Post-Launch Lessons Learned from the AIRS Science Data Processing System

Evan M. Manning*, Steven Z. Friedman, Albert Y. Chang
Jet Propulsion Laboratory, California Institute Of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099

ABSTRACT

The AIRS Science Data Processing System, responsible for processing data from the 4-instrument AIRS suite, includes 14 separate executable programs and produces dozens of products. These executable programs and products conform to ECS standards for processing and archival at Goddard Earth Sciences DAAC. These standards include format and metadata constraints, and the PGE paradigm. Before launch the AIRS team defined and implemented all PGEs, created simulated test data, verified PGE performance with simulated and ground test data, and verified PGE integration within the GES DAAC processing and archiving systems. To support validation and continued software development, Jet Propulsion Laboratory (JPL) developed a limited shadow production system, and received all instrument data after launch. This in-house system was not designed to process and serve all data, but rather to run experimental versions of our software and to run additional non-deliverable programs in support of validation. These pre-flight preparations paid off, and the first year after launch has been very active for the AIRS science data processing group. Still, lessons can be learned from our experiences during our first year of data processing and post-launch software development. These experiences and observations may be useful to science seams developing future Earth observing instruments.

Keywords: AIRS, AMSU, HSB, Science Processing, PGE

1. INTRODUCTION

For almost as long as the flight hardware was under development, the AIRS Science Processing Software (SPS) and the Team Leader Science Processing Facility (TLSCF) Data Processing System (TDS) was being designed, developed, and tested. Both the science software and the TDS were readied for data processing activities that were to commence shortly after launch. A series of successively more complex tests were conducted during the last months before launch to stress the software and the system, to ensure that they were ready. Initially, TDS hosted data processing tests were performed, testing the system in isolation, to stress local processing capabilities and the science processing software itself. Nearer to launch, test data was streamed from the GES DAAC to JPL, simulating post-launch data at expected transmission rates. Once the data was received at JPL, the TDS was used to process and archive it locally. In all pre-launch exercises, the SPS and TDS performed well and as designed. While these tests provided assurance and comfort to its developers, there was still a question in everyone's mind whether the system would perform up to expectations when the first AIRS science data was transmitted from the Aqua spacecraft. That would depend on whether SPS and TDS could handle actual data transmitted from Aqua once the system was placed into orbit. In the end, it would boil down to whether the AIRS SPS and TDS would be robust enough to deal with on-orbit data volumes and unexpected data conditions after launch.

The AIRS Science Processing Software consists of several programs or *Program Generation Executives* (PGEs) and Process Control Files (PCFs). The PGEs themselves contain executable code and/or scripts that perform computational operations, while the PCFs provide run-time commands and file staging information. AIRS PGEs and PCFs conform to

* evan.m.manning@jpl.nasa.gov; phone 1 818 354-1172; http://www.jpl.nasa.gov/airs/; Jet Propulsion Laboratory, California Institute Of Technology.

ECS* standards for programming structure and design. AIRS PGEs are organized into the following categories: Level 1A – unpacking satellite data and geolocation; Level 1B – calibration; and Level 2 – basic science product generation. In addition a series of specialized PGEs are used to develop (1) Summary Browse Products (selected Level 1B and Level 2 products) and (2) Match-Up Products, a specialized sampled data set used for validation campaigns. The AIRS SPS has been designed to fully conform to ECS standards for PGE construction and utilizes PCFs to provide runtime directives and file staging information at run time. Additionally, the AIRS SPS utilizes ECS toolkit routines for standard I/O, time and date conversions, geolocation, and metadata construction.

The TDS, used for limited data processing at JPL, was designed to emulate DAAC data processing and archival operations. A specialized suite of hardware and software was designed and built to provide the capability to process Level 0 data through Level 2 and archive all standard and intermediate data products, utilizing the same PGEs developed for use at the GES DAAC. Development of TDS itself posed several challenges, but its operation was instrumental in delivering functioning PGEs to the GES DAAC before launch and in continued development and enhancement of PGEs after launch. In addition, the TDS provided the AIRS Validation Team with a processing environment that could be used to develop all data necessary for validating AIRS Science Data Products.

