Evolution of Software-Only-Simulation at NASA IV&V

http://www.nasa.gov/centers/ivv/jstar/ITC.html

Justin McCarty
Justin.McCarty@TMCTechnologies.com

Justin Morris
Justin.R.Morris@Nasa.gov

Scott Zemerick
Scott.Zemerick@TMCTechnologies.com

NASA IV&V Facility
100 University Drive
Fairmont, WV 26554
• **Introduction to Software-Only-Simulation**
  • Process and approach for simulation and hardware modeling

• **Independent Test Capability (ITC)**
  • Jon McBride Software Testing & Research Lab (JSTAR)
  • Infrastructure, Deployment, and Users
  • Technologies Developed

• **Development Evolution of Spacecraft Simulators**

• **Closing Remarks**
  • Lessons Learned
Software-Only-Simulation
Introduction
Software-Only-Simulation

Introduction

• Software-Only-Simulation is a complete software representation of modeled hardware components and software emulators

• Together, the components form a complete spacecraft simulator

• Software-Only-Simulator provides complete control of CPU, Time, and Memory
  – Can stop all execution for debugging.
  – Can peek/poke memory, perform fault injection

• Spacecraft simulator used for:
  – Independent Testing (IVV)
  – Operator Training
  – Augment Project Hardware Testing
Software-Only-Simulation

Introduction

Simulator Components

Modeled Spacecraft Hardware
- cPCI, Spacewire, 1553, Ka-band, S-band, FPGAs, etc

Instruction Set Simulator
- Simics/Qemu
- ppc750, ppc401, LEON3, etc

Unmodified Flight Software Binary

Operational Ground System

Independent Testing

- Halt Entire System
- Memory Analysis
- Fault Injection

Operator Training

- Spacecraft Scenarios
- What-IF Scenarios
Software-Only-Simulation
Introduction

Simulator Development Process
NASA IV&V Independent Test Capability (ITC)
Introduction
Charter

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
• ITC Develops System Simulators
  – Experts in **Hardware Modeling** and Distributed Simulation
  – Experts in Simulator & Software Integration
• Cloud-based infrastructure using server and desktop virtualization

• Large scale simulator deployments

• Hardware-in-the-loop and software-only test environments

• Integration of COTS and GOTS software tools to support V&V activities
Virtualized Deployment
ITC Technologies
NASA Operational Simulator (NOS)

- Software-only simulation architecture
- Capable of executing unmodified flight software
- Custom layered-architecture middleware
- Dynamic interception capability
- Reusable software modules and scripts
- Virtual machine deployment
NOS Feature Set

Plug-and-Play Hardware Models

Use of Operational Ground Systems Software

Instrument Model Framework

Instrument1
- Subaddress HandlerA → FunctionA
- Subaddress HandlerB → FunctionB
- ...
- Subaddress HandlerN → FunctionN

InstrumentX
- Subaddress HandlerA → FunctionA
- Subaddress HandlerB → FunctionB
- ...
- Subaddress HandlerN → FunctionN

Internal Bus Monitoring

NOS Middleware

Specialized Layers

Base Layer Communications

Deployment & Maintenance

Virtualization
NOS Middleware

Overview

✓ Offers re-usable communication mechanism
  • Ensures consistent and correct data passing

✓ Provides synchronization between distributed applications

✓ Flexible and extensible design
  • Can be extended to incorporate any communication protocol

Features

✓ Transport agnostic
✓ Cross platform C++ implementation
✓ Robust User API
✓ Specialized User API Layers
  • MIL-STD-1553B
  • ESA SpaceWire
  • Discrete Signals
  • Time Synchronization
✓ Interception allows for V&V analysis
  • No modification to software-under-test
NOS Middleware Architecture

- System Under Test
  - MIL-STD-1553
  - Space Wire
  - Discrete
  - Time Synchronization
  - Additional Protocols as Needed

- NOS Core Middleware with Interception Capability

- I/O Interface Layer

- System Monitoring
  - Bus Analyzer
Virtual Oscilloscope
  – Virtual CompactPCI (cPCI) Analysis
  – Board-Level Signal Analysis

Virtual MIL-STD-1553 Bus
  – Bus Controller with XML Defined Schedules
  – Remote Terminal
  – Bus Monitor/Logger
  – PASS3200 Software Emulator

Virtual SpaceWire Router
NOS Dynamic Interception

Normal Data Flow

Node A <-> NOS <-> Node B

Interceptor

Block

Modify
Evolution of ITC Spacecraft Simulators
Evolution of ITC Spacecraft Simulators

