



NYU

**TANDON SCHOOL
OF ENGINEERING**

Autonomous Drone Delivery System For Disaster Relief Management

Business, Requirements and Analysis Specification - Version 1.2

Document Number: Business, Requirements and Analysis Specification-003

• Team Number: C4

• Team Members:

Name	Email
Maxwell Aliapoulios	mma570@nyu.edu
Anuj Bakshi	aab735@nyu.edu
Rigved Kale	rrk320@nyu.edu
Jesse Lew	jl8429@nyu.edu
Zhongzhou Sun	zs941@nyu.edu

Revision History

Date	Version	Description of Change
3/11/2017	1.0	Initial document
4/3/2017	1.1	Added Requirements
4/17/2017	1.2	Added Analysis

Table of Contents

1	Introduction
1.1	Purpose
2	Scope
2.1	Identification
2.2	Bounds
2.3	Objectives
2.4	System Overview
3	Business Requirements
3.1	Technology
3.2	Economics
3.3	Regulatory and Legal
3.4	Market Considerations
3.5	Risk and Alternatives
3.6	Human Resources and Training
4	Context Diagram
5	Requirements
5.1	Functional Descriptive Requirements
5.2	Requirement Use Cases
5.3	Use Case Diagrams
5.4	Non-Functional Descriptive Requirements
6	Component Architecture
6.1	Component Descriptions
6.2	Component Architecture Diagram
6.3	Class Diagrams
6.4	Class Relationship/Interaction Diagrams
6.5	Events
6.6	Activity/State Section
6.7	State Logic
6.8	Behavior
7	Dictionaries
7.1	Class
7.2	Method
7.3	Attribute
7.4	Relationship
7.5	Key Events

This page intentionally left blank.

1. INTRODUCTION

1.1 Purpose

The purpose of this software is to save lives. To do so, we hope to provide aid organizations and governments with an alternative and effective option for disaster relief: an Autonomous Drone Delivery System (ADDS). Post-disaster, when roads are out and there are no other means of delivering water, food, medicine, clothing, and other supplies, we hope our software will be able to fill that role. This software would be best utilized by large fleets of drones. There is an emphasis on developing software for low cost drones with small hardware requirements so that they are dispensable. The software for the drone would be made proprietary in order to be able to be sold separately from the drone. This software would greatly increase the speed and scope in which any supplies could be delivered to an area, and many lives could be saved as a result.

2. SCOPE

2.1 Identification

This version of the ADDS is 1.2. Major changes to the software and documentation will be named X.0 (1.0, 2.0, 3.0, etc). Minor changes to the software and documentation will be named 1.X (1.0, 1.1, 1.2, etc).

2.2 Bounds

The software shall include both autonomous flight control and manual override options. The software shall include automatic movement, stabilization, and altitude control. The software shall include basic power management. The software shall make use of an onboard GPS for pathing and tracking. The software shall allow an onboard camera to be turned on, off, and rotated if possible. The software shall include radio communication for instructions and data.

The software shall be limited to specific hardware platforms and may not function on alternative hardware. However, one of our project goals is flexibility, so this software shall provide functionality for multiple common and inexpensive pieces of hardware. The hardware supported by this software is:

Flight Controller: XRacer F303, Naze32, Kiss FC

Altitude Control: Maxbotix XL-MaxSonar, LeddarTech LeddarOne

Camera: GoPro Hero, Amcrest AMC960HDC36-W

2.3 Objectives

As stated earlier, the main goal of this project is to provide aid organizations and governments with an effective option for disaster relief. Post-disaster, when roads are out and there are no other means of delivering water, food, medicine, clothing, and other supplies, we hope our software will be able to fill that role and save lives. We intend to accomplish this objective by preparing effective and versatile software that can be utilized to create fleets of autonomous delivery drones.

Project deliverables are as follows:

Deliverables	Dates
Project Proposal	02/15/2017
Project Business Specifications	03/15/2017
Project Requirements	04/05/2017
Project Analysis	04/19/2017
Presentation	05/03/2017

2.4 System Overview

After each drone lands (or when the system is first started), the drone needs to be physically brought over to a loading/charging station. It will ask the main terminal via radio for both a drop-off destination and a landing destination. The ADDS software will allow different GPS destinations to be set for each drone and will keep track of them in the main terminal. If GPS is down, simple flight plans based on distance and direction will be uploaded to the drones.

At this point, the drone can be loaded with its supplies. The drone will send its current power/battery charge data to the main terminal. The weight of the package will also be requested by the main terminal. The main terminal will calculate whether the drone has enough power to make the entire flight. If it has enough power, it will prompt the main terminal user (the ‘drone orchestrator’) to approve the flight.

Once this prompt is issued, all charging wires need to be disconnected from the drone. The orchestrator will give flight approval to the ADDS and the drone will take off on its own and calculate a path to its drop-off destination. It will release its package when it arrives and then calculate and head to its landing destination. It will be able to be tracked and will remain in communication with the main terminal throughout its delivery using radio signals. If it goes out of range, the main terminal will still have a copy of its flight plan and can predict the timing of its expected drop-off and expected return to radio signal range.

If the drone does not have enough power to make it to the destination, an appropriate indicator will be displayed and either some of the weight of the package will need to be removed or a new destination will be requested.

If complications occur during the flight, a fall back controller onboard the drone will automatically stop the drone, calculate a trajectory to the landing site if GPS is down, and head

toward the landing site. Manual control can be implemented on a secondary terminal at any time as long as there is communication via radio.

Last but not least, a software maintenance team will be installing the software on the terminals and the drones. They will be running various pre-written tests to make sure that all parts of the software are functioning properly and communicating.

3. BUSINESS REQUIREMENTS

3.1 Technology

The popularization of “hackable” quadcopters, and other propellerized drones, incentivizes this software to be highly adaptable. This means that the hardware utilized will be highly dependent on the use case and operator. The firmware or drone orchestration software will be highly feature adaptable. If the user would like to utilize, for example, the fall back manual control setting then their drone hardware must have the minimum required hardware for that feature, i.e. an antenna and a camera on board.

Some of the hardware that the software features will utilize is the following:

- basic quadcopter with a frame, propellers, motors, legs, landing pads, GPS, and a motherboard is the bare minimum.
- Another required feature is the main computer to install the orchestration software on with access to the internet or GPS satellite link.
- An optional feature can make use of an onboard camera.
- An optional feature can make use of an antenna and receiver for a second layer of communication.

Impact of the new software to governments in high natural disaster areas will be a low cost solution to efficiently delivering disaster relief supplies in hard to reach areas. This will allow governments and organizations to save money on the delivery process, and therefore provide more supplies to those in need. The low cost of the bare minimum hardware requirements makes it so that many drones can be part of a fleet and therefore deliver to more people in larger areas.

3.2 Economics

The economics of the project are macroscopic as well as microscopic in nature. There shall be various upgradable plans like:

1. **Basic** - The software’s basic functionality will include pick up/drop locations for medical supplies, food, water etc.
2. **Premium** - Other features will be included like landscape reporting using computer vision technologies or image processing, information broadcast etc. in addition to the basic features.

- The upgradable plans will make the software flexible with respect to the available services provided by various drones in the market.
- Ease of flexibility and integration techniques will make it simpler to attract customers, potential investors and venture capitalists.

- The software banks upon intangible assets like time and energy and tangible assets like human lives and money in times of crisis.

3.3 Regulatory and Legal

The main regulatory and legal issues will revolve around airspace laws. In the United States, the Federal Airspace Administration (FAA) is responsible for keeping US airspace safe, efficient and under control. Therefore, for the proposed drone software solution would need to be in full compliance with any of the FAA's regulatory requirements.

That being said, the FAA may be quick to comply with any disaster relief activity. The drone operation would propose a compliance documentation before any drone disaster relief activity. This implementation would ensure that the proposition notifies all relevant regulatory parties are aware of the airspace activity without blocking any aid or disaster relief.

This proposal model can be implemented with other airspace regulatory institutions in other countries as well. The assumption and intuition here being that so long as the drone activity is abiding by airspace laws, and the regulatory institution is notified, then the relief will be appreciated and complied.

3.4 Market Considerations

The software is primarily targeted towards disaster relief management organizations which shall be governments or NGO's. Other customers could be research groups, technology enthusiasts, students etc. Considering the increasing popularity of unmanned vehicles, drones have become the new age toys as were remote controlled cars once (ie, quadcopters). The software shall thus be considered for secondary activities as well.

