Open Software Standards for Next-Generation Community Satellite Software Packages

June 2017

David Hogan
Alex Werbos, Scott Zaccheo, Craig Oliviera, Dan Hunt, and Erik Steinfelt

Atmospheric and Environmental Research, Inc.
Outline

• The Challenge: Developing Next-Generation Systems
• The Solution: Open Standards for Community Processing
• Building an Open Standard
  – Componentized Architecture
  – Metadata Model
  – Data Fusion
  – Choosing Appropriate Interfaces
• Prototype Implementation: AER Algorithm Workbench
• The Path Forward
The Challenge: Developing Next-Generation Systems

The Solution: Open Standards for Community Processing

Building an Open Standard
  - Componentized Architecture
  - Metadata Model
  - Data Fusion
  - Choosing Appropriate Interfaces

Prototype Implementation: AER Algorithm Workbench

The Path Forward
The Challenges

• New satellites and instruments offer new capabilities but with increasing complexity, data volumes and data rates
• Speed introduction of new and improved algorithms
• More complex algorithms including data fusion from multiple instruments and other data sources
• Tighten the Research-to-Operations-to-Research loop
• Broaden the algorithm developer community
• Add increasing capability for collaboration
• Flexibly employ new computing technologies such as cloud computing
• The Challenge: Developing Next-Generation Systems

• The Solution: Open Standards for Community Processing

• Building an Open Standard
  – Componentized Architecture
  – Metadata Model
  – Data Fusion
  – Choosing Appropriate Interfaces

• Prototype Implementation: AER Algorithm Workbench

• The Path Forward
Solution: Open Standards for Community Processing

• Overall Objective
  – Open frameworks that allow easy integration and test of complex, high-performance algorithms from multiple contributors

• Support R2O2R Cycle
  – Simplify transition of software from development to operational environments, as well as permitting new and updated software to be tested in development environments and returned to operations

• Flexible Integration of Complex Algorithms
  – Powerful tools for integration of algorithms across multiple missions and infrastructures, including modification of spectral characteristics and cadences

• Reduce Manual Configuration
  – Complex and evolving systems require minimized manual intervention to prevent error and reduce overhead

Open interfaces facilitate robust interoperable systems
• The Challenge: Developing Next-Generation Systems
• The Solution: Open Standards for Community Processing
• Building an Open Standard
  – Componentized Architecture
  – Metadata Model
  – Data Fusion
  – Choosing Appropriate Interfaces
• Prototype Implementation: AER Algorithm Workbench
• The Path Forward
Key Solution Elements

• Algorithm component architecture
  – Algorithms must be transferable to different environments
  – Well-defined interfaces for compatibility with a broad range of infrastructure and support software

• Metadata model
  – Describes key component characteristics in machine-readable form
  – Complete system configuration is expressed as data-driven model
  – Layered design facilitates adaptation to different environments

• Data Fusion controller
  – Coordinates algorithm and system configurations to schedule execution
  – Has generic interfaces for monitoring multiple asynchronous data streams in varied environments
Algorithm Component Architecture

- User Workstations and Laptops
- Local Servers
- Cloud Processing System

- Execution Interface
- Scientific Logic
- Data Model Interface

- Algorithm Metadata
- Algorithm Descriptor Interface

- System Analysis Tools
- Automated Integration
- Visualization Tools

- File-based infrastructure
- Distributed Data Servers

Remote Sensing Systems, Applications and Technologies

2017 CSPP/IMAP Users' Conference
Component-Based Design

• All data processing elements of the system are encapsulated as components
  – Data processing algorithms, importers, and exporters are wrapped in common interfaces
  – Standard approach allows portability

• Components provide complete package for integration
  – All software, accessed using standard interfaces
  – Metadata describing functionality

• Components are reusable
  – Multiple processing infrastructures, workstation vs. server vs. cloud
  – Adapt to new platforms and sensors
  – Use common tools for visualization and analysis
Algorithm Metadata Model

• Describes software using programmatically-accessible interfaces
  – Enables automation and flexible tools that can be reused in multiple contexts
  – Move toward shared algorithm baselines that do not require re-engineering for each new system, leverage previous development

• Layered approach allows reuse in different contexts
  – Science layer describes inherent capabilities of software components, is used wherever that software is run
  – System layer describes abstract interconnections between processing components, allows processing systems to be ported between multiple environments
  – Instance layer describes precise mapping of system to concrete computing resources, allows configuration to be effectively CM’d and audited, rather than relying on brittle startup scripts

• Engineering to provide enough commonality to be useful
  – Avoid overgeneralization that offloads work to configuration rather than system
Data Fusion Controller

High-Rate Data

Low-Rate Data

Model Data

Complex Data Fusion Application

Application Component Execution Metadata

1 2 3 4 5 6 7 8 9 10 11 12

1 2 3 4

1 2

2 1

2 1

2 1

2 1

Time
Data Fusion Controller

• Controller monitors multiple data streams and coordinates the execution of data processing components to meet user requirements
  – Ingests system metadata model
  – Listens for signals from processing infrastructure indicating data availability
  – Generates signals indicating that a particular component should be run when its predecessors are available
• Flexible design allows extension and infrastructure upgrade
  – Easily change data cadence
  – Generic signals interface allows interoperability with both simple and high-performance infrastructures
Interface Design Challenges

