

Supporting Tests of Autonomy: Autonomy Requirements Tester (ART)

Eugene V. McMahon
S&K Global Solutions (NASA)
Houston, TX

CONTRIBUTORS

- Carroll Thronesbery, PhD “Principal Investigator”
- Ayman Qaddumi, MS “Software Developer”
- Michael Merta, MA “Human Computer Interaction Designer”
- Mike Monahan, “Business Developer”
- Eugene McMahan, “Testing Designer”

Topics

- SBIR and the ART project
- High-level Diagram
- Test Runner
- Requirements in XML Format
- Generating Test Plan from Requirements
- ART Examples
- Advantages
- Innovations
- Next Steps

NASA SBIR

NASA Small Business Innovation Research (SBIR) project titled “Autonomy Requirements Tester” (ART) with the following goals:

- Design eXtensible Markup Language (XML) **schema** to define data models to support app-level testing
- Describe potential approaches for **semi-automatic** test generation
- Design **displays** that support the management of requirements, test designs, and test results
- Develop a Concept of Operations (**ConOps**) for the use of ART that employs the following scenarios:
 - Capture autonomy requirements
 - Generate test specifications
 - Execute the test specs
 - Report results

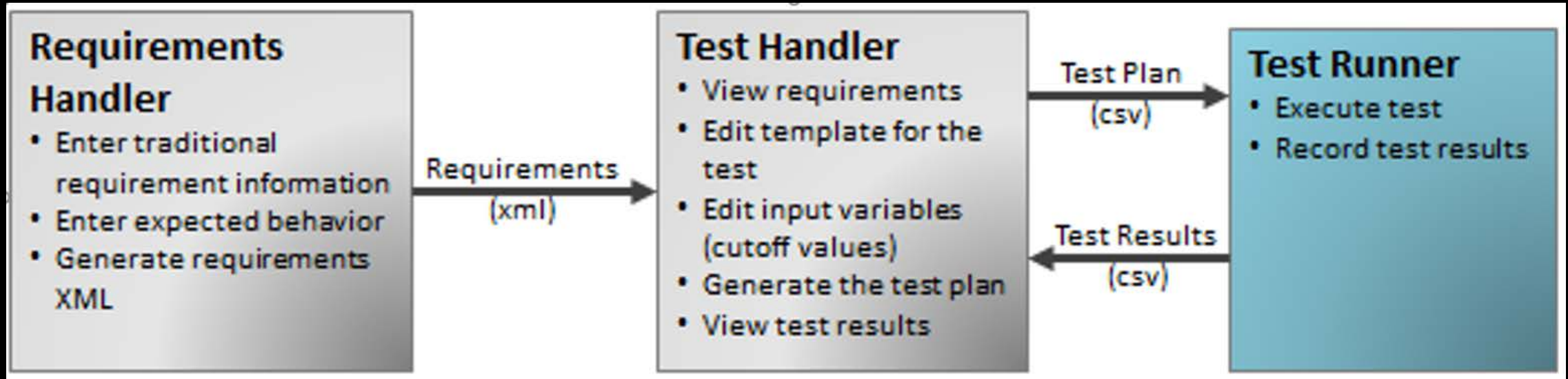
Recent Survey

A recent survey of software developers revealed some of their top issues:

1. Requirements that are confusing or incomplete
 2. Modifying software that is not documented, difficult to understand, or difficult to see the relationship between requirements and software
 3. Unrealistic expectations or deadlines
- This SBIR project directly addresses the first two issues

