A Systematic Approach to Mission and Scenario Planning for UAVs

Niloofar Shadab
Institute for Systems Research
University of Maryland
College Park, Maryland 20740
Email: nshadab@umd.edu

Huan Xu
Institute for Systems Research
Department of Aerospace Engineering
University of Maryland
College Park, Maryland 20740
Email: mumu@umd.edu

***Abstract*—As unmanned autonomous vehicles (UAVs) are being widely utilized in military and civil applications, concerns about mission safety and how to integrate different phases of mission design are growing significantly. One important barrier to a cost-effective and timely safety certification process for UAVs is the lack of a systematic approach for bridging the gap between understanding high-level commander/pilot intent and implementation of intent through low-level UAV behaviors. In this paper we demonstrate an entire systems design process for a representative UAV mission, beginning from an operational concept and requirements and ending with a simulation framework for segments of the mission design, such as path planning and decision making in collision avoidance.**

I. INTRODUCTION

Unmanned autonomous vehicles (UAVs) are becoming increasingly utilized in military and civilian application due to their potential to provide improved capabilities while increasing manpower efficiency [1]. Current and future domestic applications for UAVs include search and rescue, weather forecasting, law enforcement, border patrol, firefighting, disaster response, precision farming, commercial fisheries, scientific research, aerial photography, mail delivery, infrastructure monitoring and emergency management [2]. As a result of the prevalence of UAVs, particularly in civilian applications, there are growing concerns with regard to the safe integration of UAVs into the national airspace. The safety and reliability of UAVs are highly reliant on their capability to avoid emergency situations in order to have a safe flight. However, the lack of a systematic approach for mission validation and verification can lead to mission failure as current methods cannot properly integrate high-level controls with low-level commands. Using a systematic approach for solving high-level problems and tracing them into the lower level problems can reduce the risk of failure and catastrophe [3].

The goal of this paper is to demonstrate an entire system design process for a representative UAV mission. Beginning from an operational concept and requirements and ending with verification through simulation, we demonstrate how model-based systems engineering tools can be used to capture high-level design coupled with low-level constraints.

A. Model-Based Systems Engineering

Systems engineering is an interdisciplinary approach used in various projects to enable the realization of a successful system and reduce the risk of encountering problems during system operation. A systems engineering approach to a project includes analyzing and deriving stakeholders' needs, documenting requirements and continuing with system design while considering the complete problem and validating the system to ensure it can satisfy stakeholders' needs in an efficient, cost-effective and high quality manner. A model-based system engineering methodology uses formalized applications of modeling to achieve this [4]. In this methodology, system requirements, structure and behavior can be visualized in the conceptual phase of system development as well as later in the life cycle. This method can help systems engineers to provide different representations of a system from the standpoint of corresponding concerns and issues of a system [5]. To clarify the importance of systems engineering applications in our problem, we demonstrate the complexity of the mission design and challenges that designers may encounter in mission planning and how they can leverage systems engineering approaches for planning a safe mission.

In mission and scenario planning, clients have specific requirements for accomplishing their mission, and as a result, various users such as Air Traffic Control and ground controllers, may interact with the flight mission. In addition, conceptual operations in different flight states and UAV behaviors and structure can play important roles in organizing the mission. In developing a mission plan for a UAV, one should address different design challenges such things as platform route, sensor modeling, communication, navigation, threat analysis and 4D visualization. Integrating all these requirements, verifying the entire complex system, as well as reducing failure risk and improving mission safety requires a systematic approach to mission planning. This approach allows us to capture this complex system and detect possible faults and malfunctions in each phase of the system. All of this leads us to conclude model-based systems engineering is a good solution for modeling all required states of our UAV flight.

In this paper we utilize a model-based systems engineering approach to capture the requirements, provide a high-

level solution to our mission planning problem, and map the generated models into the mathematical models. In Section II, we formulate our problem and demonstrate the mission requirements and operational concepts. Section III discusses system architecture as well as functional and behavioral analysis. In Section IV, we implement our architectural design in simulation and evaluate and validate the results with the mission requirements. Finally, we conclude and address future work in Section V.

II. Problem Formulation

Based on our stakeholder's (Millennium Engineering and Integration Company) requirments, we were assigned to design and implement a "Situational Awareness and Response Guidance Module"(SARGM) for use onboard Unmanned Air Vehicles (UAVs). The SARGM shall incorporate the following functions: (1) execute a prelaunch-uploaded mission plan, (2) identify anomalies (including UAV flight-rule violations, midair collisions, component failures and loss of communication link events) and (3) respond to such anomalies by generating revisions to the baseline mission plan in a manner that minimizes hazards to human life and property. The SARGM shall incorporate the Collision Avoidance algorithms that we are trying to develop.

A. concept description

Safe operation of Unmanned Air Vehicles (UAVs) operated by commercial and military entities in the National Air Space (NAS) is envisioned to require autonomous situational awareness and safe response to situations and anomalies that may constitute hazards to human life and property. The hazardous situations and anomalies may result from loss-of-command-link, violation of flight rules, departure from flight plan, UAV component failures, failure to respond to AT directives, and the need to sense and avoid nearby air traffic. Therefore, software algorithms onboard the UAV must detect and identify them. In addition, other onboard software algorithms must decide the "safe response" to each identified situation or anomaly, wherein determining such responses requires knowledge of map position, obstacle and terrain features. The onboard "safe response" software must incorporate decision support to either terminate the UAV flight or alter the UAV's onboard flight plan in accordance with the selected response.

Safety certification remains a challenge for UAVs because of the gap between understanding commander/pilot intent and implementation of intent through low-level UAV behaviors [6]. A lack of appropriate systematic methods prevents high-level autonomous systems from being widely fielded. Therefore, new techniques for standardized and formalized requirements specification and mission planning of UAVs are needed that take into account discrete decision-making and can be integrated with flight simulation software in order to verify the overall system. For this purpose we explain the mission overview and some of its high-level and low-level mission requirements in order to bridge the gap between functions and operational concepts of the UAV using model-based systems engineering.

B. Project Objective

In this project we tried to design and implement a "'Situational Awareness and Response Guidance Module"'(SARGM) for use onboard Unmanned Air Vehicles (UAVs). The SARGM shall incorporate the following functions:

1) Execute a prelaunch-uploaded mission plan
2) Identify anomalies (including UAV flight-rule violations, midair collisions, component failures and loss-of-command-link events)
3) Response to such anomalies by generating revisions to the baseline mission plan in a manner that minimizes hazards to human life and property

In conclusion, the purpose of the proposed research is to create a Preliminary Design of an Autonomous Intelligent Flight Management System for UAVs that incorporates autonomous situational awareness and safe response to situations and anomalies that may constitute hazards to human life and property.

