Software Requirements Specification

for

CubeSat

Version 2.0 approved

Prepared by

Dagrac, Mark
Ha, Kit
Tahod, John
Tobgui, Pierre
Ung, Jonathan

The Aerospace Corporation

9/16/2019
# Table of Contents

Table of Contents...................................................................................................................<1>
Revision History.....................................................................................................................<2>

1. Introduction................................................................................................................<3>
   1. Purpose...............................................................................................................<3>
   2. Intended Audience and Reading Suggestions.................................................<3>
   3. Product Scope....................................................................................................<3>
   4. Definitions, Acronyms, and Abbreviations..................................................<5>

2. Overall Description....................................................................................................<6>
   1. Product Perspective........................................................................................<6>
   2. Product Functions...........................................................................................<8>
   3. User Classes and Characteristics..............................................................<8>
   4. Operating Environment..............................................................................<8>
   5. Design and Implementation Constraints.....................................................<8>
   6. Assumptions and Dependencies...................................................................<9>
   7. Apportioning of Requirements.....................................................................<9>

3. External Interface Requirements.............................................................................<10>
   1. User Interfaces..............................................................................................<10>
   2. Hardware Interfaces.....................................................................................<10>
   3. Software Interfaces......................................................................................<11>
   4. Communications Interfaces.........................................................................<11>

4. Requirements Specification....................................................................................<12>
   1. Functional Requirements............................................................................<12>
   2. External Interface Requirements................................................................<14>
   3. Logical Database Requirements..................................................................<14>
   4. Design Constraints......................................................................................<14>

5. Other Nonfunctional Requirements......................................................................<16>
   1. Performance Requirements..........................................................................<16>
   2. Safety Requirements....................................................................................<16>
   3. Security Requirements................................................................................<16>
   4. Software Quality Attributes.......................................................................<16>
   5. Business Rules.............................................................................................<16>

6. Other Requirements..............................................................................................<17>

Appendix A: Glossary............................................................................................................<17>
## Revision History

<table>
<thead>
<tr>
<th>Team Members</th>
<th>Date</th>
<th>Reason for Change</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dagraca, Ha, Tahod, Tobgui, Ung</td>
<td>9/16/2019</td>
<td>Initial release</td>
<td>v1.0</td>
</tr>
<tr>
<td>Dagraca, Ha, Tahod, Tobgui, Ung</td>
<td>11/1/2019</td>
<td>Revision</td>
<td>v2.0</td>
</tr>
</tbody>
</table>
1. Introduction

The CubeSat Deorbiting vehicle is an autonomous vehicle created to assist failed satellites. The software allows the CubeSat not only to reach its target, but also to identify how far away the object is. Additionally, the software is able to detect instances in which a target should be approached with caution or avoided completely. Finally, the software allows the CubeSat the capacity to identify situations in which an object should be approached gradually or slowly.

1.1 Purpose

The purpose of this document is to assert the requirements for CubeSat as well as to outline the projects design. This document will explain in detail the software that will be implemented to the CubeSat. CubeSat developers will be the intended user for this document. The use for this document will guide the user to know what the CubeSat software is possible/not possible of achieving.

1.2 Intended Audience and Reading Suggestions

This document is intended for The Aerospace Corporation, and instructors.

1.3 Product Scope

Software Products Produced:

**Ground System (GCS)** - used to control/monitor CubeSat that will help to rendezvous with a satellite in need of repair or possible assistance to deorbit. In addition to assisting satellites, it can help deorbit space junk/debris as well.

A. GCS’s Functionality:

   I. Store sensor data from CubeSat into database. Use that data to display a graphical interface to visualize CubeSat sensor data. Sensors include LIDAR sensor, 9 DOF (degree of freedom) sensor. As well as a camera to assist with image recognition.

   II. The GCS with communicate directly with the CubeSat.
III. The GCS will issue commands to the CubeSat.

**SimPlat** - is used to simulate movement of the CubeSat in space.

B. SimPlat’s Functionality:

   I. Accept thruster commands from CubeSat and replicate zero gravity movement using omnidirectional platform for example if CubeSat request forward thruster the CubeSat will speed up and maintain speed until the opposing thruster is fired.

