CubeSat Flight Software Workshop

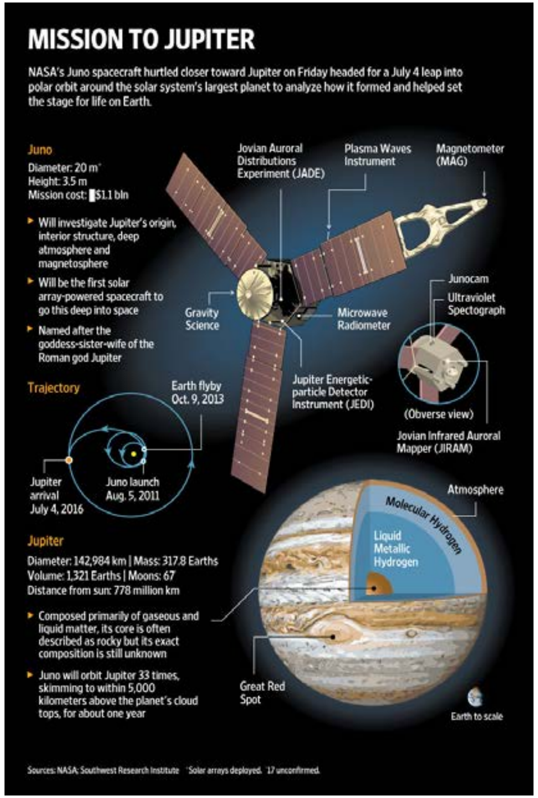
Flight Software System Engineering And Architecture

Timothy Canham
Flight Software Engineer
June 3, 2019

Jet Propulsion Laboratory
California Institute of Technology

Flight Software System Engineering and Architecture

- **Definition** – Designing the various functions of the flight software and their interactions with the spacecraft and each other
- Understanding the mission objectives
  - What does the mission want to accomplish?
- Understanding the spacecraft system
  - How is the spacecraft design going to enable accomplishing the objectives?
- Concept of Operations
  - How are you going to operate the spacecraft to achieve the objectives?



MISSION TO JUPITER

NASA's Juno spacecraft hurtled closer toward Jupiter on Friday headed for a July 4 leap into polar orbit around the solar system's largest planet to analyze how it formed and helped set the stage for life on Earth.

Juno

Diameter: 20 m*
Height: 3.5 m
Mission cost: $1.1 bln

- Will investigate Jupiter's origin, interior structure, deep atmosphere and magnetosphere
- Will be the first solar array-powered spacecraft to go this deep into space
- Named after the goddess-sister-wife of the Roman god Jupiter

Jupiter

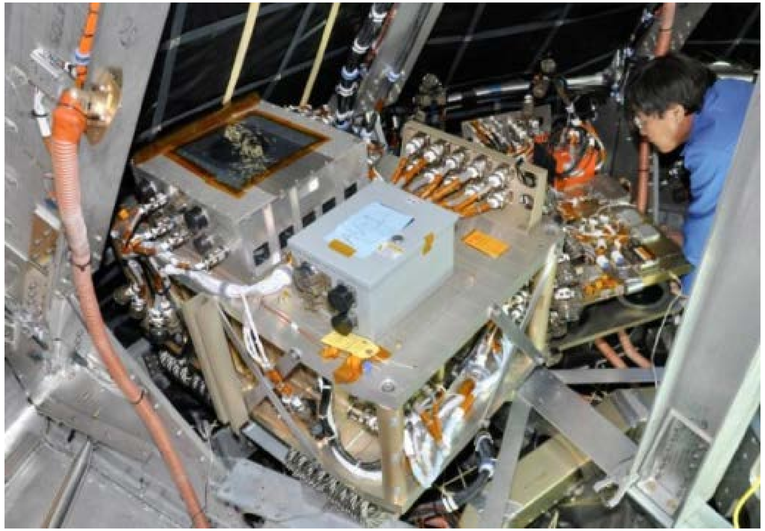
Diameter: 142,984 km | Mass: 317.8 Earths
Volume: 1,321 Earths | Moons: 67
Distance from sun: 778 million km

- Composed primarily of gaseous and liquid matter, its core is often described as rocky but its exact composition is still unknown
- Juno will orbit Jupiter 33 times, skimming to within 5,000 kilometers above the planet's cloud tops, for about one year

Sources: NASA; Southwest Research Institute *Solar arrays deployed. 17 unconfirmed.