C. Mission Overview

In order to determine the operational and functional requirements, it is essential to define representative mission and flight plan scenarios in order to identify, clarify and analyze users' requirements. These are focused on a variety of tasking scenarios characterized by FAA class airspaces A-G [7]. For this paper, we chose a loitering scenario and captured both high-level and low-level mission requirements related to this phase of flight. Table I shows the main goals of the mission, which are as follows:

1) Autonomous Flight: Achieve controlled take off, flight, loitering and landing.
2) Cover Entire Search Area: Determine target location within defined distance (50ft), fly the search area.
3) Obstacle Detection and Avoidance: Carry out Air Traffic Control (ATC) requirement to remain well clear of other traffics.

While this surveillance application is highly in demand due to its potential to be used in civil applications such as disaster response, firefighting, search and rescue [8], the approach used in this paper can also extend to the rest of the flight phase as well as to other scenarios.

D. Operational Concepts

After creating the main goals and mission scenarios, the next step is to provide use case diagrams for our system of interest, the UAV's mission planner, to capture the system and sub-system's behavior. Use-case diagrams are developed using the main goals of a system and show what the users want the system to do. Systems engineers can derive system requirements from them and their flows of actions [9]. Thus, we developed two use-case diagrams for the purpose of this example. One is a high-level system's use case diagram in which we show the overall tasks of mission planner in the

TABLE I
MISSION OVERVIEW

Mission Overview	High-Level Requirements	Low-Level Requirements
1	UAV shall approach the pre-planned maneuver point	UAV shall take images from loitering location
2	UAV shall fly at 5000ft altitude	UAV shall pass specific waypoints during loitering
3	UAV shall loiter for 1 hour	UAV shall Determine target location within defined distance (50ft)
4	UAV shall resume the flight path along the border	UAV shall detect all intruders within 100ft in loitering phase
5	UAV shall climb back after 1 hour loiter	UAV shall avoid up to 3 intruders at the same time

sequence of actions that the user might interact with. The second is an obstacle avoidance use-case which is a lower level use-case diagram for our mission. We go through the details of each diagram in the following paragraphs.

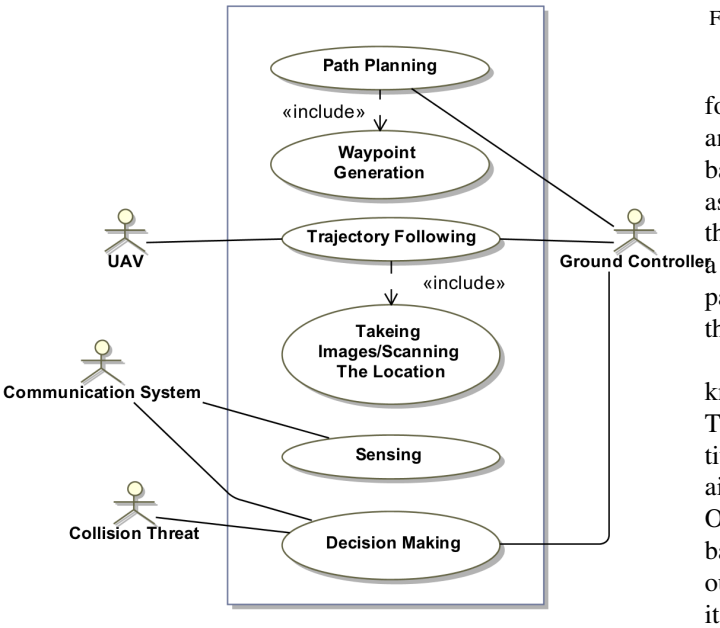

Fig. 1. High-Level Use Case Diagram for Overall Loitering Scenario

Figure 1 depicts all the states of the mission planner. These are (1) path planning, (2) trajectory following, (3) sensing, and (4) decision making for emergency situations. In Figure 1, the users of the mission are demonstrated. One type of user is the UAV communication systems, which allows the system to send and receive data from sensors and ground controls. Another type is the ground controller, who monitor, manage and track the mission and are ready to act in emergency situations. Ground controllers also have permission to cancel or change the mission on board if it is necessary.

To accomplish the mission, the UAV must be safe from any plausible threat. Therefore, the UAV should have a reliable collision avoidance system for all mission states, including the loitering scenario we focus on. This leads us to create a sub use case diagram that effectively shows the actions of the collision avoidance system and how it interacts with actors. Figure 2 is our sub-level use case diagram for the case in which the UAV should avoid obstacles.

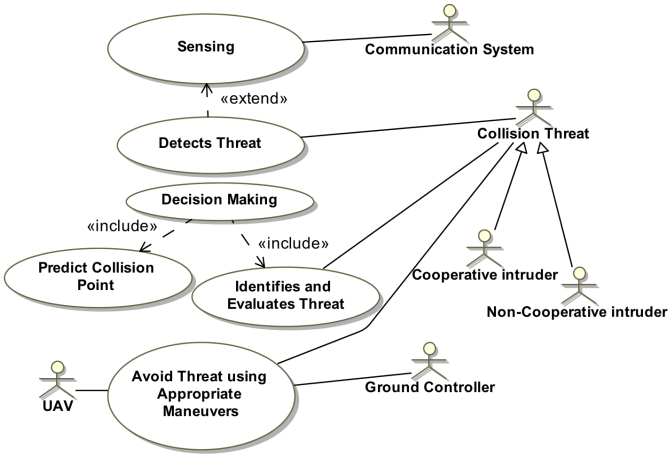

Fig. 2. Use Case Diagram for Collision Detection and Avoidance Scenario

As shown in Figure 2, while the UAV is loitering and following its trajectory, it should also sense and detect threats and use an appropriate maneuver to avoid collisions. However, based on the type of threats, the detection and avoidance aspects need to meet different sets of requirements. To simplify this, our simulation only looked at the set of requirements for a non-cooperative intruder, which is explained in the following paragraph. The simulation will be discussed in more detail in the software module section.

It should be noted that there are two types of intruders, known as cooperative intruders and non-cooperative intruders. The difference between these two types is that in the cooperative scenario, the UAV can cooperate with the intruder (another aircraft or UAV) and they can work together to avoid collision. On the other hand, non-cooperative intruders encompass birds, balloons, and other intruders that cannot communicate with our UAV. As a result, the UAV must avoid them entirely by itself. This paper focuses on the non-cooperative scenario to simulate the results and develop the algorithm.

