NASA Operational Simulator for Small Satellites (NOS³)

NASA IV&V Independent Test Capability (ITC) Team
Mr. Scott Zemerick
Scott.Zemerick@TMCTechnologies.com
Agenda

STF-1 Intro
- ITC Intro
- NASA IV&V CubeSat
- C&DH FSW Architecture

NOS³
- V1.0
- Architecture
- Simulators

Conclusion
- Next Steps
- Questions
Independent Test Capability (ITC) Introduction
Acquire, develop, and manage adaptable test environments that enable the dynamic analysis of software behaviors for multiple NASA missions.

Dynamic Analysis is performed on flight software to verify software behavior.
NASA IV&V Independent Test Capability (ITC)

**JIST**

*JWST Integrated Simulation & Test*

**S3**

*SLA Software-Only-Simulator*

---

**JSTAR**

Jon McBride Software Testing & Research Lab

- **Automation and virtual deployment**
- **Small Sats**
- **QEMU RAD750 Model**
- **Wind River Simics Modeling**

---

**NOS**

NASA IV&V
5000 NASA Blvd, Fairmont, WV 26554
www.nasa.gov/centers/ivv
Simulation-to-Flight (STF-1) Introduction
Simulation-to-Flight (STF-1)

NASA IV&V ITC & West Virginia University (WVU) 3U Cubesat

- NASA Cubesat Launch Initiative (CLI) proposal submitted and accepted – NASA will pay manifest for future launch
- First WV Cubesat
- ITC is responsible for C&DH hardware/software, integration (hw/sw), and all testing
- WVU is responsible for payload hardware and software
- STF-1 is a “GSFC Cubesat” – partnering with GSFC/WFF and Dellingr Cubesat Team
- Current Launch Ready Date is August 2016 – not yet manifested – prefer polar orbit

Simulation-to-Flight (STF-1)

- **Primary Objective** – Showcase simulation technologies developed at IV&V
- **Secondary Objectives** – WVU Research into space weather, rad-hard materials, navigation instruments (GPS and IMUs), and camera

Sponsored by:

[Logos of the sponsors]
Simulation-to-Flight (STF-1)

STF-1 Flight Software / Hardware Design

- Working closely with the GSFC Dellingr 6U cubesat team
- FSW is Core Flight System (cFS)
  - Dellingr reuse, specifically on the radio cFS application
- ITC designed solar panel PCBs (Dellingr-based)
- Most hardware same as other GSFC cubesats

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Onboard Computer</td>
<td>Received</td>
</tr>
<tr>
<td>Solar Cells</td>
<td>Received</td>
</tr>
<tr>
<td>Power System</td>
<td>Ordered – 10 Week Lead Time</td>
</tr>
<tr>
<td>Chassis</td>
<td>Ordered – Unknown Lead Time</td>
</tr>
<tr>
<td>ITC Designed Solar Panel PCBs</td>
<td>Designed – Out for Quote</td>
</tr>
<tr>
<td>Radio</td>
<td>Ordered – 6 Month Lead Time</td>
</tr>
<tr>
<td>Clean Room</td>
<td>Procured and Setup for Ribbon Cutting</td>
</tr>
<tr>
<td>Deployable Antenna</td>
<td>Ordered – Unknown Lead Time</td>
</tr>
<tr>
<td>Camera</td>
<td>Received</td>
</tr>
</tbody>
</table>
Anatomy of STF-1

Camera
- Mounted to a PC104 protoboard
- Optional filters to provide earth science data

CADET Radio
- Half duplex UHF
- Low power design
- Store and Forward architecture
- 4GB memory buffer
- Up to 22 Mbps data rate

ISISpace UHF/VHF Antennas
- Deployable antenna system
- Four alloy tape antennas
- Up to 55cm in length
- Includes 30mm diameter center through-hole for pass-through

ISISpace Chassis
- Modular structure
- Each unit can be assembled independently
- COTS component
- Compatible with P-POD Cal-Poly specifications

Inertial Measurement Unit (IMU)
- Micro Electro-Mechanical Systems
- Accounts for errors through calibration
- High quality inertial sensing with a MEMs IMU cluster

LCSEE
- Two PC104 Boards
- Fits directly into stack without modification
- Three different LED carriers with different shielding levels

Physics Payload
- Particle detector
- VLF receiver
- Plasma Probe

GomSpace Nanomind A3200
- High-performance AVR32
- 512KB build-in flash
- 125Mb NOR flash
- 32MB SDRAM
- I²C, UART, CAN-Bus

Novatel OEM625 GPS
- On-orbit reprogrammable
- Precise orbit determination
- Open loop tracking
- Science data products: 100-Hz phase, TEC, S4

2 x ClydeSpace Batteries
- Lithium Polymer
- 80 Watt Hours Total
- Two independent boards for redundancy
- Internal heaters

ClydeSpace Electrical Power System (EPS)
- 10 command-able power switches
- Provides 3.3V, 5V, and 12V
- Optimized for Low Earth Orbit (LEO)
- Three independent battery charge regulators
Simulation-to-Flight (STF-1)
Simulation-to-Flight (STF-1)
Simulation-to-Flight (STF-1)

FSW Architecture
NASA Operational Simulator for Small Satellites (NOS$^3$) Introduction
What is NOS$^3$?

- A software test bed for small satellites
- Based upon STF-1 hardware, but sufficiently generic
- Easily-interfaces to cFS FSW, but cFS not required
- Currently open-loop, closed loop planned
- Openly distributed solution Ready-to-Run (RTR) – Looking for Users!
- A collection of Linux executable and libraries

What is it used for?

