected via two drop down menu boxes containing lists of Locations on the System. 3. The User is also prompted to name the corridor. 4. The System creates the corridor and confirms the creation to the user.
Post Conditions	The new corridor has been added to the System.
Alternative	4a. The User tries to add a corridor with matching Locations. 4a.1 The System informs the user that the same Location has been used for both ends of the corridor and allows the user to re-enter the Locations.

Delete Corridor

Use Case	**Delete Corridor**
Actors	User, System
Pre Conditions	The user is logged in and wishes to remove one of the existing corridors from the System.
Success Scenario	1. This use case begins when the user is on the main screen and wishes to delete a flight corridor. 2. The User clicks the “Delete Corridor” button and the system presents the User with a list of flight corridors. 3. The User selects a corridor from the list and the System asks the User if they have made the correct selection. 4. The User clicks “yes” and the System deletes the corridor. 5. The System then confirms to the User that the corridor has been deleted.
Post Conditions	The corridor is removed from the System.
Alternative	4a. The user clicks “no” when asked if they are sure. 4a.1 The System returns the User to step 3 and the user re-selects a corridor.

Create Flight

Use Case	**Create Flight**
Actors	User, System, Drone
Pre Conditions	The User is logged in and wishes to add a new flight to the System.
Success Scenario	1. This use case begins when the User is on the main screen and wishes to add a new flight to the System. 2. The User clicks the “Add Flight” button and the System presents the User with a form containing drop down menus for Departure Location, Arrival Location and Drone, along with a field to enter the departure time. 3. The User fills in the details of the flight and submits the flight. 4. The System adds the flight to the flight list and informs the User that the flight has been successfully created.
Post Conditions	The User receives confirmation from the System that the flight has been created.
Alternative	3a. The Drone chosen by the User is scheduled for another flight at the chosen time. 3a.1 The System informs the User that the Drone will be in use at that time and allows the user to select a drone from a list that will be free at the desired flight time.

Cancel Flight

Use Case	**Cancel Flight**
Actors	User, System, Drone
Pre Conditions	The User is logged in and wishes to cancel a flight on the system.
Success Scenario	1. This use case begins when the User is on the main screen and wishes to cancel a flight. 2. The User clicks the “Cancel Flight” button and is presented with a current list of flights on the System. 3. The User chooses a flight from the list and clicks “cancel”. 4. The System checks if the flight is currently active. 5. If the flight is not active the System removes the flight from the list and confirms the deletion to the User.
Post Conditions	The flight is cancelled by the system.
Alternative	5a. The selected flight is active. 5a.1 The System informs the User that the flight is active and asks if the User is sure they want to cancel the flight.

	5a.2 The user “confirms” and the System instructs the drone to return to its Origin. 5a.3 The System deletes the flight from its list and informs the user the flight has been cancelled.

View Drone

Use Case	**View Drone**
Actors	User, System
Pre Conditions	The User is logged in and wishes to view data on a drone on the System.
Success Scenario	1. This use case begins when the User clicks the “Drones” tab. 2. The System displays a list of drones on the System in a menu beside the map. 3. The User may click on any of the drones. 4. The System displays specific data relating the drone on screen.
Post Conditions	The User can view data on a drone.
Alternative	

View Location

Use Case	**View Location**
Actors	User, System
Pre Conditions	The User is logged in and wishes to view a Location on the network.
Success Scenario	1. This use case begins when the User clicks the “Locations” tab. 2. The System displays a list of Locations on the System in a menu beside the map. 3. The User may click on any of the Locations . 4. The System displays specific data relating the Location on screen.
Post Conditions	The User can view data on a location on the network.
Alternative	

View Corridor

Use Case	**View Corridor**
Actors	User, System
Pre Conditions	The User is logged in and wishes to view a corridor on the network.
Success Scenario	1. This use case begins when the User clicks the “Corridors” tab. 2. The System displays a list of Corridors on the System in a menu beside the map. 3. The User may click on any of the Corridors. 4. The System displays specific data relating the Corridor on screen.
Post Conditions	The User can view data on a corridor on the network.
Alternative	

Receive Drone Data

Use Case	**Receive Drone Data**
Actors	System, Drone
Pre Conditions	The System is connected to the Drone and the Drone is powered on.
Success Scenario	1. This use case begins when the System requests data from the Drone relating to position, heading, speed, altitude, pitch, yaw and battery life. 2. The drone responds by sending the data to the system.
Post Conditions	The System receives the data from the Drone
Alternative	

Manage Flight

Use Case	**Manage Flight**
Actors	System, Drone
Pre Conditions	The System has a flight plan and and needs to direct a drone to its destination

Success Scenario	1. This use case begins when the System receives a flight plan from the list of flight plans. 2. The system calculates the most efficient route for the drone to take to its destination and creates a flight plan. 3. When the correct time for the flight is reached the System passes the flight plan to the Drone Controller. 4. When the Drone Controller sends confirmation that the flight is completed, the System removes the flight from the list.
Post Conditions	The Drone completes its flight
Alternative	

Control Drone

Use Case	**Control Drone**
Actors	System, Drone
Pre Conditions	The Drone is powered on and connected to the System
Success Scenario	1. This use case begins when the Drone Controller receives a flight plan from the Flight Manager. 2. The System sends a data request to the Drone and waits for a response. 3. When a response is received if there are no obstructions, the controller issues control commands to the drone in accordance with the flight plan. 4. When the flight plan is completed, the Drone Controller informs the Flight Planner that the Flight is complete.
Post Conditions	The Drone has completed its flight plan successfully.
Alternative	2a. The Drone fails to return data. 2a.1 The Drone automatically returns to Flight Origin. 3a. The System fails to send commands to the Drone. 3a.1 The Drone automatically returns to Flight Origin. 3b. The Drone encounters another drone with priority(lower battery first, then greater travel distance). 3b.1 The System instructs the Drone to increase altitude and wait. 3b.2 When the priority drone has passed, the System instructs the Drone to decrease altitude and continue.

5. FURPS+

5.1 Functionality

Functionality: “is assessed by evaluating the feature set and capabilities of the program, the generality of the functions that are delivered, and the security of the overall system.” (1000sourcecodes.com, 2020)

The system will allow the user to CRUD (Create, Read, Update and Delete) drones to the system. It will also allow the user to CRUD Locations and flight corridors. The system will allow the user to CRUD flights which the system will manage automatically once created. The user will also be able to view all current flights via a map which will display current data on all drones, Locations and flight corridors on the system.

5.2 Usability

Usability: “is assessed by considering human factors, overall aesthetics, consistency, and documentation.” (1000sourcecodes.com, 2020)

The system should have a straightforward layout that is easy for the user to navigate. The user should be able to easily add new drones to the system. The user should find it easy to add new Locations to the system. The addition of corridors between Locations should be easy for users to accomplish. Setting up a new flight should be as simple as selecting the Location and desired flight time, the system should be able to arrange other details like the chosen drone and flight corridor pathing. The system should be able to clearly communicate the health of individual drones to the user for basic maintenance, such as battery replacement.