III. ARCHITECTURAL DESIGN

In order to capture functional aspects of the system, we demonstrate and emphasize the system architecture, including system behavior and structure in detail. In this section we discuss the necessary specifications the system should have in order to meet the users' requirements. To begin, we explain the desired inputs and outputs. Then we will walk through the system's functional (operational) requirements to achieve those outputs.

A. Functional Concepts

In our use case diagrams, there are two important components for mission planning, one being path planning and

trajectory following, and the other having the capability to avoid potential collisions. In order to combine these two components and demonstrate how they can be related to each other, we present the following functional diagram (Figure 3).

Figure 3 is a context diagram that shows how the system interacts with the environment, their interfaces, and the flow of information. We consider documented requirements and a UAV model as inputs for the mission planner system [N.B. In this paper, we do not consider the problem of modeling the UAV dynamics, instead we use it as an input and a constraint for UAV mission planning]. Additionally, the simulation software can simulate the functionality of the system. Then the hardware codes can be developed in order to provide 4D visualization of the mission and integrate them into UAV system.

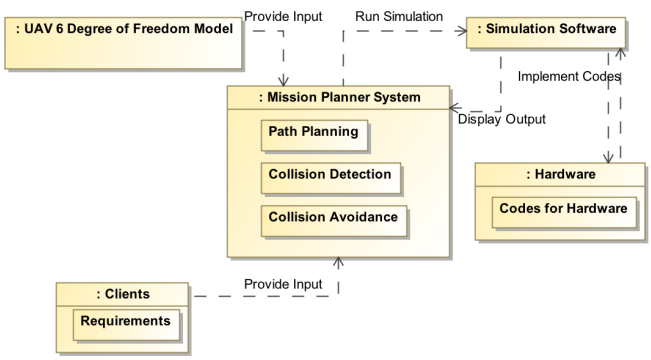

Fig. 3. System Context Diagram showing interactions between system and environment

In Figure 3, the functions of each step and how they connect with each other is derived. In this figure, the interoperability of the path planning, sensing and collision detection and avoidance subsystems can be seen. All the information about UAV model is used as input for the path planning and waypoint generation phase. For collision avoidance, the UAV needs to acquire information about unpredictable obstacles and turbulences from sensors, and use that to predict if there is a risk of an upcoming collision. Then, based on the prediction, it needs to avoid the threat and re-plan the trajectory within a specified time interval. Sometimes sensor data sent to the control station necessitates changes in the scenario planning, which must set updated to the system. In this case, reliability and safety of the sensor are the key parameters for a successful mission. However, in this paper we do not address the sensor parameters and functionality. Instead, we assume that the sensor is capable of sensing and sending all required data for the path planning, collision detection and collision avoidance, and developing sensor models will be the subject of the future work.

B. Behavioral analysis

In the previous sections, we considered the users' requirements and how it should interface with the system. Next, we develop the system's behaviors and functions. For this purpose, activity diagrams are helpful to visualize the steps, actions, and the parts of the systems that carry out the actions. In the collision detection and avoidance segments, the sequence and flow of actions are important for managing requirements and consistency between design and requirements. We provide the reader with some sections of the activity and sequence diagrams related to the non-cooperative collision detection and avoidance.

Figure 4 describes the collision detection system, which highly relies on the history of the tracked obstacle to estimate the future trajectory. It shows that the detection section shall predict the point of collision and the time to reach that point in order to provide sufficient information for the collision avoidance section to avoid threats. This prediction is based on time-based information about the history of the obstacle's trajectory and velocity. Sensor systems will ensure the whole system about the possibility of getting this kind of information. If there is sufficient time-based information, the system will estimate the future trajectory of the obstacle. Then it can predict the closest approaching point for UAV and obstacle.

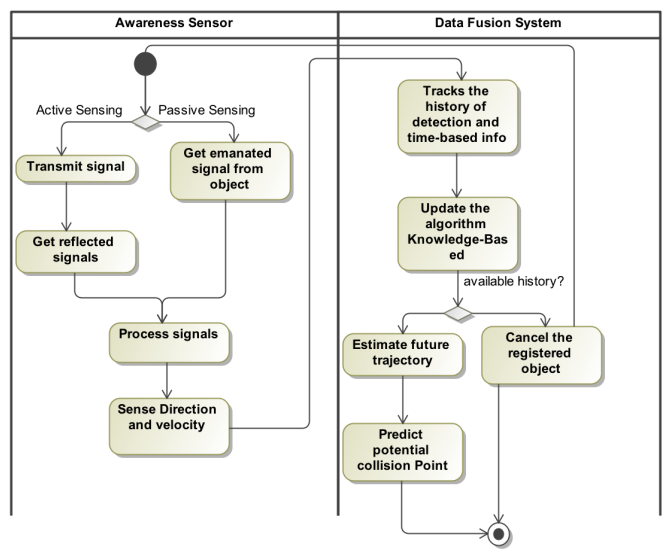

Fig. 4. Activity Diagram for collision avoidance showing flows of activities

In Figure 5, the collision avoidance activity, which occurs after detection, is shown. In particular, it shows the steps of determining the best avoidance maneuver and how it applies the maneuver to avoid the obstacle, as well as necessity of re-planning and waypoint generation for some small time intervals to avoid the obstacles. The declaration system uses detection information about distance and time to closest approaching point to evaluate if the UAV can avoid the obstacles or not. After this evaluation, system can provide a no-flight area for the UAV by considering other obstacles and select the most appropriate maneuvers for avoiding. In the next section, we demonstrate how collision avoidance algorithm that we use can determine these maneuvers.