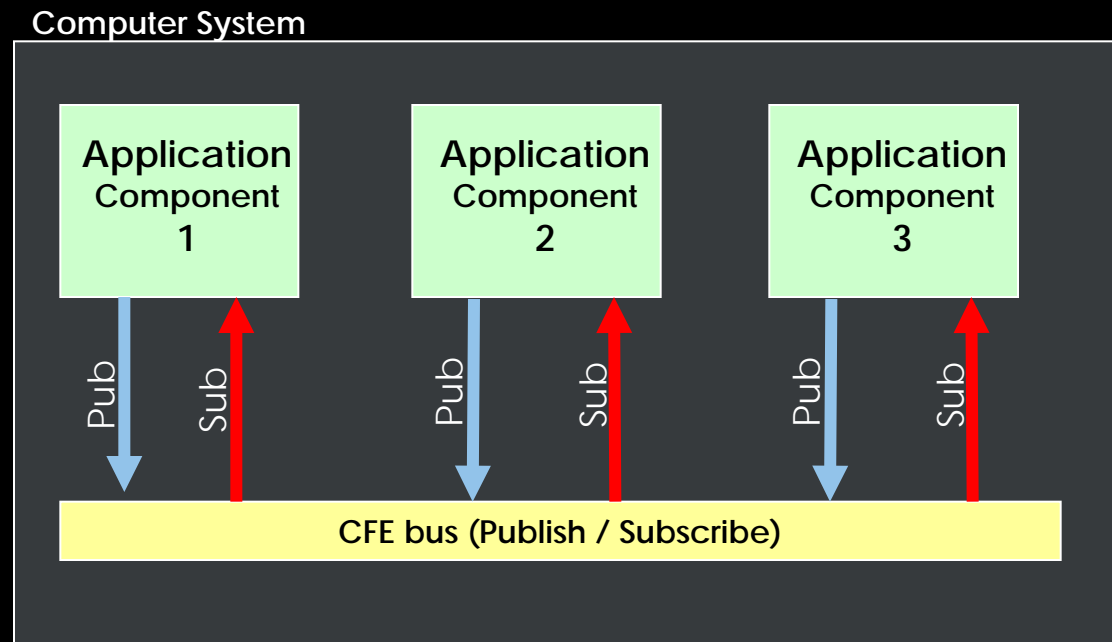
ART Flow Diagram

High-level diagram of Autonomy Requirements Tester (ART)



Publish / Subscribe Software

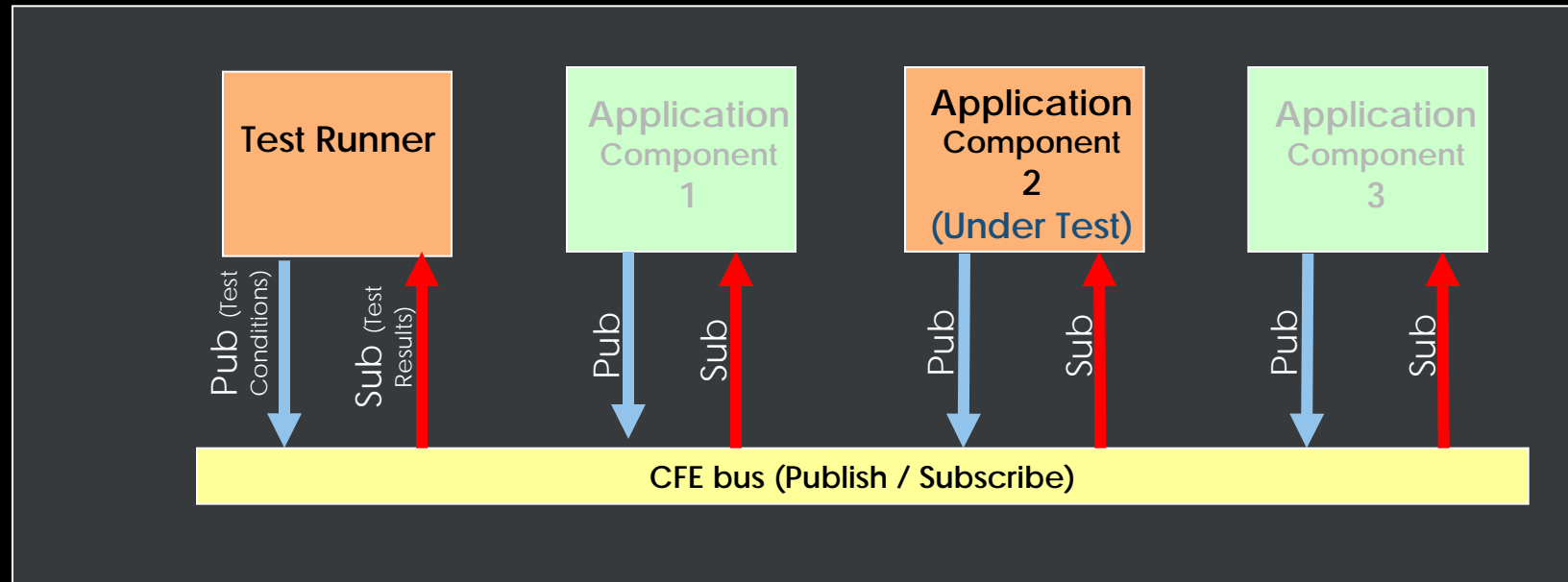
- Examples: cFS and ROS
- Component based design
- Publish-subscribe message communication to make component apps independent



Test Runner Software

- **Publish**: Send specific test data to Application Under Test
- **Subscribe**: Receive test results from Application Under Test

Computer System



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
```

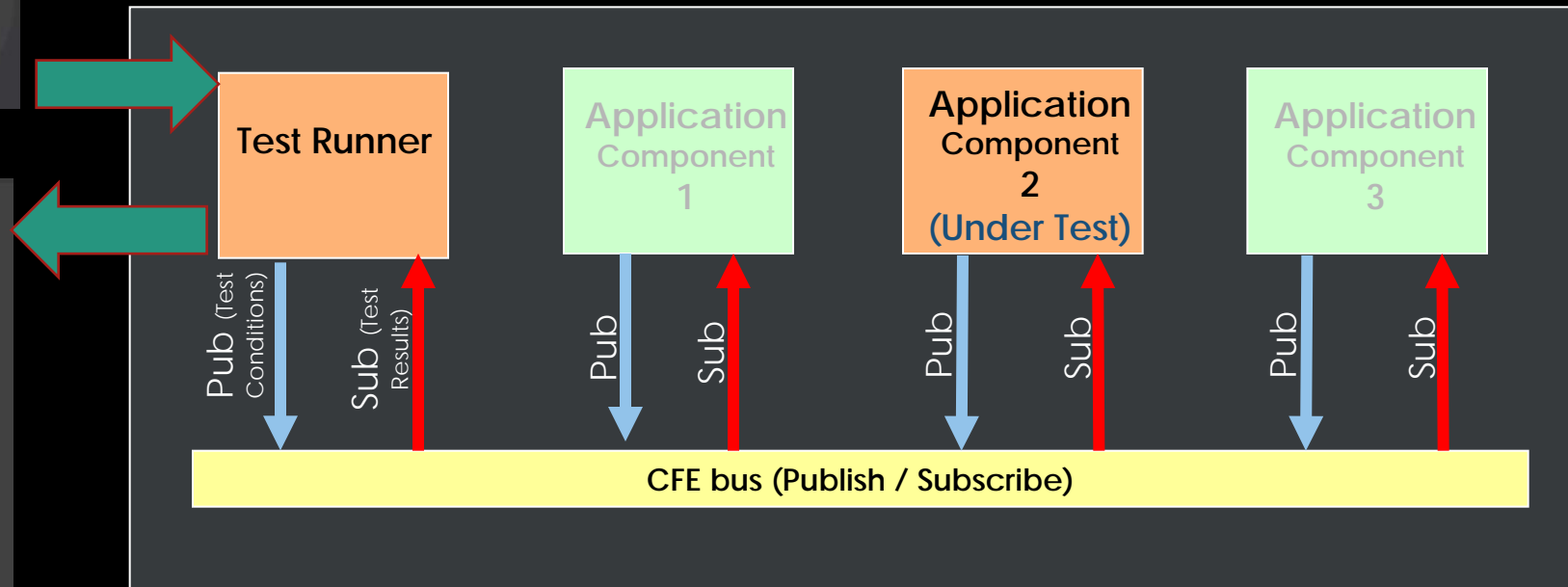
```
Test2
...
```

Test Results

