Efficient Software Verification Process along with Newly Designed Software Architecture

Shusaku Yamaura
Institute for Education on Space
Wakayama University

NanoSatellite Symposium @Nagoya
11th October 2012
Contents

• Introduction
• Approach
• Software Architecture Design
• Verification Process
• Result (Current Status)
• Conclusion
Introduction (Background)

• UNIFORM satellite is now under development, and on-board software is being developed from the scratch.

• In satellites development, verification of software is time-consuming and thus critical in the development schedule.
How can we make the on-board software better?

How can we test on-board software efficiently?
Approach

• Software architecture and the verification process is designed at the same time.

• Software architecture is designed to have two axes layer: vertical layer and horizontal layer. Vertical layer is mainly consisted of hardware dependent part and hardware independent part. Horizontal layer is mainly consisted of project-inherent part and re-usable part.

• Software verification process is divided into 3 steps: Model-In-the-Loop Simulation, Software-In-the-Loop Simulation and Hardware-In-the-Loop Simulation.
Software Architecture Design

- **Algorithm Layer**
  - **H/W Independent Part**
    - e.g. Attitude control algorithm
    - Mode/execution control framework
    - Time/event driven sequence framework
  - **H/W Dependent Part**
    - Convert data type between H/W dependent part and H/W independent part
    - For input and output from outside computer
    - Operating system

- **Sequence Frame Layer**

- **I/O Layer**

- **Common Platform Layer**
  - **Uncommon Library**
  - **Common Library**
## Software Verification Process

<table>
<thead>
<tr>
<th>Test configuration</th>
<th>① MILS (Model In-the-Loop Simulation)</th>
<th>② SILS (Software In-the-Loop Simulation)</th>
<th>③ HILS (Hardware In-the-Loop Simulation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Things to test</td>
<td>Control Logic, mode transition</td>
<td>C code</td>
<td>C code on OBC, Sensor/actuator drivers</td>
</tr>
<tr>
<td>Tools</td>
<td>MATLAB/Simulink</td>
<td>MATLAB/Simulink, I/F soft</td>
<td>MATLAB/Simulink, PXI System, LabVIEW</td>
</tr>
</tbody>
</table>

Environment & dynamics model, sensor models and actuator models developed in MILS (Model In-the-Loop Simulation) are used for following simulation, such as SILS (Software In-the-Loop Simulation) and HILS (Hardware In-the-Loop Simulation). What’s changed is inside “AOBC”.

① MILS: The GNC algorithm made of MATLAB/Simulink in the AOBC is simulated.
② SILS: Actual software written in C in the AOBC is simulated. I/F between the C-code and Simulink need to be properly defined.
③ HILS: Actual software on actual OBC is simulated. PXI system and LabVIEW are used for the I/F with OBC.
MILS: Model-In-the-Loop Simulation

- MILS is for testing algorithm (logic) of software.
- To make the simulation “closed loop”, simulation models have been developed with MATLAB/Simulink. (Detail is shown in the next slide.)

<table>
<thead>
<tr>
<th>Test configuration</th>
<th>MILS (Model In-the-Loop Simulation)</th>
<th>SILS (Software In-the-Loop Simulation)</th>
<th>HILS (Hardware In-the-Loop Simulation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td><img src="image1" alt="MILS Diagram" /></td>
<td><img src="image2" alt="SILS Diagram" /></td>
<td><img src="image3" alt="HILS Diagram" /></td>
</tr>
<tr>
<td>PXI System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simulation Environment

Actuator Models
- Motor-on/off
- RotMode
- open/close
- WheelMom
- MTQin
- GeoMag

Environment Models & Satellite Dynamics
- Julian Date
- SatPos

Sensor Models
- Sun
- Mag
- q
- Error
- Cov
- NSAS1
- NSAS3
- GasVol
- Volt
- GPSOut

OBC
- Out2b
- error_flag
- NSAS1flag
- NSAS3flag
- dir
- alt
- day
- V

ACBC & Sensors
SILS : Software-In-the-Loop Simulation

- SILS is for software “coding” verification. Actual C-code is tested if it works properly as designed in MILS.
- Hardware independent part is mostly completed at this point.
- “Shared memory” is used to interface C-code and Simulink. (Detail is shown in the next slide.)