Global Precipitation Measurement (GPM) Operational Simulator (GO-SIM)
Closed-loop simulator including unmodified operational ground system, unmodified flight software, environmental simulator, and science instrument simulators

James Webb Space Telescope (JWST) Integrated Simulation and Test (JIST)
Simulator that demonstrates reusable NOS technologies can be applied to other NASA missions

Deep Space Climate Observatory (DSCOVR)
Turn-key modeling effort for spacecraft C&DH
Evolution of ITC Spacecraft Simulators

GPM Operational Simulator (GO-SIM)
## GPM Operational Simulator
### GO-SIM

### Components
- COTS Emulator
- Primary Instrument Simulations (GMI/DPR)
- GPM Ground System
- GSFC Goddard Dynamic Simulator (GDS)
- NOS Middleware
- GPM Hardware Models

### Capabilities
- Load and run unmodified flight software binaries
- Execute test flight scripts
- Single-step debugging
- Inject errors via ground system and NOS middleware
- Stress system under test

---

*NASA Software of the Year Honorable Mention 2012*
GO-SIM Architecture

ASIST Ground System with FEDS

SCOMM Simulator

RAD 750 Emulator GPM FSW

SpaceWire Router

NOS Middleware

Dynamics Simulator (GDS)

Instrument Simulator (GMI)

Instrument Simulator (DPR)

KEY

TCP/IP
1553
SpaceWire
1553 & SpaceWire

NASA IV&V Independent Test Capability 23
Evolution of ITC Spacecraft Simulators

James Webb Space Telescope (JWST)
Integrated Simulation and Test (JIST)
JWST Integrated Simulation and Test (JIST)

- Software-only spacecraft simulator
- Flexible environment to support V&V activities
- Unmodified ground system and scripts
- Unmodified software-under-test binaries
- Integration of COTS, GOTS and in-house developed components
- Custom hardware models
- Automated Testing Framework
- Fault Based Testing
JIST Architecture

ECLIPSE CCTS
Ground System

Ground System Simulators (MTTS/TCTS)

Comm Cards

RAD 750 Emulator

PPC 405 Emulator

NOS Middleware

Instrument Simulations (DSIM)

Solid State Recorder Simulation

Dynamics Simulator

RAD750 Emulator ISIM FSW

KEY

<table>
<thead>
<tr>
<th>Arrows</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>←→</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>←</td>
<td>1553</td>
</tr>
<tr>
<td>⬛⬜⬜⬜</td>
<td>Shared Memory</td>
</tr>
<tr>
<td>←→</td>
<td>1553 &amp; SpaceWire</td>
</tr>
</tbody>
</table>
Evolution of ITC Spacecraft Simulators

Deep Space Climate Observatory (DSCOVR)
DSCOVR Architecture
# Simulator Level-of-Effort Comparison

<table>
<thead>
<tr>
<th>Year Usage</th>
<th>Simulator</th>
<th>Effort</th>
<th>Prototype (Basic C&amp;DH)</th>
<th>Complexity</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-2014</td>
<td>GO-SIM</td>
<td>2 FTEs</td>
<td>6 Months</td>
<td>Medium</td>
<td>IV&amp;V, GPM Project Testers Launch Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012 - Ongoing</td>
<td>JIST</td>
<td>2 FTEs</td>
<td>4 Months</td>
<td>Very High</td>
<td>IV&amp;V, JWST Test Labs, JWST Operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013 - Ongoing</td>
<td>DSCOVR</td>
<td>1 FTE</td>
<td>2 Months</td>
<td>Low</td>
<td>DSCOVR Testers DSCOVR Operations</td>
</tr>
</tbody>
</table>
Evolution Lessons Learned

- Establishment of a reusable simulation architecture has proven to save costs and reduce future effort.
- Automate tests and deployments as much as possible as it allows for engineers to focus on more challenging tasks.
- Hardware modeling should focus on the minimum needed in order for the flight software to execute. Establish this baseline then augment to support full V&V dynamic testing using an iterative process.
- Spend considerable time writing unit tests for the hardware models. When things go wrong, debugging is very difficult.
- Integration of simulators to form a system will require significant development labor, cost, and time.
Contact Information

• Web Page
  – http://www.nasa.gov/centers/ivv/jstar/JSTAR.html

• E-Mail
  – Justin.McCarty@TMCTechnologies.com
  – Scott.Zemerick@TMCTechnologies.com
  – Justin.R.Morris@nasa.gov

• Contact us for...
  – Demonstrations of test beds
  – Middleware usage agreements
  – Simulator development
  – Hardware modeling
  – V&V Services, HWIL Testing, Performance Testing