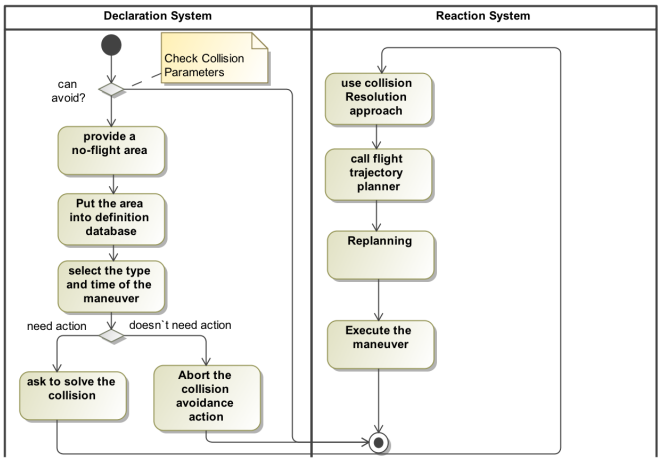

Fig. 5. Activity Diagram for collision avoidance showing flows of activities

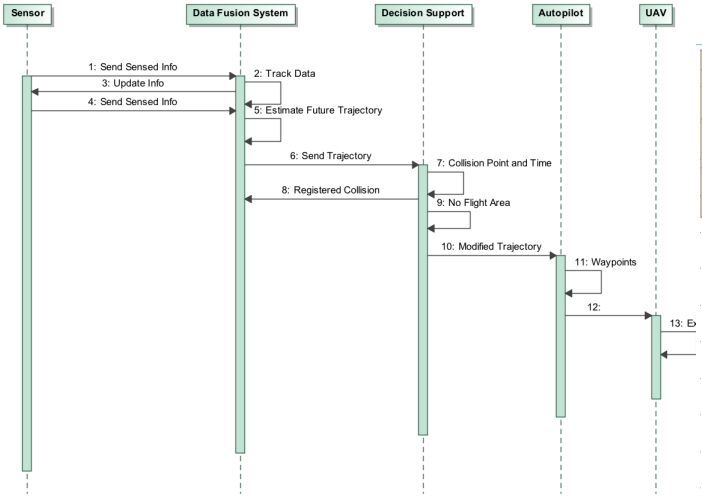

Fig. 6. System Sequence Diagram- Sequence of actions between sub-systems

In Figure 6, the sequence of actions for non-cooperative collision avoidance is shown. This figure helps us to determine the procedure of the collision avoidance and the priority of different steps. Sensors receive information about the location of obstacles and send that information to the data fusion block for processing. Then, if the information if sufficient, the system tracks the data and estimates the future trajectory. Afterwards, the predicted trajectory is used for determining collision time and collision point. The next step is to provide a no-flight area for UAV and modify its trajectory for some amounts of time. Then, the new waypoints are generated by autopilot and UAV can execute appropriate avoidance maneuver.

C. Structural Analysis

Having explained functionality and behavior, we now demonstrate the system's structure by using a block definition diagram. A block definition diagram is useful for showing the system's module and can be used in software development and simulation of the system. Block definition diagrams can accept values, parts, operations and attributes, which allow it to be easily converted to code. In this paper we focus on the procedure and provide the reader with an example of simulation results that are extracted from the block definition diagram. This is shown in Figure 7, which depicts the functionality of the entire system.

As discussed earlier, one of the first steps in mission planning is generating trajectories. Other parts of the system are sensing, collision detection and collision avoidance subsystems needed for mission safety. All parts have their own operations and values. For example, in collision detection, the system shall predict the obstacle's trajectory in the near future and detect if there will be a collision based on the speed and trajectory of the UAV. The collision avoidance part shall make the decision on how to avoid the potential collision by changing speed, turning radius and altitude. Figure 7 shows each of these parts in mission planning. In each block, constraints demonstrate the method and formula for developing codes for each block

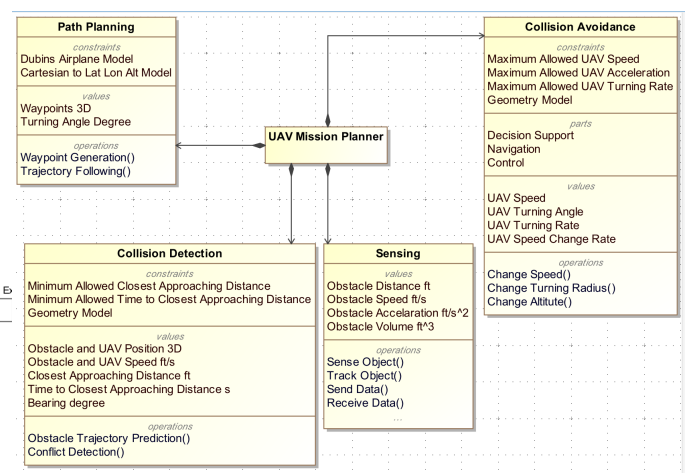

Fig. 7. System Block Definition Diagram that shows four modules of the system. These modules are: (1) path planning, (2) collision detection, (3) sensing, and (4) collision avoidance.

IV. SOFTWARE MODULE

To integrate a complex system consisting of different subsystems, simulations are used to analyze the system's behavior and verify its correctness. The simulation system is derived from the system's model and tries to capture all aspects of the system's model. However, in the real world, the entire system cannot be simulated as there are always many constraints for simulating all parts of the system. Therefore, it is important to clarify what the goals of the simulation system are and what parts of the system are going to be simulated. As it is shown in Figure 3, simulation software is developed from the mission planner model to simulate different segments of the mission planner model. In the previous sections, we provided a semi-formal model for a mission planner system and in this section, we provide simulation systems for path planning and collision avoidance parts of our developed model.

A. Simulation Module Structure

In order to show the flow of the data and the structure of the simulation, we develop the simulation internal block diagram which shows the interaction between parts of the simulation software and how data is transferred between them. Figure 8 shows the internal structure of our simulation system. The simulation parts are as followed: (1) Mission Planner User Interface, (2) Mission Planner, (3) Mission Manager, (4) Mission Recovery, (5) Mission Planner Display Engine. Mission Planner User Interface and Mission Planner Display Engine are related to the users' inputs, outputs and the simulation results. The Mission Planner determines the UAV's trajectories, waypoints and search area. The Mission Manager is responsible for determining possible collisions and calculating collision parameters such as point of closest approach and time to point of closest approach. The Mission Recovery calculates required speed change, turn rate and altitude change for avoiding obstacles. In Figure 8, the flow of information between these parts is also depicted.