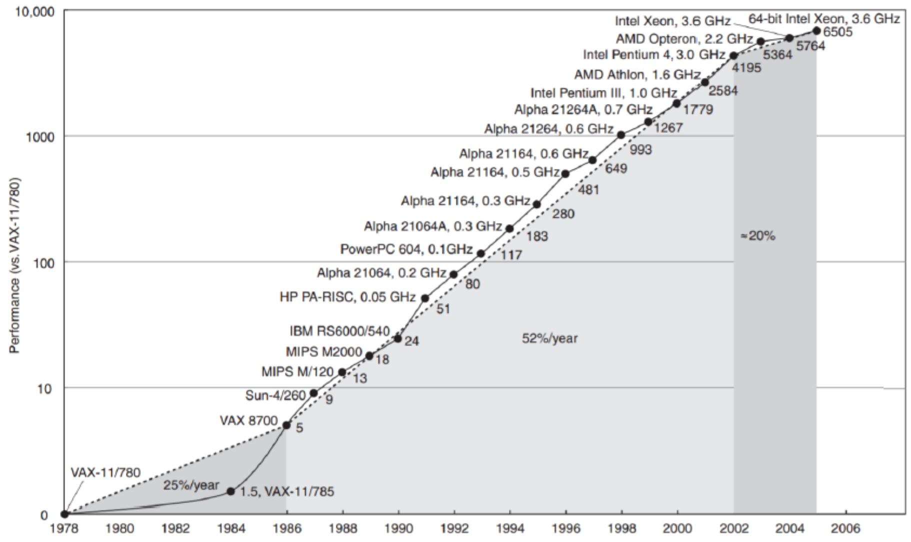
Flight Software System Engineering and Architecture

- Understanding the interfaces to the system
  - How does it interact with the environment?
  - How does it interact with payloads?
  - What is the avionics platform?
  - What is the execution environment?
    - RTOS, Non-RTOS, bare metal


Orion

Flight Software System Engineering and Architecture

- Understanding the requirements
  - Performance
    - Deadlines
    - Computing load
    - Data throughput
  - Resources
    - Energy
    - Memory
    - Storage
  - Activities
    - Behaviors to complete mission
  - Testability



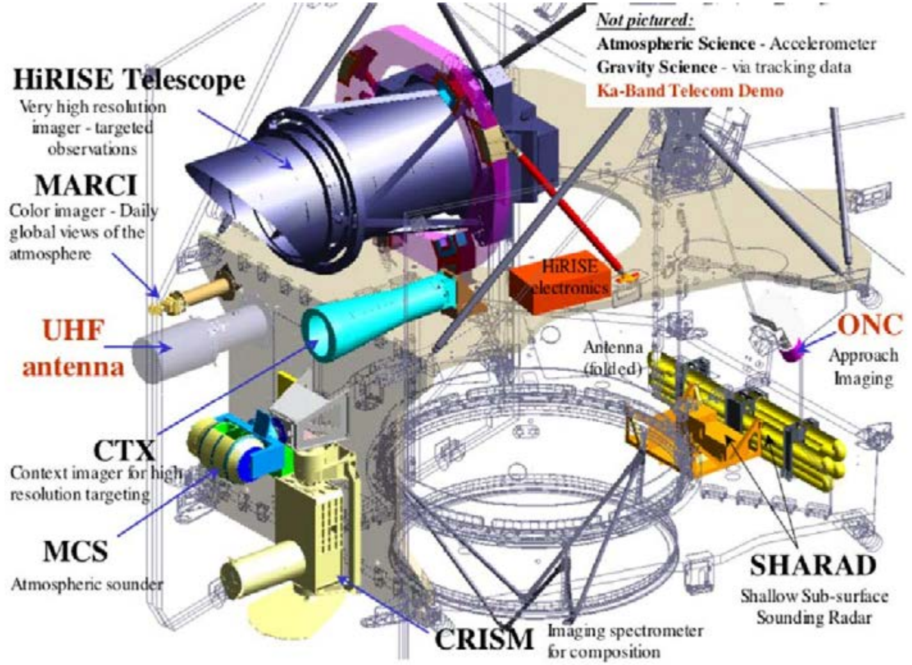
Moore’s Law

Spacecraft Subsystems – Hardware Layering

Peripheral
(Payload, Power,
Telecom, etc)

