Employing the SAF Standard in the Defense Domain

Mike Berenato

SSTC Conference May, 2011
<table>
<thead>
<tr>
<th>1. REPORT DATE</th>
<th>MAY 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. REPORT TYPE</td>
<td></td>
</tr>
<tr>
<td>3. DATES COVERED</td>
<td>00-00-2011 to 00-00-2011</td>
</tr>
<tr>
<td>4. TITLE AND SUBTITLE</td>
<td>Employing the SAF Standard in the Defense Domain</td>
</tr>
<tr>
<td>5a. CONTRACT NUMBER</td>
<td></td>
</tr>
<tr>
<td>5b. GRANT NUMBER</td>
<td></td>
</tr>
<tr>
<td>5c. PROGRAM ELEMENT NUMBER</td>
<td></td>
</tr>
<tr>
<td>5d. PROJECT NUMBER</td>
<td></td>
</tr>
<tr>
<td>5e. TASK NUMBER</td>
<td></td>
</tr>
<tr>
<td>5f. WORK UNIT NUMBER</td>
<td></td>
</tr>
<tr>
<td>6. AUTHOR(S)</td>
<td></td>
</tr>
<tr>
<td>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</td>
<td>Mission Solutions Engineering, 304W 304 W Route 38, Moorestown, NJ, 08057</td>
</tr>
<tr>
<td>8. PERFORMING ORGANIZATION REPORT NUMBER</td>
<td></td>
</tr>
<tr>
<td>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</td>
<td></td>
</tr>
<tr>
<td>10. SPONSOR/MONITOR’S ACRONYM(S)</td>
<td></td>
</tr>
<tr>
<td>11. SPONSOR/MONITOR’S REPORT NUMBER(S)</td>
<td></td>
</tr>
<tr>
<td>12. DISTRIBUTION/AVAILABILITY STATEMENT</td>
<td>Approved for public release; distribution unlimited</td>
</tr>
<tr>
<td>13. SUPPLEMENTARY NOTES</td>
<td>Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License</td>
</tr>
<tr>
<td>14. ABSTRACT</td>
<td></td>
</tr>
<tr>
<td>15. SUBJECT TERMS</td>
<td></td>
</tr>
<tr>
<td>16. SECURITY CLASSIFICATION OF:</td>
<td></td>
</tr>
<tr>
<td>a. REPORT</td>
<td>unclassified</td>
</tr>
<tr>
<td>b. ABSTRACT</td>
<td>unclassified</td>
</tr>
<tr>
<td>c. THIS PAGE</td>
<td>unclassified</td>
</tr>
<tr>
<td>17. LIMITATION OF ABSTRACT</td>
<td>Same as Report (SAR)</td>
</tr>
<tr>
<td>18. NUMBER OF PAGES</td>
<td>29</td>
</tr>
<tr>
<td>19a. NAME OF RESPONSIBLE PERSON</td>
<td></td>
</tr>
</tbody>
</table>

Standard Form 298 (Rev. 8-98)  
Prescribed by ANSI Std Z39-18
Aegis Open Architecture (AOA)

- Aegis Open Architecture goals
  - Provided opportunity to modernize the Aegis Weapon System design
  - Re-architect applications to improve openness and modularity
  - Employ COTS products when possible and practical for system services
  - Unify a fault-tolerant design across subsystems
Open Architecture System Management (OASM)

- **OASM Goals**
  - Improve consistency of and access to system status information
  - Modernize approach to managing COTS computing equipment
  - Modernize the approach to Application Management
    - Provide framework for fault-tolerant application designs
    - Employ COTS high-availability software if possible and practical
    - Adhere to industry standards
OASM Application Management Scope

• Configure applications & computing nodes into a system
  – Define SW hierarchy, redundancy models, SW assignment to nodes

• Launch, monitor, terminate SW components

• Recover from HW & SW failures
  – Form & monitor computing cluster (from configured nodes)
  – Assign/re-assign active & standby roles for SW components
  – Clean up after erroneous SW component terminations
OASM Application Management Scope (continued)

- Attempt repair (restart/reboot) to reinstate SW components & nodes

- Interface to management clients
  - Provide SW & node cluster structure & status
  - Provide administrative controls (startup, shutdown, repair)