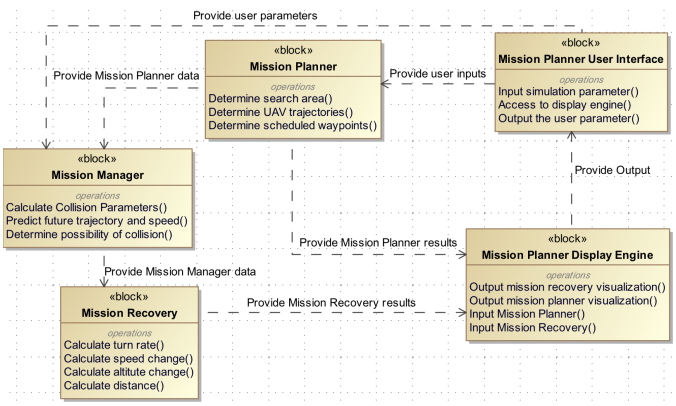

Fig. 8. Simulation Internal Structure and the Flow of Information between different parts of the simulation software

B. Module Implementation

In this section, we demonstrate a simulation that represents how our UAV mission planning system meets the customer's requirements and achieve the required functionality. Our requirement analysis and the artifacts that we created allow us to identify relevant data for the simulation and organize our numerical simulation. For this purpose, first we show the trajectory generation for the overall mission, then we go through the obstacle avoidance and generalize it by simulating collision avoidance for 3 obstacles at the same time (which was one of the mission requirements). It should be noted that in this paper, we do not show the sensing part and we assume that the obstacles are already sensed.

1) Path Planning Module: After modeling our simulation system, the first step in developing our simulation is to create the UAV trajectory based on the mission scenario. In Figure 9, the UAV loiters in some specific locations and passes specific waypoints. In order to generate this path, we used Dubins airplane method [11] which allows us to generate an optimal solution to the path planning problem.The very basic Dubins Airplane equation of motions are as follows.

$$\begin{aligned} \dot{r_n} &= V\cos\psi \\ \dot{r_e} &= V\sin\psi \\ \dot{r_d} &= u_1 \qquad |u_1| \leq 1 \\ \dot{\psi} &= u_2 \qquad |u_2| \leq 1 \end{aligned} \tag{1}$$

$$\begin{pmatrix} \dot{r_n} \\ \dot{r_e} \\ \dot{r_d} \end{pmatrix} = \begin{pmatrix} V\cos\psi\cos\gamma \\ V\sin\psi\cos\gamma \\ -V\sin\gamma \end{pmatrix} \tag{2}$$

These equations are based on the assumptions that there are some constraints on airspeed V, flight path angle γ^c and the bank angle ϕ^c just the same as the Dubins Car algorithm. So the Dubins Airplane should satisfies these constaints:

$$\begin{aligned} \phi^c &\leq \bar{\phi} \\ \gamma^c &\leq \bar{\gamma} \end{aligned} \tag{3}$$

For developing our path planning module, we derived path planning simulation Functional requirements for a non-cooperative scenario, some of the important ones are listed bellow:

1) Generate waypoints between main waypoints using Dubins Path algorithms.
2) Develop candidate paths between main waypoints.
3) Choose the optimal Dubins path between each of two waypoints among all possible generated Dubins paths.
4) Develop 3D trajectory for UAV using Dubins algorithms for 3D environment.
5) Transfer waypoints XYZ coordinate to altitude-latitude-magnitude coordinate using the transforming equations.

The Dubins airplane path between two nodes can be derived using dubins motion primitives. So here you can see the possible path between two specific nodes can be created using the combinations of three Dubins Airplane paths. The path is generated by including 4 nodes, which between each two node, a Dubins Airplane path is generated, then the generated Dubins Airplane paths combine to each other and create the desired path between start and end waypoints.

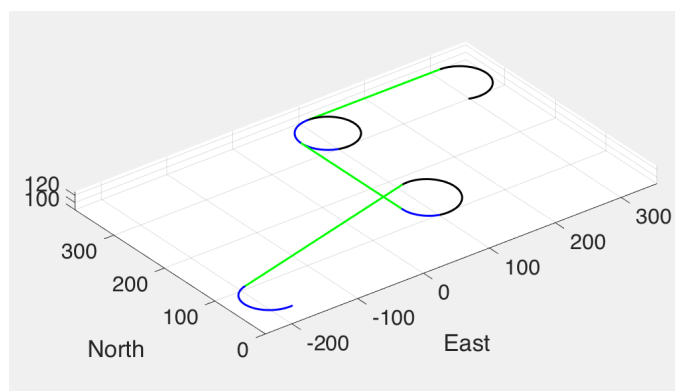

Fig. 9. Path Planning and Waypoint Generation for the overall loitering scenario. start node: [0, -200, -125], end node: [300, 300, -100], the middle nodes are [100, 100, -100] and [300, 100, -100].

2) Collision Detection and Avoidance Module: After completing the path planning and waypoint generation, we need to calculate how the UAV can successfully detect and avoid the obstacles that may appear in its path. We used the Geometry model [12] to calculate how the UAV detects the collision and how it decides to avoid the obstacle. Both Collision Detection and Avoidance algorithms are used the differential geometry concepts. These algorithms can be used for one or multiple collisions at the same time. They are also used the principals of airborne collision avoidance systems confirming to TCAS. This study limits the analysis to non-cooperating UAVs and intruders. Some of the assumptions that are considered for developing this algorithm are

- Vehicle dynamics are presented by point mass in Cartesian coordinates on R^2.
- The threats are non-cooperative and non-maneuvering.
- The threats have been sensed by the UAV's sensors so the deterministic positions and velocity vector of the intruders are determined. So the UAV can predict the future trajectories of threats based on the current position and velocity vectors and their linear projections.

Considering that the intruder is sensed by the sensors, the UAV establishes a sightline between itself and the intruder. This sightline vector is given by

$$r = r_a - r_u. \tag{4}$$

considering the assumptions that the velocity of both intruder and UAV is constant, then the differential of equation 4 is

$$\dot{r}t_s + r\dot{\theta}_s n_s = v_a t_a - v_u t_u \tag{5}$$

where n_s is the basis vector normal to the sightline for UAV and the t_s and t_a are the basis vectors along to the sightline for UAV and intruders, respectively. In the figure bellow you can see the deferential geometry related to the UAV and intruder.