I/O
Device

Processor



Not pictured:
Atmospheric Science - Accelerometer
Gravity Science - via tracking data
Ka-Band Telecom Demo
HiRISE Telescope
Very high resolution imager - targeted observations
MARCI
Color imager - Daily global views of the atmosphere
UHF antenna
CTX
Context imager for high resolution targeting
MCS
Atmospheric sounder
HiRISE electronics
Antenna (folded)
ONC
Approach Imaging
SHARAD
Shallow Sub-surface Sounding Radar
CRISM
Imaging spectrometer for composition

MRO Spacecraft

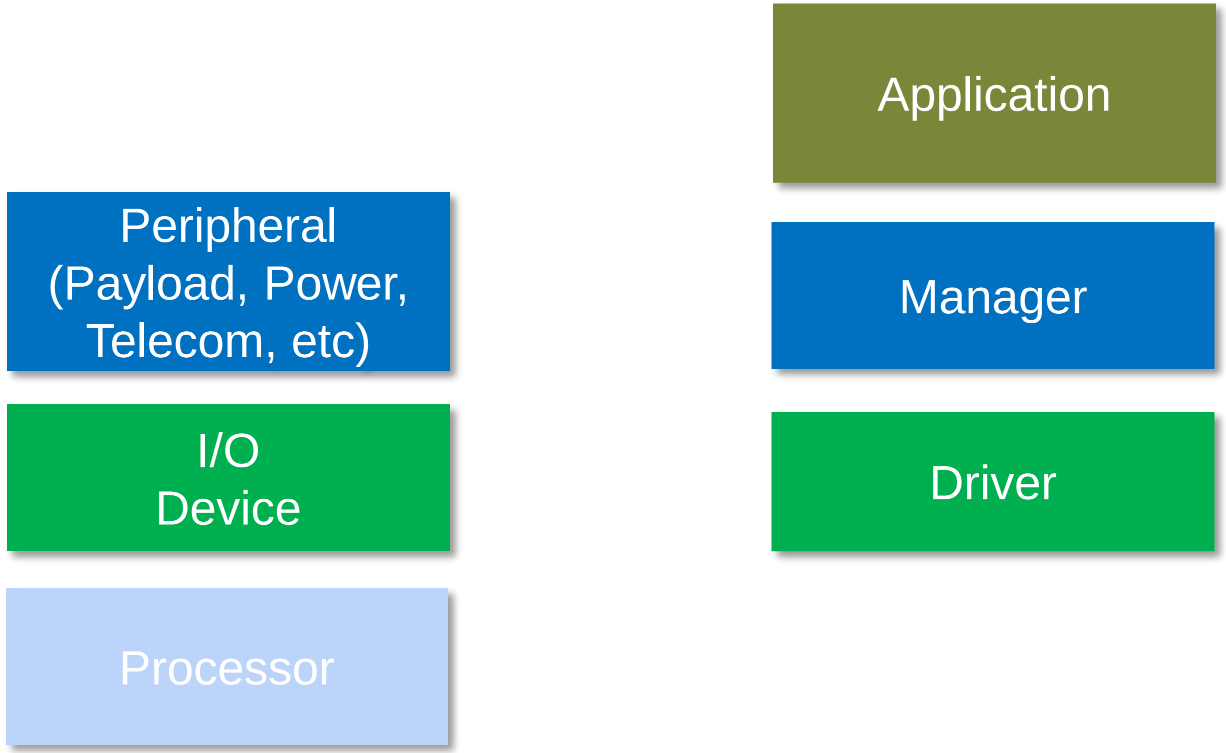
Flight Software System Engineering and Architecture

- Hardware layers are designed (for the most part) to be modular and reusable
  - Peripherals can be reused
  - I/O can be reused
  - Processors can be reused
  - Layers can be tested independently
  - Clear interfaces
- Software
  - Software should be layered for the same purpose
  - Modularity
  - Reusability
  - Testability