3.5 Risks and Alternatives

Business Risk:	Dependant on companies with same business model in market.
Probability:	Low
How discovered:	Analysis of competition in market
Responsible Party Status:	Development and Marketing teams
Mitigation Plan:	Develop clean software with minimum bugs. Aggressively market the software as a reliable software.
Operational Risk:	Dependant on types of Drones and their distinguishing features
Probability:	Low
How discovered:	Each drone is different from the other. Part of the software required for each individual customization
Responsible Party Status:	On-site team Technology team
Mitigation Plan:	Training of on-site tech team enabling proper installation of software
Technology Risk:	Dependant on functioning of software as desired.
Probability:	High
How discovered:	Analysis of product and varieties of Drones available
Responsible Party Status:	Software developers
Mitigation Plan:	Intensive Unit and Integration testing of software.
Economic Risk:	Dependant on cost due to untimely development of software.
Probability:	Medium
How discovered:	Time vs cost graph in design and implementation section of Software development lifecycle model.
Responsible Party Status:	Software development team
Mitigation Plan:	Conducting regular audits and monitoring the productivity

3.6 Human Resources and Training

Initially:

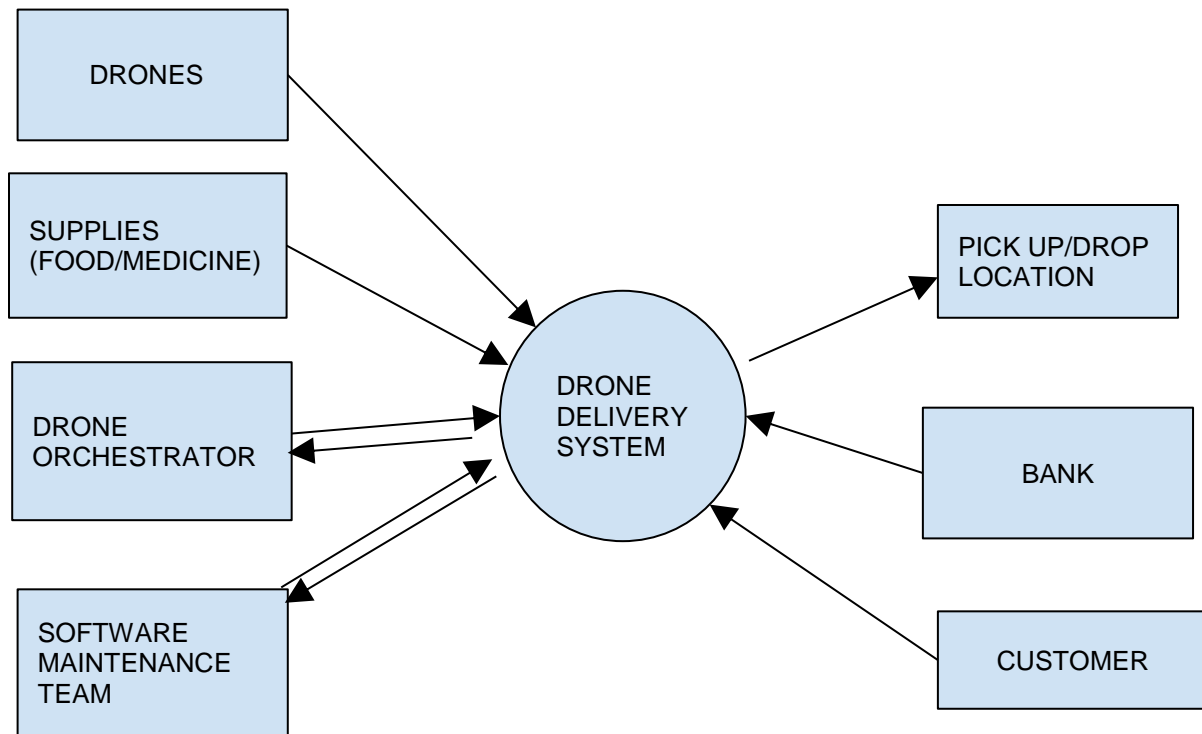
- Training shall only be required for unit testing. Testers shall be trained to unit test the software in the development phase.

Later on:

- Training is required for 'on-site' setup team for installation of software on drones. Considering variety of specifications of each drone, training for enabling and disabling features is also required.
- Training is also required for marketing team for them to understand the exact functioning and features of working of the software.

4. CONTEXT DIAGRAM

High-level context diagram identifying system boundaries



5. REQUIREMENTS

5.1 Functional Descriptive Requirements

The following are the functional descriptive requirements of the system:

Section ID	Requirement Description
5.1.1	The system shall allow the drone orchestrator to set the pick-up/drop location.
5.1.2	The system shall allow drones to autonomously listen to orchestration flight controls.
5.1.3	The system shall receive status feedback from the drones.
5.1.4	The system shall allow to manually configure the flight path with regards to direction and altitude.
5.1.5	The system shall allow software maintenance to update the functionality and debug the drones.
5.1.6	The system shall allow payments to be taken.

5.1.1

The drone orchestrator shall have the ability to control the location that the fleet picks-up and drops off the material. The location can be set through relative positioning. The material can be anything from medicinal supplies to water.

5.1.2

This takes the work away from the operator from having to decide the optimal flight path and instead this calculation and decision is performed by the fleet's onboard software.

5.1.3

This allows knowledge of when a drone in the fleet needs an update. The orchestrator must also know when a drone in the fleet crashes, receives an error, malfunctions or any other issue. This allows the orchestrator to account for every piece of hardware and delivery status.

5.1.4

The drone orchestrator and the software maintenance team shall be to control the directions and the altitude of the drones manually. It can be one drone or multiple drones. In the event that the drones need to be controlled at a specific level, this requirement gives the ability to manually decide what drone does what.

5.1.5

The system shall allow to process payments when basic software is provided or software upgrades take place.

5.1.6

The basic version of the software shall have pick-up/drop location input using a preloaded map and onboard sensors. The upgraded versions will have GPS, Camera, and WiFi functionalities. The software shall have the ability to update itself in accord with the customer's requirements.

5.2 Requirement Use Cases

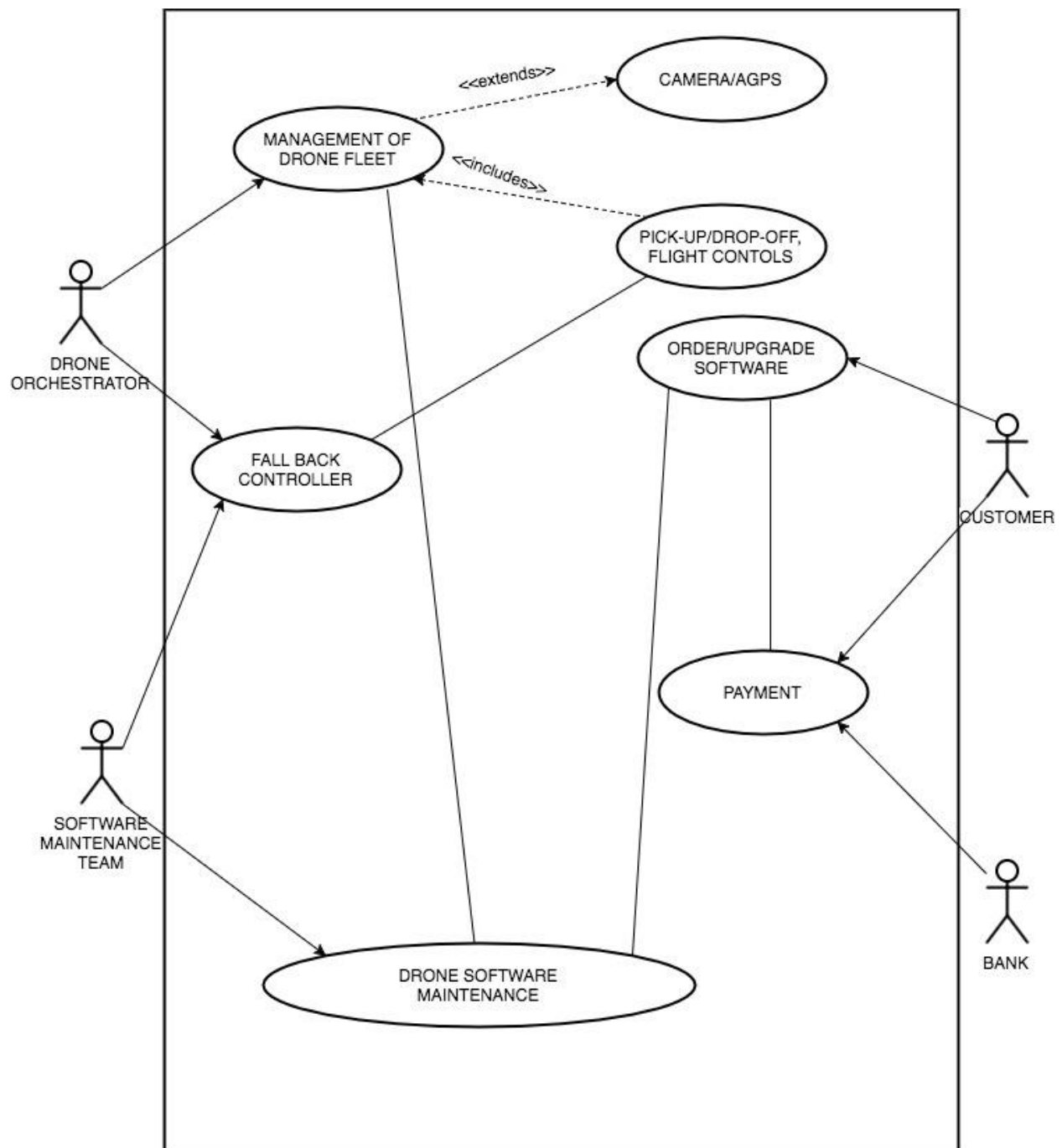
Following are the actors in the Autonomous Drone Delivery System.