```
Test1
<pass>

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

Computer System



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
```

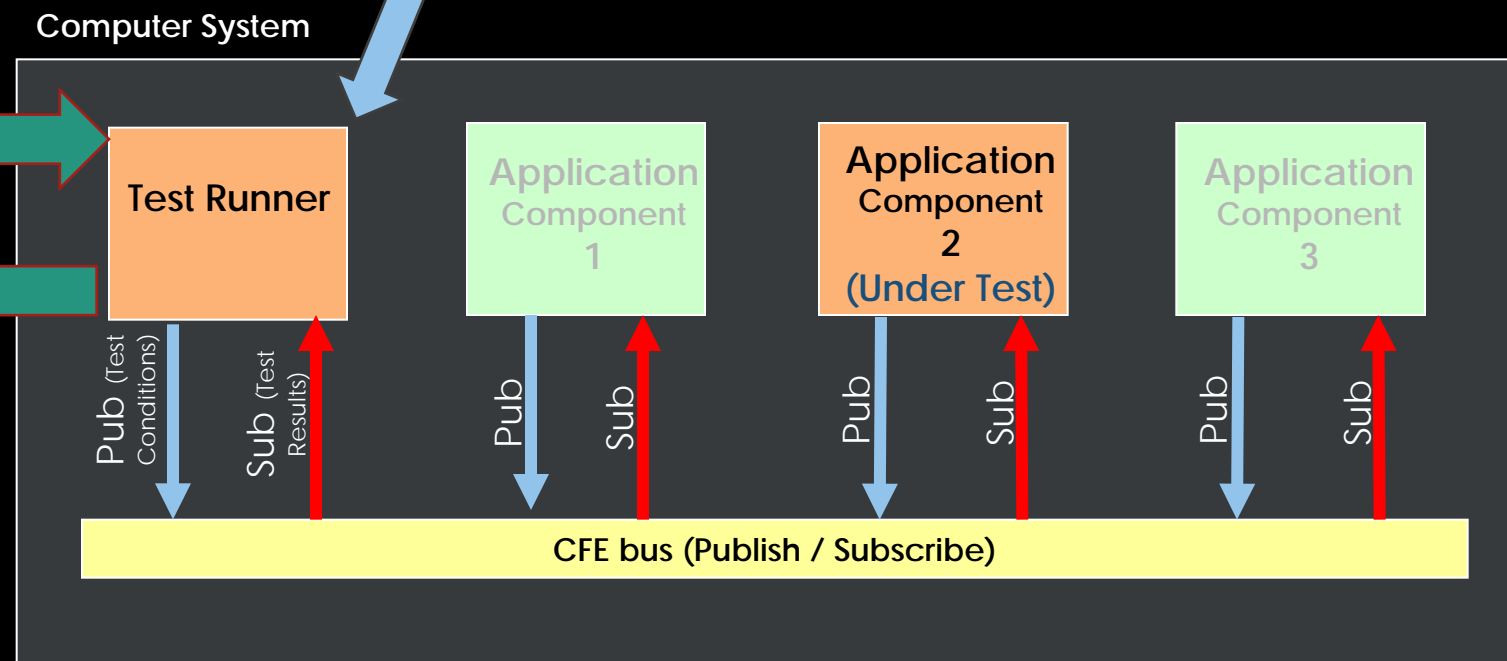
```
Test2
...
```

Could potentially act
as H/W simulator

Test Results

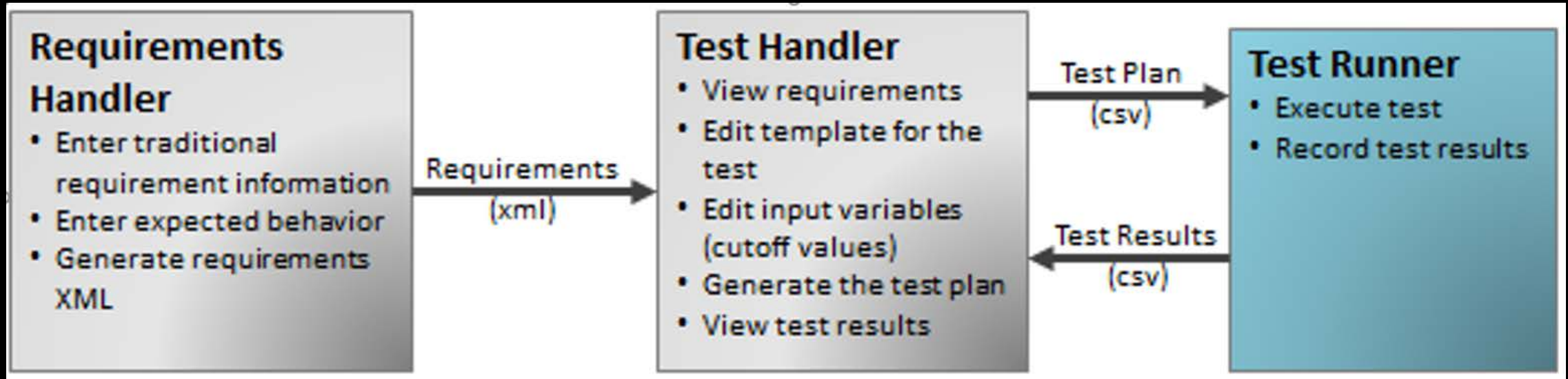
```
Test1
<pass>