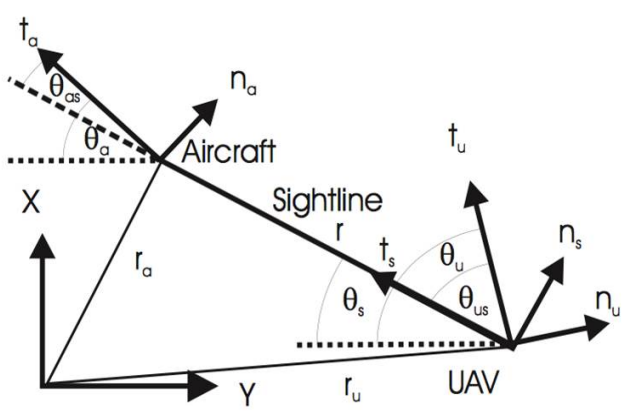

Fig. 10. Geometry of the UAV relative to the threat in the Cartesian coordinate

Components of the relative velocity vector along and normal to the sightline are as follows. These equations are derived from the 5 and dot product of t_s and n_s to the 5 equation, respectively

$$\begin{aligned} \dot{r} &= v_a t_s.t_a - v_u t_s.t_u \\ r\dot{\theta}_s &= v_a n_s.t_a - v_u n_s.t_u \end{aligned} \tag{6}$$

We can also derive the relative acceleration along and normal to the sightline by modifying equation 5 and using the Serret-Frenet[] equations. So we have

$$\begin{aligned} (\ddot{r} - r\dot{\theta_s}^2) &= v_a^2 k_a t_s.n_a - v_u^2 k_u t_s.n_u \\ (r\ddot{\theta}_s + 2\dot{r}\dot{\theta}_s) &= v_a^2 k_a n_s.n_a - v_u^2 k_u n_s.n_u \end{aligned} \tag{7}$$

These equations define the geometry concept of UAV relative to an intruder and show the geometry kinematics. Both detection and avoidance algorithms are developed based on this geometry kinematics. Functional Requirements For developing detection and avoidance module, functional requirements should be identified and trace to the software architecture and structure. The final software should address all defined functional requirements. Both detection and avoidance modules, must have specific functionality to achieve desired goals. Here, we provide a set of some of these requirements. Collision detection simulation functional requirements for a non-cooperative scenario are as followed:

1) Determine closest distance between intruder and UAV.
2) Determine time to closest distance between intruder and UAV.
3) Compare the closest distance with safety circle (minimum allowed distance between intruder and UAV)
4) Compare time to closest distance with look-ahead time
5) Determine if collision will occur
6) Do all the procedure for each of detected targets.

Collision avoidance simulation functional requirements for a non-cooperative scenario are as followed:

1) Determine the speed rate.
2) Determine the heading angle rate.
3) Determine the velocity at each time step
4) Determine the angle at each time step.
5) Check if at each time step the distance between intruder and UAV remains above safety circle.
6) For multiple collisions at the same time, the system shall consider the maximum relative heading angle among all heading angles between target and UAV
7) For multiple collisions at the same time, the system shall consider the union of all conflict sectors as a conflict resolution.

The detection algorithm that we used, works based on the UAV's minimum and maximum allowed speed, acceleration and turn rate as well as look ahead time and minimum allowed distance between UAV and obstacles. In Figure 10, the UAV calculates the closest point of approach and the time to closest point of approach for each of the approaching obstacles. In this simulation, the assumption is that the obstacles' velocity and heading angles are constant during the detection period.

There are different scenarios in which, a collision might occur. Based on the sensor information and the future prediction of threads, the collision avoidance conditions can differ

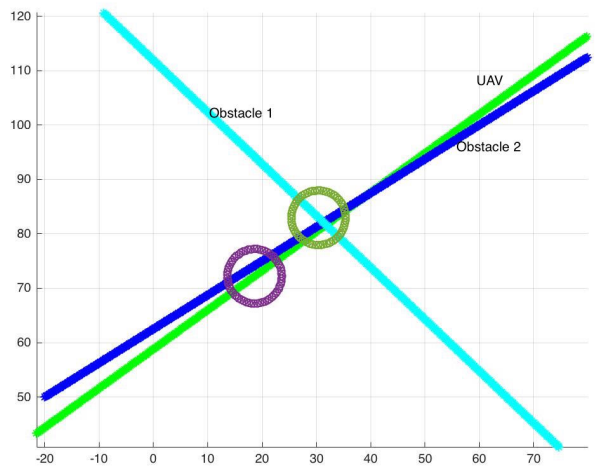

Fig. 11. Collision Detection of two Obstacles at the same time. The circles indicate the collision points where the distance between UAV and the obstacle is less than the minimum allowed distance for UAV and obstacle.

from each others. As a result, avoiding a situation in which UAV or obstacles don't have constant velocities or linear trajectories, requires different avoidance actions in comparison to situation in which velocities are constant and trajectories are linear. The more different possibilities considered, the more reliable collision avoidance system we have. In table II, we considered some of possible scenarios based on UAV and obstacles' specifications where a_a and V_a are acceleration and speed of thread respectively and a_u and V_u are acceleration and speed of UAV, respectively. In this paper we aim to provide simulation results for some of these scenarios and compare the results with each other in order to analyze how different the avoidance maneuver can be based on different situations.

TABLE II
POSSIBLE COLLISION SCENARIOS

Number of Threads	V_a	V_u	a_a	a_u
One	Constant	Constant	0	0
One	Constant	Changing	0	Constant
One	Constant	Changing	0	Changing
One	Changing	Changing	Constant	Constant
One	Changing	Changing	Changing	Changing
Multiple	Constant	Constant	0	0
Multiple	Constant	Changing	0	Constant
Multiple	Constant	Changing	0	Changing
Multiple	Changing	Changing	Constant	Constant
Multiple	Changing	Changing	Changing	Changing

Avoiding all these collision situations depends on some physical and mechanical constraints of UAVs. For having a successful avoidance maneuvers, it is very important to examine extreme values of UAV specifications. These boundary values include but not limited to maximum and minimum UAV's achievable speed, maximum allowed UAV's turn rate, maximum heading angle, maximum and minimum UAV tangent acceleration. These parameters may limit the maneuverability and agility of UAV. Since these factors are some design factors, they should be either designed based on the importance of UAV's desired collision avoidance capabilities or they provide restrictions for UAVs in order to avoid collisions. Although, the collision avoidance algorithms used in this paper, are practical in avoiding obstacles, as they are developed based on UAV's physical constraints, they cannot provide UAVs with efficient avoidance maneuvers in all possible scenarios. Therefore, it is very critical to identify the situations, in which UAV is not agile enough or maneuverable enough to do required maneuvers. In the data structure diagram that is shown bellow, how the physical constraints of UAV can affect on efficiency of the avoidance algorithm, is shown. Figure 12 shows all types of data for avoidance module. These include input data, internal data and output data. Having this information of data, one can effectively develop software module in a way that the data flow and required steps for getting desired outputs will be considered and implemented.

