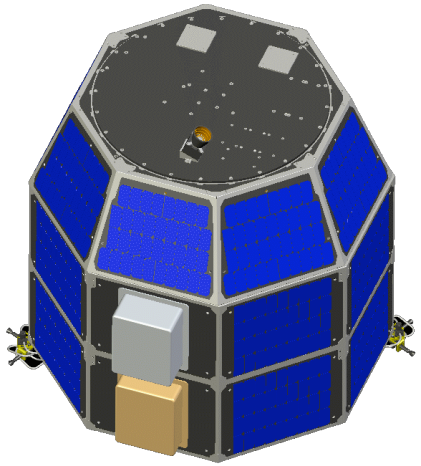
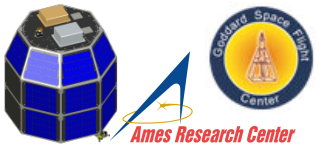


Modular Infrastructure for Rapid Flight Software Development

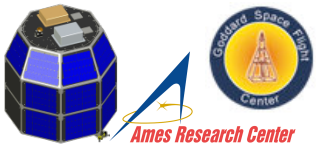
Howard Cannon
Craig Pires

Overview



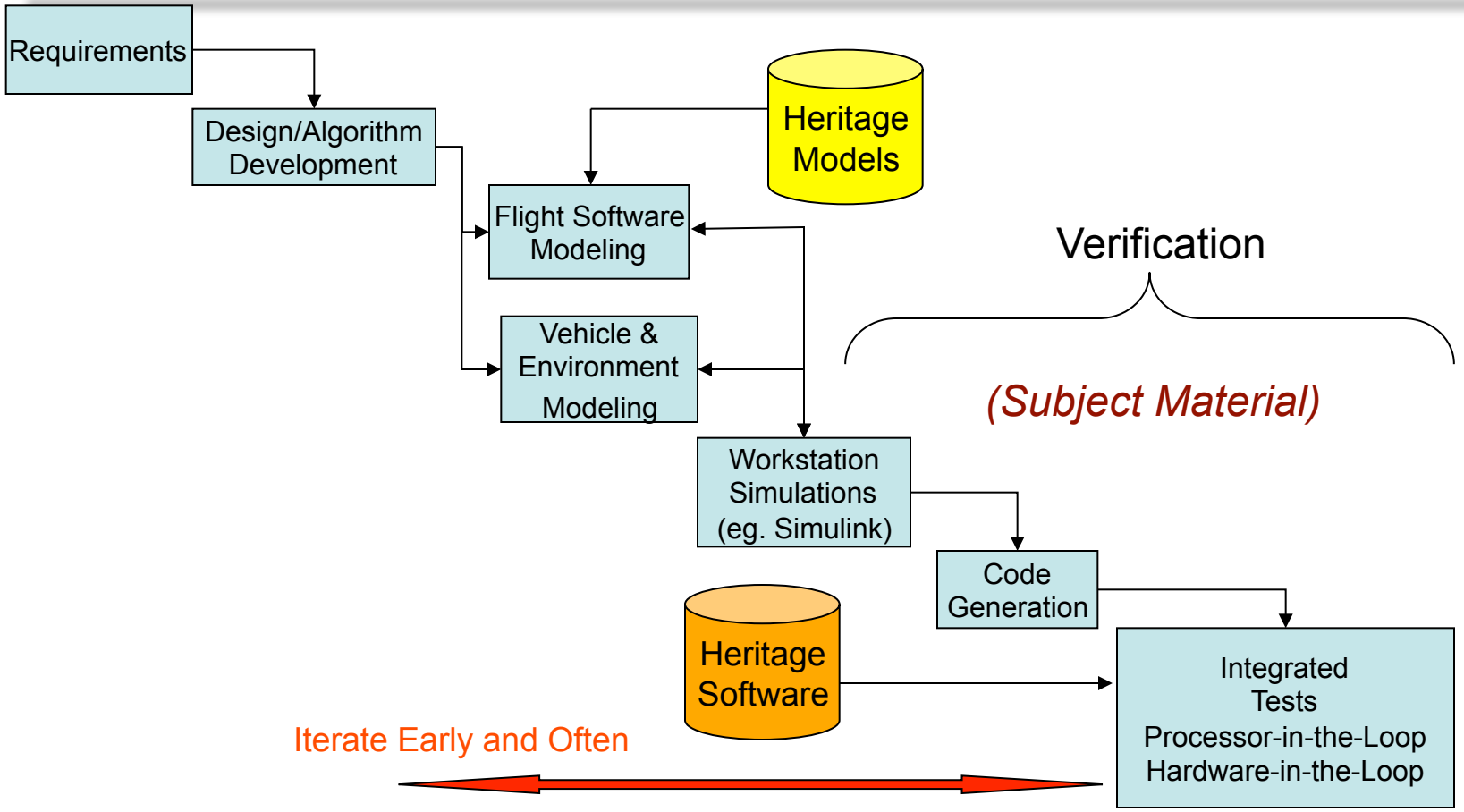
- History
- Simulink Modeling
- CFE
- Modular integration of Simulink into cFE