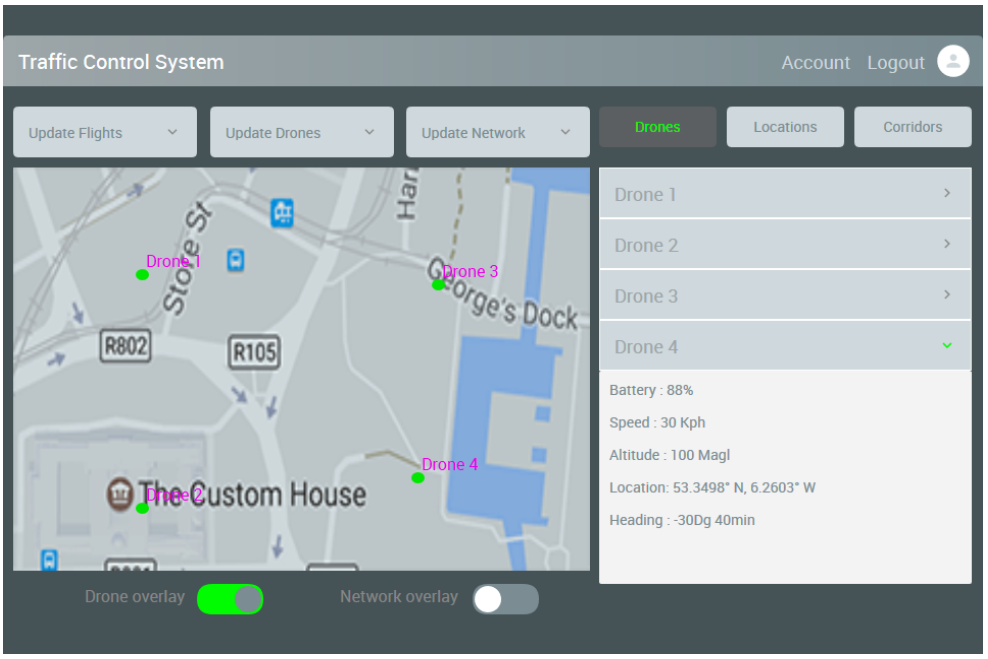


Fig 1. Example User Interface Layout

5.3 Reliability

Reliability: “is evaluated by measuring the frequency and severity of failure, the accuracy of output results, the mean-time-to-failure (MTTF), the ability to recover from failure, and the predictability of the program.” (1000sourcecodes.com, 2020)

The system should be able to maintain communication with drones for at least 99% of data communication intervals. The system should be able to accurately judge the health of drones to avoid all but catastrophic failure 99.9% percent of the time. The system should be able to accurately relay drone data to the user via the in application map 99% of the time. In the event of loss of contact with a drone. The system should be able to easily handle upscaling with the addition of more drones, Locations and corridors.

5.4 Performance

Performance: “is measured by processing speed, response time, resource consumption, throughput, and efficiency.” (1000sourcecodes.com, 2020)

The system should be able to load the user to the main screen within 30 seconds of being launched. The system should be able to process data from multiple drones in real time and respond accordingly within 100ms to individual drones, and should not see a notable drop in performance as the number of drones increases. Scalability for multiple drones will be vital to ensure the application runs smoothly.

5.5 Supportability

Supportability: “combines the ability to extend the program (extensibility), adaptability, serviceability—these three attributes represent a more common term, maintainability—in addition, testability, compatibility, configurability, the ease with which a system can be installed, and the ease with which problems can be localized.” (1000sourcecodes.com, 2020)

The system should be able to allow for the easy addition of new drone types in the future. To achieve this the system should have a generic use of drone calls built independently of the individual drone APIs which would allow for easy integration of new APIs.

5.6 +

The “+” in FURPS+ can be used to represent any additional requirements or constraints on the software. (Ottinger, and Langr, 2009)

Security: Ideally, there would need to be a way to securely connect to the drones to avoid them being intercepted mid flight. As it stands with parrot drones all that is needed to connect to them is their IP address. Unfortunately implementation of such a security feature would require a firmware update to the drone itself, which is not possible under the conditions of access to the drones used on the project.

6. Similar Products

As the concept of delivery drones is an emerging technology with large business potential, there are a number of high profile projects in development currently (Buckley, 2016)(Gelinas, 2019). However there was no information to be found on the specific functionality of any system that would have similarities to this one. This may be due to the potential business value of such a system and a reluctance to share any valuable information.

References

1000sourcecodes.com, (2020), *Software Engineering-FURPS*, [online]Available at:https://www.1000sourcecodes.com/2012/05/software-engineering-furps.html[Accessed 23 November 2020]

Buckley, S. (2016), *Domino's starts delivering pizza by drone, but only in New Zealand,* Engadget, [online]Available at: https://www.engadget.com/2016-11-16-dominos-starts-delivering-pizza-by-drone-but-only-in-new-zeala.html[Accessed 1st Nov 2020]

Gelinas, J. (2019), *Look, up in the sky! It's my package. Amazon to start drone delivery 'within months',* Komando, [online]Available at: https://www.komando.com/shopping/look-up-in-the-sky-its-my-package-amazon-to-start-drone-delivery-within-months/571255/[Accessed 2nd Nov 2020]

Ottinger, T., and Langr, J., (2009), *FURPS+*, Agile in a Flash, [online]Available at:http://agileinaflash.blogspot.com/2009/04/furps.html[Accessed 23 November 2020]