Lessons Learned from Development of Web-Based Mission Operations Software

Nathan Chow, Logan Pulley, Michael Lembeck
Laboratory for Advanced Space Systems at Illinois, University of Illinois at Urbana-Champaign

INTRODUCTION

The Steven R. Nagel Mission Operations Center (MOC) software system currently under development within the University of Illinois at Urbana-Champaign’s Aerospace Engineering Department’s Laboratory for Advanced Space Systems at Illinois (LASSI) utilizes a web-based application architecture to meet the unique requirements of orbiting CubeSats. Earlier mission operations solutions supporting LASSI’s CubeSats were decentralized and dedicated to individual missions. The new MOC software system is designed with several primary goals: to incorporate all aspects of mission operations, from commanding to data visualization and trend analysis; support for traditional workstations as well as mobile device like smartphones and tablets to increase data accessibility; and it addresses maintainability, expandability, and flexibility to maximize capability and to support evolving requirements for all present and future LASSI’s CubeSats.

This paper enumerates the features of the web-based MOC software system and presents a high-level description of its current architecture. Also discussed is the development pipeline that is used to implement parts of our Agile-like design philosophy for continuous, rapid, and responsive development.

WEB APPLICATION FEATURES

Expandable and Flexible

While the overall data storage method and associated logic layer are expected to remain largely static between missions, the front-end is constantly evolving based on payloads and customer feedback. Web-based applications neatly separate the server-side back-end and client-side front-end operations.

A critical feature of front-end and back-end separation is a standard data access mechanism. Both REST API and WebSocket communications protocols facilitate data access.

Maintainable

LASSI’s MOC software maximizes usage of mainstream and open-source web technologies. It is developed on the popular Django framework which provides much of the base code and services needed to run a web server. Front-end static files are written using the common trio of JavaScript, HTML, and CSS. These tools should be familiar to many web-developers with extensive documentation publicly available online.

Accessible

Data visualization through the MOC webpage is not restricted to consoles in the lab. Users with authorized accounts can access satellite data anywhere they have an internet connection.

SERVER-SIDE ARCHITECTURE

The primary job of the MOC web server is to handle incoming telemetry downlinked from the satellite and passed through a parser, handling WebSocket and HTTP requests to view the data from web page clients. Models define PostgreSQL database tables and relationships described as Python classes in the Django framework. Raw telemetry from the downlink chain is sent to the parser, which makes POST requests to the web server with timestamped data. The Django framework processes the POST requests and stores the information in the database. It also pushes telemetry immediately to the client through the WebSocket connection. The client web page can also make HTTP requests for submitting form data or requesting telemetry stored in the database.

CLIENT-SIDE ARCHITECTURE

A client-side web page is the presentation layer and main interface between LASSI operators and data downlinked from CubeSats. It supports visualization of data from user selected satellite, subsystems, and time range. The client-side architecture separates functionality into several layers to improve organization and maintainability. A publish and subscribe (pub/sub) bus facilitates communication between the different layers. With this system, modules do not directly talk to each other and instead pass information, such as UI events and data requests, through messages. The pub/sub bus eliminates the need for tightly integrated inter-modular communication, increasing the modularity and scalability of the front-end architecture.

DEVELOPMENT OPERATIONS

Development Pipeline

Development on the MOC software system follows an Agile-like process. Source control, issue tracking, and automated CI/CD are primarily executed using GitLab tools. GitLab’s issue tracking features are used to manage and track work on new features and bugs. Every issue receives its own merge request. Our development pipeline evolves as the project grows. Our initial process involved only development testing. As the code base and team grew, code review, automated testing, and linting were introduced to ensure quality and robustness. Our development pipeline is now a critical part of the speedy and consistent delivery of the MOC software system to production.

Containerization

MOC utilizes Docker to containerize MOC applications. During the build stage of the CI/CD pipeline, applications and services that are a part of the MOC software system are packaged into images and pushed to either the development or production server Docker registries. From there, the latest images are automatically pulled and executed in containers.

CONCLUSIONS

Our incremental development philosophy means that the MOC software system is able to support LASSI’s upcoming missions as students continue implementing new features to place a centralized, modern, and easy-to-use mission operations and control solution at the hands of LASSI’s operators.

ROADMAP

Development of the MOC software system is a work in progress with new features under development and bug fixes addressed through testing. While we have not yet implemented the full intended feature set, the current version is ready to support operations of LASSI’s first missions. The infrastructure for handling downlinked telemetry from the CubeSat bus systems and payloads, as discussed in the section on server-side architecture, is in place. On the client-side web page, data presentation pages for viewing requested telemetry in graphic and tabular formats are available.

Ongoing and future work include optimizations on existing features such as the data visualizations page and better support for viewing and interaction on smartphones and tablets. An interface allowing LASSI operators to send spacecraft commands from the MOC software system web page is also planned. Work to support mission-specific requirements will also continue alongside the development of enhancements and new features.

CONTACT

Nathan Chow
Email: nwchow2@illinois.edu
Phone: (703) 226 9277

DEVELOPMENT PIPELINE