Modular Model Based Design



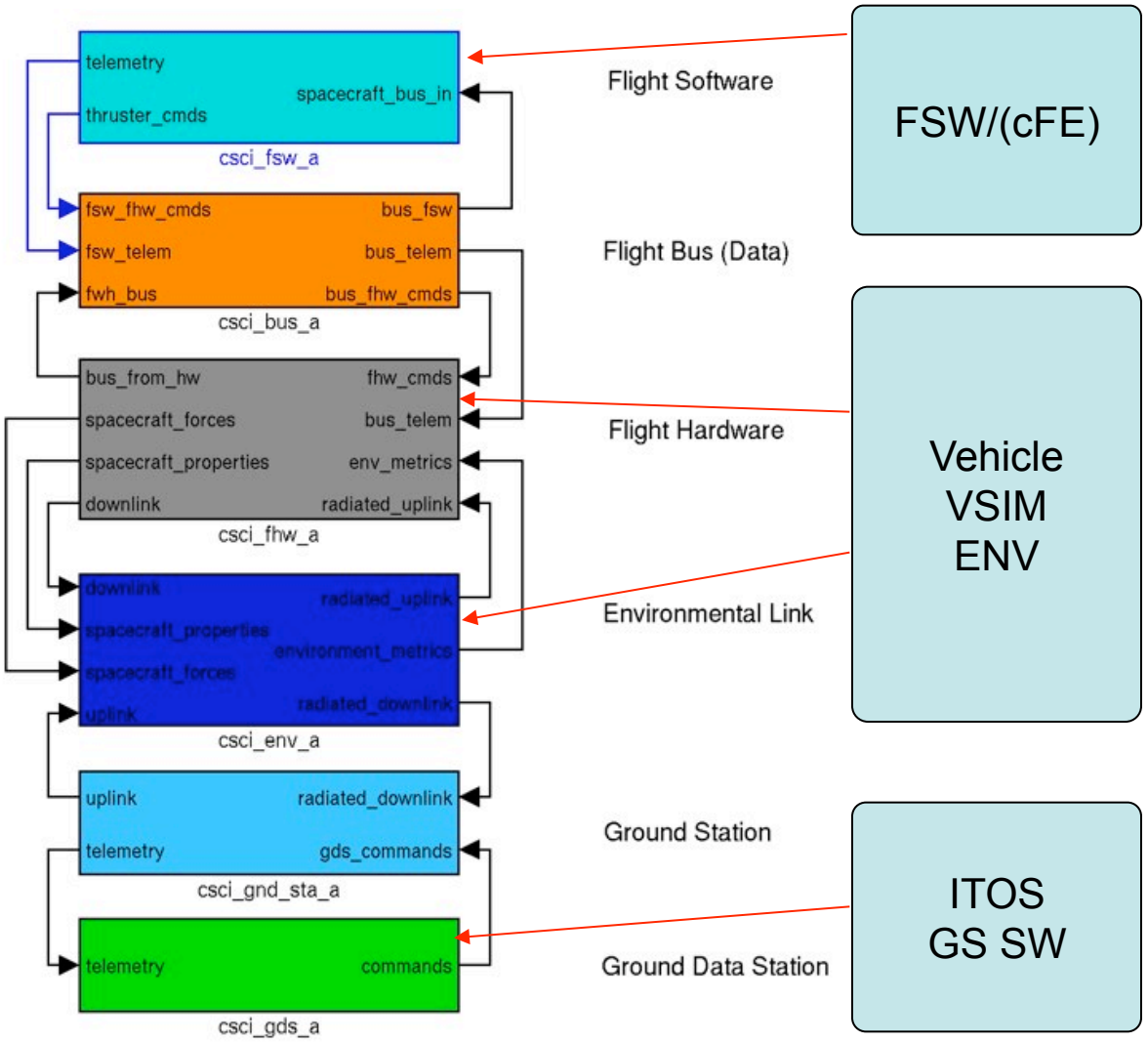
- LADEE History
  - Small Sat Investigation
  - HTV Development
  - Next Step LADEE -> STV
- Model Based Development
  - Started with NI MatrixX/System Build
  - Current Mathworks Simulink/RTW EC
- Working w/CFE – Modular Approach

FSW Process Overview

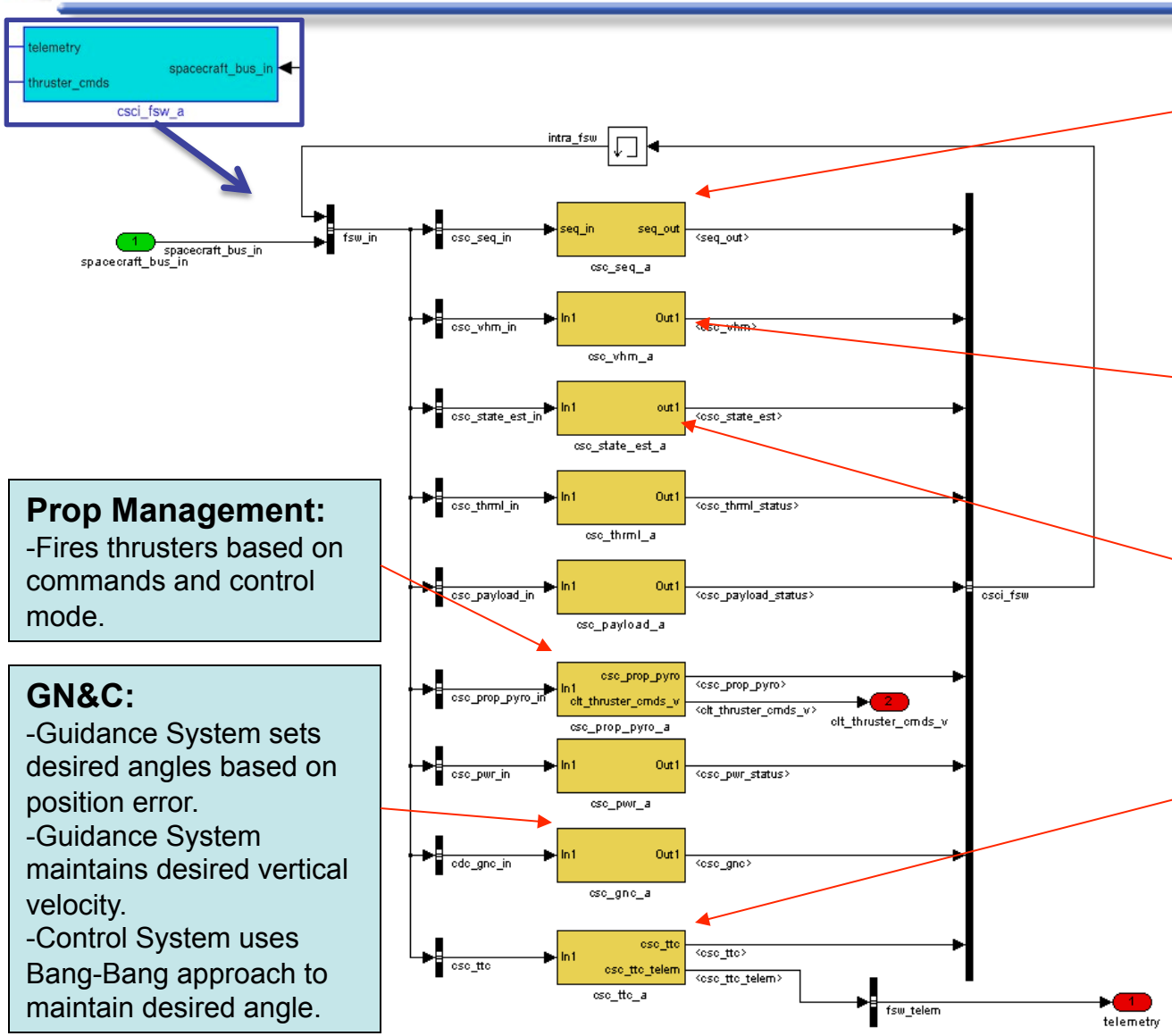


- Using Model Based Development Approach
  - Develop Models of FSW, Vehicle, and Environment in Simulink
  - Automatically generate Software using RTW/EC.
  - Integrate with hand-written and heritage software.
  - Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)

Simulink HTV Architecture



Simulink FSW Model



Command Processing:
-Receives commands via CDH (TCP/IP or RS422).
-Compiled in script allows flexible sequencing.
-Processes and Sets Control Modes.

Vehicle Health Monitoring:
-Command Checking
-Sensor Limit Checking
- Hardware status

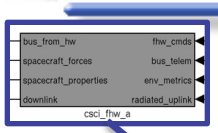
State Estimation:
-Receives sensor data.
-Low Pass Filters
-Auto generated Kalman Filter.

Telemetry:
Passes data to the CDH so that it can be transmitted via TCP/IP or RS422.