- Drone orchestrator: The Drone orchestrator enables the setting up of pickup and drop-off locations for delivering material for all drones. Additional functionalities of drone such as Camera and GPS are managed by Drone orchestrator. Manages several drones with the help of Drone fleet management system. The Drone orchestrator also receives feedback from fall back controller enabling the correct assignment of drones.
- Software Maintenance team: The team receives feedback from fall back controller. The team modifies and tests software accordingly.
- Customer: The customer orders for new copies or upgradation of existing software copies. The customer also initiates the payment.
- Bank: Bank or any third party organization verifies and completes the transaction between customer and the system.

Following are the use cases used in Autonomous Drone Delivery System.

- Management of drone fleet: The Drone orchestrator lists the locations and assigns the number of drones required to be sent to specific locations. Additional functionalities such as cameras and GPS are also controlled using this system. It also checks with Drone software maintenance system whether a flaw has been corrected.
- Set Pick-up/Drop-off: The Drone orchestrator sets and monitors the drones with respect to their pickup and drop-off locations.
- Fall Back Controller: Software maintenance team receives feedback from fall back controller. It enables the software maintenance team to know about possible warnings and errors in the software. It also notifies the error messages to Drone orchestrator in order to better assist the orchestrator with assignment of drones.
- Software maintenance team: The software maintenance team modifies and tests the software in order to avoid bugs or errors occurring in the software. It is also responsible for repairs and emergency situations. It also provides a feedback to Management of drone fleet and notifies the customer if an upgrade is required on the software.
- Order/Upgrade Software: The customer selects the quantity of software copies and the version of the software to be ordered or upgraded in accordance with the number of drones.
- Payment: The customer makes payment for the software through a third party organization such as a bank which verifies and confirms payment.

5.3 Use Case Diagram



Use Case Descriptions

MANAGEMENT OF DRONE FLEET	
Description	The drone orchestrator uses this system to manage several drones together.
Actors	Drone Orchestrator
Pre Conditions	Multiple drones to be sent at multiple locations.
Basic Flow	<ol style="list-style-type: none">1. Locations are selected where supplies need to be delivered or picked up from.2. Drones are assigned to locations based on Drone's ID nos.3. Cameras and other additional functions are controlled during the flight.4. Receives notification on availability of drone or modified version of software from Drone software maintenance.
Alternative Flow	NA
Post Conditions	The system shall update assignment information and deselect drones if an error occurred during the flight run.
Special Requirements	None
Extension Points	Controlling and monitoring additional functional features such as camera and GPS.

PICK-UP / DROP-OFF, FLIGHT CONTROLS	
Description	The Drone Orchestrator sets the pickup and drop-off locations for the drones. Supplies need to be either picked up or dropped off to particular locations.
Actors	Drone Orchestrator
Pre Conditions	There is a need for transportation of supplies.
Basic Flow	<ol style="list-style-type: none">1. Pickup location is set in order to pickup supplies.2. Drop-off location is set for transporting the supplies.3. On assignment of drones are monitored in-flight.
Alternative Flow	<ol style="list-style-type: none">1. Drop-off location is set if supplies are already present in the current location.2. Drones are monitored in-flight.
Post Conditions	Drones shall be assigned to the respective locations set.
Special Requirements	None
Extension Points	None

FALL BACK CONTROLLER	
Description	Provides feedback to Drone Orchestrator in case of an error so that drone is not assigned a location until it is repaired. Software maintenance team is also provided a feedback of flight run.
Actors	Drone Orchestrator, Software maintenance team
Pre Conditions	The drone shall undergo a flight run.
Basic Flow	<ol style="list-style-type: none">1. An error is encountered during the flight run.2. Drone Orchestrator is notified.3. A report is sent to the software maintenance team.
Alternative Flow	<ol style="list-style-type: none">1. A warning is encountered during the flight run.2. Drone Orchestrator is notified.3. A report is sent to the software maintenance team.
Post Conditions	The software shall be modified and tested to resolve such warnings or errors.
Special Requirements	None
Extension Points	None

SOFTWARE MAINTENANCE TEAM	
Description	Software Maintenance is done for ease of system maintenance, software version installation, repairs, emergency cases etc.
Actors	Software Maintenance Team
Pre Conditions	A fleet of drones in operation.
Basic Flow	1. The Order/Upgrade Software provides software version information. 2. The drone fleet provides information about ongoing activities 3. The software maintenance team handles cases of discrepancies
Alternative Flow	N/A
Post Conditions	After fixing the drone discrepancies, that drone is kept under observation for a small period of time.
Special Requirements	None
Extension Points	None

ORDER / UPGRADE SOFTWARE	
Description	The customer orders the software version or if he/she already has the basic version installed he/she will purchase the upgraded version.
Actors	Customer
Pre Conditions	Upgrade option is only available if the customer has already purchased a version of the software beforehand. No pre conditions for a first time buyer.
Basic Flow	<ol style="list-style-type: none">1. The customer orders a version of the software2. Information like name of organization, email, etc. is obtained3. The payment information is made available to the customer
Alternative Flow	<ol style="list-style-type: none">1. The customer upgrades a previous version2. The payment information is made available to the customer
Post Conditions	The customer information is stored in the system.
Special Requirements	None
Extension Points	None

PAYMENT	
Description	The customer makes payment for the software through a bank which verifies and confirms payment.
Actors	Bank and Customer
Pre Conditions	The customer shall need to complete the Order/Upgrade Software step.
Basic Flow	1. The system generates a receipt with regards to the version. 2. The payment is made through debit/credit card/cheque. 3. The authentication of the payment is done by the Bank.
Alternative Flow	NA
Post Conditions	The payment information is stored in the system.
Special Requirements	None
Extension Points	None

5.4 Non-Functional Descriptive Requirements

System Capabilities, conditions, and constraints

The system shall be able to connect to applied drones using both radio and WiFi connection.

The system shall be able to handle and process requests both automatically and manually.

The system shall support manual control through a main terminal.

The system shall provide concurrency across all terminals.

The system shall support a satellite connection while GPS is used.

The system shall provide security for all messages sent to and from the drones.

Physical Resource Requirements

Equipment: A computer shall be essential for the team. The system needs a computer to install the software with an uninterrupted power supply and network connectivity.

Insurance: Used for protecting businesses from losses due to any type of events that can occur during the normal course of the business including insurance for property damage, legal liability, and employee-related risks.

Network: Enterprise level local area network is recommended for the server.

Drones: Drones are needed. They shall contact with the main terminal.

Computer Hardware Requirements

Processor: Multicore processors that are each 3GHz or higher.

RAM: 4GB or higher.

HDD: Preferred to install on system where RAID is preconfigured.

Database: 1TB database space.

Display: 1024x768 to 1920x1080 shall be supported.

Network: 100 KBps connection minimum between clients and servers.

Computer Hardware Resource Requirements

Keyboard and mouse shall be used to input instructions for drones.

Monitor shall be used for displaying pictures taken by drones.

Radio transmitter hardware shall be included in our system, used for communication.

Computer Software Requirements

Operating system: Our software prefers UNIX based operating system on the server front comparing with Windows because it is more flexible on most machines, more stable, and possesses much greater processing power than Windows. However, Windows Server OS can also be supported for our software to meet demands of particular drones.

Flight Control Software: This depends on the Flight Controller used on different drones:

XRacer F303: open source Cleanflight software

Naze32: open source BaseFlight software

Kiss FC: open source MinimOSD, Arduino OSD software

Database: Relational database such as MySQL and Oracle are supported for this system, because QPS at level of thousand per second can meet the requirement. Our system is highly CPU bound for calculating newest GPS data rather than memory bound and without much scalability and transaction requirement, SQL > noSQL.

Computer Communications Requirements

Either Internet connection or Radio communication shall be provided and guaranteed for our system.

Radio communication shall be supported between modern drones and the software system.

Transmitter and receiver need to be tuned to the same frequency and are paired using an RFID (radio frequency identification).

WiFi shall also be supported by the software to broadcast video to the main computer.

