The University of Western Ontario

**Department of Electrical and Computer Engineering**

**Software Engineering Program**

# SE4450 - Software Design II Project Proposal

# Cubesat Flight Software

By Stephen Amey, Hyunjoon Kim, Anthony Teixeira, Rakul Janatharan

Faculty Supervisor: Dr. Ken McIsaac

Date: 2019-10-04

## Introduction and Problem Statement

The Canadian CubeSat Project (CCP) was announced in April 2017. The CCP is providing students and professors in 15 selected post-secondary institutions with an opportunity to engage in a real space mission. Through this national initiative, professors and students are offered the unique opportunity to design and build their own miniature satellite called a CubeSat. The CCP timeline outlines design to take place between January 2019 and August 2020, construction between August 2020 and September 2021, and testing/launch integration between September 2021 and November 2021. Student teams across Canada are now hard at work to design and build their CubeSats. Western University is in collaboration with Nunavut Arctic College and partnered with the CCP for this project. Western aims to launch a 2U CubeSat carrying two novel 180° VR cameras provided by Canadensys in order to create full 360° images viewable with a virtual reality headset. By late 2021, Western will work with NanoRacks to integrate the CubeSat into their NanoRacks dispenser pod, which will then be placed as one of the payloads on the rocket. The rocket will launch and deliver the CubeSat payloads to the International Space Station (ISS), where they will be prepared and ejected from the station. Each team of the CCP will then operate their satellites and conduct science according to the objectives of their missions, which could last up to 12 months.

The CCP's objectives are to increase students' interest in STEM, particularly in space domains, develop students' expertise in space domains, give students hands-on experience and prepare them to enter the job market, and advance space science and/or technology. Alongside these objectives, the CubeSat has future applications in Earth observation and space exploration for years to come.

A CubeSat is a complicated machine with many critical parts, and the flight software is integral to its functionality. Western needs a unique implementation for the CubeSat flight software. Western has just finished the preliminary design phase and now requires a team to develop the onboard flight software system that will interface with the other major subsystems of the CubeSat (attitude determination and control (ADCS), power, communications, and payload). Our team aims to absolve this requirement.

## Background Information

### What is a CubeSat?

A CubeSat, classified as a nanosatellite, is typically built of small discrete ‘Units’. These units are cubes of volume 10cm x 10cm x 10cm which gives them their name. Such satellites can be 1U (1 unit), 2U, 3U, or any number of units. In comparison to larger satellites, CubeSats are relatively low cost which makes them the ideal solution for testing new and novel technologies. As such, it is a way for small organizations and student bodies to engage in a real space mission while not having to bear the massive costs that are typically associated with space. CubeSats typically contain all of the major subsystems that are found on a large satellite such as communications, power systems, onboard computing, attitude determination and control, but with the added challenge of fitting it all into a compact space.

### History of CubeSats

In 1999, Jordi Puig-Suari of Cal Poly and Bob Twiggs of Stanford proposed a reference design for the CubeSat. These CubeSat requirements aimed to develop skills (for graduate students) necessary for the design, manufacturing, and testing of small satellites intended for low Earth orbit (LEO) that perform scientific research functions. The first launched CubeSat was placed into orbit in June 2003 via a Russian Eurockot. Until 2013, academia was responsible for most CubeSat initiatives. After 2013, more than half of CubeSat launches were produced by non-academic sources (ex. commercial projects, amateur projects). Some countries’ first national satellites are CubeSats. As of right now, a total of roughly 100 CubeSats have been placed into orbit. Many more of these satellites are planned to come. The CubeSat's concept has become popular mainly due to its simplicity, inexpensiveness, and ability to be built quickly.

### Review of past CubeSat projects

One of the first Canadian student teams to launch a CubeSat was the University of Alberta. Their CubeSat, the Ex-Alta 1, was successfully deployed on May 27th, 2017. The purpose of its mission was to collect information on space weather. This project was done in collaboration with the Canadian Space Agency providing the launch services, similar to the current Canadian CubeSat Project and the opportunity provided to Western University.

The logistics of the software is an important point for all CubeSat projects. Past projects have implemented their solutions using various operating systems such as Windows, Linux, freeRTOS. The needs of the software is typically a driving factor in the OS chosen, where there may be needs such as concurrent processing, data handling and storage, task scheduling and priority, communication between systems, and real-time data needs. The ITASAT 1 CubeSat built by Instituto Tecnológico de Aeronáutica (ITA) launched and successfully deployed in 2018 utilized freeRTOS.