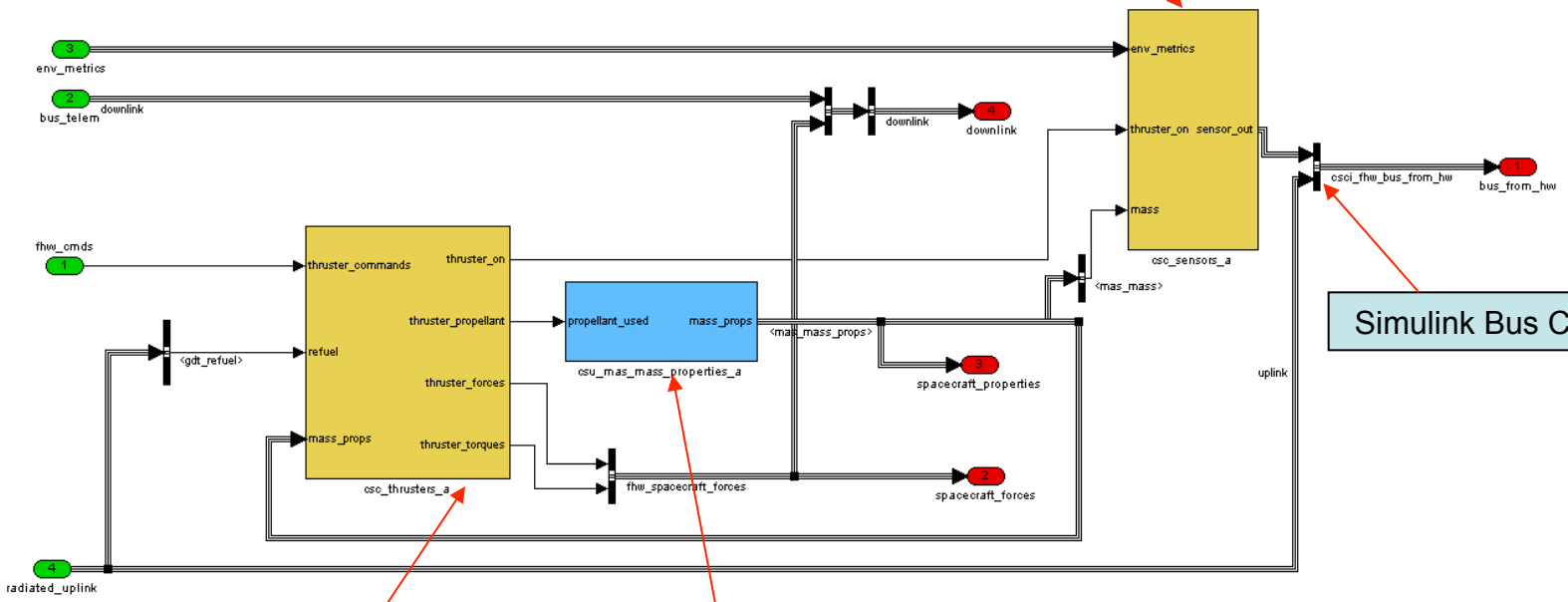
Prop Management:
-Fires thrusters based on commands and control mode.

GN&C:
-Guidance System sets desired angles based on position error.
-Guidance System maintains desired vertical velocity.
-Control System uses Bang-Bang approach to maintain desired angle.