Remote control shall be done through either WiFi or radio communication.

GPS shall be supported for ping satellites for location data from devices and pre-programming routes. Drone can be cut loose and fly in sequence to each of the GPS locations identified.

Environmental Conditions

Climate: Drone delivery is highly dependent on climate changes. Our software and manual control shall work under normal conditions, without rain or wind greater than 10 mph.

System Performance characteristics

High throughput: The system shall be able to support high throughput for large number of queries per second through different clients as drones or materials.

Parallel processing: The system shall be able to process requests from clients parallelly.

Data integrity: The system shall be able to guarantee data integrity, TCP is used for most data transmission to provide reconnecting and handshakes.

Availability: Our system shall be minimize any crash or downtime because it is a life saving system and shall provide good availability to avoid possible accidents.

Security: Our system shall be secure enough to avoid unauthorized access from hackers.

Safety Requirements

The software shall be in full compliance with any of the Federal Airspace Administration's (FAA) regulatory requirements to keep all the routing and operation of the drones safe enough to fly.

Security and Privacy Requirements

Radio and WiFi communication shall be kept safe between drones and our software to avoid unauthorized access or damage to our system.

WiFi Security is guaranteed by WiFi Protected Access (WPA).

System Human Interfaces

Main terminal shall provide user interface which can be accessed by management teams. This terminal can be opened in different PC installed this software simultaneously or controlling different drones without interfering with each other.

User interface shall include live video for manual control of specific drones besides those essential instruction like drop off or land at certain destination.

Software maintenance team shall be provided particular human interfaces for management, this interface is also shared by software developing and test engineer which requires particular permission.

System Maintainability

Software maintenance team shall install the software on the terminals and the drones. They shall run various pre-written tests to make sure that all parts of the software are functioning properly and communicating.

The frequency of doing maintenance testing depends on the feedback of the system.

If new features are added to the software, test scripts shall be revised and new feature tests will be added.

System Quality Factors

Functionality: Basic functionality of the software shall include autonomous and manual flight controls.

Reliability: System reliability should be guaranteed in both hardware and software, otherwise extra costs will be needed. Reliability should be addressed during the design stage and continually reevaluated throughout the testing process.

Maintainability: The system shall be maintainable, with pre-written tests for all hardware. The system shall support any number of drones.

Portability: Our software shall be portable and can be installed on various operating systems and drones, which makes it highly flexible.

Design and Construction Constraints

Different hardware is needed for developing and testing of our system.

Communication with FAA and satellites.

Budgets needed for developing and testing.

Specific performance requirements for specific hardware.

Completion date.

Other constraints include legal, environmental, social or third party constraints.

Life Cycle Model

The spiral model shall be used. It will begin with risk management and prototypes, then design, coding, integration, testing, and implementation. We will keep maintaining and adding new features to our system iteratively. Agile development may be used after releasing several versions of our products and grow big enough for our team.

Policies and standards - Methods, tools, and techniques

The software shall fulfill all use requirements.

The software shall try to protect the hardware (drones) it is installed on.

The software shall work on multiple platforms/types of hardware to provide versatility.

Personnel-Related Requirements

Engineering team: Include Software development engineer: Write developing docs, coding and help testing the software. Junior and Senior software engineer should work together to finish this; Software test engineer: Write test cases and do unit, integrated and performance tests;

Hardware engineer: Mainly for testing compatibility and help find solution for it.

Marketing team: Consider about marketing risks, do market research, get feedback from customers and communicate with developing team to add new features to software.

Financial team: Responsible for analyzing and calculating all financial details and cost for developing, testing, and marketing research.

Software maintenance team: Responsible for installation and debugging of on-site software.

Responsible for testing the drone software after each delivery and moving the drones back to the loading site if they pass all tests or to the maintenance/repair area if the drone fails any tests.

Training-Related Requirements

Unit and integrated test training is required for the software. Software test engineer should be recruited and trained to be familiar with the software during the development process and test all the functions by designed test cases.

On-site setup team should be trained for installation of software and enable and disable features for different drones. Different training teams can be trained targeting different drone requirement.

Marketing team should be trained for understanding features of the software.

Every team should have trainer and should collaborate with software development engineer to finish the training for new employees.

Logistics-Related Requirements

The drones shall be transported from the landing site to their loading site where they shall be recharged. This shall be done by the maintenance team.

The landing site and loading site shall be in close proximity to maximize delivery time.

The loading site shall have access to electrical outlets or generators for recharging the drones.

Packaging Requirements

The software shall be packaged as 2 downloadable .exe files off a secure website (one file for the orchestrator console and one file for each drone).

The software shall require a one-time registered user activation code provided via automated email after purchase before it can be downloaded.

Precedence and Criticality Requirements

Due to the high cost of a fleet of drones, the development of the Fall Back Controller will have the highest precedence to attempt to save as much hardware as possible should any issues arise. Safety requirements and Security and Privacy requirements will also take precedence over other parts of the system to prevent the drones from being hacked.

All other parts of the software have the same precedence/criticality.

Other non-functional Requirements

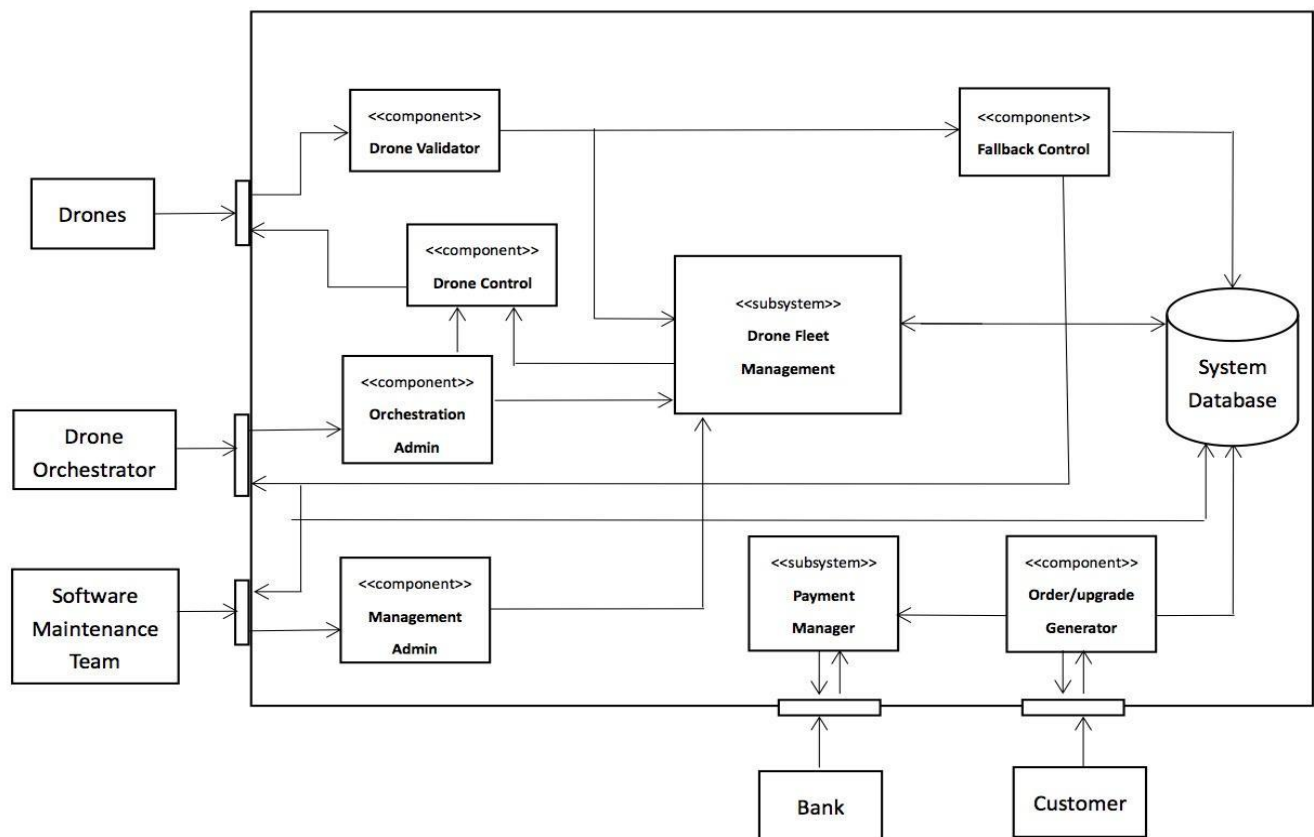
There are no other non-functional requirements.