<table>
<thead>
<tr>
<th>Test configuration</th>
<th>① MILS (Model In-the-Loop Simulation)</th>
<th>② SILS (Software In-the-Loop Simulation)</th>
<th>③ HILS (Hardware In-the-Loop Simulation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td></td>
<td></td>
<td>PXI System</td>
</tr>
<tr>
<td>display</td>
<td></td>
<td></td>
<td>OBC</td>
</tr>
</tbody>
</table>
SILS Configuration

- Actuator Models
- Environment & Dynamics
- Sensor Models
- Shared Memory
- SDK Structure
- Virtual SDK (C-code)

Files:

- AOCS_process.c
- AOCS_process.h

Shared Memory Connections:

Virtual SDK (C-code) <-> AOCS_process.c & AOCS_process.h <-> Sensor Models
HILS : Hardware-In-the-Loop Simulation

- HILS is for verification of H/W and S/W integration. On-board computer is used as the “hardware” here.
- Shared memory and serial port is used for interface between OBC and environment. (Detail is shown in the following slides.)

<table>
<thead>
<tr>
<th>Test configuration</th>
<th>① MILS (<em>Model In-the-Loop Simulation</em>)</th>
<th>② SILS (<em>Software In-the-Loop Simulation</em>)</th>
<th>③ HILS (<em>Hardware In-the-Loop Simulation</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td>PXI System</td>
<td>OBC</td>
<td></td>
</tr>
</tbody>
</table>
HILS Configuration

- Actuator Models
- Environment & Dynamics
- Sensor Models

Simulink on PXI (or HILS PC)

Serial Port & Shared Memory

SDK (C-code)

AOCS_process.c
AOCS_process.h

OBC

SDK Structure

RS422 etc.

SDK Structure

SDK (C-code)
Simulator part is installed on PXI System. LabVIEW is used to interface with OBC. PC1 is for OBC monitor and debug.

**Good**: Real-time OS is used, and it’s originally planned configuration.
**Bad**: PXI system is bulky and a little expensive. It needs to stay in certain place.

USB hub and signal converting cables are used instead of PXI System. Simulator part is installed on PC2. PC1 is for OBC monitor and debug.

**Good**: Easy to run simulator wherever there is PC with simulator models and cables.
**Bad**: It is hard to used as real-time simulation.
Software Verification Process Summary

- **H/W independent**
  - Design verification (MILS)
  - Coding verification (SILS)
  - Component Integration verification (Open-Loop with components)

- **H/W dependent**
  - Design verification (Design review)

- **Software design and coding verification (HILS)**

I/O Layer:
- Compo A Driver
- Compo B Driver
- Compo C Driver
- Compo D Driver

Common Platform Layer:
- Operating System

Sequence Frame Layer:
- Mode supervising function

Guidance and Navigation:
- Guidance 1
- Guidance 2
- Navigation 1
- Navigation 2
- Navigation 3
Result Summary

- UNIFORM satellite and the software is under development.
- Software architecture design and part of coding is completed. And AOCS algorithm layer design and the verification is mostly completed in MILS.
- SILS and HILS environment is almost ready to be used as software verification.
- Rest of tasks on software development is coding and the verification of them.
Conclusion

• Software architecture design and the verification process is jointly considered for UNIFORM satellite.

• Software is divided into horizontal layer and vertical layer, and the reusability is taken into account.

• Software verification process is mainly divided into MILS, SILS and HILS for efficient test and debug.

• Software coding and the verification is in progress but the simulation environment is ready to be used.