Integrating Tests of Autonomy with SW and People: Autonomy Requirements Tester (ART)
AIAA-HSI ATS 2017
Carroll Thronesbery, Ayman Qaddumi, Michael Merta, Eugene McMahon, Mike Monahan
Background: core Flight System (cFS)

• Autonomy Requirements Tester (ART) is designed to be compatible with core Flight System (cFS)

• cFS architecture and software
  • Software platform developed by Goddard Space Flight Center (GSFC)
  • Reusable software framework across multiple projects
  • Set of reusable software applications
  • Dynamic run-time environment (real-time constraints)
  • Layered software
  • Component-based design
  • Publish-subscribe message communication to make component apps independent

• ART can easily be modified to fit any pub-sub architecture
Autonomy Requirements Tester (ART): Human Centered View

• A tool to support software developers
  • Especially flight software

• People Tasks:
  • Capture autonomy requirements
  • Generate test specifications
  • Execute the test specs
  • Report results
  • Iterate for test-driven development
cFS-based Autonomy Requirements Tester (ART) project: System View

NASA Phase I SBIR sponsored by ARC & JSC

• Design eXtensible Markup Language (XML) schema to define cFS data models to support app-level testing

• Describe potential approaches for semi-autonomous test generation

• Design displays that support the management of requirements, test designs, and test results

• Develop a Concept of Operations (ConOps) with scenarios illustrating how ART supports people tasks

• Develop and demonstrate feasibility prototype
Represent Data for Progression from Requirement to Test Results

**Autonomy Requirement**
When a person is near the robotic arm, its movements should be reduced in speed.

**Test Objective (Behavior)**
When any sensor detects a range of 3 feet or less:

IF (range to human <= 3 feet)
THEN switch to slow arm motion mode (which reduces the highest arm speed to no greater than 0.5 feet per second)

**Expected Results**
Arm_Motion_Mode = slow

**Test Input**
IR sensor range
IR_range <= 2.9 feet

**Publish Msg**
IR_Range_To_Human
IR_range <= 2.9 feet

**Subscribe Msg**
Arm_Mode_Change
(expect value to be: Arm_Motion_Mode = slow)

**Test Input**
Camera estimated range
Camera_Est_Range = 2.9 feet

**Expected Results**
Expect values of Arm_Motion_Mode = slow
Test Runner Uses CFS Pub/Sub Architecture
Test Runner Reads Test Spec, Produces Test Results

Test spec (script data) (Adaptation of ATML)

Test1
Input Message:
  x=5
  y=2
  z=7

Expected results message:
  xx=1
  yy=3
  zz=1

Test2
...

Test Runner

Test Results

Test1
  <pass>

Test2
  Expected
    xx=1
    yy=3
    zz=1
  Observed
    xx=0
    yy=3
    zz=1
  ...

Software being tested
Data Model Based on IEEE Standards
adopted by the Institute of Electrical and Electronics Engineers (IEEE) as a standard (IEEE Std 1671-2010)
Generating the Test Plan from Requirements

Observation from Phase I exercise with APL Solar Probe Plus type autonomy requirements

- Similarities from one requirement to the next
  - Often a tiered response, when first tier doesn’t correct the issue, go to the next tier
  - Rule based behavior: If \{condition\} then \{response\}
- Similarities enable the formation of a \textbf{template} (illustrated on next page) for generating the test
- Some additional \textbf{parameters} are needed in addition to the template (illustrated on the page after that)
Generating Detailed Test Plans

- Detailed test plan can become tedious to specific in every detail
- Templates for a given set of requirements can remove some of the tedium
- Can encourage test-driven development
Parameter Entry to Enable Test Generation from Template

Enter Design Values To Construct Initial Test

<table>
<thead>
<tr>
<th>AUT-3</th>
<th>Monitor Battery Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Autonomy shall perform the following tiered response if the battery temperature is above a pre-defined limit:</td>
</tr>
<tr>
<td></td>
<td>A) Soft-reset the PSC</td>
</tr>
<tr>
<td></td>
<td>B) Power-cycle the PSC</td>
</tr>
<tr>
<td></td>
<td>C) Switch the PSC (via a CIM side switch)</td>
</tr>
</tbody>
</table>

Enter values from design file...

- **M**: 5
- **N**: 6
- **Persistence (m of n)**: 9
- **Max fire count**: 2
- **Priority**: Enabled
- **Initial rule state (enabled/disabled)**

- **battery_temp**: Battery temperature variable name (default from rqts xml)
- **160**: pre-defined limit
- **PSC_reset_cmd**: Soft-reset the PSC command name (default from rqts xml)
- **PSC_pwr_cycle_cmd**: Power-cycle the PSC command name (default from rqts xml)
- **change_CIM_side_cmd**: Switch CIM side command name (default from rqts xml)
Template Contents for Generating Details of Test Specification

<table>
<thead>
<tr>
<th>AUT-3</th>
<th>Monitor Battery Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Autonomy shall perform the following tiered response if the battery temperature is above a pre-defined limit:</td>
</tr>
<tr>
<td></td>
<td>A) Soft-reset the PSC</td>
</tr>
<tr>
<td></td>
<td>B) Power-cycle the PSC</td>
</tr>
<tr>
<td></td>
<td>C) Switch the PSC (via a CIM side switch)</td>
</tr>
</tbody>
</table>