2. THE AIRS SUITE OF INSTRUMENTS AND MISSION

AIRS is a key facility instrument launched on the EOS AQUA platform on May 4, 2002. The AIRS instrument on Aqua is collocated with the EOS Advanced Microwave Sounding Unit (AMSU–A) and the Humidity Sounder for Brazil (HSB). When combined, these three instruments provide the capability to obtain temperature and humidity profiles through the atmosphere, unsurpassed by any other EOS instruments in orbit today.

The AIRS instrument measures upwelling Infrared radiances at 2378 frequencies ranging from 3.5 μm to 15.4 μm. A limited number of visible/near-infrared (VIS/NIR) channels are also present to provide diagnostic support for the temperature and humidity sounding. The AIRS scan geometry produces a ±49.5° cross-track scan swath with a 1.1° instantaneous field of view to provide twice-daily coverage of the globe from a 705-Km altitude, on a 1:30 P.M. sun synchronous orbit. Each AIRS scan line is completed in approximately 2.7 seconds and consists of 90 footprints, with each footprint covering 45 km diameter area. For AMSU-A, the scan rate is once every 8 seconds, three times longer than it takes for a single AIRS and HSB scan. Thus, while AIRS and HSB scan three lines for a total of three lines of data for a total 270 scanned footprints, AMSU-A scans a single line of data consisting of 30 footprints., with each footprint covering 45 km. This geometry, scan, and footprint a characteristic leads to the alignment of these instruments on board the Aqua platform as illustrated in Figure 1.

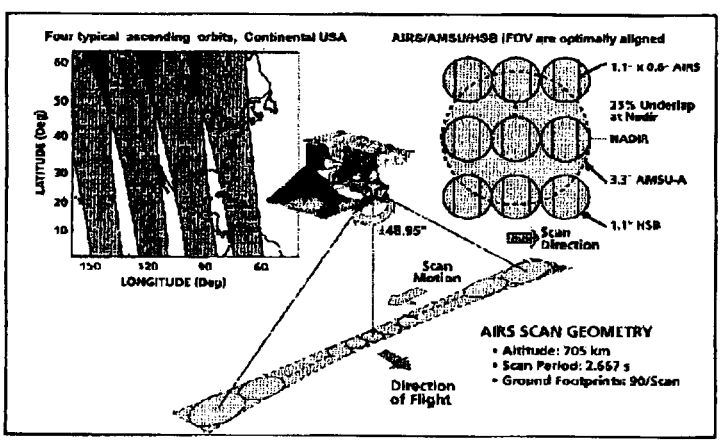

Figure 1. Scan geometry for AIRS, AMSU-A and HSB.

The EOS Data and Operations System (EDOS) at the GSFC DAAC processes Aqua spacecraft data to Level 0 products, resulting in the generation of Production Data Set (PDS) files of instrument science and engineering packets. EDOS processing removes redundancy and time orders the data

*ECS is the EOSDIS (Earth Observing System Data and Information System) Core System, is both a process and a system for defining how PGEs should be constructed to perform science data processing at NASA DAACs. For AIRS, the Goddard Space Flight Center (GSFC) Earth Sciences (GES) DAAC processes and archives all AIRS data.

packets. The Level 0 data is then sent to the GES DAAC for data processing. There, each AIRS PDS file, containing two hours of instrument data, is processed by the AIRS SPS. Some of the Level 0 PDS data is sent to JPL where it is also processed by the JPL TDS as a shadow operation.

PDS files are processed by the SPS software to generate 6-minute granules of Level 1A and Level 1B data for each of the AIRS instruments. Level 1A and Level 1B AMSU granules contain 45 scans of AMSU data, while Level 1A and Level 1B granules of AIRS and HSB data, with their higher scan rates, contain 135 scans of AIRS and HSB data respectively. For AMSU-A, each 6-minute granule contains 1350 footprint observations (gridded 45 observations along track by 30 observations cross track) covers an area of approximately 45 x 45 kilometers of the surface of the Earth. Later, during Level 2 processing, each *retrieval footprint* utilizes one footprint of AMSU-A data and 3 x 3 footprints of HSB and AIRS data to retrieve the required geophysical parameters as specified in Table 1.

3. AIRS SCIENCE PROCESSING SYSTEM (SPS) ARCHITECTURE

The AIRS SPS, developed at JPL, is delivered to GES DAAC for generation, archive, and distribution of AIRS standard products. After validation, users of the AIRS science products can access AIRS products through the GES DAAC's Data Pool (on-line cache) and Web-Hierarchical Search and Ordering Mechanism (WHOM) or the EOS Data Gateway (EDG). The design of the AIRS SPS takes advantage of the definition of standard product size (6-minute granule) in order to develop a system that is easily distributable over a heterogeneous operational environment and is also capable of taking advantage of parallel processing of science data packets. AIRS science data products are generated in three distinct phases:

Phase 1 - Level 1A processing is performed by three Product Generation Executables (PGEs) for AIRS, AMSU, and HSB instruments. Each PGE performs the following for each instrument:

- Synchronizes Level 0 data packets
- Unpacks and reformats Level 0 data packets
- Converts engineering parameters to engineering units
- Performs geolocation (i.e. calculates observed location, altitude, etc.)
- Merges and archives engineering and calibration data
- Puts science ground footprints from instruments into swath format and generates Level 1A swath products.

Phase 2 - Level 1B processing is performed by four PGEs for AIRS (Infrared, and visible), AMSU, and HSB. Each Level 1B PGE reads the Level 1A science and calibration data from appropriate instrument and calculates the radiances. Corrections for polarization, non-linearity, and drift are handled in this phase.

Phase 3 - Level 2 processing is performed by a single PGE. This PGE reads Level 1B data from all instruments and performs a mathematical process called "*retrieval*" that calculates temperature and humidity profiles as functions of atmospheric pressure.

Processing at each of the three phases described above is performed by PGEs. Each PGE is a standalone Unix process that is activated to perform a task. Multiple instances of the same PGE (process) can be activated concurrently to perform the same processing on different granules of science data. The AIRS SPS architecture takes advantage of this capability as well as the design of the AIRS standard science granule size to enable parallel processing of its data. The architecture of the major AIRS SPS PGEs allows for end-to-end processing of independent granules, with no or little

Level 2 Product	Units	Accuracy
Temperature profile	Kelvin	1.0K
Humidity Profile	gm/kgm	20%
Total Perceptible Water	mm	5%
Fractional Cloud Cover	None	0.05
Cloud Top Height	Km	0.5km
Cloud Top Temperature	Kelvin	1.0K
Land Skin Surface Temperature	Kelvin	1.0K
Land Surface Spectral Emissivity	None	0.05
Ocean Skin Surface Temperature	Kelvin	0.5K
IR Spectral Cloud Emissivity (3-16μm)	None	0.05
Total Ozone Burden	Dobson	15%
Cloud-Cleared Brightness Tem.	Kelvin	1K

Table 1. AIRS Standard Products

dependence on other adjacent granules for data (Figure 2).