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



ART Flow Diagram

High-level diagram of Autonomy Requirements Tester



Requirements in XML format

Requirements

AUT-3	Monitor Battery Temperature	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)
-------	-----------------------------	---

Behaviors

- No action should be taken if Battery temp okay
- All data needs to have consistent readings over a period of time such as 5 of 6 readings must be the same
- There is a limit to the number of times a command can be sent to the hardware

XML Requirements

```
<?xml version="1.0" encoding="UTF-8" ?>
<Requirements xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cmn="http://www.omg.org/spec/ReqML/20140401/ReqML.xsd" >
  <Requirements>
    <Requirement id="AUT-3" requirementId="AUT-3" />
    <Parents>
      <Parent id="BC-1" />
    </Parents>
    <Categories>
      <Category deviceHealth />
      <Category interfaceHealth />
      <Category telemetryMonitoring />
    </Categories>
    <Title>Detect Loss of PSC Telemetry</Title>
    <Rationale>Protects against operating unit without having access to telemetry</Rationale>
    <Contexts>
      <Text>Autonomy shall perform the following tiered response if PSC telemetry is stale (not updating)</Text>
      <Condition>PSC telemetry is stale (not updating)</Condition>
      <Subject>Autonomous system</Subject>
      <Actions>
        <Action>Soft-reset the PSC</Action>
        <Action>Power-cycle the PSC</Action>
        <Action>Switch the PSC (via a CIM side switch)</Action>
      </Actions>
      <Object>PSC</Object>
      <Value>false</Value>
    </Contexts>
  </Requirements>
</Requirements>
```

Generate Test Plan From Requirements

Observation from Solar Probe Plus 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 template** that could be reused for generating tests
- Some additional parameters are needed in addition to the template. A **display** was developed to collect this data.

Template For Generating Test Specification

AUT-3	Monitor Battery Temperature	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)
-------	-----------------------------	---

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 2^d action
 3. Inject fault corrected by 3^d action
 4. Inject unrecovered fault

Data Entry To Enable Test Generation From Template

Enter Design Values To Construct Initial Test

AUT-3	Monitor Battery Temperature	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)
-------	-----------------------------	---

Enter values from design file ...

M	<input type="text" value="5"/>	N	<input type="text" value="6"/>	Persistence (m of n)
	<input type="text" value="9"/>			Max fire count
	<input type="text" value="2"/>			Priority
	<input type="text" value="Enabled"/>			Initial rule state (enabled/disabled)
	<input type="text" value="battery_temp"/>			Battery temperature variable name (default from rqts xml)
	<input type="text" value="160"/>			pre-defined limit
	<input type="text" value="PSC_reset_cmd"/>			Soft-reset the PSC command name (default from rqts xml)
	<input type="text" value="PSC_pwr_cycle_cmd"/>			Power-cycle the PSC command name (default from rqts xml)
	<input type="text" value="change_CIM_side_cmd"/>			Switch CIM side command name (default from rqts xml)

