Introduction
This work describes the Algorithm Workbench, a set of software interfaces and tools that allow users to effectively create, document, execute, and update software data processing systems. This software is specifically designed to reduce the overhead involved in research to operations, and enable direct sharing of science algorithms across development, testing, and production environments. Each component, shown below, is part of a complete data processing system, which is designed to be adaptable and customizable to user needs in both operational and development contexts.

Algorithm Descriptor Database (ADDB)
- Programmatically-accessible database that stores information about algorithms and datatypes
- All algorithm inputs and outputs are stored in abstracted form, allowing the end-user to assign physical types and grids
- All system development driven by automated tools rather than manual management
- Automated analysis can describe and verify all inputs and data flows, avoiding the problem of a missed data product amongst the hundreds involved in an operational system
- Document generation tools can read ADDB information to create data dictionaries, component listings, etc. By generating these pre-constructed, overhead and error involved in manual management are reduced.
- Fragment-based architecture for ease of distribution
- ADDB information is packaged in a fragment file with the associated algorithm code.
- Individual algorithms can be selected for use without requiring dependencies on unrelated ones.
- No central repository to be distributed and managed

Algorithm Architect Tool
- User can select desired outputs and automatically generate algorithm trees to produce the desired outputs.
- Allows user to select input data files and directly run the displayed tree using the Algorithm Driver
- Exploration features give user the ability to trace dataflows, examine algorithms, and validate that all inputs are present and correct

Inspect and Analyze
- Java-based tool with operational legacy: previous version used to monitor product quality in GOES-R ground segment
- Offers automated re-projection between grids, pan, zoom, and sub-scene sharing
- Capable of automatically updating as new data arrive, allowing users to monitor live processing
- Layered software architecture allows inputs to be gathered from a variety of formats (NetCDF, HDF, etc.) and multiple different infrastructures (file system, database)

Adapting GOES-R Algorithms to Himawari Data
Using the AER algorithm workbench, the operational GOES-R algorithms have been executed, without any code changes, on newly-released Himawari AHI data. By maintaining the same code base across multiple missions, the latest updates can be shared, and users can be provided consistent product outputs across the globe.

To prepare the algorithms for run, a simple mapping was created from the AHI (Himawari) channels to ABI (GOES-R) channels, and provided as an input to the algorithm workbench. In addition, semi-static and ancillary inputs (NWP, Reynolds global SST, snow masks, etc.) were prepared for the associated observation times and sub-satellite position.

In this scenario, we ran the core cloud algorithms, Cloud Mask, Cloud Type and Phase, and Cloud Top Height and Temperature. The results are shown below.