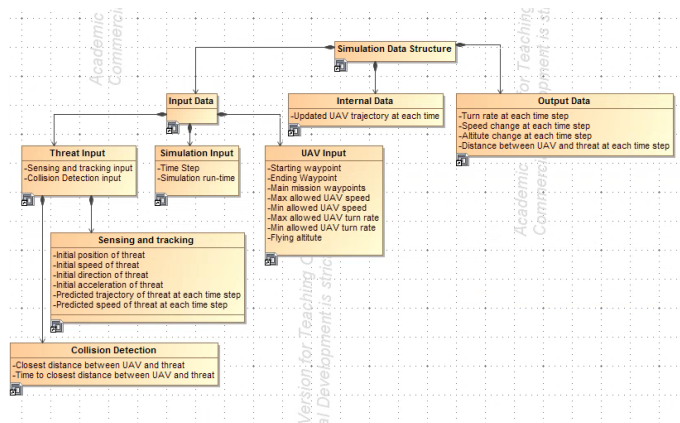

Fig. 12. Data structure for collision avoidance software module containing input data, internal data and output data of the module

The other important fact that should be evaluated when getting simulation results, is the time required for UAVs to process the commands and apply maneuvers. So the gap time between understanding commands and executing them, can cause some inaccuracies in calculation of required time to avoid collision. In order to mitigate those inaccuracies, the amounts of response lag in executing required command, should be determined. This response lag will vary for different UAVs. However, in this paper we assumed ideal situation which means there is no lag between system's command and UAV execution.

Figure 13 is based on an assumption that both UAV and obstacle's velocities are constant and the equation of motion for obstacle is linear. In Figure 13, we see that the UAV changes its heading angle to avoid an obstacle. In this specific scenario, the UAV does not need to change its speed to avoid the obstacle, although it may need to in some other cases. The second plot in Figure 13 shows how the UAV and obstacle are able to remain far enough from each other. The first plot depicts the changes of UAV heading angle in order to avoid

the obstacle. As it was mentioned before, this obstacle is non-cooperative, so the UAV shall do all avoiding maneuvers on its own.

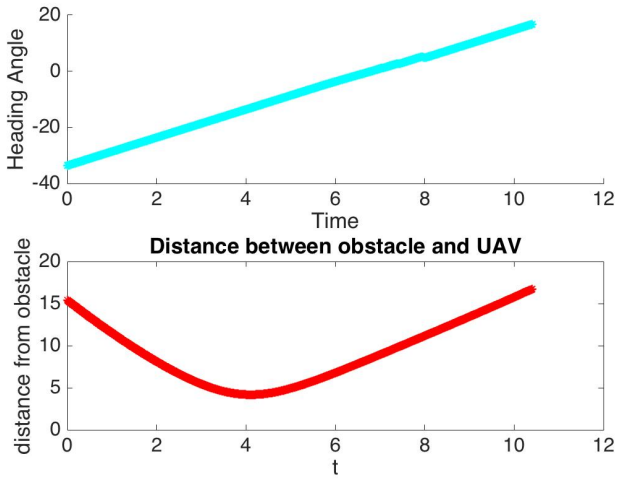

Fig. 13. Collision Avoidance from one Obstacle- UAV heading angle (top), velocity (middle), and distance between obstacles and UAV (bottom)

3) Multiple Collision Avoidance: The next step in developing our simulation is to generalize the collision avoidance part from avoiding one obstacle into avoiding three obstacles. For this purpose, we simulate the scenario in which the UAV should simultaneously avoid multiple obstacles [13]. Figure 14 indicates a situation in which the UAV should change both its turning angle and speed to avoid the collision. In this scenario, velocity and heading angle of obstacles are constant so they follow linear trajectories. Also UAV has constant speed and heading angle. This situation describes one of the possible collision scenarios UAV should avoid.

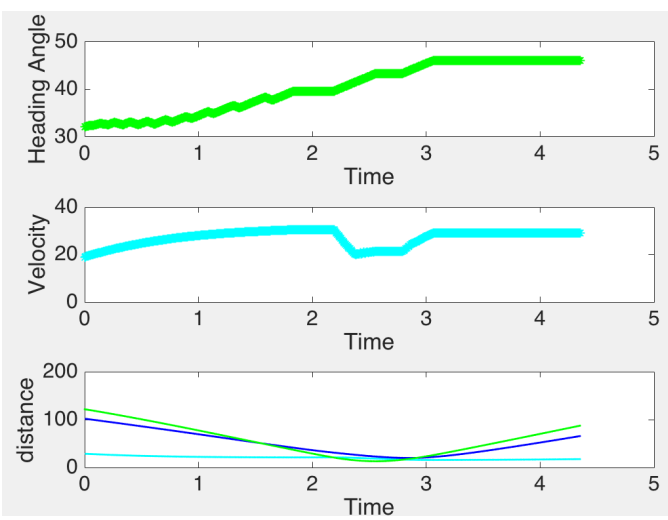

Fig. 14. Multiple Collision Avoidance- UAV's Velocity and Heading Angle Change During Time

In order to be certain that our avoidance algorithm works properly, we capture the distances between all three obstacles and UAV all the time . Figure 15 shows how the obstacles and UAV are far enough from each other based on the minimum allowed distance between UAV and obstacles. This distance is also one of the requirements.

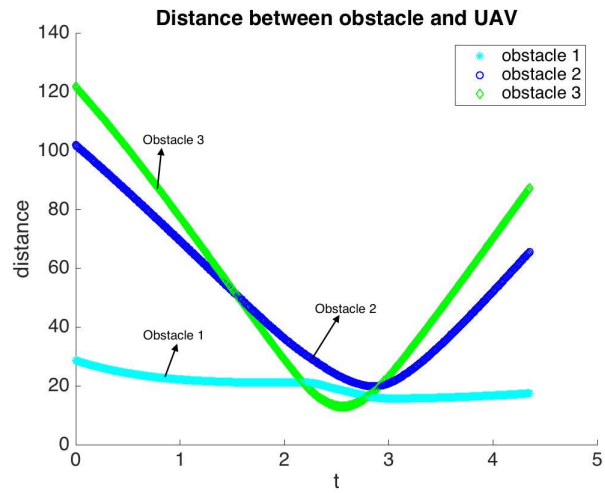

Fig. 15. Multiple Collision Avoidance- UAV and All Three Obstacles During Avoidance Time. 10m was considered as the minimum allowed distance

The other possible collision scenario that is shown in table II, is the situation in which, obstacles and UAV's velocities change during collision time. The next simulation results capture a scenario of having 3 obstacles which each of them has a constant tangent acceleration. So, it means that we still have future prediction of obstacles and UAV's trajectories. As it is shown in Figure 16, speed and turning angle rate