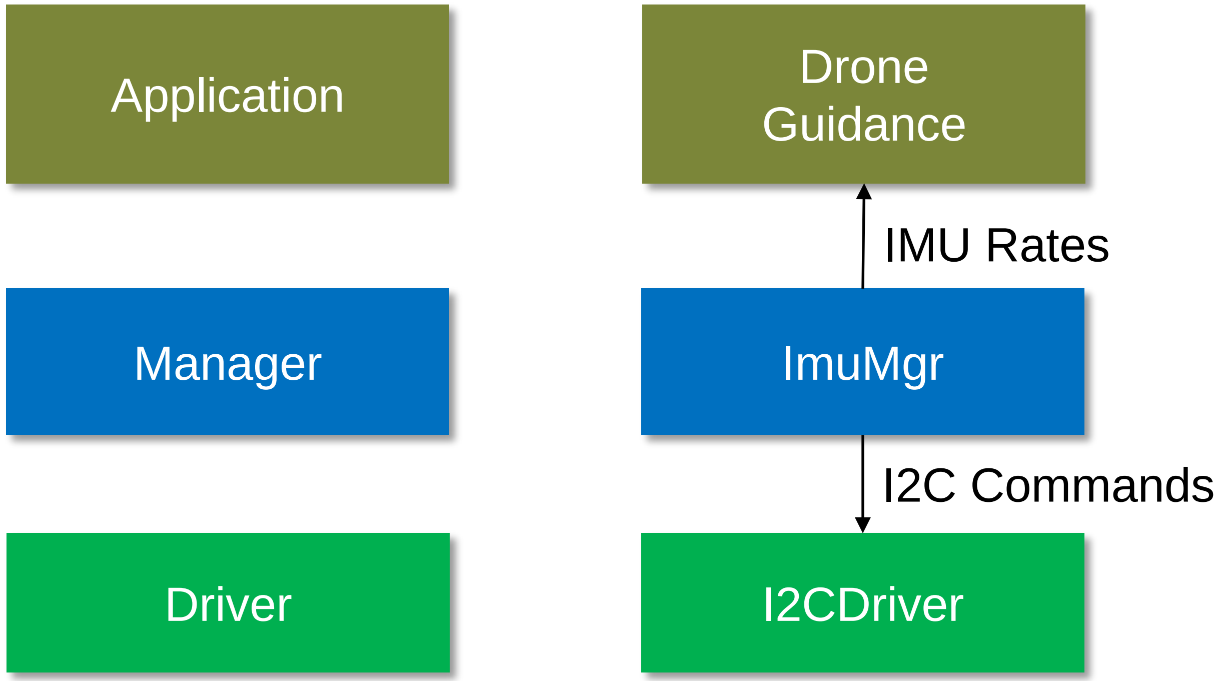
Software System Architecture

- Decompose software into modules
  - Separation of function
  - Definition of interfaces
  - Behavioral characteristics
    - Rate groups
    - Concurrency
    - Data flow
  - Test at the unit level
  - Ownership

Spacecraft Subsystems – Software Layering



Example - IMU



Adafruit I2C IMU

Software Layers

- Driver
  - Dedicated to a particular hardware interface
  - Written to interact with that device only
    - Interrupt driven, polling
  - Should not need to know use of hardware
  - Provide abstracted interface
- Manager
  - Manages a particular peripheral
  - Uses the interface driver at abstract level
  - Should not know the use of the peripheral
  - Also provide abstracted interface
- Application
  - Implements mission specific use of the peripheral (and others)
  - Should not know about the driver to get to the peripheral
- Calls should be down the layers; no upcalls

Software Layers

- Layering allows:
  - Separation of concerns
    - Only implement logic for that layer
  - Reusability
    - Reuse software modules on other projects
  - Replacement
    - Abstraction allows layers to be replaced with other implementations with same interface
    - New driver or manager
    - Simulation
  - Testing
    - Layers can be tested at their level with stubs for lower levels
  - Fault protection
    - Faults can be handled at the appropriate layers

References (shortened via bitly.com)

- Juno - https://yhoo.it/2QmFedE
- Orion - https://bit.ly/2YVga0B
- Moore’s Law - https://bit.ly/2K4nZgc
- MRO - https://bit.ly/2VUQH5D
- IMU - https://bit.ly/2jF4rlo

Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov