F`Software Framework

A Small Scale Component Framework for Space

Jet Propulsion Laboratory,
California Institute of Technology

1/18/2018

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology,
under a contract with the National Aeronautics and Space Administration.

This software has been approved for open source release under NTR #49404.

What is F`?

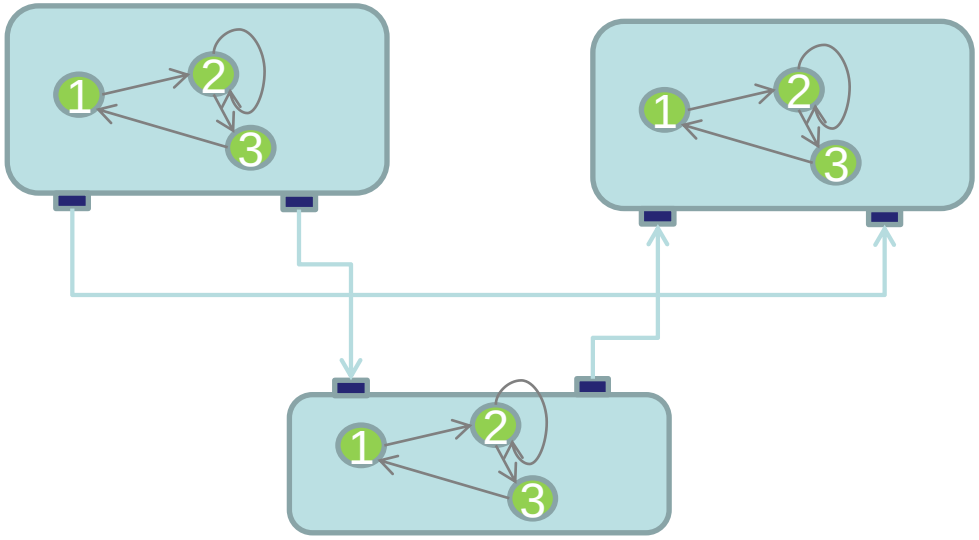
- F` Flight Software Framework
  - Targeted for instruments, CubeSats and other smaller platforms
  - Currently baselined for JPL Sphinx Leon3 Avionics SOC
- A component-based architecture as well as a software framework to support it
  - Uses the concept of software components
- Designed from the ground up to be compact and reusable
- Includes framework, code generators, build tools, Command/Telemetry GUI, and unit test environment
- Designed to make it easier for developers to concentrate on mission-specific logic rather than common implementation patterns.

Where is it being used?

- Development
  - Developed under JPL technology exploration task (2013)
  - Matured under a number of JPL projects (2014-2017)
  - Using established JPL flight processes/analysis tools
- Flew on RapidScat (2014-2016)
  - Radar experiment on ISS
  - Very stable with no reported software bugsf
- Flying on Asteria (Cubesat)
  - Asteroid detection technology demonstrator
- In development for:
  - Mars Helicopter Technology Development
  - Lunar Flashlight (Cubesat)
  - NEAScout (Cubesat)
- Available on GitHub
  - Reference example can be run on Linux, MacOS, Cygwin and most embedded ARM processors (e.g. Raspberry Pi)
  - https://github.jpl.nasa.gov/FPRIME/fprime-sw.git

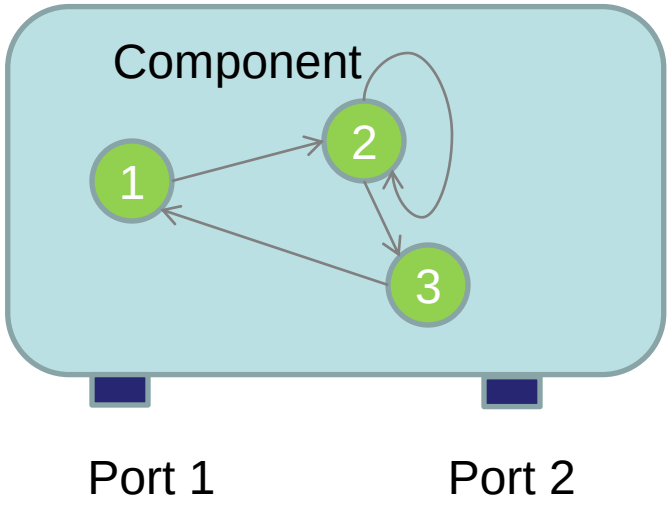
F`: A Reusable Component Architecture

- Consists of components (behaviors) and ports (interconnections for data)
- Components are not dependent on other components, so can be easily reused.
- Components to fulfill different requirements (simulation vs. actual) can be substituted, even at run time.
- Components can have generic roles (commanding, telemetry, storage) which are not dependent on specific applications.