## Objectives and Scope

### Goals and Scope

The goal of the project is to construct a CubeSat Flight Software (CFS) which will control the activities for the CubeSat that will be launched in 2021. Software for a testing board will also be developed in order to test how the CubeSat reacts given different sensor input. While the CubeSat is in orbit, there will be certain times of the day when it will meet with one of the ground stations and be capable of communicating. The command ground station will send out commands to the CubeSat, and the CFS will parse these commands and execute it while the CubeSat is in the orbit. Actions executed by the commands needs to be implemented into the CFS which has access to the subsystem components of the CubeSat. By the end of the project the CubeSat should:

- Initiate startup sequence on power-up (de-tumble and deploy antennas)
- Receive commands from the ground station such as taking pictures, rotate the CubeSat, send images to the ground station
- Send telemetry when in range of a ground station
- Read data from CubeSat sensors (photodiodes, temperature, GPS, electrical power system)
- CubeSat should rotate into charging position when necessary
- The CubeSat should passively roll to maintain even heating over all surfaces
- Occasionally orient towards GPS constellation satellites in order to update orbital parameters

First, the CubeSat will receive the command and start taking pictures of its surroundings. Second, the CubeSat will send the picture to the ground station. Third, CubeSat will send the telemetry data it has received from the sensors. Transmitting data and pictures from the CubeSat is only available when it is in the range of the ground station. However, the transmission cannot be done instantly, so CubeSat will stop and be able to resume its transmission from where they left off in the previous transmission. Lastly, CubeSat will rotate on its orientation by command (e.g. to take pictures). The CubeSat will also rotate on its own in the case of low power in order to charge its batteries. These actions will be scheduled by the CFS and regularly done when it is not in contact with the ground station. Also, these actions have different priorities which requires interrupts to be available. Interrupts can occur in multiple situations, such as when the CubeSat is taking pictures but it still has unsent data, so while in contact with the ground station it will simultaneously start transmitting data.

The scope of the project is that the time to fully transmit a single image from CubeSat to the ground station may take over two weeks, thus the software must be built to avoid transmission errors as much as possible. In the case of no transmission failures, once entirely received the images should be complete 180 degree images that will be stitched together on the ground in order for a complete 360 degree image from the CubeSat. Along with this, The CubeSat will also be sending basic telemetry from its systems and sensors, however because of its relatively small size compared to the images, the telemetry transmissions will be completed quickly.

The additional testing board software will simulate output from the various sensors located on the CubeSat and other subsystem components, namely the photodiodes, temperature sensors, antenna release, electrical power subsystem, communication subsystem, payload cameras, and the GPS unit.

## **Constraints**

The lifetime of the CubeSat will be roughly a year. The orbit will degrade over time due to various factors (atmospheric particles, orbit perturbations, solar pressure), and eventually reenter the atmosphere to get destroyed. Therefore, CubeSat must maintain accurate orbital parameters and must constantly update its orbital location to know when it will be destroyed.

The CubeSat Flight Software should be able to operate consistently over at least a year. There must built-in redundancy for data used during operation, as the CubeSat will experience strong solar radiation which may cause memory upsets.

The use of a task scheduler with priority must also be used to ensure tasks are properly executed at appropriate times. The CubeSat must also maintain a constant rotation in order to ensure even surface heating, except for when commanded to do otherwise. Ground station commands must also take precedence over the automated tasks of the CubeSat.

## Methodology/Process

The software development life cycle (SDLC) is an industry wide used practice that aims to help organize the completion of software projects with a high degree of effectiveness while meeting/exceeding customer expectations. Throughout this project our team will be using the Agile Scrum SDLC. The scrum method was chosen in particular as this management framework will help to implicitly organize our team with its clear-cut and defined artifacts, roles, and meetings. We are hoping to create cross functional teams that would create a shippable software product in sprint durations of two weeks. Below is a breakdown of how we will allocate and follow the selected parts of the scrum framework that works best amongst our team:

### Meetings

| Meeting Type | What it is? |
| --- | --- |
| Sprint Planning Meeting | This meeting will be used to discuss which product backlog items (PBI) will be selected for the next sprint. The PBI should be broken down into further sprint tasks. The selected PBI should be added to the sprint backlog. |

| | |
|---|---|
| Daily Scrum | An online meeting should occur every two days discussing the status of the sprint and PBI chosen. Further problems should be discussed. The sprint burndown list should be updated as needed. |
| Sprint Review Meeting | At the end of the sprint, a demonstration of the working software should be shown to the entire team. Product backlog should be further update and discussions of pros and cons shall occur. |

**Artifacts**

| Artifact Type | What it is? |
|---|---|
| Product Backlog | The product backlog should be a ranked list of desired functionality according to the team and project manager. This list may change if new requirements are found. |

| | |
|---|---|
| Product Backlog Item | A breakdown of each part of the product backlog, to help explain what is to occur after the completion of each item. Each PBI should be broken down into a few sprint tasks that may be written in user stories. |
| Sprint Backlog | Committed PBI's for the sprint should be added to the artifact. This will ensure that everyone is aware of which items are to be completed by the end of the sprint. |