- Provide High Availability (HA) services to running SW components
  - Component cooperates with OASM in monitoring its own health
  - Component receives its HA state (active or standby) from OASM
  - Component issues & subscribes to notifications via OASM
    - Abnormal & state-change events
  - Component can obtain any subsystem’s availability state from OASM
Open Standards Analysis

- A set of evaluation criteria was established

- Each standard was evaluated in terms of its ability to meet that criteria

  - Example criteria:
    - Ability to logically group applications
    - Ability to start/stop applications in groups
    - Allow user specification of application dependencies
    - Support application failovers

- A Service Availability Forum (SAF) standard was ultimately selected based on the evaluation criteria
Service Availability Forum (SAF)

- The SAF is a consortium that promotes open standards for mission critical systems

- The SAF’s goals align well with DoD goals:
  - Faster time to market for applications
  - Reduced life-cycle costs
  - Simplified introduction of “best in breed” software components
  - “There is no upside to downtime”

- SAF spec is comprised largely of two specifications:
  - Hardware Platform Interface - HPI
  - Application Interface Standard - AIS
## SAF Application Interface Specifications

<table>
<thead>
<tr>
<th>AIS Specifications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Membership Service</td>
<td>Availability Management Framework</td>
</tr>
<tr>
<td>Notification Service</td>
<td>Event Service</td>
</tr>
<tr>
<td>Checkpoint Service</td>
<td>Log Service</td>
</tr>
<tr>
<td>Information Model Management</td>
<td>Messaging Service</td>
</tr>
<tr>
<td>Naming Service</td>
<td>Platform Management Service</td>
</tr>
<tr>
<td>Security Service</td>
<td>Software Management Framework</td>
</tr>
<tr>
<td>Timer Service</td>
<td>Lock Service</td>
</tr>
</tbody>
</table>

Visit [http://www.saforum.org](http://www.saforum.org) for all SAF specifications & tutorials
OASM Application Management and the AIS

- OASM App Mgmt is based on the Service Availability Forum (SAF) Application Interface Specification (AIS), version B.02.01, January 2005

- The AIS Availability Management Framework (AMF)
  - Specifies a high availability model, redundancy models, auto-repair behaviors
  - Specifies an application API for HA management

- The AIS Notification Service was also implemented
Key Concepts in AIS

• Computing Cluster
  – Physical nodes that correspond to actual machines

• Components
  – Generally one or more software processes (as in Unix/Linux)
  – Lowest level of managed entity recognized by AMF

• Service Units
  – Composed of one or more components
  – Assigned service instances (work units) by the AMF
  – Can be configured to have redundant instances for fault-tolerance

• Service Groups
  – Collection of Service Units that protect service instances against failures
  – Characterized by a SG Redundancy Model (2N, M+N, etc.)
Example Cluster

Cluster 1 contains:
- 6 Components
- 4 Service Units
- 2 Service Groups
- 2 Nodes

Node 1
- Service Unit 1
  - Component 1
  - Component 2
- Service Unit 2
  - Component 3

Node 2
- Service Unit 1
  - Component 1
  - Component 2
- Service Unit 2
  - Component 3

Service Group 1
- Service Unit 1
  - Component 1
  - Component 2
- Service Unit 2
  - Component 3

Service Group 2
COTS Product Evaluations

• A set of evaluation criteria was established

• Products were initially evaluated with test applications in an unclassified environment
  – Evaluation included both functionality and performance

• Examples of important criteria:
  – Extensibility of the product
  – Ability to set thread priorities
  – Customizable logging levels
  – Vendor’s presence in the marketplace

• A demonstration of candidate products was provided for the stakeholders
COTS Product Evaluation

• Ultimately GoAhead’s SelfReliant product was chosen

• A SAF compliant product was on GoAhead’s roadmap but their product development schedule and our ship delivery schedules did not align

• In reality we needed the product before it was commercially available
OASM Development

• In order to meet our schedule a custom solution was developed

• An OASM value-add layer was created to insulate the tactical applications from the underlying COTS product

• This layer provided:
  – SAF APIs to the tactical components
  – The ability to add instrumentation within the API
  – An adaptive layer for future COTS product insertion
OASM Development

- The OASM product extended the functionality of the basic COTS product in the following areas:
  - System Configuration files used to create the System Model are validated and parsed at system initialization
  - Nodes receive requests to start/stop applications from a centralized manager
  - Data recording of system events, state changes, etc.
  - System model data is published via SNMP for subsequent use by a control agent
Management-Client Interface

• Library of C++ classes providing access to OASM Management information

• Management information obtained via SNMP according to SAF MIBs

• The Management-Client interface provides access to a repository of system SAF objects
  – Finer granularity of information

• This interface provides the ability to modify an object’s state within the model from an outside source
OASM Status Tracking Service

• A System Status Tracking capability was created to allow components to monitor other component’s status.