**CubeSat** - CubeSat is used to gather data from its surroundings to the GCS.

C. CubeSat’s Functionality:

   I. Collects data from the sensors attached.

   II. Sends the data to the Ground Control System for storage and display.

   III. Uses data with machine learning to generate movement commands that:

       1. Will orient the CubeSat to track a target.
       2. Follow a course to intercept a target.
       3. Establish and maintain a distance in relation to a target.

IV. Sends movement commands to SimPlat.
1.4 Definitions, Acronyms, and Abbreviations

DFD - Data Flow Diagram
SRS - Software Requirements Specification
GCS - Ground Control System
DB - Database
2. **Overall Description**

The SimPlat’s purpose is to carry the CubeSat payload and simulate motion in space in response to commands to move or spin on a horizontal plane. This vehicle will be provided to the computer science team. It uses Omni Wheels for full two-dimension mobility including the ability to rotate in place. When the vehicle receives a signal that the Cub is thrusting to move in a direction, the SimPlat will accelerate in that direction while the force is in effect. When the thrust stops, the vehicle will hold its velocity and continue to move at that speed and direction until countered. To stop the vehicle, an opposite force will have to be applied to reverse the velocity vector precisely. The SimPlat will obey wireless commands to move. It will report its own actions to the GCS.

The SimPlat consists of:
- Omni Wheel vehicle with platform and CubeSat mount
- Wireless two-way communication
- Onboard computer processor
- Software to report status to GCS, process commands from GCS, and react to motion telemetry from Cube

The prototype of this vehicle was developed by the 2018-2019 CSULA Senior Design Team in the computer science area and utilized machine learning techniques to learn how to move around obstacles. The project targeted anomaly prediction and prevention. The CS team will program the vehicle to move, transmit, and receive appropriately.

2.1 **Product Perspective**

The CubeSat Deorbiting Vehicle is designed to simulate a satellite in space that can identify objects, identify potential damaged satellites, and maneuver to the satellite to perform deorbiting operations on the object. A key difference however, between a normal satellite and the CubeSat product is that while a normal satellite is designed to work in 3-D space with zero gravity and large distances in mind, the CubeSat will instead work on a 2-D plane and ignore the gravity portion as all objects will be moving on a horizontal plane. For modeling purposes, distances are
scaled down and an emphasis on target tracking and automated movement is made to demonstrate the constraints

The CubeSat project is divided into three systems: CubeSat, SimPlat, and Ground Control System.

CubeSat serves as the brain of the entire project and similarly, it has multiple functions. It acts as a monitoring device that is able to take video, measure distance to objects in front of it, and record changes in speed and acceleration of the apparatus it is attached to. Besides gathering information, the CubeSat also acts like a gps that can define an optimal path to a target destination with the information that it has. Finally, much like a controller, the CubeSat is able to send commands to the SimPlat in order to move the vehicle towards an intended destination or target.

SimPlat is the system that moves the entire project physically on a 2-D plane. The SimPlats only job is to receive commands from the CubeSat and like an R/C car, it will move as where the CubeSat tells it to.

The Ground Control System serves as a monitoring tool for the user, allowing the user to view data that is collected from the CubeSat. The Ground Control System will also be fitted with the ability to send simple commands to the CubeSat in semi-autonomous mode.

Internet connectivity is not required for the project to function but a local network such as through a router, is required for the CubeSat to connect with the other components of its system.

### 2.2 Product Functions

#### 2.2-1 CubeSat

- **Object Detection**
  - Able to identify targets at a distance. This means that it is able to determine whether something is an object of interest or something to be avoided.
  - Able to calculate the distance and speed of distant objects from the CubeSat in real time.

- **Intelligent Maneuvering**
  - Able to calculate the predicted movement of a moving target object.
  - Able to plan a route to intercept the moving target.
  - Once near the target, able to safely maintain a distance from the target to perform deorbiting operations.

- **Data Collection**
- A live video feed is collected

2.2-2 **SimPlat**
- **Movement**
  - Able to move in any direction 360 degrees around it.
  - Once moving in a speed and direction, it is able to maintain this speed and direction.

2.2-3 **Ground Control System**
- **Monitoring**
  - Live camera feed from the CubeSat’s camera
  - Live updates of the CubeSat’s velocity and acceleration
- **Emergency Stop**
  - Able to send a command to SimPlat to stop immediately.

