F Prime: An Open-Source Framework for Small-Scale Flight Software Systems

Robert Bocchino, Timothy Canham, Garth Watney, Leonard Reder, and Jeffrey Levison

32nd Annual Small Satellite Conference
August 9, 2018
Utah State University

Copyright © 2018 California Institute of Technology
Government sponsorship acknowledged
Introduction

• Developing flight software (FSW) is challenging
• Especially so for small spacecraft
  – Small budgets
  – Ambitious goals
• Challenges include
  – Compressed schedules (especially test)
  – Inadequate resources
  – Poorly-specified interfaces
  – Under-specified and changing requirements
Introduction

Developing Flight Software (FSW) for Small Spacecraft

• Options
  1. Develop FSW from scratch
  2. Adapt FSW from a previous mission
  3. Use a multi-mission FSW framework

• We contend that option (3) is best
  – Option (1) is too expensive and/or compromises quality
  – Option (2) can work, but it is not ideal
    • Unless FSW is designed for reuse, it is difficult to reuse
    • Developers must re-engineer it for the new mission
The F Prime FSW Framework

Overview

• Free and open-source; developed at JPL
• Tailored to small-scale systems
  – CubeSats, SmallSats, instruments
• Comprises several elements
  1. A component-based architecture
  2. A C++ framework providing core capabilities
  3. Modeling tools for specifying models and generating code
  4. A collection of ready-to-use components
  5. Testing tools for unit and integration testing
• Runs on a wide range of hardware platforms
• Runs on several OSs (e.g., Linux, Mac OS, VxWorks)

https://github.com/nasa/fprime
The F Prime FSW Framework

Architecture

• Based on components, ports, and topologies
  – **Component**: A unit of FSW function (like a C++ class)
  – **Port**: A point of connection between component instances
  – **Topology**: A directed graph of instances and connections

• Component instances
  – Communicate only through ports
  – Have no compile-time dependencies on other components

• Port connections
  – Are typed and statically specified
  – May be synchronous or asynchronous

*Provides structure to FSW applications*
*Enables automatic checking of correctness properties*
*Enhances reusability of FSW components*
The F Prime FSW Framework

**C++ Framework**

F Prime auto-generates a C++ base class from a high-level specification.

The developer fills in handler functions with application-specific code.

---

**Base Class**

- **Port 1**
  - Asynchronous
  - virtual port1Handler(…) = 0
  - port1Handler(…) {
    - ...
    - }

- **Port 2**
  - Synchronous
  - virtual port2Handler(…) = 0
  - port2Handler(…) {
    - ...
    - invokePort3 (…)
    - ...
    - }

---

**Thread**

- **Port 3**
  - Developer-written
  - Auto-generated

---

**Implementation Class**
The F Prime FSW Framework

Modeling and Code Generation (Components)

MagicDraw Model ➔ XML Specification

```
<component name="CmdDispatcher"
          kind="active"
          namespace="Svc">
  <import port_type>
    Fw/Cmd/CmdPortAi.xml
  </import port_type>
  ...
  <comment>
    A component for dispatching commands
  </comment>
  <ports>
    <port name="compCmdSend"
          data_type="Fw::Cmd"
          kind="output"
          max_number="$CmdDispatcherCommandPorts">
      <comment>Command dispatch port</comment>
    </port>
    ...
  </ports>
  ...
</component>
```

C++ Base Class ➔ Ground Dictionaries
The F Prime FSW Framework

Modeling and Code Generation (Topologies)

MagicDraw Model  ➔  XML Specification  ➔  Ground Dictionaries

Python code for F Prime ground data system

Mission-specific ground data system formats

```xml
<assembly name="Ref">

<!--
Svc/CmdDispatcher/CmdDispatcherComponentAi.xml
</import_component_type>

<import_component_type>
Svc/CmdSequencer/CmdSequencerComponentAi.xml
</import_component_type>

<instance namespace="Svc"
  name="cmdDisp"
  type="CmdDispatcher"
  base id="121"
  base id window="20"/>

<instance namespace="Svc"
  name="cmdSeq"
  type="CmdSequencer"
  base id="541"
  base id window="23"/>

<connection name="Connection37">
  <source component="cmdSeq"
    part="cmdResponseOut"
    type="CmdResponse" name="9"/>
  <target component="cmdDisp"
    part="compCmdStat"
    type="CmdResponse" name="9"/>
</connection>

</assembly>
```
The F Prime FSW Framework

Reusable Components

• F Prime comes with over 20 reusable components (and counting)
• The components provide many standard FSW behaviors
  – Commanding
  – Events and telemetry
  – Ground interface
  – File system
  – Memory management
  – Generic data storage
  – Parameters (updatable constants)
  – Time
  – Health
  – Assertions and fatal events
• Fully unit-tested and ready to go
Provides a simple solution for testing F Prime components
The F Prime FSW Framework

Ground Data System and Integration Testing

Flight System

- F Prime Deployment

Ground System

- Graphical User Interface (GUI)
- Python API
- Automated Integration Tests
- Threaded TCP Socket Server

Provides a simple solution for testing F Prime deployments
The F Prime FSW Framework

The Ground Data System GUI

Commanding View

Telemetry View

Strip Chart View
Experience with F Prime

Missions and Projects

- We have used F Prime on several space missions
  - **ISS RapidScat** scatterometer (flew)
  - **ASTERIA** CubeSat (flying now)
  - **Mars Helicopter** (in development)
  - **Lunar Flashlight** CubeSat (in development)
  - **Near Earth Asteroid (NEA) Scout** CubeSat (in development)

- We have used F Prime for research and education
  - JPL R&D project on autonomous FSW
  - Collaborations with several universities

- F Prime can reduce the cost of developing FSW
  - Facilitate sharing and reuse between projects
  - Let developers focus on mission-specific code
Enhancements in Progress

Making F Prime Better

• Modeling and code generation
  – New input language and visualizer for F Prime models
  – It will be free and easier to use than MagicDraw/SysML

• Testing of F Prime components
  – Tools for automatically picking test inputs
  – Tools for generating tests from high-level specifications
  – Integrated model checking with Spin

• Ground data system
  – XTCE ground dictionaries
  – Mobile user interface
  – Improved server using ZeroMQ
Conclusion

• Developing FSW for small spacecraft is hard

• F Prime can help
  – Architecture
  – Direct code reuse
  – Development ecosystem

• F Prime is a flight-proven technology

• Several enhancements are in progress

https://github.com/nasa/fprime