XML Test Plan

```
<td:Action>
  <td:Action ID="aa2" name="Initialize" xsi:type="td:SessionAction">
    <td:Description>Set initial state (rule fire count set to 0)</td:Description>
    <td:Parameters>
      <td:Parameter ID="aa2p1" name="RULE_FIRE_COUNT">
        <td:Description>Rule fire history.</td:Description>
        <td:Value>
          <td:Data xsi:type="cr:Integer" value="0"</td:Data>
        </td:Value>
      </td:Parameter>
    </td:Parameters>
    <td:Behavior>
      <td:Description>Set initial state. The schema can possibly be extended to acc
    </td:Behavior>
    <td:Outcome>
      <td:Outcome ID="aa2o1" value="Done">
        </td:Outcome>
      </td:Outcome>
    </td:Outcome>
  </td:Action>
  <td:Action ID="aa2" name="Set_Temp_Below_Threshold"
  xsi:type="td:SessionAction">
    <td:Description>Rule not firing check.</td:Description>
    <td:Parameters>
      <td:Parameter ID="aa2p1" name="BATTERY_TEMP">
        <td:Description>Battery temperature.</td:Description>
        <td:Value>
          <td:Data xsi:type="cr:Integer" value="34">
            </td:Data>
          </td:Value>
        </td:Value>
      </td:Parameter>
    </td:Parameters>
```