Simulink Flight Hardware Model



Sensor Models
-Analogs (Temperature, Pressure)
-LN200 IMU
-VIZ Camera System

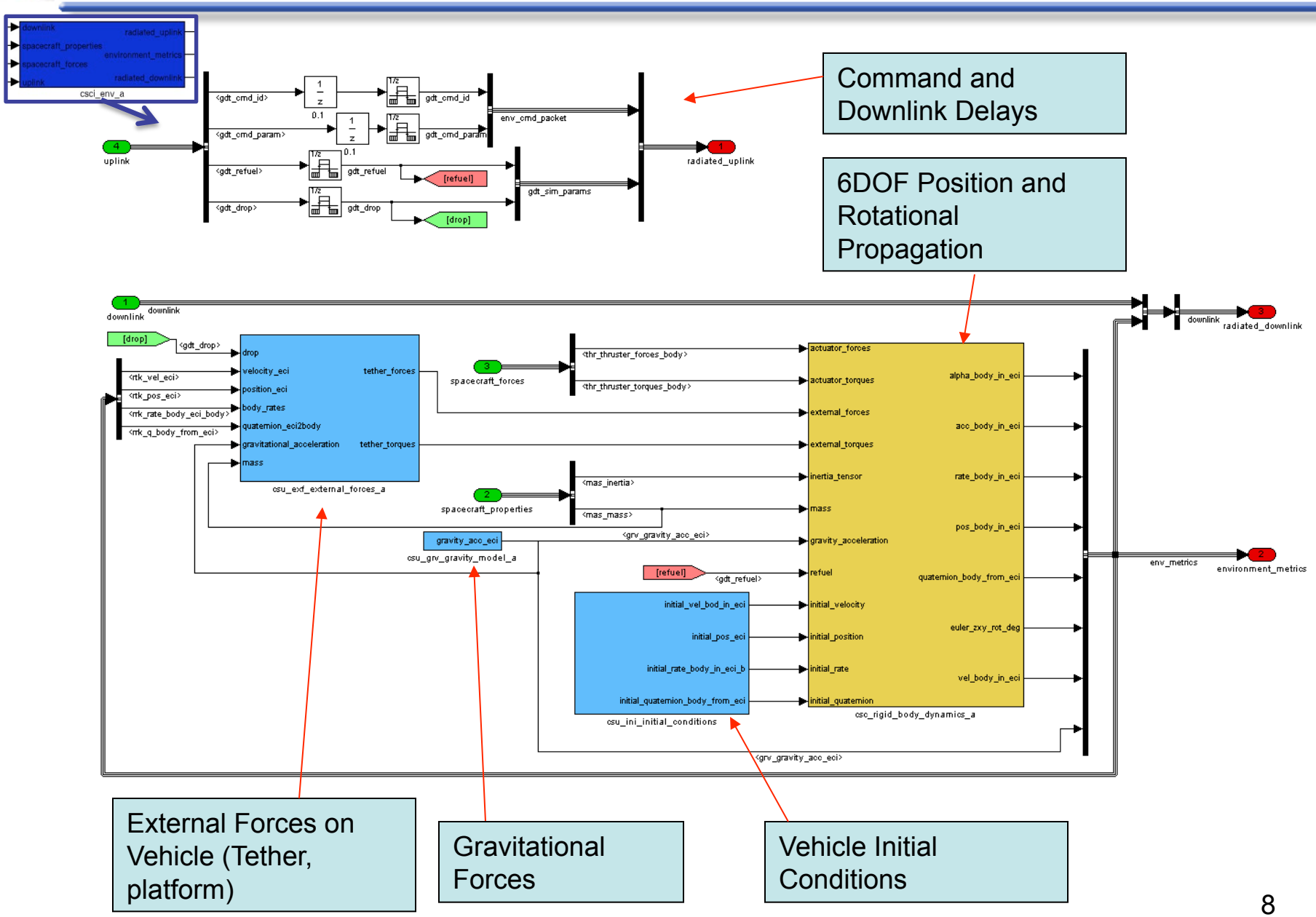


Simulink Bus Creator

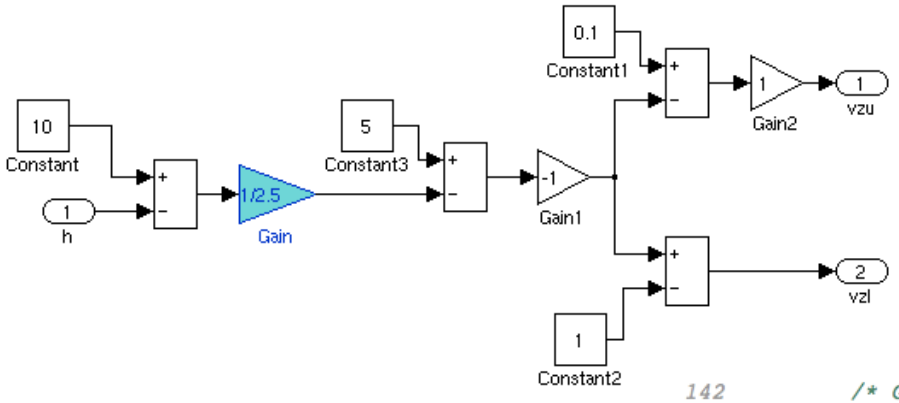
Thruster dynamic forces and torques.

Mass and Inertia Characteristics of Vehicle

Simulink Environment Model



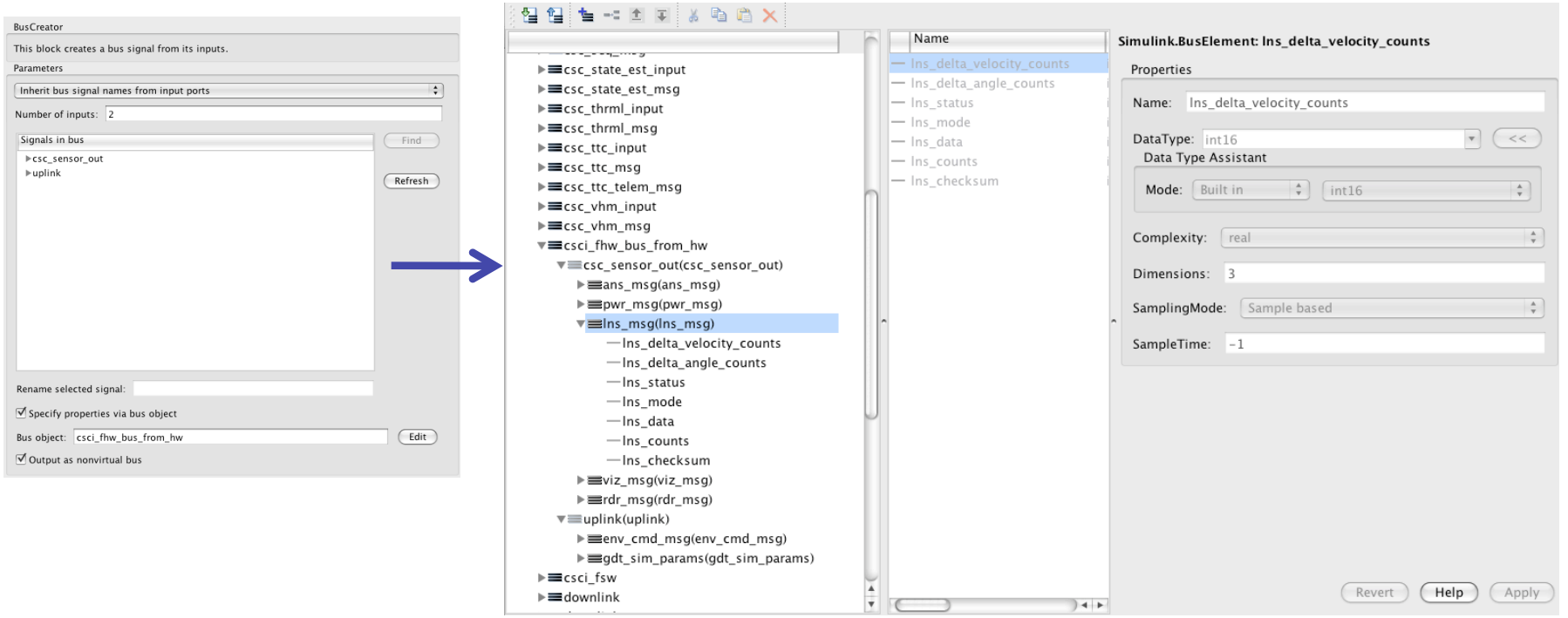
Automatic Code Generation




```
142         /* Gain: '<S16>/Gain1' incorporates:
143          *  Constant: '<S16>/Constant'
144          *  Constant: '<S16>/Constant3'
145          *  Gain: '<S16>/Gain'
146          *  Sum: '<S16>/Sum'
147          *  Sum: '<S16>/Sum3'
148          */
149         rtb_Gain1_o = (5.0 - (10.0 - rtb_h_cg) * 0.4) * -1.0;
150         rtb_Switch_g_idx = 0.1 - rtb_Gain1_o;
151         rtb_Switch_g_idx_0 = rtb_Gain1_o - 1.0;
```

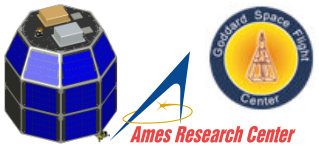
- Simulink supports two way trace-ability between models and generated code
- Code Easy to read, well commented

Simulink Bus



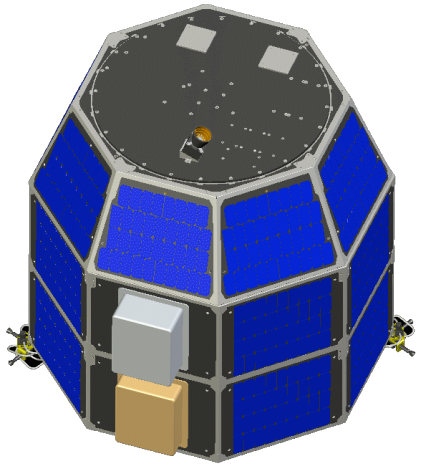

```
'lns_msg', ...
  '', ...
  sprintf(''), { ...
    {'lns_delta_velocity_counts', 3, 'int16', -1, 'real', 'Sample'}; ...
    {'lns_delta_angle_counts', 3, 'int16', -1, 'real', 'Sample'}; ...
    {'lns_status', 1, 'int16', -1, 'real', 'Sample'}; ...
    {'lns_mode', 1, 'int16', -1, 'real', 'Sample'}; ...
    {'lns_data', 1, 'int16', -1, 'real', 'Sample'}; ...
    {'lns_counts', 3, 'int16', -1, 'real', 'Sample'}; ...
    {'lns_checksum', 1, 'int16', -1, 'real', 'Sample'}; ...
  } ...
```

cFE – Core Flight Executive



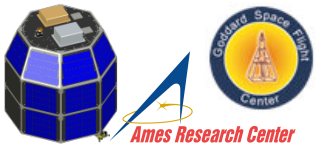
- Goddard Space Flight Center Developed
- Derived from Legacy Missions
- Flexible infrastructure for Space Flight Software
- Components:
  - Executive Services
  - Event Services
  - Time Services
  - Table Services
  - Software Bus Services

