Designing and building flight software is a deeply involved task, often made more complicated by lack of hardware support and few preexisting software solutions. Leveraging modern framework designs and patterns has the potential to reduce development complexity, ensure greater safety and increase software reusability.

**PORTABILITY LAYERS**
Central to designing a reusable framework is the idea of cross-platform compatibility. The framework must understand and provide layers of abstraction for differences in hardware platforms. The operating system (Linux or FreeRTOS) is included in the framework and must be ported to each platform. The OS abstraction layer provides a singular interface for common OS functionality. The hardware abstraction layer speaks with various peripheral interfaces.

**SERVICES LAYERS**
A major component of the framework is providing functionality which is common to most small satellite missions. In order to maximize flexibility and modularity, these common features are implemented in the form of individual services. Each service can exist independently within a framework-driven system. The system provides a common data bus used to connect services and exchange communications.

**HIGH LEVEL FRAMEWORK**
Designing a software framework requires an evaluation and understanding of the surrounding context and requirements. Flight software operates in the context of a small satellite mission. Within this context are three main components: mission hardware, common flight software functionality and mission specific functionality. A well designed framework will provide reusable common functionality and can be leveraged for building mission specific features.

**SATELLITE SOFTWARE STACK**
The framework requirements must then be examined and broken down into architecture layers which allow for cross-platform compatibility and modular software development. The mission application will rely on and extend the functionality supplied by these layers.