1. Set nominal spacecraft system state
2. Verify autonomy takes no action
3. Inject fault
4. Verify faulted state (optional, especially level 0)
5. Verify autonomy response
6. Repeat steps 3-5 through all possible iterations
7. For tiered rule:
   1. Inject fault corrected by 1st action
   2. Inject fault corrected by 2d action
   3. Inject fault corrected by 3d action
   4. Inject unrecovered fault
1\textsuperscript{st} Part of Detailed Test Plan

Initialize

1. Set nominal spacecraft system state
   • set rule\_fire\_count(AUT3) to 0
   • set battery\_temp = 159, every second

2. Verify autonomy takes no action
   • Wait 12 sec (m*2)
   • Verify rule\_fire\_count(AUT3)=0

Succeed on Tier 1 Response

1. Inject fault
   • set battery\_temp = 161, every second

2. Verify tier 1 autonomy response
   1. Wait 7 sec (n+1)
   2. Verify PSC\_reset\_cmd
   3. Verify rule\_fire\_count(AUT3) = 1

3. Success
   1. set battery\_temp = 159, every second
   2. Wait 15 sec (m * 3)
   3. Expect rule\_fire\_count(AUT3) = 2
   4. Verify that there is no:
      1. PSC\_reset\_cmd
      2. PSC\_pwr\_reset\_cmd
      3. change\_CIM\_side\_cmd

Succeed on Tier 2 Response

Succeed on Tier 3 Response

Unrecovered Function

Initialize

Succeed on Tier 1 Response

Succeed on Tier 2 Response

1. Inject fault
   • set battery\_temp = 161, every second

2. Verify tier 2 autonomy response
   1. Wait 7 sec (n+1)
   2. Verify PSC\_pwr\_reset\_cmd
   3. Verify rule\_fire\_count(AUT3) = 2

3. Success
   1. set battery\_temp = 159, every second
   2. Wait 15 sec (m * 3)
   3. Expect rule\_fire\_count(AUT3) = 2
   4. Verify that there is no:
      1. PSC\_reset\_cmd
      2. PSC\_pwr\_reset\_cmd
      3. change\_CIM\_side\_cmd

Succeed on Tier 3 Response

Unrecovered Function
# Autonomous Rule System

## Requirements Overview

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Test Plans</th>
<th>Test Results</th>
<th>Requirement AUT-3 Monitor Battery Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ID</td>
<td>Title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-1</td>
<td>Detect Loss of Telemetry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-2</td>
<td>Detect Invalid Telemetry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-3</td>
<td>Monitor Battery Temp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-4</td>
<td>Monitor Battery State of Charge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-5</td>
<td>Detect Critically Low State of Charge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-6</td>
<td>Battery Heater Power On</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUT-7</td>
<td>Battery Heater Power Off</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Details

- **ID**: AUT-3
- **Name**: Monitor Battery Temp
- **Categories**: Device Health, Thermal Monitoring
- **Parents**: SC-1
- **Rationale**: Protects against a potential PSC control fault that could cause excessive battery temperature.

- **Text**: Autonomy shall perform the following tiered response if the battery temperature is above a pre-defined limit: A) Soft-Reset the PSC, B) Power-cycle the PSC, C) Switch the PSC (via a CIM side switch).
- **Condition**: Battery_temp > 160
- **Subject**: Autonomous system
- **Tiered Action(s)**: Soft-reset the PSC, Power-cycle the PSC, Switch the PSC (via a CIM side switch)
- **Object**: PSC
- **Limit Value**: 160
- **Constraint**: N/A
# Viewing Details from Test Plan

<table>
<thead>
<tr>
<th>Step ID</th>
<th>Step Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Initialize</td>
</tr>
<tr>
<td>Step 2</td>
<td>Tier 1 Response</td>
</tr>
<tr>
<td>Step 3</td>
<td>Tier 2 Response</td>
</tr>
<tr>
<td>Step 4</td>
<td>Tier 3 Response</td>
</tr>
<tr>
<td>Step 5</td>
<td>Unrecovered Function</td>
</tr>
</tbody>
</table>

## Test Runner - Active Requirement: AUT - 3

### Step 2: Tier 1 Response

1. Set battery temp = 161, every 1 second

### Verify tier 1 autonomy response

1. Wait 7 seconds
2. Verify PSC_reset cmd
3. Verify rule_fire_count(AUT3) = 1

### Success

1. Set battery temp = 159, every 1 second
2. Wait 15 seconds
3. Expect rule_fire_count(AUT3) = 1
4. Verify commands
A Graphical View of the Test Plan

Test Runner - Active Requirement: AUT - 3

Flowchart

Step 1
- Initialize
  - Set Nominal Spacecraft System State
  - Verify autonomy takes no action

Step 2
- Tier 1 Response
  - Inject Fault
  - Verify Tier 1 response
  - Determine Success

Step 3
- Tier 2 Response
  - Inject Fault
  - Verify Tier 2 response
  - Determine Success