2.3 **User Classes and Characteristics**

The Aerospace Corporation is the primary user for this product. Prospective clients may also be additional users who may want to utilize the design are secondary users.

2.4 **Operating Environment**

The CubeSat Deorbiting Vehicle utilizes two Raspberry Pi running Raspbian as the operating system. Controls the CubeSat from the Ground System (GCS) will be through a website.

2.5 **Design and Implementation Constraints**

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Component Type:</th>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thrusters / SimPlat</td>
<td>Hardware</td>
<td>No actual propulsion of real thrusters, servo motors used instead.</td>
</tr>
<tr>
<td>Image Recognition</td>
<td>Software</td>
<td>Using the small raspberry pi results in high CPU load resulting in slower processing.</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>Hardware</td>
<td>Data transfer from one RaspberryPi to another takes time to send and process data.</td>
</tr>
</tbody>
</table>
2.6 Assumptions and Dependencies

- For Computer vision we will be relying on OpenCV
- We will assume an artificial velocity and acceleration speed for the SimPlat for demonstration purposes only.
- The connection between GCS and CubeSat will be at the discretion of the team.

2.7 Apportioning of Requirements

- Simulating 2D rotation in space will be delayed until SimPlat’s revision.
3. **External Interface Requirements**

3.1 **User Interfaces**

The GUI for Ground Control will be made in HTML with panels that will be created using a program called Grafana, allowing users to access Ground Control from any device with an internet browser. Using Grafana, panels will be created that will link to the Influx database and display appropriate data from the CubeSat. The entire layout of Ground Control will be done in HTML with JavaScript handling the backend and a node server handling communication between the CubeSat and JavaScript.

3.2 **Hardware Interfaces**

A Raspberry Pi will serve as a router for the project that the Ground Control System, CubeSat, and SimPlat will be connected to. By having all three connect to the same router, they will be able to communicate with each other. Using the router, the SimPlat will be able to receive commands from both the CubeSat and Ground Control System.

- **Emergency Stop** - The Ground Control System will be able to send an emergency stop command directly to the SimPlat through the router, resulting in SimPlat to cease all movement.
- **Manual Movement Commands** - The Ground Control System will allow users to send manual movement commands to the SimPlat via CubeSat. When the user chooses a movement command, Ground Control System will send the movement command to CubeSat which will then send the command to SimPlat.

The CubeSat itself is a Raspberry Pi 4 which has a camera built on. With the camera is a provided library available for use in python which will allow us to turn the camera on and have it send a live feed to the user. The live feed itself will be sent to Ground Control System by having the live feed running on a HTML page running on a local python server.

The SimPlat is utilizing a Raspberry Pi 3 that have servo motors connected to it to simulate movement. The Raspberry Pi will be running scripts that will utilize the ServoBlaster library in python, allowing the Pi to control the servos connected to its GPIO pins. By using ServoBlaster, we will be controlling the SimPlat’s wheels by adjusting the speed and direction the SimPlat moves.
Attached to the CubeSat is a 9 Degrees of Freedom sensor, or 9dof for short. The 9dof is a sensor that measures acceleration, magnetic field, and orientation and angular velocity using the built in 3-axis accelerometer, 3-axis magnetometer, and 3-axis gyroscope. The 9dof has a python library for us to use to extract its data and store it in our database.

3.3 Software Interfaces

We will be using Grafana v6.3.5 to design the panels that will display data received from the CubeSat, HTML and JavaScript to design a web page that will act as the Ground Control System for users, and InfluxDB v1.7 as the database. The scripts for the SimPlat and CubeSat will be coded in Python. Using the InfluxDB library in python, we’ll be writing a python script to get data from the CubeSat into the Influx database. The OpenCV library will be used for image recognition while the Raspberry Pi camera is active on the CubeSat. With the data flowing in from InfluxDB, Grafana will be utilized to create panels that will display the data being received from the CubeSat in some visual form for users to see. These panels will be implemented into the Ground Control display by using HTML taGCS.

3.4 Communications Interfaces

Socket.IO and InfluxDB’s connections both use WLAN to perform any communications. Since all components of this project will be connected to one local router, any information that is sent between Socket.IO or InfluxDB will be done through the router’s network.
4. **Requirements Specification**

4.1 **Functional Requirements**

4.1 Design Module: CubeSat

4.1-1 The CubeSat shall relay sensory data to the GCS.