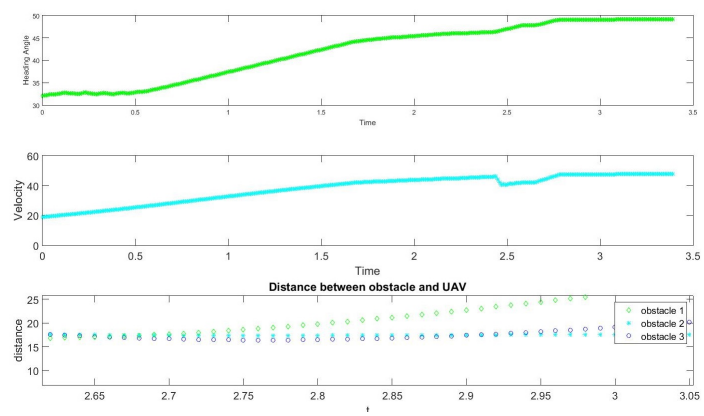

Fig. 16. Multiple Collision Avoidance- For the situation in which speeds are not constant and are changing with some constant tangent accelerations. In this simulation result, tangent acceleration for all obstacles is 5m/s^2

curves are sharper than the ones in Figure 14. We can notice from the figure that UAV has not yet reached to its boundary values for turn rate, velocity or acceleration. That means it still should be able to avoid some obstacles with higher speed or acceleration. In order to be certain that UAV is capable of doing avoidance maneuver, we developed a test case in which, UAV is examined by its boundary values. We tested if UAV

uses its ultimate physical and mechanical capabilities such as maximum turn rate and speed rate, it will be able to avoid obstacles or not! If it passes this test, then we can provide the UAV with the best avoidance maneuver using collision avoidance algorithms. We believe that these analysis are so critical as sometimes sensors cannot sense obstacles in a right time, or detection part detects collisons with some delays. In this situation, UAV has less time to avoid obstacles and this problem would be more serious if there are multiple threats in the UAV's trajectory. As an example, we provided another simulation results in which we considered a scenario that UAV flies in a low speed and the time to closest distance is less than the previous scenarios. As UAV should avoid all three collisions at the same time, it may not even use its maximum capabilities for maneuver, cause there should be a balance in distance between UAV and all obstacles and it cannot consider just one collision at a time to avoid.

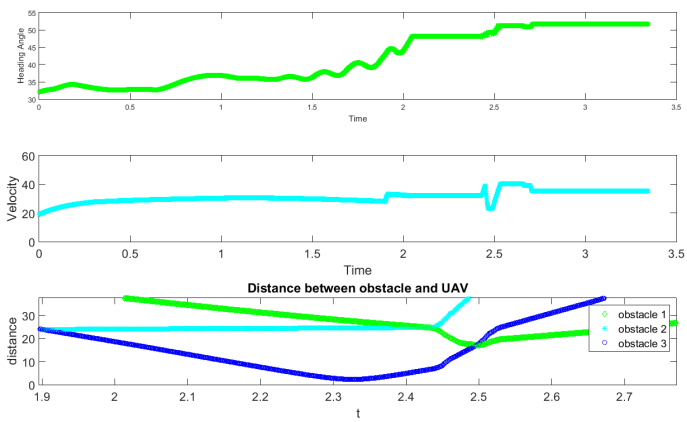

Fig. 17. Multiple Collision Avoidance- For the situation in which UAV fies in lower speed and has to do avoidance maneuvers. The speed of UAV is[16 10] and obstacles accelerations are [-3 5], [10 -4], [5 4] respectively

V. DISCUSSION AND FUTURE WORK

A model based systems engineering approach applied to this projects helps us in formalizing requirements analysis and requirement identification. System artifacts ensure the consistency between design and requirements so the simulation results and final mission planning design will satisfy the stakeholders' needs.

In future work, we plan to develop algorithms for cooperative intruders and verify the results using the method proposed in this paper. Moreover, we will explore the details of the detection and sensing parts, which would include some challenges about sensor systems, tracking objects and trajectory predictions. Thus, sensor modeling would also be considered for our future works to determine if the system can satisfy detection and sensing requirements. Also, for designing a mission, communication and navigation accuracy play important roles in accomplishing a safe mission. Considering these parts would be challenging and increase the uncertainty of the results.

ACKNOWLEDGMENT

The authors would like to thank Millennium Engineering and Integration Company for supporting this research and Matthew David Solomon for his comments and discussion on this paper.

REFERENCES

[1] Bachkosky, J. M., et al. Roles of unmanned vehicles. No. NRAC-03-1. NAVAL RESEARCH ADVISORY COMMITTEE ARLINGTON VA, 2003.

[2] Civil UAV capability assessment, NASA, 2006. Interim report, http://www.nasa.gov/centers/dryden/research/civuav/index.html (last accessed 21/3/2011).

[3] Andrew P. Sage, William B. Rouse, Handbook of Systems Engineering and Management, John Wiley & Sons, 2009

[4] INCOSE Systems Engineering Vision 2020, September 2007, Version 2.03

[5] IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. September 2000

[6] Evans, Andrew R. "The hazards of unmanned air vehicle integration into unsegregated airspace." The University of York, York (2006).

[7] Aviation Handbook, FAA Regulation and Policies, 2014, http://www.faa.gov/regulations_policies/handbooks_manuals/aviation/pilot_handbook/media/PHAK%20- %20Chapter%2014.pdf

[8] A. Stenger, B.Fernando, M. Heni, Autonomous Mission Planning for UAVs, A Cognitive Approach, German Aerospace Congress, Berlin, Germany, 2012.

[9] Muhairat, Mohammad I., and Rafa E. Al-Qutaish. "An approach to derive the use case diagrams from an event table." 8th WSEAS,?Int. Conference on Software Engineering, Parallel and Distributed Systems. 2009.

[10] Data Integration Glossary, U.S. Department of Transportation, August 2001.

[11] Beard, Randal W., and Timothy W. McLain. "Implementing dubins airplane paths on fixed-wing UAVs." Contributed Chapter to the Springer Handbook for Unmanned Aerial Vehicles (2013).

[12] White, B. A., Shin, H. S., and Tsourdos, A., ?UAV obstacle avoidance using differential geometry concepts?, IFAC World Congress 2011, Milan, Italy, 2011.

[13] Shin, H. S., White, B.A., and Tsourdos, A., ?Conflict detection and resolution for static and dynamic obstacles?, Proceedings of AIAA GNC 2008, August 2008, Honolulu, HI, AIAA 2008-6521