- Interfaces between infrastructure and computational software must be chosen carefully, balancing between generic and specific capabilities
- Overhead and risk are incurred with both too-generic and too-specific solutions

**Overly-Generic Interface**

- Small number of built-in functions
- Integration between user software relies on convention
- Requires large amounts of glue code and application-specific configuration
- Pushes burden of implementation on users

**Overly-Specific Interface**

- Infrastructure has large number of application-specific methods and interfaces
- Evolving system needs may require large changes to infrastructure and user software
- Certain types of software may not match conceptually, and be completely unable to fit
Choosing Effective Interfaces

- AER has hard-won experience in developing ground system interfaces
- Lessons learned inform careful selection of interface capabilities to minimize overall cost and risk

Balanced Interface

- Infrastructure has carefully-chosen methods to support broad categories of user needs
- Easy to add new user software, infrastructure has data-driven capabilities to adapt
- Smaller overall software footprint, less configuration required for working system
• The Challenge: Developing Next-Generation Systems

• The Solution: Open Standards for Community Processing

• Building an Open Standard
  – Componentized Architecture
  – Metadata Model
  – Data Fusion
  – Choosing Appropriate Interfaces

• Prototype Implementation: AER Algorithm Workbench

• The Path Forward
• AER has been developing cutting-edge operational technologies for decades
  – Data Model approach
    • AER has championed the use of common data model interfaces (DMI)s on numerous civilian and DOD LEO/GEO atmospheric/environmental remote sensing programs
  – GOES-R Demonstrated Large-scale operational capability
  – Single common Data Model Interface allowed shared development between distributed AER and Harris teams
  – Updates to algorithm science are easily tested by science developers, and the software is directly integrated into operations

• AER Continues to develop Ground System technology based on lessons learned
  – Reduce overhead and system configuration time
  – Advanced capabilities for multi-nodal and cloud-based computation
AER Algorithm WorkBench: Flexible, Multi-Platform Science

- Standardized interface for algorithms and infrastructure allows both to be swapped without code changes
- Algorithms in different languages run in the same environment
- Users can change/add algorithms, reformat data types and sectors using standard interfaces
The AWB provides all software necessary for processing data
- Ingests unprocessed inputs
- Automatically partitions data
- Provides data to each algorithm
- Collates and processes outputs into user-specified sectors and formats

Scales to meet user needs
- Automatically uses available cores
- User-selectable processing areas
- Tunable to maximize throughput vs. latency
- Runs on laptops, workstations, cloud servers
- Suitable for one-off and continuous processing systems
AWB Multi-Mission Platform

- Algorithm Components adapt easily to diverse missions
  - Simple data reformatters can adapt algorithms to new sources
  - Entire algorithm processing chain can be reused
- Reusing algorithm code across missions fosters collaboration
  - International communities can use the same science and provide more consistent data for users
  - New missions benefit from invested cost in algorithm development

Cloud Mask
Source: Himawari 8 AHI

SST Composite
Source: Simulated ABI

6.4 µm Radiance
Source: Simulated FCI FDHSI
Remote Sensing Systems, Applications and Technologies

AWB Tools for developing new and migrating existing algorithms

- Complete algorithm package has several elements
  - Science code
  - Data interface
  - Algorithm metadata file

- Assemble algorithm packages
  - C++ and Python code can natively interface
  - Other languages can use provided wrappers to easily cross the language barrier
  - Packages require metadata file describing inputs and outputs

- Algorithm packages can be immediately used in AWB cloud processing system

C++/Python Algorithm
AWB C++ Interface
Algorithm Metadata File
Algorithm Package

Fortran/C Algorithm
Provided AWB Wrappers
AWB C++ Interface
Algorithm Metadata File
Algorithm Package
AWB in the Cloud

• Tools for system management
  – Web console can manage all features
  – Upload new execution scripts to continuously process data
  – Instantiate new servers

• Tools for analysis, visualization and data distribution
  – Use built-in google maps visualization to observe data live during generation
  – Data can be distributed to diverse tools using THREDDS
AER Algorithm Architect Tool

- Visualization of science algorithms and data flows
- Dynamically-generated, built from component metadata
- Allows users to inspect and analyze systems
- Demonstrates viability of data-driven execution
• The Challenge: Developing Next-Generation Systems
• The Solution: Open Standards for Community Processing
• Building an Open Standard
  – Componentized Architecture
  – Metadata Model
  – Data Fusion
  – Choosing Appropriate Interfaces
• Prototype Implementation: AER Algorithm Workbench
• The Path Forward
The Path Forward

• Key action areas
  – Community awareness of open software principles
  – Avoid proliferation of multiple standards
  – Prevent vendor lock-in
  – Begin assembling user requirements from various stakeholders

• Lessons learned
  – OTS Software is essential for building systems, but does not solve the problems alone
  – Robust software requires accommodating diverse workflows
  – Configuration can become more burdensome than the software that uses it
Remote Sensing Open Processing Workshop

• We are looking to put together a workshop for interested parties
  – Algorithm Developers
  – System Integrators
  – Government and Organizational Representatives

• Potential Venue
  – 2018 AMS Annual Meeting (Austin, TX)

• Points of Contact:
  – David Hogan (Business) dhogan@aer.com, 781-761-2270
  – Alexander Werbos (Software) awerbos@aer.com, 781-761-2322