6. COMPONENT ARCHITECTURE

Component (Component/Package/Subsystem) Architecture

6.1 Component Descriptions

High level component architecture diagram below:



6.2 Component Architecture Diagram

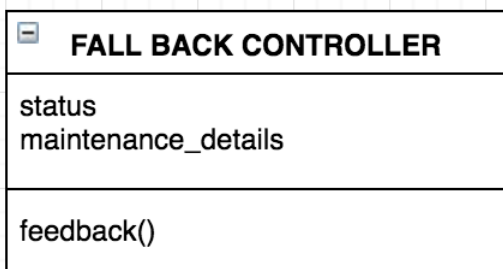
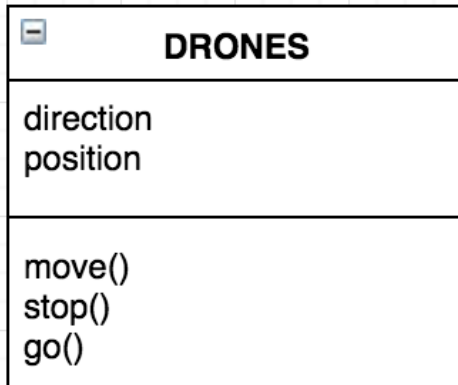
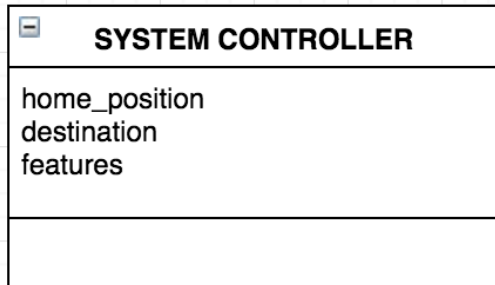
Identify and describe the functionality each system component or subsystem


- **Drone Validator:** Responsible for validating every input signal and response from a drone that is to be considered part of the fleet. This component ensures that the only communication that goes comes into the system is from a drone that is considered a valid part of the fleet.
- **Drone Control:** This component is responsible for reading, processing and distributing drone controls to the fleet from the orchestration administration. It also receives input from the drone fleet management subsystem because it needs to be aware of inputs from the drone's themselves.
- **Orchestration Admin:** This administration component allows the orchestrator to dictate control commands that are received by the drone control component. This is the main interface between what movements/actions the drones make and the actual orchestrator.
- **Management Admin:** This administration component allows the software maintenance team to perform any management operations involving the fleet. This involves taking care of drone status, and being aware of certain tasks that need to be performed.
- **Drone Fleet Management:** The subsystem that is responsible for managing all the drones in the fleet. It does this by tracking the status of each drone, and passing input to any receiving components about drone feedback and orchestration controls. It has connections to the validator so that all potential drone communication passes to it but through the validator first. It also has a connection to the orchestration admin so that the orchestrator can have full visibility and control into the drone fleet management subsystem.
- **Fall Back Controller:** This controller component allows the both the drone orchestrator and software maintenance actors to take control of a drone in the event that manually control is required. It has a connection to the system database so that the drone fleet management system can be aware of any fall back issues and use that information when calculating any fleet management decisions.
- **Payment Manager:** This will be an entire subsystem dedicated to maintaining the payment operations with the bank and the order/upgrade generator. It will know who has paid for what, what kind of payment to process with the banks, and will be responsible for any monetary information.


- **Order/upgrade Generator:** This component generates orders and upgrades so that the entire system can be aware of which features should be dictated to who and so that the payment manager knows who to chart for what.


6.3 Class Diagrams

The following are the Individual Class Diagrams with attributes and methods:



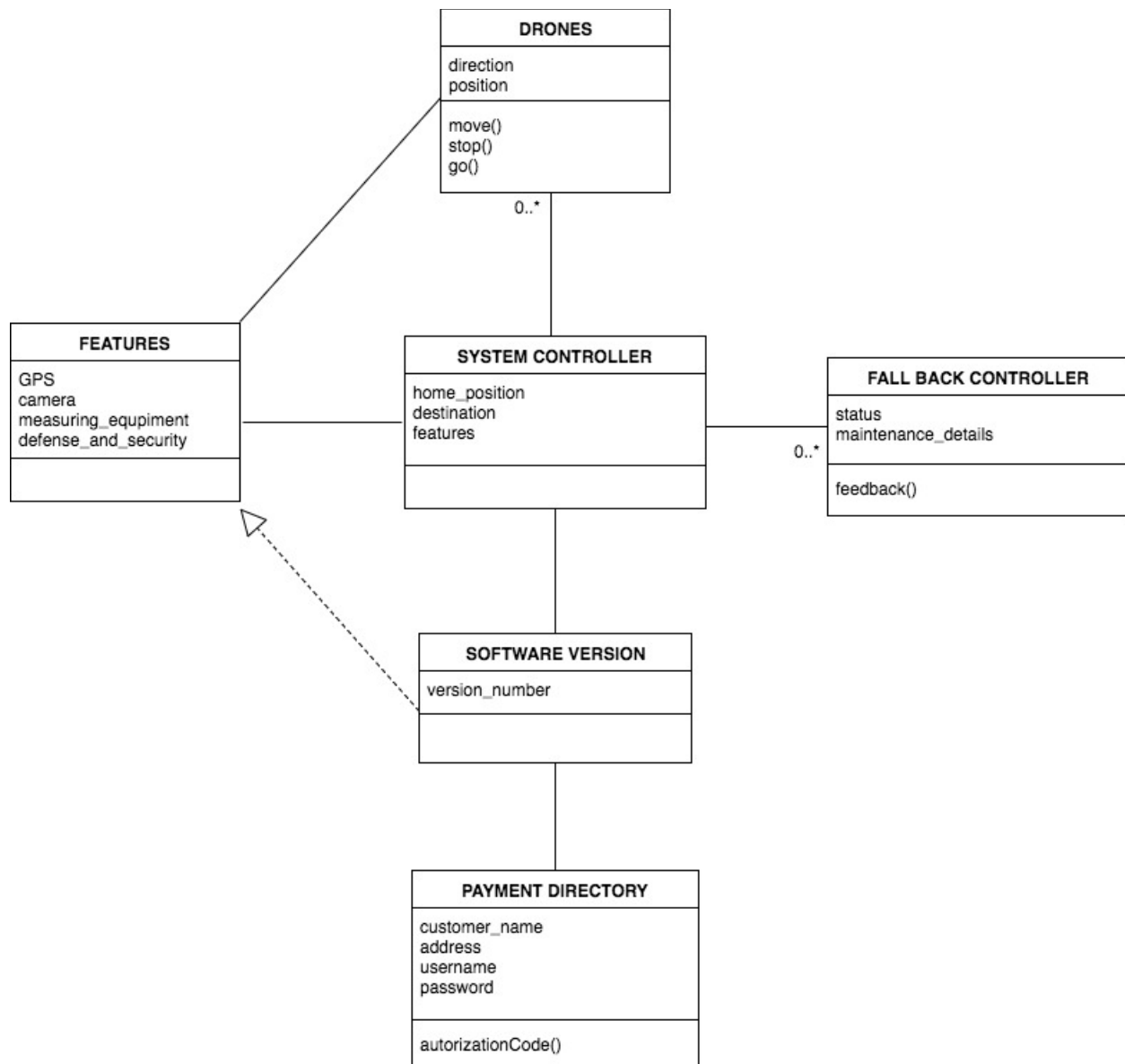
 FEATURES
GPS camera measuring_equipiment defense_and_security

 PAYMENT DIRECTORY
customer_name address username password
authorizationCode()

 SOFTWARE VERSION
version_number

6.4 Class Relationship/Interaction Diagrams

Class Interaction Diagrams



6.5 Events

Following are the events to which the system responds :-

Drone Validator:

- Checks if inputs and responses are valid or not.
- Alerts fallback controller if input or response is invalid.

Drone Control:

- Processes and distributes commands to the fleet.

Orchestration Admin:

- Enables Orchestrator to dictate commands to the drone.

Management Admin:

- Checks drone status
- Changes drone status on repair.

Fall Back Controller:

- Enables Orchestrator to manually control the drone
- Informs Drone Orchestrator if any issues arise which need to be handled.
- Informs Software maintenance team of issues that arise during flight.
- Conveys Drones fixed status back to Drone Orchestrator.

Order/Upgrade Generator:

- Receives orders from the customer.
- Checks if Customer information and version of software purchased is present in database.
- Updates Database if the customer is new or new software version is to be installed.

Payment Details and Check:

- Provides payment details of the customer to the bank.
- Enables bank to validate customer's payment credentials and perform successful transaction.

6.5.1 Motives

Drone Validator:

A very important task of validating inputs and responses is handled by the Drone validator. It makes sure that the validity of both inputs and responses is maintained which otherwise may lead to errors. In case of errors the fallback controller is alerted.

Drone Control:

The Drone Orchestrator gives commands to the entire fleet. These commands must be distributed to all drones in the fleet as well as processed by each drone. Therefore Drone control ensures these commands are carried out properly.