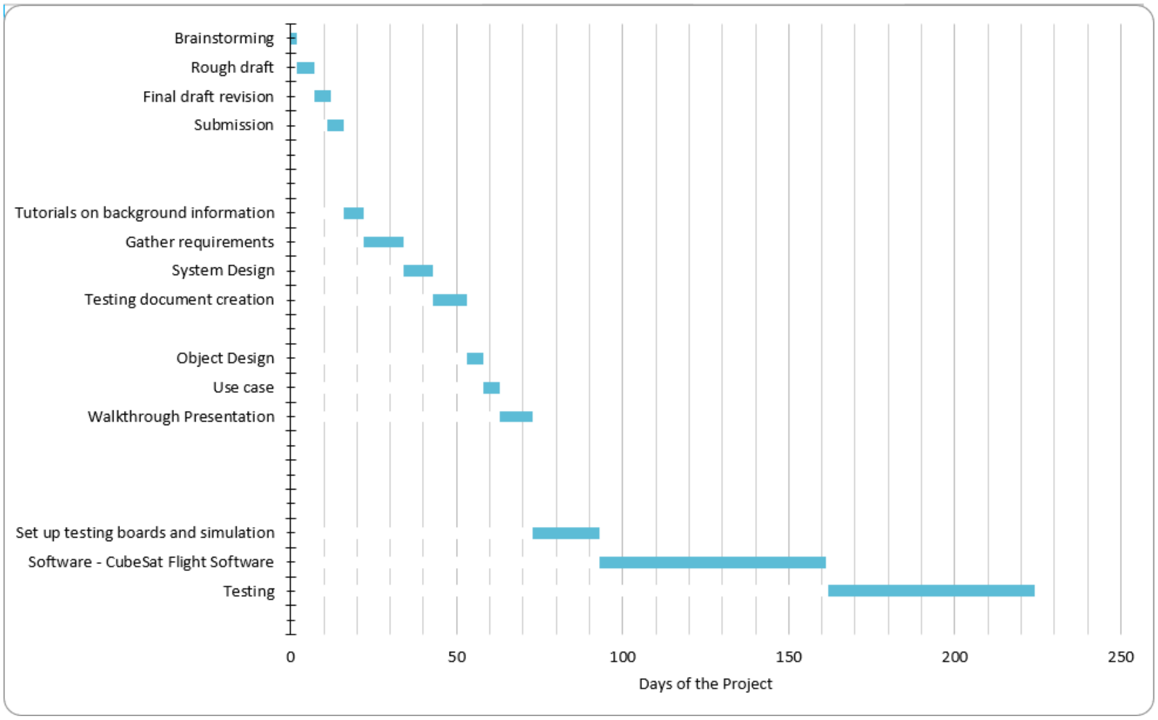
All team members should equally participate with all artifacts, meetings and roles. Responsibilities will be rotated throughout the project duration.

## Deliverables

- Project Proposal
- Project Walkthrough Presentation
- Mid-term Report
- Testing Board Software
- Working Prototype
- Final Product for demo
- Final Presentation
- Final Report

# Project Plan

## Gantt Chart (Start: Sept 19 2019, End: April 30 2020)




## Matrix of Responsibilities

| | Rakul | Stephen | Anthony | Hyunjoon |
|---|---|---|---|---|
| **Project Proposal** | | | | |
| Brainstorming | X | X | X | X |
| Rough draft | X | X | X | X |
| Final draft revision | X | X | X | X |
| Submission | | | | X |
| **Software Requirements & Design specification** | | | | |

| | | | | |
|---|---|---|---|---|
| Tutorials on Background information | X | X | X | X |
| Gather requirements | X | X | X | X |
| System Design | X | X | X | X |
| Testing document creation | X | | | X |
| Object Design | | X | X | |
| Use case | X | | | X |
| Walkthrough Presentation | X | X | X | X |
| **Build** | | | | |
| Set up testing boards and simulation | X | X | X | X |
| Software - CubeSat Flight Software | X | X | X | X |
| **Testing** | | | | |
| Unit Testing | X | | | X |
| Integration Testing | | X | X | |
| Documentation | X | X | X | X |
| **Final Report and Presentation** | | | | |
| Preparation for Demo | X | X | X | X |
| Demo for class | X | X | X | X |
| Preparation for Public Presentation | X | X | X | X |
| Final Report - rough draft | X | X | X | X |

| | | | | |
|---|---|---|---|---|
| Final Report - submission | X | X | X | X |