F`: A Framework for quick development

- F` provides a C++ framework and code generator that encapsulates:
  - Thread management
  - Inter-Process communication (IPC)
  - Commanding
  - Telemetry
  - Parameters
- Developer specifies common patterns in simple XML.
  - Code generator generates boiler-plate code.
  - Developer concentrates on domain-specific code.
  - Framework invokes user code automatically



F`: A Framework for reuse

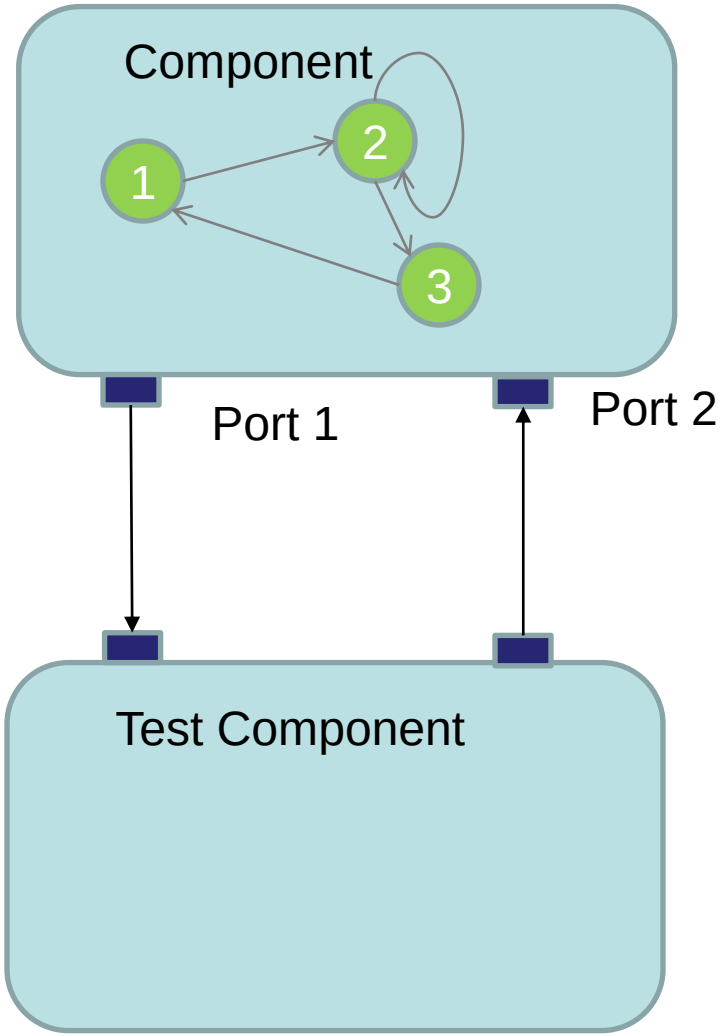
- Over time, a library of reusable components are being built:
  - For common facilities:
    - Rate group management
    - Command dispatching/sequencing
    - Telemetry storage
    - Ground interfaces
  - For specific hardware platforms:
    - Device drivers
    - Radios
    - GNC devices
    - Operating system adaptations
- A reusable ground system can be used
  - Framework has uniform data representations
  - Can be adapted to existing ground systems
    - Runs on JPL multi-mission ground system
  - Python-based lightweight ground system is provided with code

F`: A Portable Framework

- Code base is in portable, embedded C++
- Has abstraction layer for OS facilities such as:
  - Threads
  - Synchronization
  - Files
  - Time
- Data products are stored and transmitted in a portable representation
  - Allows interaction with ground system no matter the processor architecture
- Has been run on the following processor architectures:
  - X86, PPC, ARM, MSP430, Leon3
- Has been run on the following OSes:
  - VxWorks, RTEMS, Linux, MacOS, Cygwin, Raspberry Pi Raspbian
- Very compact
  - Framework classes ~1K compiled

F`: A Framework for testing

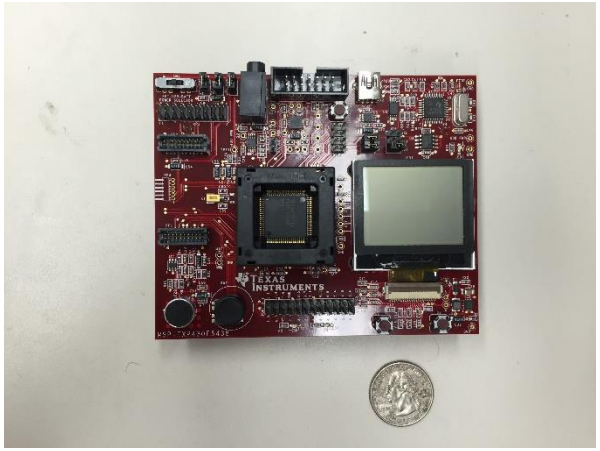
- F` components are decoupled from each other, so unit testing is easier
- F` code generator generates counterpart test component that can be connected.
- Test component “knows” the interfaces, commands, and telemetry
- Tester can invoke generated C++ functions to exercise component interfaces, commands.
- Telemetry automatically decoded and stored for checking in test component.



F`: A Flight-ready Framework

- In 2015-2016, C&DH components were taken through flight software processes
  - Design, coding and testing reviews with LARS tools and code coverage
    - Design and code reviewed by peers
    - Code scrubbed by static analyzers (e.g. Coverity)
    - 100% coverage except certain assertions (default switch, etc)
    - Delivered with repeatable automated unit tests
  - Includes:
    - Rate Groups
    - Command handling
    - Command Sequencer
    - Telemetry Processing
    - Parameter storage
    - Event handling
    - File Uplink/Downlink
    - Telemetry Database
    - Health Monitor
    - File Manager
    - Socket “Ground” interface

Both Ends of the Scale



TI MSP430 Microcontroller
- 24K RAM
- 64K Flash

Rack Mount PC
- Quad-core Xeon
- 8GB RAM
- Hard disk