4.1-2 The CubeSat shall use lidar sensors to detect obstacles in its path.

4.1-3 If an obstacle is 35 cm away from the CubeSat the CubeSat shall maintain a safe distance.

4.1-4 The CubeSat shall recognize a target object and be able to rendezvous with it.

   4.1-4-1 The CubeSat shall have two ways of recognizing target objects. (manual and autonomous mode)

   4.1-4-2 The CubeSat shall be able to receive a destination from the user move towards that area autonomously.

   4.1-4-3 The CubeSat shall be able to recognize aruco markers and navigate towards that object.

4.1-5 The CubeSat shall process and execute received commands from the GCS.

4.1-6 The CubeSat shall receive commands via socket from GCS.

4.1-7 The CubeSat shall send thruster commands to the SimPlat to move.
4.2 Design Module: SimPlat

4.2-1 The SimPlat shall imitate movements of objects in 2D space.

4.2-2 The SimPlat shall receive movement commands from the CubeSat.

  4.2-2-1 The SimPlat shall move in the opposite direction at a constant velocity when a thruster is fired in one direction.

  4.2-2-2 The SimPlat shall increase velocity if additional firing of thruster in same direction occurs.

  4.2-2-3 The SimPlat shall reduce velocity if thruster is fired in opposite direction.

4.3 Design Module: Ground Control System (GCS)

4.3-1 The GCS shall send commands to the CubeSat using socket based communication.

4.3-2 The GCS shall send emergency “Stop” command to the SimPlat.

4.3-3 The GCS shall receive sensory data from the CubeSat.

4.3-4 The GCS shall display information from the database.

4.3-5 The GCS shall display sensory data on a web-based dashboard (including the camera feed).

4.3-6 The GCS shall be developed using Grafana if possible.

4.3-7 The GCS shall display camera feed showing target as it is identifies and approaches target.
4.2 External Interface Requirements

1.1 Grafana

- The GCS shall incorporate Grafana’s dashboards.
- The GCS shall require data input from the database to display results on grafana.

4.3 Logical Database Requirements

1.1 Sensory Data

- The CubeSat shall send the sensory data from the 9dof to the influx database.
- The CubeSat shall send the sensory data from the lidar sensor to the influx database.
- The sensory data in the database shall be used to display information on the GCS.

4.4 Design Constraints

1.1 Hardware limitations:

1.1-1 Power:

- The system shall consist of two Raspberry Pi’s, which are limited to power.
- The system shall not be able to have high powered devices plugged into Raspberry Pi’s USB ports because it may cause a shortage/burnout.
- The SimPlat shall be powered by a 4100 mA 3 cell battery which is limited to the amount of power it can give, as well as the duration it can give that power for.

1.1-2 Memory:

- The CubeSat Raspberry Pi 4 is limited to 4GB of RAM.
- The SimPlat Raspberry Pi 3 B + has 1 GB RAM.
- The CubeSat and SimPlats Raspberry Pi’s cannot have RAM added to them.
1.1-3 Motion:

- The SimPlat shall be an omni-wheel vehicle that has full two-dimensional mobility and can rotate in place but cannot move and rotate simultaneously.
- The SimPlat shall be limited by the max speed of the servos.

1.1-4 Sensors:

- The CubeSat’s lidar sensors shall be limited to 30 cm. The sensor is unable to determine if an object is closer than 30 cm away.
- The lidar shall have a max distance it can read of 12 meters.
5. **Other Nonfunctional Requirements**

5.1 **Performance Requirements**
- The ground control system shall display sensory data and the camera feed from the CubeSat in real time.
- The CubeSat shall be able to process computer vision data at a fast enough rate to be able to determine live motion.

5.2 **Safety Requirements**
The ground control system shall incorporate an Emergency stop button that will send a direct message to SimPlat to stop its wheels.

5.3 **Security Requirements**

N/A

5.4 **Software Quality Attributes**

This product should be designed using object-oriented principles.

5.5 **Business Rules**

N/A
6. Other Requirements
N/A

Appendix A: Glossary

Refer to section 1.4 of section 1.