**Horizon Simulation Framework**

**Development of User Interface and Testing Harness**

William Albertini and Jacob Amezquita

---

### HSF Introduction

The Horizon Simulation Framework (HSF) is a tool which can be used in the planning portion of a mission design. The simulation algorithm relies on user supplied information about the system, and creates a list of schedules that the system may perform. It uses an exhaustive approach and explores all possible schedules in order to assess the feasibility of a mission. This style of dynamic modeling allows for cross disciplinary uses. Notable research utilizing this software includes UAV swarm vehicle performance, CubeSat astronomy mission modeling, and autonomous thermal soaring. While it has been used primarily in aerospace applications, it can be extended to model most time driven systems.

---

### HSF User Interface

Developing a user interface (UI) is of paramount importance across various domains and applications due to its pivotal role in enhancing the user experience. It is instrumental in bridging the gap between complex underlying systems and end-users, ensuring that technology is accessible and usable. The problem was that HSF didn’t have a UI. Therefore, creating a one was essential for optimizing user experience and facilitating effective interaction with technology. However, it poses challenges such as addressing adapting to changing requirements, ensuring compatibility, gathering feedback, and ensuring security. The UI also needed to serve as a host for the HSF. It would display each subsystem and their dependency on other systems.

Creating the UI involved using two programming languages, JavaScript and C#, which were both developed with HyperText Markup Language (HTML). With plenty of diverse examples of UIs given from Bootstrap and Electron, the initial dashboard was a simple display that allowed the user to navigate through different pages. Although this was useful, it did not have the functionality to parse XML files and display each subsystem with their respective attributes. To resolve this, I used the Node.js library to be able to properly show each subsystem given from the initial output of HSF. Future iterations will be able to not only display the data but create graphs and models from it.

---

### HSF Testing Harness

The simulation software is written in C# and is precompiled before use. It utilizes an open-source project, IronPython, to allow integration of Python code into the .NET framework. This presents the possibility of the scripted code breaking the compiled HSF program. Errors are not produced natively by the Python interpreter and can be difficult to debug. This makes testing Python methods/objects difficult, as they rely on native HSF objects. To solve this problem, a basic web app was developed as a proof of concept for assisting the user in building usable Python code. The app uses the Python library Flask, which is a micro web application framework. The application provides a text editor feel and assists with subsystem creation and testing in a Python runtime.

---

In order to model a given system, HSF must be provided three documents from the user. These documents include a target deck (tasks which the system will perform over the course of simulation), simulation parameters, and model parameters. The documents require an XML format and some specific naming connotations, making them tedious to write. The software can also accept scripted accessories, such as subsystems, written in Python3. This allows for large amounts of user customization, and in the case of subsystems, is guided by a subsystem abstract class written in C#. The software is currently set up to rely on native HSF objects. To solve this problem, a basic web app was developed as a proof of concept for assisting the user in building usable Python code. The app uses the Python library Flask, which is a micro web application framework. The application provides a text editor feel and assists with subsystem creation and testing in a Python runtime.

---

This research was funded by Lockheed Martin.