Ames Research Center



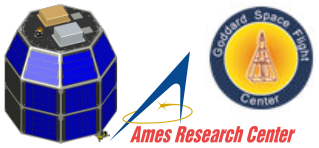
cFE Simulink Architecture

cFE Simulink App Process Goals



- Utilize cFE with no changes
- Code Gen. to automate process
- Subsystem Blocks generate to cFE Apps.
- Simulink Apps/Blocks Communicate via cFE Software Bus

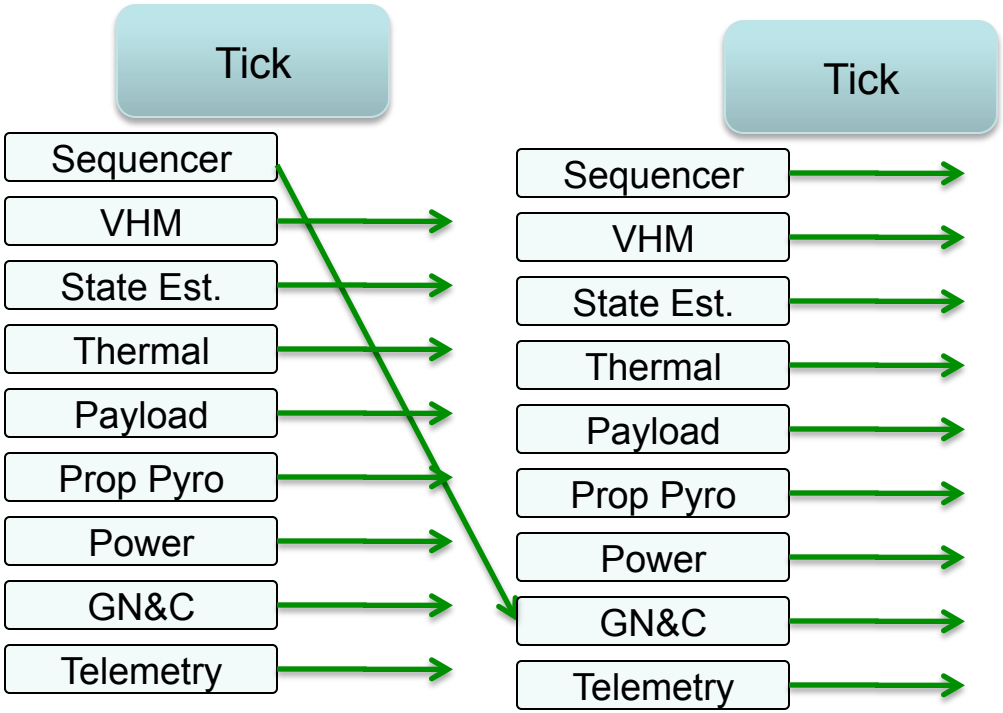
cFE Simulink App (Task)



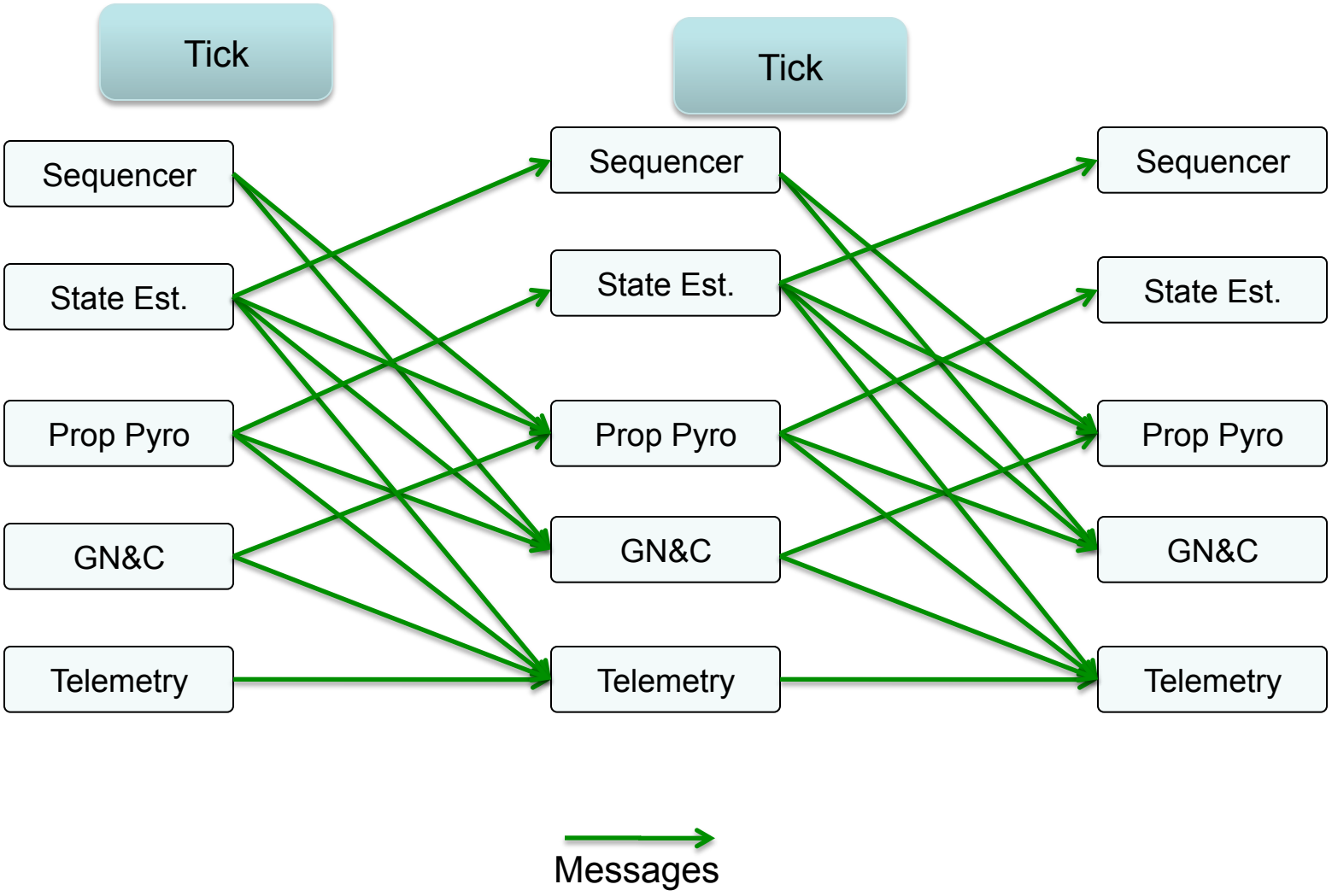
- Simulink Bus translates to cFE Message
- RTW/EC generates Task Description
- Master Timer Generates “Tick” to Schedule (Step) Apps and generate Output Message(s)
- Receive Structure Msgs update local App Input Values
- Apps also Respond to Other Command and Housekeeping Messages