- FSW early-development – NOS$^3$ provides real-world inputs to FSW
- FSW V&V – Testing FSW, invalid inputs, behavior, stress conditions
- FSW Integration – Used for early-app development and payload team integration
- Mission Planning – Example: power analysis
NOS$^3$ Ready-to-Run (RTR)

Leverage ITC virtual deployment technologies

- ITC does NOT distribute virtual machines
- Virtual machines are built on-the-fly by the user
- Deployment Steps
  - Obtain files ITC
  - Install virtual machine provisioner such as Virtual Box
  - Run 1 Command – generates virtual machine
  - Login to virtual machine and build cFS with RTR script

Ready-to-Run (RTR) for...

- cFS development environment
- NOS$^3$ environment
- Ground system software
- Software integration testing
## NOS$^3$ v1 Included Simulators

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Hardware Modeled</th>
<th>Sim Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnetometer</td>
<td>Honeywell HMC5843</td>
<td>FSW data source for development</td>
</tr>
<tr>
<td>Electrical Power System (EPS)</td>
<td>Clydespace Gen III</td>
<td>Power analysis – software control of switches</td>
</tr>
<tr>
<td>GPS</td>
<td>Novatel</td>
<td>FSW data source for development and software commanding of GPS</td>
</tr>
<tr>
<td>Camera</td>
<td>ArduCam Mini OV2640 SPI/I2C</td>
<td>FSW data source for development and large data packet handling</td>
</tr>
</tbody>
</table>
# NOS$^3$ Components

<table>
<thead>
<tr>
<th>Component</th>
<th>What is it?</th>
<th>How is it used?</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA Operational Simulator (NOS) Engine</td>
<td><em>Engine</em> is ITC-developed message passing middleware designed specifically for use in simulation. Includes time synchronization, data manipulation, and fault injection.</td>
<td>Serves as the NOS$^3$ glue to tie all components together into a common interface to FSW</td>
</tr>
<tr>
<td>Hardware Model</td>
<td>A model for a specific piece of flight hardware, often focusing on the inputs/outputs of the device from the FSW perspective.</td>
<td>Serves as virtual hardware in order to provide FSW with an accurate representation of its data</td>
</tr>
<tr>
<td>42</td>
<td>42 is an open-source general purpose simulator developed at NASA Goddard Space Flight Center for spacecraft attitude and orbit dynamics.</td>
<td>Serves as an <em>Environment Data Provider</em> – chosen to provide magnetic field data and positional data as inputs to the magnetometer and GPS simulators</td>
</tr>
</tbody>
</table>

![Diagram of NOS$^3$ Components](image)
NOS$^3$ Components

FSW

- NOS Engine Interface
- NOS Engine Message
- NOS Engine Interface
- NOS Engine Message
- NOS Engine Interface
- NOS Engine Message

Hardware Model
- time, input bytes
- output bytes
- time, input bytes
- output bytes
- time, input bytes
- output bytes

Environment Data Provider
- environmental data
- environmental data
- environmental data
- time
STF-1 FSW + NOS$^3$

No FSW code changes

OS Abstraction Layer (OSAL)

Linux

FreeRTOS

Flight Hardware

STF-1

cFS

NOS$^3$
STF-1 FSW + NOS³

No FSW code changes

OS Abstraction Layer (OSAL)

Linux (x86, ARM/Pi)

FreeRTOS (AVR32)

STF-1

cFS

NOS Engine

Flight Hardware

Hardware Models

Environment Data Provider

NOS³
STF-1 FSW + NOS$^3$

No FSW code changes

STF-1

cFS

Hardware Lib

OS Abstraction Layer (OSAL)

Linux (x86, ARM/Pi)

FreeRTOS (AVR32)

CMake

Cross-platform Make

NOS Engine

NOS Engine

Hardware Models

42

Flight Hardware

NOS$^3$
## Future Plans – NOS$^3$ v2

<table>
<thead>
<tr>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Axis Gyroscope Simulator (In Progress)</td>
<td>InvenSense MPU-3300</td>
</tr>
<tr>
<td>Temperature Sensors (In Progress)</td>
<td>I2C Temperature Sensors</td>
</tr>
<tr>
<td>Electrical Power System (EPS) Sim Maturity (In Progress)</td>
<td>Add battery sim to Clydespace Simulator</td>
</tr>
<tr>
<td>UHF Radio Simulator (under consideration)</td>
<td>L3 Cadet Radio</td>
</tr>
<tr>
<td>Visualization / User Interface</td>
<td>Provide the user with a generic NOS$^3$ user-interface.</td>
</tr>
<tr>
<td>Integrate with Ground System Software</td>
<td>Currently looking into COSMOS and ITOS.</td>
</tr>
<tr>
<td>Tighter 42 Integration</td>
<td>Programmatically sync FSW time to 42 time so that NOS$^3$ hardware models and FSW are in sync</td>
</tr>
</tbody>
</table>
NOS³ Work In Progress

Ground System Software (ITOS, COSMOS)

Commanding & Telemetry

STF-1

OS Abstraction Layer (OSAL)

Linux (x86, ARM/PI)

Hardware Adapter

Hardware Lib

42

FreeRTOS (AVR32)

Flight Hardware

NOS Engine

Hardware Models

NOS³ UI & Control
NOS$^3$ Work in Progress
NOS$^3$ Visualization
NOS³ Ground System Integration
Questions?
Backup Slides
NOS$^3$ FSW Architecture
NOS$^3$ FSW Architecture
NOS$^3$ 42 Integration