Step 4
- Tier 3 Response
  - Inject Fault
  - Verify Tier 3 response
  - Determine Success

Step 5
- Unrecovered Function
  - Cycle Tiered Response
  - Verify rule disabled

Back to Step Details
# A View of the Test Results

**Test Runner - Active Requirement:** AUT - 3

<table>
<thead>
<tr>
<th>Date &amp; Time</th>
<th>Test Plan Name</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/11/2016 – 9:34 AM</td>
<td>Initial autonomous rule test</td>
<td>Failed ✗</td>
</tr>
</tbody>
</table>

**Initial autonomous rule test Results**

- **System:** Autonomous Rule System
  - **Date:** Aug. 11, 2016 – 9:34 AM
- **Personnel**
  - Operator: Smith Johnson, sysop@company.com, 239-234-4321
  - Quality Assurance: Jane Smith, qa@company.com, 239-234-4321

**Test Plan:** Initial autonomous rule test

**Description:** Performs macro command tests to verify that appropriate commands are being sent when the AUT rule is triggered.

**Outcome:** Failed

- ✔ Step 1: Initialize
- ✔ Step 2: Tier 1 Response
- ✔ Step 3: Tier 2 Response
- ✗ Step 4: Tier 3 Response
  - 1. Inject Fault
  - ✗ 2. Verify tier 3 autonomy response
    - ✔ 1. Wait 7 seconds
    - ✗ 2. Verify change CIM side
TestRunner Detailed Results

Requirement ID 3
Requirement Name ID AUT-3
Requirement Description Monitor Battery Temperature
Parent Requirement SC-1
Rationale Protects against a potential PSC fault
Send Telem parameter: 1
Tier A
Data Point Nominal 159.000000
Data Point Injected Fault 161.000000
The number of seconds to send continuous data to AUT: 12
Limit Value 160.000000
Logical (Mathematical) Operator GT

```
SIM_AUT_APP: Received a Message. Calling SIM_AUT_Process_Received_SB_Msgs()
Received test data from the Test Runner!!! Calling SIM_AUT_runTestCase()
Sim Aut: Received PSC packet from Test Runner APP - Message ID = 0x889
Sensor Value sent by CDH is: 159.000000 !!!

1980-012-14:03:20.32327
Time Stamp (MET Seconds) : 1000.1368466176
* * * * * * * * * * * * * * * * * * * * * * *

......................... Number of Execution Seconds Elapsed : 2

SIM_AUT_APP: Received a Message. Calling SIM_AUT_Process_Received_SB_Msgs()
Received test data from the Test Runner!!! Calling SIM_AUT_runTestCase()
Sim Aut: Received PSC packet from Test Runner APP - Message ID = 0x889
Sensor Value sent by CDH is: 159.000000 !!!

1980-012-14:03:21.24989
Time Stamp (MET Seconds) : 1001.1073303552
* * * * * * * * * * * * * * * * * * * * * * *
EVS Port 66/1/CFE_TIME 21: Stop FLYWHEEL

......................... Number of Execution Seconds Elapsed : 3

SIM_AUT_APP: Received a Message. Calling SIM_AUT_Process_Received_SB_Msgs()
Received test data from the Test Runner!!! Calling SIM_AUT_runTestCase()
Sim Aut: Received PSC packet from Test Runner APP - Message ID = 0x889
Sensor Value sent by CDH is: 159.000000 !!!

1980-012-14:03:22.24989
Time Stamp (MET Seconds) : 1002.1073303552
* * * * * * * * * * * * * * * * * * * * * * *
EVS Port 66/1/CFE_TIME 22: Stop FLYWHEEL

......................... Number of Execution Seconds Elapsed : 4
```

...
ART Emphases

• Focus of Phase I (completed)
  • Hard, real-time autonomy (low level of autonomy) and cFS
  • Test-driven development
  • Unit test for app in publish-subscribe architecture

• Advantages of this approach
  • Start test driven development early
  • Ease the expression of autonomy requirements in terms of expected behavior
  • Support pre-integration testing – unit testing and regression testing
  • Make integration testing time more productive – no logic errors in software
  • During integration, if software changes are required
    • If software changes are required
    • Make the changes
    • Re-run the pre-integration test to ensure no errors were inadvertently entered
    • Resuming integration testing
Innovations

• Represent requirements and link with intended behaviors for testing the requirements

• Formal data models for requirements, behavioral expectations, test specifications, and test results

• Use of template to drive the elaboration of test specifications

• Support for test driven development

• Integration of the testing mechanism with the operational environment to support (CFS users)
  • Enabled by modular architecture w/ pub-sub communications scheme
  • No change to the unit under test between testing and operations
  • Paves the way for runtime checkout routines for selected apps (e.g., sensors for deep-space science operations)

• Reporting of test results – similar appearance to specifications, still linked to requirements
Next Steps

• Identification of how to support higher levels of integration testing:
• Identification of how to support additional types of autonomy requirements:
• Identification of additional options for semi-autonomous test generation:
• Proof-of-principle prototype of ART
• Evaluation of proof-of-principle prototype