Orchestration Admin:

The Drone Orchestrator is provided with an interface through which it can communicate with the drones to perform certain actions. The Orchestration admin enables this communication from the Orchestrator to the drone control system.

Management Admin:

This event constantly monitors the status of a drone. In the process of repair a drone is removed from the directory of available fleets and assigned a status of not available. As soon as a drone is repaired the management admin changes the status of drone to available. Therefore, the main motive is to keep a track of the availability of drones.

Fall Back Controller:

The FallBack control is used to inform the Drone Orchestrator and Maintenance team of unusual or unexpected occurrences leading to crashes, bugs and errors. It is also used to enable manual control to make sure the drones are not lost in cases of errors in automation.

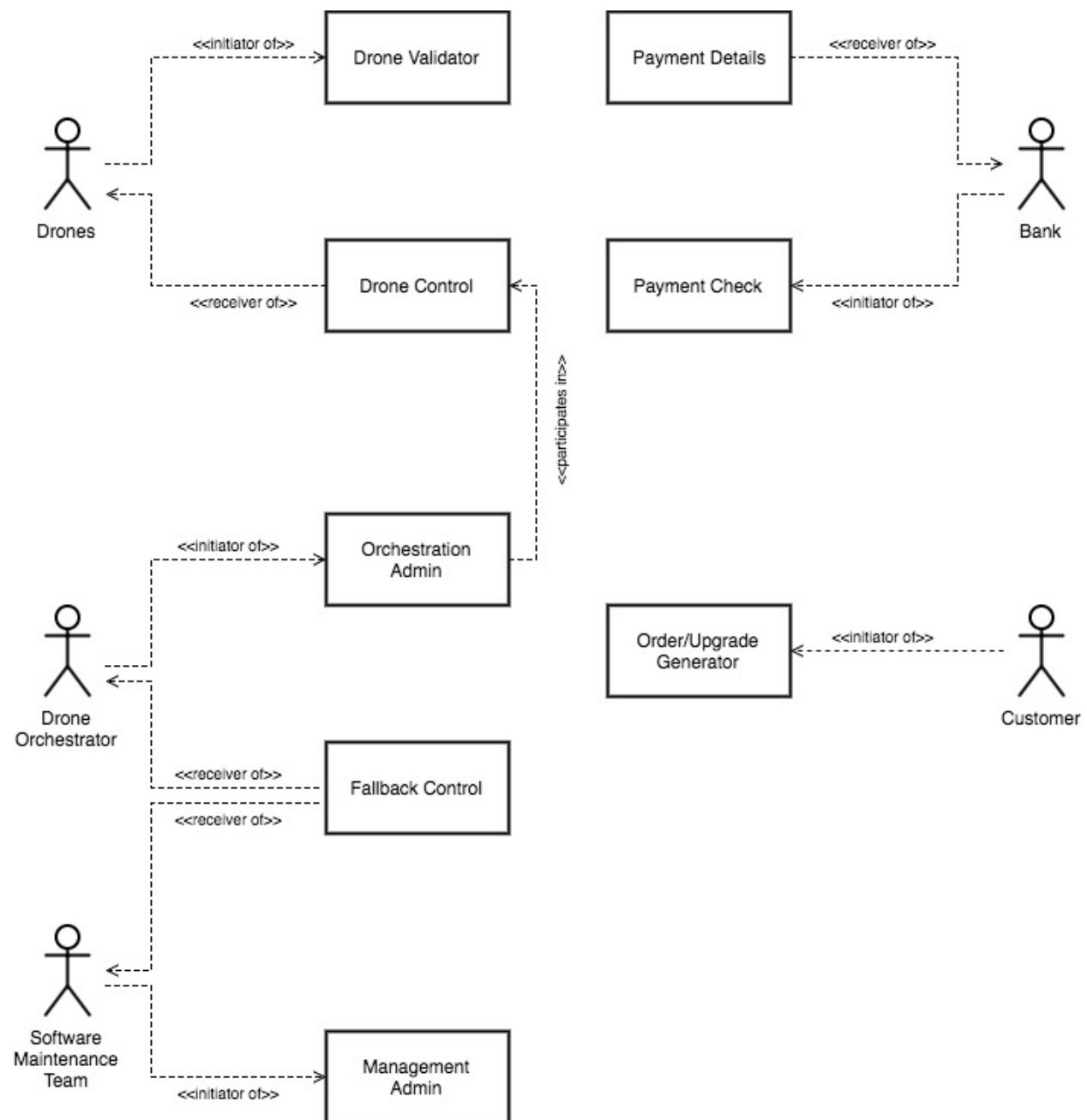
Order/upgrade Generator:

This event enables a customer to place an order for new or upgrade of software. Once payment has been successfully made the generator also updates the database with new details on either software version or user or both.

Payment details and check:

The payment details of the customer are provided to the bank so that it can carry out the transaction successfully. This ensures that there is no fraud with respect to payments. The payment process is therefore carried out successfully before granting the customer an access to the software.

