Open Experimental Platform (OEP) for Mission Flight Software

Peter Glück
Garth Watney
Martin Gilbert

Autonomy and Control Section
Jet Propulsion Laboratory
Agenda

- Motivation
- Description
- Progress
- Issues
Motivation - Background

Background

- JPL’s Autonomy Lab preserves the DS1 Flight Software executing closed-loop with a spacecraft and dynamics simulation.
- Researchers and technologists need access to a platform for experimentation
  - TPF testbed will reuse the core of the DS1 Flight Software
  - Technology providers for new Remote Agents (IDEA) and V&V techniques need realistic Flight Software for their development.

Typical User Requirements

- Closed loop – full spacecraft and dynamics simulation
- Instrument, rebuild, execute
- Non mission-specific is OK
- Virtual real-time is OK
  - Preserves the ordering of events without concern for the timing of the events.
Motivation - Historical Perspective

- Cassini AACS (A7 release) had 520 Acceptance tests pertaining to Fault Protection
  - Involved injecting a Fault in the system and testing appropriate responses
  - 450 were performed on FSDS (Virtual real-time solaris workstation)
    - testing correct functional code execution
    - algorithmic
    - complex fault injection difficult to administer on real hardware.
  - 70 were performed on CATS (real-time processor with some flight-like hardware)
    - hard real-time issues
- Industry partners use workstation-based flight software validation environments

http://qmsworkshop5.gsfc.nasa.gov/
Motivation - Eliminate The Bottleneck

To reduce the bottle-neck on limited resources.
Motivation - Objective and Approach

- **Objective**
  - Provide a desk-top version of Flight Software as an easily accessible software package.
  - Enable users to have full access of the software for instrumenting, building and executing.

- **Approach**
  - Port to network-free environment.
  - Provide portability by Inserting an OS Adaptation Layer (PACE) into the DS1 Flight and Simulation software.
  - Build a layer to support both real-time and virtual real-time environments.
Motivation Summary

- To solve the problem of software accessibility
  - Limited resource
  - Complex operating procedures
  - Location specific (Autonomy Lab only)
  - Operating System specific (VxWorks)
  - Build procedure is unavailable for "outsiders"

- The problem: Accessible Mission Flight Software is effectively closed to "outsiders"
What Is OEP?

- What is the OEP?
- Flight Software in a box
  - Adapted from DS1
  - Portable (C code)
  - Accessible

http://qmsworkshop5.gsfc.nasa.gov/
Actual Real-Time Environment (ARTE)  
OEP Architecture
Virtual Real-Time Environment (VRTE)
OEP Architecture

Users
Configuration File:
- VRTE
- FSW Version
- Sim Version
- Telemetry View

http://qmsworkshop5.gsfc.nasa.gov/
Actual Real-Time Environment

VMIC reflective memory (c-PCI)

PPC-603
- SIM
  - OEPAL
  - PACE
  - OS (VxWorks)

PPC-750
- FSW
  - OEPAL
  - PACE
  - OS (VxWorks)

TIP
remote TCL

TIP
cmd/tim

NFS Server
File System

Sun Mac or PC

ITS
Integrated Test System

- Replaces ALAB ITL utility for OEP
- TK-based
- Loads the Referee and Remote TCL plug-in modules
- Launches SIM and FSW as TIP windows in ARTE
- Launches SIM and FSW as child processes in VRTE
- Interactively guides user through start-up and Launch Sequence
- Configuration Parameter File for configuring test session
  ➢ Fully-commented sample template included with ITS deliveries
- User console for issuing ITS, SIM, and FSW commands
- VRTE simulation control:
  ➢ start, pause, resume, step, warp, quit
- Log files

http://qmsworkshop5.gsfc.nasa.gov/
OEP Adaptation Layer

- Motley of functions for non-PACE'able OS calls . . .
  - Common (albeit non-PACE'able) calls: ioctl(), select(), etc.
  - OS-specific (e.g. VxWorks-only) calls: vxMemProbe(), logMsg(), etc.
  - Incompletely-PACE'd calls (e.g. supported for some OS'es, not all)
  - ARTE vs VRTE file system issues: open(), mkdir(), etc.
  - ARTE vs VRTE timing issues: nanosleep(), clock_gettime(), etc.
  - Centralized "convenience" functions to avoid duplicating "glue" code

- OEPAL will be heavily-laced with conditional compilation
  - (#if) directives
  - Differentiate between ARTE and VRTE
  - Differentiate OS (Solaris, Linux, etc)
  - Other environmental factors

- OEPAL is "thin" in ARTE and "thick" in VRTE.
VRTE Threading Model

- OEP makes 2 categories: **application** and **service** threads
- OEPAL keeps a table for tracking all the application threads
  - Their states ("waiting" vs "running")
  - OS-specific "thread-id"
  - Owning process (SIM or FSW)