• This service composes SAF status into overall Up/Down for tactical components:
  – Higher level of granularity of information.

• The Status Tracking Service is not a SAF capability, but a derived capability crafted from SAF constructs.
Notification Service

- The Notification Service is used by a component to send a system event notification to interested subscribers.
- Notifications have built-in semantics that further define the type and reason for the notification. Our implementation utilized two SAF notification types:
  - Alarms – examples:
    - Fatal exceptions
    - Loss of critical resource
    - Threshold limitations exceeded
  - State Change – examples:
    - Initialization Complete
    - Channel Up
    - Operational Mode (Tactical, Training)
Notification Service

- Our implementation utilized DDS as the message transport
- Notifications are data recorded to aid in root/causal analysis
- Common classes of notifications were provided for users
- Reader utility classes are provided to allow for search and retrieval of notification data
Implementing the Product

- An OASM Integrated Product Team was established that was comprised of a design team, development team and integration team

- Weekly meetings with the tactical product areas were held to discuss requirements and the SAF adoption process

- The design team worked to ensure that the OASM requirements supported the product area requirements

- The development team created a product that provided the necessary abstractions between the SAF API and the SelfReliant product

- The integration team focused on lab activities and working with each product area to ensure successful integration of the product
Integrating OASM into the Tactical Product Areas

• Integrating OASM into the individual product areas was a challenging task

• Each product area contained a unique resource management implementation

• Some product areas were already componentized and easily adopted the new resource management scheme

• There were also several legacy applications that required some significant rework to integrate with OASM
Integrating Open Architecture Components

- Tactical applications that were designed with an open architecture philosophy were easily integrated

- Component based architectures fit well within the SAF

- Component dependencies were easily established

- Recovery designs were crafted from SAF capabilities
Integrating Legacy Components

• A subset of tactical applications were leveraged from prior development efforts

• These applications were based on legacy designs that contained proprietary resource management solutions

• Each application was modified to accept the new service
  – Remove legacy system management solution
  – Adding the new APIs

• In most cases the tactical applications had implemented a service layer for their resource management capability

• We found that legacy recovery requirements fit well within the SAF
Integrating Other Applications

• Some components could not accept the OASM API
  – COTS products with no access to source code
  – Legacy products used on other projects that did not have OASM as the resource management service

• A custom “wrapper” was developed to address these components and serves as a proxy
  – OASM launches and monitors the wrapper
  – Executable is launched via the wrapper
  – The wrapper executes the OASM API on behalf of the application
Deploying the Product

- The OASM product was successfully deployed on the USS Bunker Hill CG-52 as part of the Navy’s Cruiser Modernization Effort.

- OASM will be deployed as a part of the Aegis Modernization efforts on destroyers and cruisers.
System Level Integration

• Integrating multiple product areas with OASM went fairly well
  – Minor network configuration changes needed

• Our system model became large and complex
  – Ultimately we revisited our node cluster organization

• OASM logs become critical for a first level assessment of multi-component failures

• Data recording of key events provides critical information for performance analysis and root cause analysis for failures
Summary

• Commercial standards may not fulfill all of your requirements
  – Standards can be augmented with “value add” services
• Look for COTS solutions that are extensible and mature
• Find vendors who are willing to modify their product when needed
• Evaluate the product in terms of its functional capabilities and performance to gain confidence in it
• Recognize that some components will have unique requirements and need to be managed differently
• Encourage design teams to meet often and openly discuss requirements
• Stand up an integration team and embed them with product developers
Acronyms

- AIS – Application Interface Specification
- AMF - Availability Management Framework
- AOA – Aegis Open Architecture
- COTS – Commercial off the Shelf
- DDS – Data Distribution Service
- HA – High Availability
- OASM – Open Architecture System Management
- SAF – Service Availability Forum
- SNMP – Simple Network Management Protocol
Mike Berenato
Software Engineer Senior Principal Leader
t +1.856.252.2057 michael.berenato@missionse.com
304 W Route 38  |  Moorestown, NJ  08057  | www.missionse.com