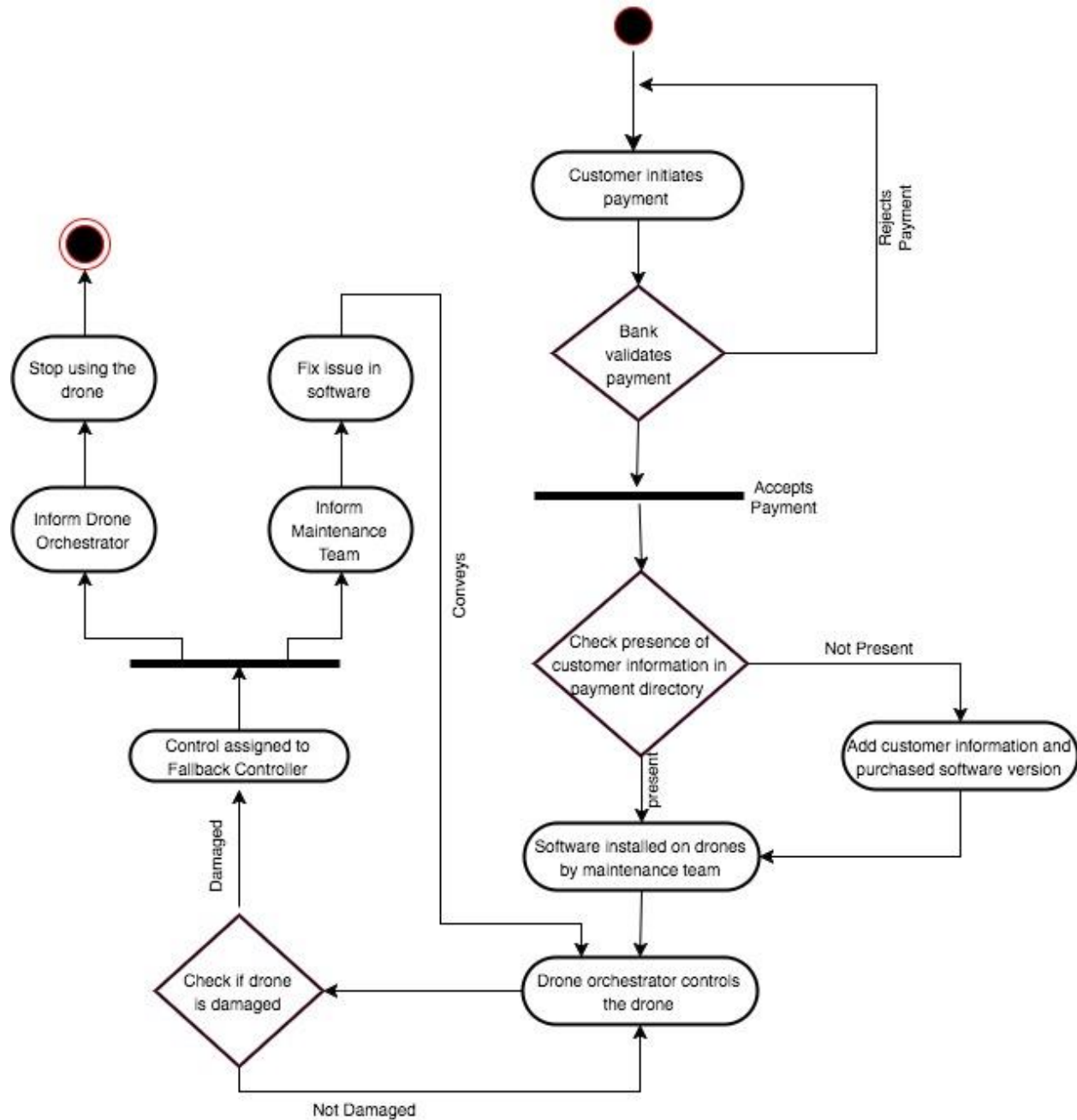
6.5.2 Event Diagrams



6.6 Activity/State Section

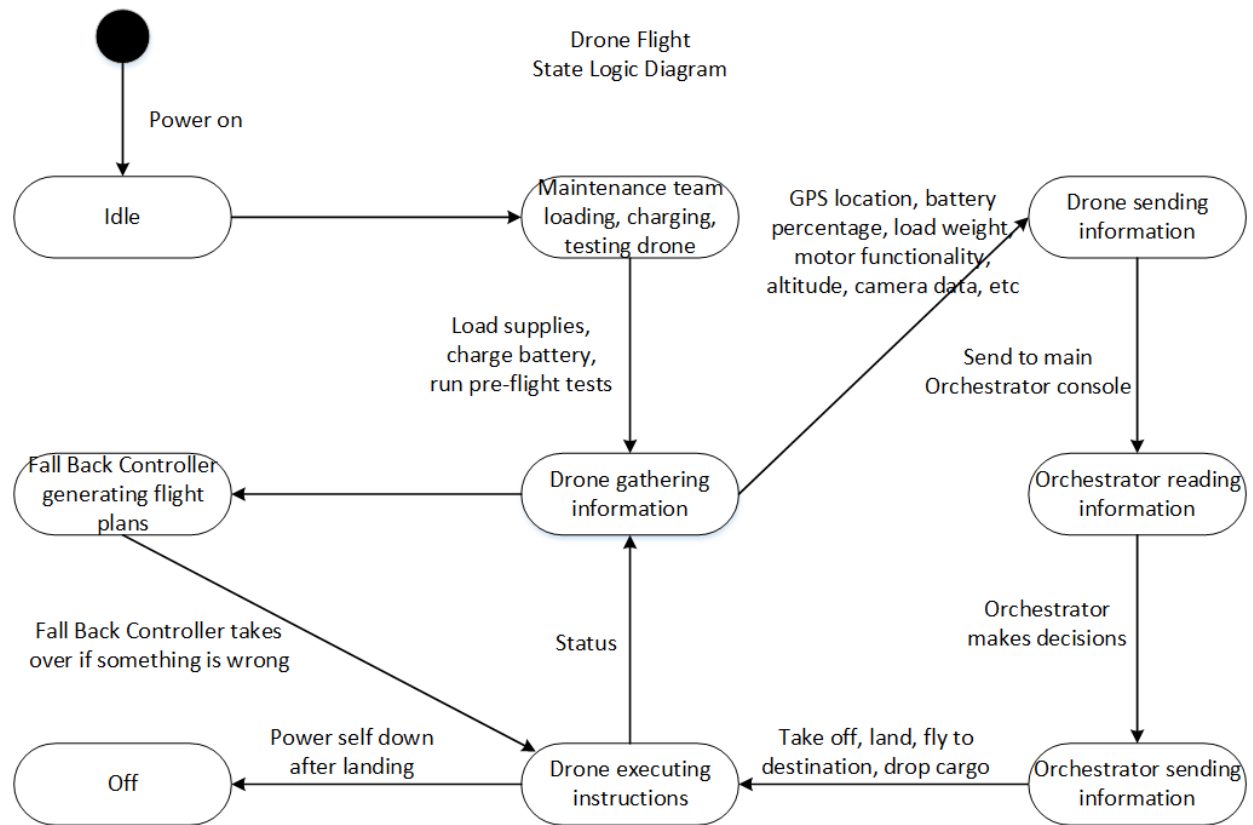
A scenario (workflow) diagram will be provided for each scenario

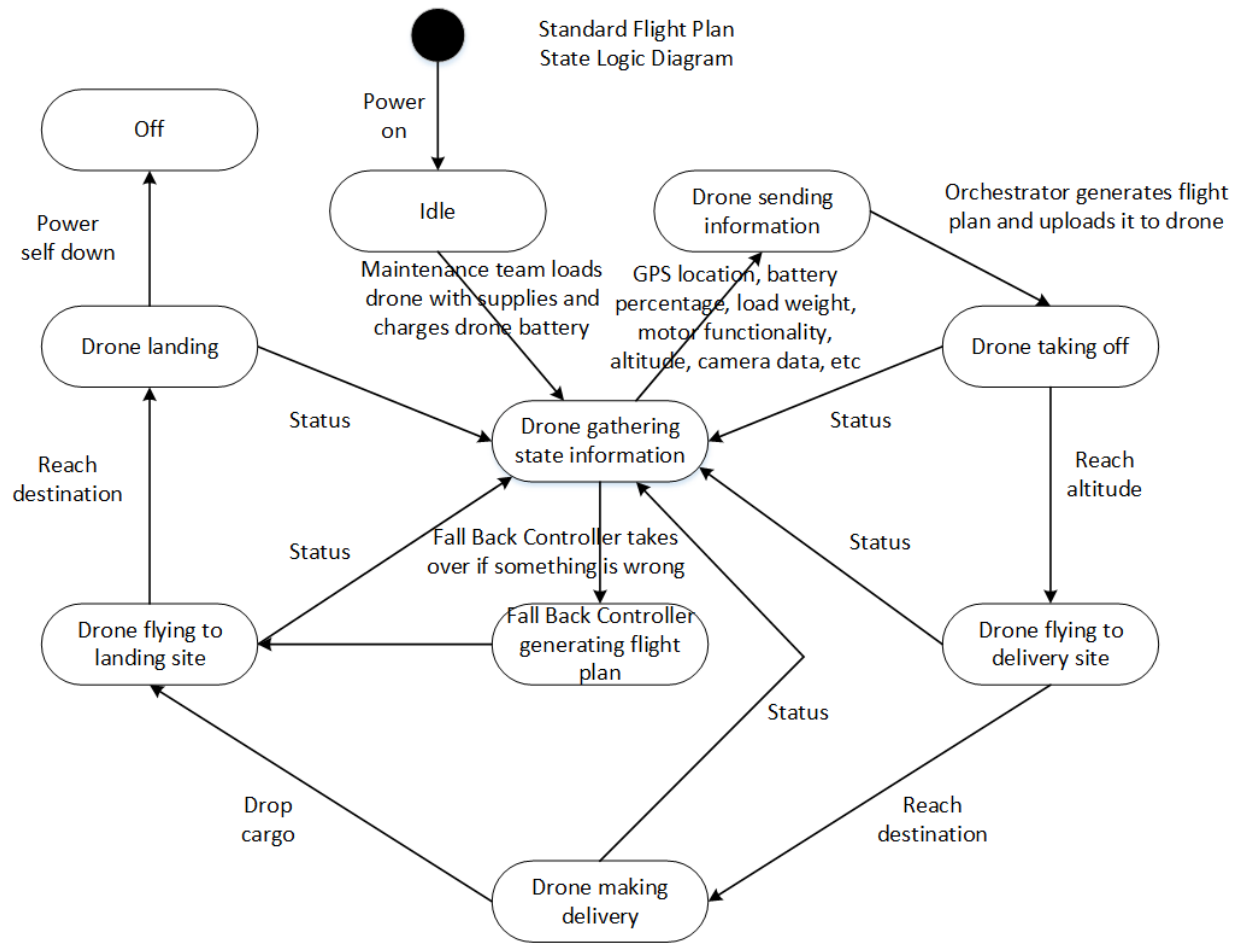
This section is started in this Analysis document and details are refined in the Design document



6.7 State Logic

Started in analysis and completed in design.