<table>
<thead>
<tr>
<th>Process</th>
<th>&quot;Application&quot; Threads</th>
<th>&quot;Service&quot; Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM</td>
<td>1:1 mapping with ARTE VxWorks tasks.</td>
<td>1 main thread 1 messenger thread</td>
</tr>
<tr>
<td>FSW</td>
<td>1:1 mapping with ARTE VxWorks tasks.</td>
<td>1 main thread 1 messenger thread</td>
</tr>
<tr>
<td>ITS</td>
<td>None.</td>
<td>1 main thread 1 uplink thread</td>
</tr>
</tbody>
</table>
VRTE Scheduler (1 of 2)

- Scheduler runs from main thread in ITS process
- Maintains To-Do List of pending scheduled events
  - when (expiration v-time)
  - who (which thread to wake up)
  - where (which process owns the thread)
- OS restricts thread signaling to within same process only, so Scheduler sends message to messenger thread in targeted process, telling it to issue SIGUSR1 to particular application thread
- ITS’s uplink thread is another customer
- Maintains centralized “virtual clock”
  - Updated as events expire
- When all application threads are in waiting state, then OK to extract and expire next-scheduled event from To-Do List
- OK for customer to remove/cancel events from To-Do List prior to expiration (e.g. message arrived prior to timing-out)
VRTE Scheduler (2 of 2)

- Distinguishes “true waiting” from “false waiting”
  - True waiting: recorded in scheduling table
  - False waiting: e.g. thread blocked temporarily by fprintf()
- Detects “zombie running” state
  - Application thread terminated abruptly and forgot to update its state
- Example customer implementation:

  Function: wait until condition “C” occurs or virtual time dT elapses.
  begin
    submit event to To-Do List (when=now+dT, who=thr_self(), where=getpid())
    until (condition “C” occurs) or (event expires) do
      update my state in table: set to WAITING
      wait for SIGUSR1 or condition “C”: whichever occurs first
    end until
    update my state in table: set to RUNNING
    if event did not occur then remove/cancel it from To-Do List
  end
Conversion Strategy

- The SIM and FSW source code is divided into sections
- For sections with behavioral differences between ARTE and VRTE:
  - Incorporate `#if` conditional compilation directives.
  - OS calls in that section are
    - PACE-converted (1st preference),
    - left alone (2nd preference), or
    - OEPAL-converted (3rd preference)
- For sections with no behavioral differences:
  - Do not add new `#if` directives
  - OS calls in that section are either
    - PACE-converted (1st preference) or
    - else OEPAL-converted (2nd preference)
3 Software packages CM’d in Yam/CVS Repository

➤ Flight Software (FSW)
  • latest version: FSW-R1-02
  • DS1 M6 baseline
➤ Simulation Software (SIM)
  • latest version: SIM-R1-01
  • Spacecraft and Dynamics simulation using libsim
➤ Ground Data Support Tools (ITS)
  • latest version: ITS-R1-02
  • Downlisten
  • Sending scmf files to the spacecraft
  • Tcl-based User Interface
The Module Soup...

"Modules" contain all software

"Packages" collections of overlapping modules

Software Repository

"External" Packages
- Adaptations
- Delivery
- System

"Internal" Packages
- Subsystem
- Development
- Test
- Demo

http://qmsworkshop5.gsfc.nasa.gov/
"Link" and "Work" Modules

Sandbox 1

Sandbox 2

Sandbox 3

Sandbox 4

Sandbox n

regression builds

development

regression testing

Source Repository

Module Release Area

http://qmsworkshop5.gsfc.nasa.gov/

14 May 2003 - Page 22
Benefits of Yam

- Provides tremendous development flexibility
- Can build a 2 minute version of even large sandboxes
- Can rapidly generate scaffolding code for development
- Avoids yo-yo effect - incremental development and I&T
- Easily handles multiple software package configurations
- Easy to mix the varied development paces of stable and new s/w
- Uses stable concurrent software development practices
- Minimal coordination overhead among developers
- Developers can choose the time for syncing up to recent releases
- Developers choose the time to make a release
- Development "hacked" code is captured and not lost
- Provides full support for development and I&T teams’ CM needs
OEP Packages

Package status

- Captured all source and build procedures
- CM'd in a CVS Repository using the Yam toolset
- Users can create a private sandbox...
- ...Or use the latest/greatest packages kept on /proj/alab
- Ported to a network-free environment (no "hard-wired" references to a specific network)
  - Delivered and works at Ames Research Center
- **OEP Users Manual**
  - How to setup and maintain sandboxes
  - Specify User configurations
  - Execute Scenarios
  - Send Commands
  - Generate new commands
  - View Telemetry/Event Reporting