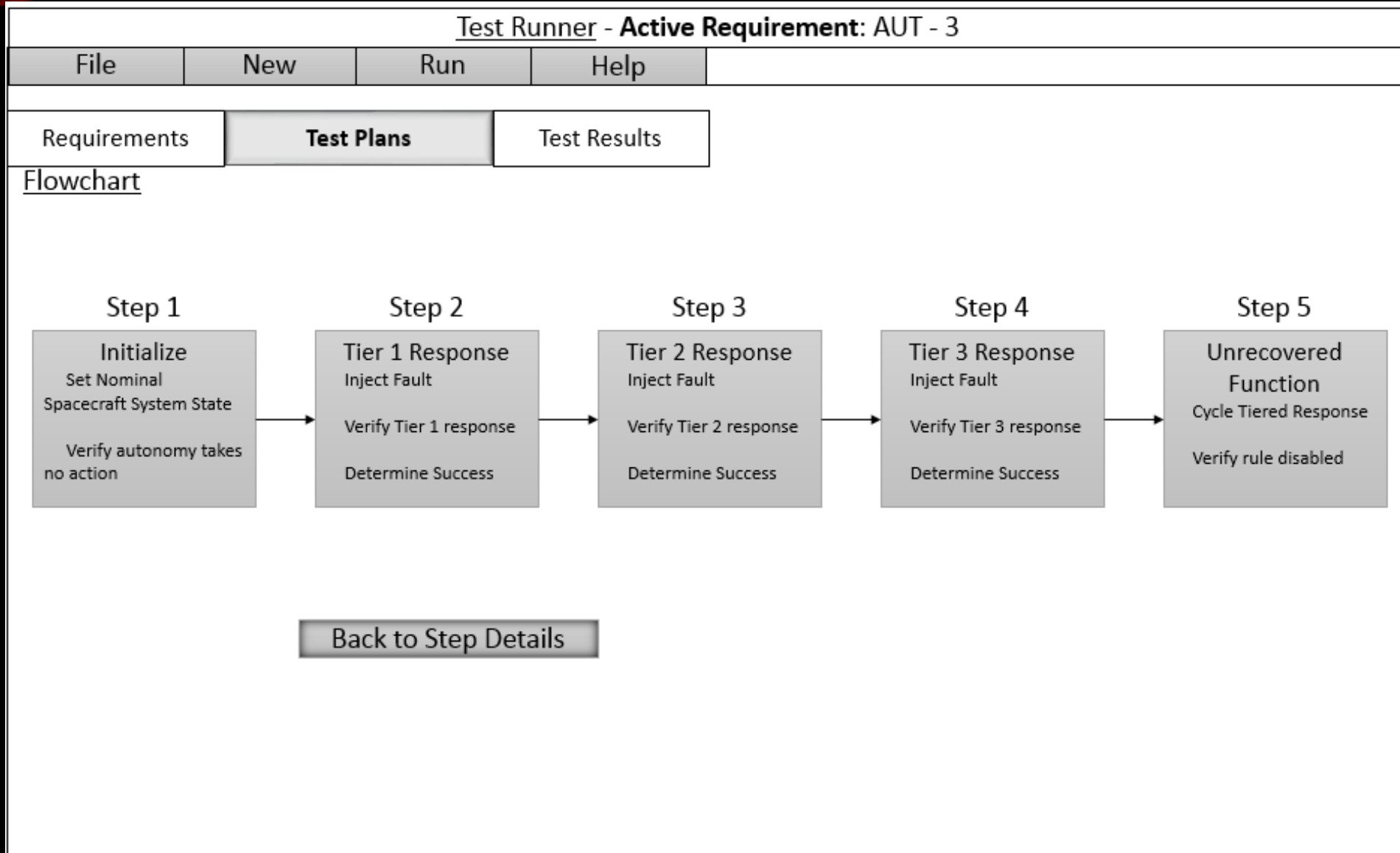
Data Model Based on IEEE Standards

adopted by the Institute of Electrical and Electronics Engineers (IEEE) as a standard (IEEE Std 1671-2010)

Examples: Browsing Requirements

<i>Autonomous Rule System</i> Requirements Overview				
Requirements	Test Plans	Test Results	Requirement AUT-3 Monitor Battery Temp	
<u>Requirement ID</u>	<u>Title</u>		ID	
AUT-1	Detect Loss of Telemetry		AUT-3	
AUT-2	Detect Invalid Telemetry		Name	
AUT-3	Monitor Battery Temp		Monitor Battery Temp	
AUT-4	Monitor Battery State of Charge		Categories	
AUT-5	Detect Critically Low State of Charge		Device Health, Thermal Monitoring	
AUT-6	Battery Heater Power On		Parents	
AUT-7	Battery Heater Power Off		SC-1	
			Rationale	
			Protects against a potential PSC control fault that could cause excessive battery temperature.	
			<u>Details</u>	
			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	<u>Battery temp</u> > 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

Examples: Test Plan (Flowchart)



Examples: Test Plan (Details)

Test Runner - Active Requirement: AUT - 3

File New Run Help

Requirements **Test Plans** Test Results

Step 2: Tier 1 Response

Sequence

Step ID	Step Name
Step 1	Initialize
Step 2	Tier 1 Response
Step 3	Tier 2 Response
Step 4	Tier 3 Response
Step 5	Unrecovered Function

[View Flowchart](#) [Add Test Plan Step](#)

Inject fault ▼

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 ▶

Examples: Test Results

Test Runner - Active Requirement: AUT - 3

File	New	Run	Help
------	-----	-----	------

Requirements	Test Plans	Test Results	Initial autonomous rule test Results
<u>Date & Time</u>	<u>Test Plan Name</u>	<u>Outcome</u>	<input type="button" value="View Flowchart"/> <input type="button" value="Export Test Results"/>
8/11/2016 – 9:34 AM	Initial autonomous rule test	Failed ❌	

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

Advantages of this Method

- Start test driven development early
- Express autonomy requirements in terms of expected behavior
- Support pre-integration testing
- Make integration testing time more productive – no logic errors in software
- During integration, if software changes are required:
 - Make the changes
 - Re-run the pre-integration test to ensure no errors were inadvertently entered
 - Resume integration testing

Innovations

- Represent requirements and link with intended behaviors for testing the requirements (Survey issue #1)
- Formal data models for requirements, behavioral expectations, test specifications, and test results (Survey issue #1)
- Use of template to drive the elaboration of test specifications
- Integration of the testing mechanism with the operational environment
 - 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 (Survey issue #2)

Next Steps

- Complete development of the ART tool and associated user interfaces
- Identify how to support higher levels of integration testing
- Identify how to support additional types of autonomy requirements
- Update overall documentation and documentation for all requirement schemas (formal data models)