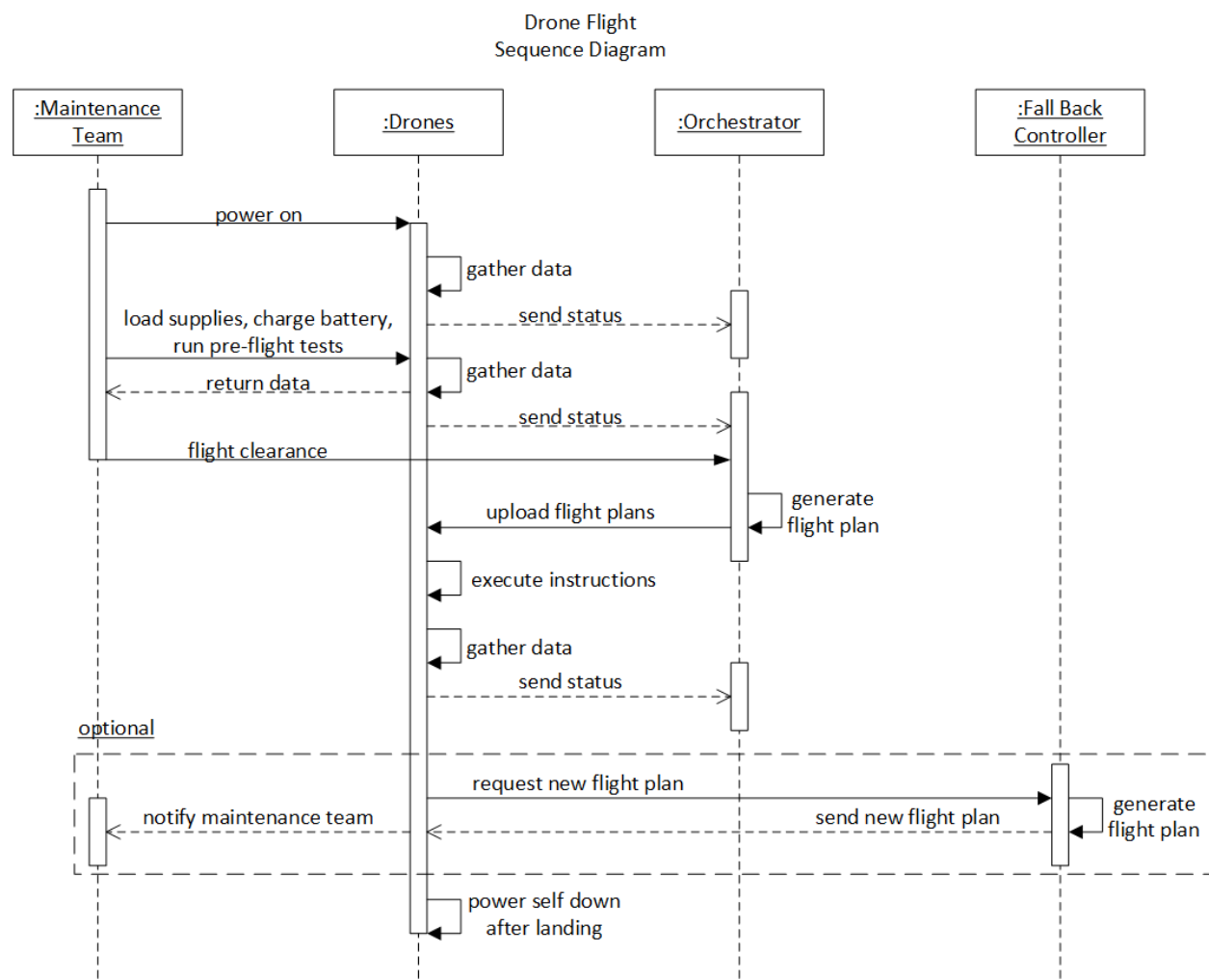


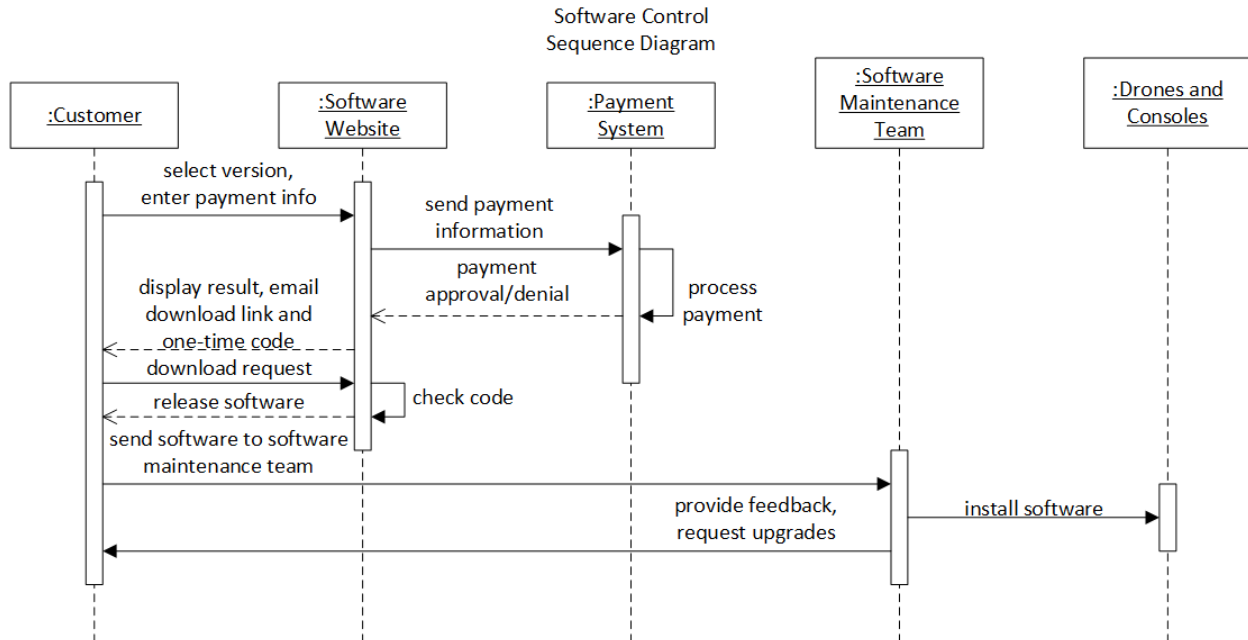
6.8 Behavior

6.8.1 Sequence Diagrams

A sequence diagram describes, for a particular scenario or use case, the events that external actors generate in the correct order and possible inter-system events. The analysis document starts the description of the high level sequence diagrams which are then refined and completed in the design phase.

The sequence diagrams for some of the important use cases are shown below:





6.8.2 Collaboration Diagrams

To be completed in design phase.

7. DICTIONARIES

This is the initial set of dictionaries.

They will be completed (refined) in the design phase.

7.1 Classes

Class Name	Description	Attributes	Method
System Controller	Contains system level control data	home_position destination features	
Drones	Represents functionality of the drone	direction position	move() stop() go()
Fall Back Controller	Contains fall back data and status of the drone	status maintenance_details	feedback()
Payment Directory	Encapsulation of payment details	customer_name address username password payment	softwareVersion() authorizationCode()
Features	Extended features of drones	GPS camera measuring_equipment defense_and_security	
Software version	Version of the latest software	version_number	

7.2 Methods

Method name	Description	Class	Arguments
move()	Setting moving details of a drone	Drones	starting point ending point
stop()	Stop a drone	Drones	
go()	Start a drone	Drones	
feedback()	Create a fall back report	Fall back controller	drone status errors
softwareVersion()	Install/Update software version	Payment directory	old_version new_version
authorizationCode()	Enter authorization code	Payment directory	code

7.3 Attributes

Attribute	Description	Class	Type
home_position	home position of drone fleet	System Controller	string
destination	destination of drone fleet	System Controller	string
features	extra features of drone fleet	System Controller	Feature
direction	drone's direction	Drones	string
position	drone's position	Drones	string
status	fall back status of drones	Fall back controller	string
maintenance_details	maintenance details of drones	Fall back controller	string
customer_name	customer's name	Payment Directory	string
address	customer's address	Payment Directory	string
username	account's username	Payment Directory	string
password	account's password	Payment Directory	string
payment	Payment	Payment Directory	int
GPS	GPS setting of drone	Features	GPS
camera	camera setting of drone	Features	Camera
measuring_equipment	measuring equipment of drone	Features	M_E
defense_and_security	defense and security of drone	Features	D_A_S
version_number	version number of software	Software version	int

7.4 Relationships

Name	Description	To_class	From_class
receives	System controller receives fall back info	Fall back controller	System Controller
controls	System controller controls drones	Drones	System Controller
receives	System controller receives extra features	Features	System Controller
updates	Software version is updated to system controller	System Controller	Software Version
determines	Software version determine what features the purchased/upgraded software is equipped with	Features	Software version
authorizes	Payment directory authorizes new software version	Software Version	Payment Directory
contains	Drone contains additional features	Features	Drone

7.5 Key Events

Event	Description	Action
Drone Validator	Ensures that the only communication that goes comes into the system is from a drone that is considered a valid part of the fleet	Validates every input signal and response from a drone that is to be considered part of the fleet
Drone Control	Receives input from the drone fleet management subsystem	Responsible for reading, processing and distributing drone controls to the fleet from the orchestration administration
Orchestration Admin	Main interface between what movements/actions the drones make and the actual orchestrator	Allows the orchestrator to dictate control commands that are received by the drone control component
Management Admin	Allows the software maintenance team to perform any management operations involving the fleet	Involves taking care of drone status, and being aware of certain tasks that need to be performed
Fall Back Controller	Allows the both the drone orchestrator and software maintenance actors to take control of a drone in the event that manually control is required	Connects to the system database so that the drone fleet management system can be aware of any fall back issues and use that information when calculating any fleet management decisions
Payment Manager	Subsystem dedicated to maintaining the payment operations with the bank and the order/upgrade generator	Knows who has payed for what, what kind of payment to process with the banks, and will be responsible for any monetary information.
Order/upgrade Generator	Responsible for purchase and upgrade of software	Generates orders and upgrades so that the entire system can be aware of which features should be dictated to who and so that the payment manager knows who to chart for what.