- **Autonomy Lab website**
  - Software Repository
  - Command Dictionary
  - Telemetry Dictionary
  - Response state charts
  - Board support manuals
  - DS1 Design documents

http://qmsworkshop5.gsfc.nasa.gov/
Development Ground Rules

- SIM and FSW shall be modified only sparingly
- Some modules (Heap, EEP, MICAS) shall retain full functionality in ARTE, and become simplified or nullified in VRTE
- Module “PACE” shall not be modified
- OEPAL shall absorb the majority of the necessary modifications
- VRTE shall allow multiple instances of itself on the same machine
  - ARTE: 1 SIM and 1 FSW on separate VME or PCI CPUs

http://qmsworkshop5.gsfc.nasa.gov/
Current Status

- Execution is fully user configurable
  - What FSW to use
  - What SIM to use
  - VRTE or ARTE
  - Target boards to use
  - Output directory for telemetry files
  - User-specific Telemetry filters
- OEP is ported to a network-free environment
  - Can be transported to other JPL and non-JPL sites
  - ARTE runs at Ames Research Center
- OEP is not “hard-wired” to specific single board computers
  - Can support multiple ARTE platforms
- Port to Solaris, Linux, and Windows in progress (VRTE)
  - “Hooks” in place
  - Design and research complete
## Completed Upgrades

<table>
<thead>
<tr>
<th>Old Autonomy Lab</th>
<th>Upgraded</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSW and SIM executables are “hard-wired”</td>
<td>Sandbox support. Acquire, modify, instrument, rebuild and execute your own FSW and SIM.</td>
</tr>
<tr>
<td>Data output directory is “hard-wired”</td>
<td>User specifies output directory in config file. Allows different directory for every run.</td>
</tr>
<tr>
<td>Manual error-prone procedure to bring up and execute a successful Launch Sequence.</td>
<td>Automated – “idiot-proof”</td>
</tr>
<tr>
<td>Target boards are “hard-wired”</td>
<td>User specifies target boards in config file.</td>
</tr>
<tr>
<td>Ground Support tools (ITL) are “hard-wired”. Lost the build procedure</td>
<td>New ITS have full control of the build procedure with Sandbox support.</td>
</tr>
<tr>
<td>Telemetry dump in one format</td>
<td>User specific – customize what telemetry of interest – specify xterm parameters for display.</td>
</tr>
</tbody>
</table>
User Community

- Used by
  - IDEA (Intelligent Distributed Execution Architecture) adaptation
  - Auto Filter tool task
  - PolySpace Verification and Validation task
  - Runtime Verification task
  - Ames Research Center
  - X2000 Performance measurement task
  - Starlight FAST (Formation Algorithm Simulation Testbed)
Deployment

Key:
- Possible
- Future
- Present
- Past

~ 7 months away
~ 3 months away

Accessibility

Open Source
- user-configurable sandbox
- Golden standard
- JPL Met

Other Sites
- JPL/non JPL
- Multiple babybeds
- Single babybed

Execution Paradigm
- Real-time
- WAWorks

Operating System
- Command/Telemetry
- Telemetry

User Interface
- MSL

Software

Environment

Resources

Applications

http://qmsworkshop5.gsfc.nasa.gov/

14 May 2003 - Page 30
Remaining Work

- Solaris VRTE OEP
  - Insert PACE layer into the Simulation software - 3 days
  - Code the Referee module (synchronization between FSW and SIM in the Virtual Real Time Environment (VRTE)) - 1 week
  - Populate the VRTE “hooks” in the ITS - 2 days
  - Incorporate VRTE support in the OEPAL module - 2 weeks
  - Incorporate updated OEPAL layer into Flight and Simulation S/W - 2 weeks
  - Test the Launch Sequence in the VRTE on a Solaris machine - 1 week
  - Total Estimate: 7 weeks

- Linux VRTE OEP
  - Build all 3 packages on a Linux machine - 4 weeks
  - Test the Launch Sequence in the VRTE on a Linux machine - 1 week
  - Total Estimate: 5 weeks

- Continue to improve user accessibility and visibility
  - Add Stethoscope for real-time data plotting
Problems and Lessons

- Funding not available to complete work
  - No project wants to foot the bill
  - Institutional funding is very competitive with other technologies
  - *Lesson: Get a commitment for multi-year funding to completion, if possible*

- Availability of key personnel
  - Martin Gilbert needed for flight project (MER)
  - *Lesson: Better to have all of someone for a short period than part of someone for a long period--they may disappear!*

- Open source adaptation layer (PACE) is no longer supported
  - Champion has graduated and abandoned the project
  - Insufficient user base
  - Technical challenges in compatibility with ACE
  - New concept being developed as a replacement, but will it last?
  - *Lesson: Reliance on open-source solutions should be contingent upon having a mature product with a well-established user community*