cFE Simulink Message Flow

- Add slide with ticks



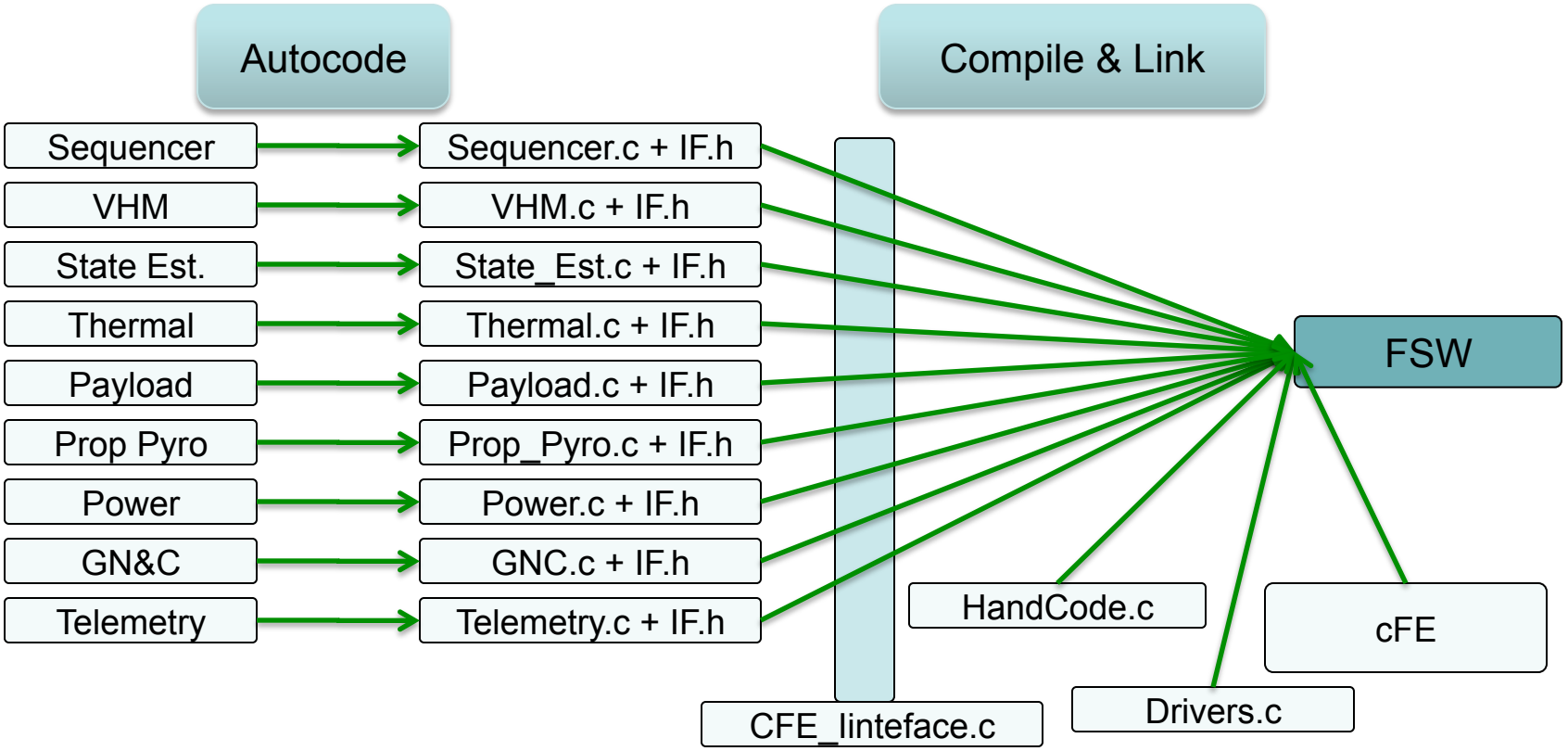
cFE Simulink Message Flow



Process Key Ideas

- Modular Process (vs. Monolythic)
  - Pros:
    - More Flexible
    - Simplifies Task Replacement
    - Easier Debugging – can look at messages between tasks
  - Cons:
    - Harder to implement
    - More overhead due to more tasks and messages
- Terminology
  - Bus (Simulink) = Message (cFE) = Packet = C Structure
  - Block (Simulink) = Application (cFE) = Task (VxWorks)
- Mathworks Template (TLC) File
  - Executed during Code Generation Process
  - Highly Flexible in creation of code

cFE Simulink Autocode Process

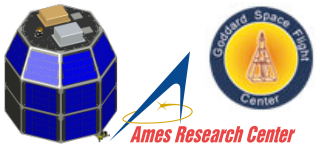


cFE Simulink App Loop

```
Struct App_Inputs In
Struct App_Outputs Out
App_Init() {
      Initialize_App_Inputs()
      Subscribe_SB_Msgs(Tick, AppMsgs,…)
      Simulink_Init(In, Out)
}
App_Main(){
      App_Init()
      while(1) {
            sb_receive_msg(msg, timeout)
            if (msg == tick) {
                  Simulink_Step(dt, In, Out)
                  sb_send_msg(Out) /* app update */
            } else {
                  If (msg == app_update) /* Process other App Msgs */
                        App_Update_Inputs(msg, Out)
                  else Process_Msg(msg) /* HK, Cmds, etc… */
            }
      }
}
```

cFE IMU App Loop

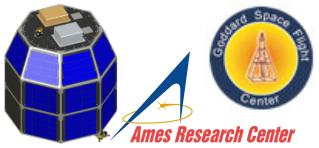
```
IMU_Main(){
      while(1) {
            struct imu_input_str imu_in
            read_msg_que(imu_in, timeout) /* VxWorks Msg Que */
            sb_send_msg(imu_msg)
            Send_tick()
      }
}

Cnt = 0;
Send_tick() {
      sb_send_msg(400HZ_Tick)    /* Do we need 400HZ Tick or key off of IMU Data? */
      if ((Cnt % 2) == 0)     sb_send_msg(200HZ_Tick)
      if ((Cnt % 4) == 0)     sb_send_msg(100HZ_Tick)
      if ((Cnt % 40) == 0)    sb_send_msg(10HZ_Tick)
      if ((Cnt % 400) == 0) sb_send_msg(1HZ_Tick)

      Cnt++;
}

/* Note: Other Apps same as IMU without the Send_tick() */
```



Backup

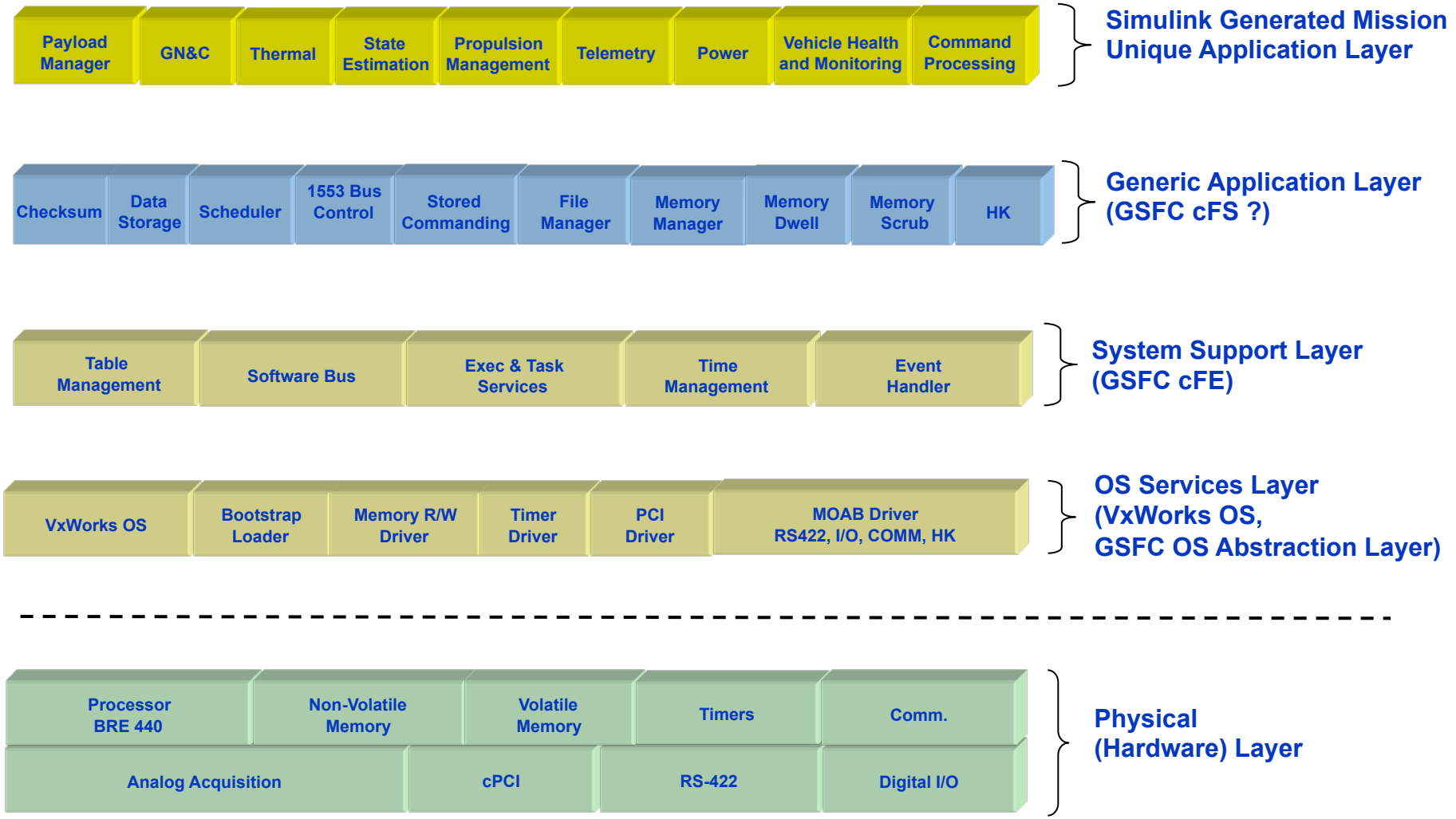
Hover Test

Motivation for Moving to Simulink

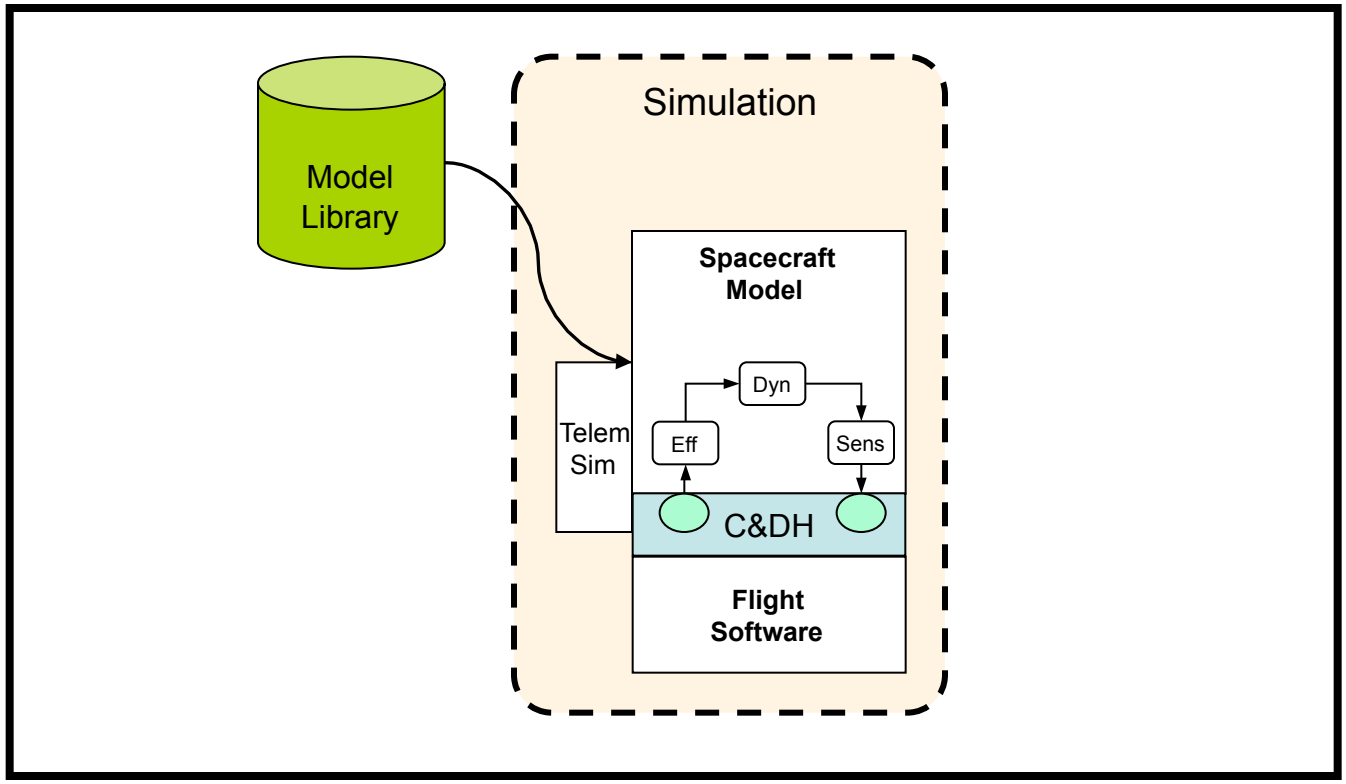


- Industry appears to be moving that direction.
- Mathworks Extensive support network.
- Mathworks tools for Requirements management, Documentation, and V&V.
- Bus concept makes model management easier.
- Monolithic SystemBuild models not conducive to Reuse and V&V.

Layered Architecture Approach

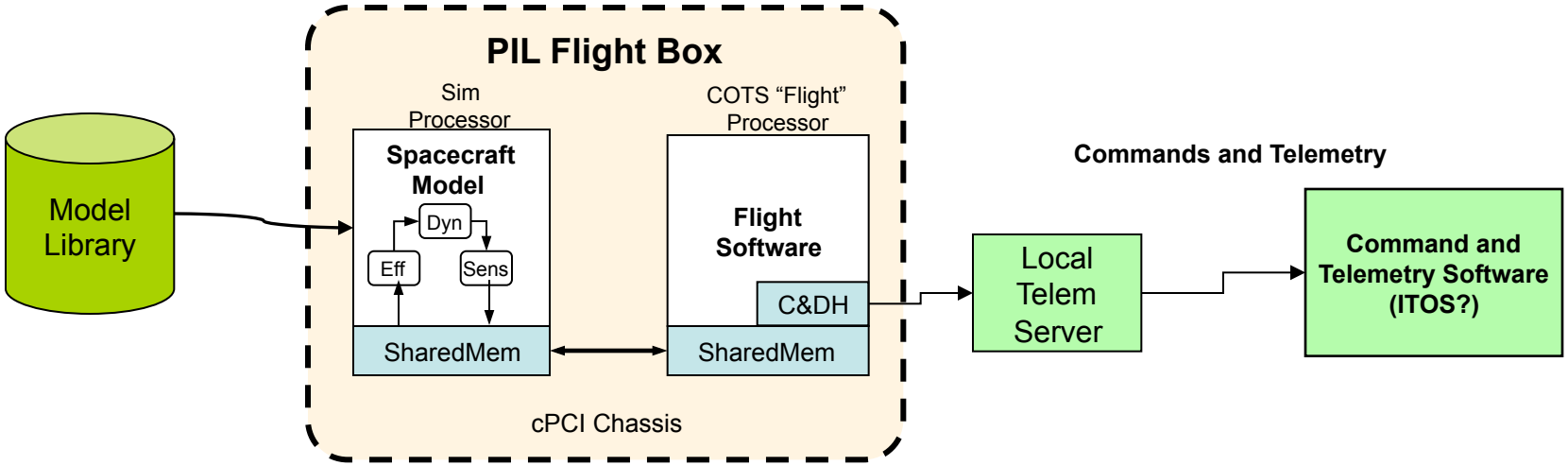


Workstation Simulation



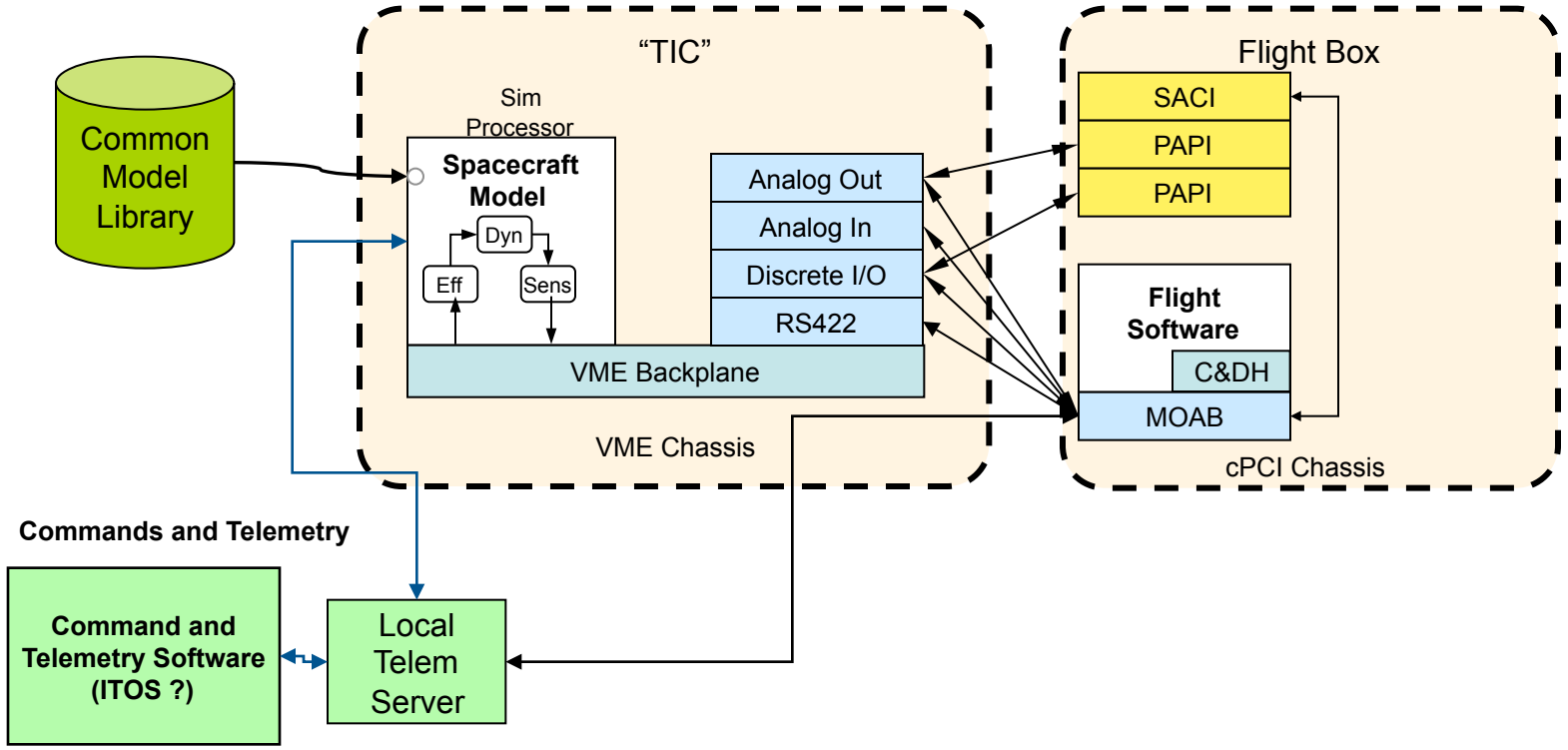
- Simulink/SystemBuild Only (No Autocode)
- Early in development process
- Algorithm Development
- Requirements Analysis

Processor-in-the-Loop Simulation



- Models autocoded and running on RT processors
- Inexpensive "flight-like" processor
- Tests autocoding process & integration with C&DH software
- Integration with Telemetry Software allows early development/testing of downlink
- Can be used for initial code size and resource utilization analysis

Hardware-in-the-Loop Simulation



- Flight code runs on Flight Avionics EDU
- Provides testing of FSW with Avionics I/O
- Definitive answers on resource utilization
- Highest fidelity simulations for verification/validation