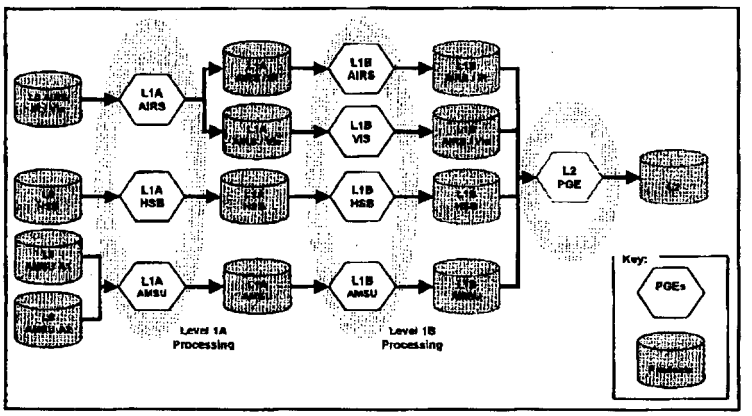

Figure 2. AIRS Science Processing System Architecture and Major PGEs.

As Level 0 data becomes available, appropriate Level 1A PGEs are activated by the production system based on production rules at the GSFC DAAC. Each Level 1A PGE receives Level 0 data from the GSFC DAAC archive and processes it into 6-minute granules of Level 1A data. Each Level 1A PGE can process data independent of other Level 1A PGEs and therefore, Level 0 data from all instruments can be processed concurrently. As shown in Figure 2, the Level 1A PGE used to process AIRS data splits the data into individual Level 1A data products, *infrared* and *visible*.

Upon availability of Level 1A products, Level 1B PGEs are activated based on production rules that govern the processing of Level 1B PGEs. Level 1B PGEs process each 6-minute granule of Level 1A data and generate 6-minute granules of Level 1B (calibrated) products. Each Level 1B PGE performs its processing independently of other Level 1B PGEs and therefore, multiple Level 1A granules can be processed concurrently. In addition, as in the case of Level 1A PGEs, multiple instances of each Level 1B PGE for an instrument (i.e. Level 1B AIRS PGE) can be active at the same time, allowing the processing of different Level 1A granules for the same instrument data concurrently. Level 2 processing requires availability of data from all instruments (AIRS, VIS, HSB, and AMSU) and therefore, it is activated when all appropriate granules of instrument data are available. For each 6-minute set of instrument granules (AIRS, VIS, HSB, and AMSU), the Level 2 PGE retrieves geophysical parameters and produces standard Level 2 products as specified in table 1. Multiple granule sets of Level 1B products can be processed at the same time by activating multiple instances of Level 2 PGE.

Each of the PGEs described above can be distributed over a network of heterogeneous CPUs. Although the operational environment at the GSFC DAAC is centered exclusively on the SGI computers, the JPL TDS operational environment as described below uses a variety of Unix and Linux based CPUs to perform the same function.

4. TLSCF DATA SYSTEM (TDS) ARCHITECTURE

The AIRS Team leader Science Computing Facility (TLSCF) Data System (TDS) provides an automated operation environment to produce AIRS science data products that are needed to support the AIRS post-launch science data calibration, verification, and validation activities scheduled to take place during the first twelve-month of mission operation. Figure 3 shows the architecture of the TDS. The TDS processes AIRS suite data similarly to the DAAC in that it utilizes standard PGEs and PCFs. However, the TDS does not use the same system architecture and controls implemented at the DAAC. Instead, a hybrid system was developed at JPL to support specific needs of the AIRS Science Team Leader and the greater Science Team.

The TDS is a complete end-to-end software solution for processing AIRS suite data, starting with a File Ingestion Subsystem, at the front end, to ingest AIRS Level 0 and other input data. The Job Planning Subsystem, that implements the production rules (i.e. prerequisites), is activated upon receiving data production request created by the TDS operator via the Job Entry Subsystem. Once the production prerequisites are satisfied, the production job is marked *ready*. The Job Scheduling Subsystem takes over at this point to create and archive the data products. Given the vast amounts of data to be processed and archived, this system would not complete without a centralized File Catalog and Management Subsystem. These elements are described in more detail below:

File Ingestion Subsystem (FIS). Upon receiving email notification from the data provider, the File Ingestion Subsystem retrieves the external data product files that were deposited by the data provider in the external data file spooling area. For each data product file, it then creates a metadata file based on the predefined metadata policy for a specific product file type and data source.

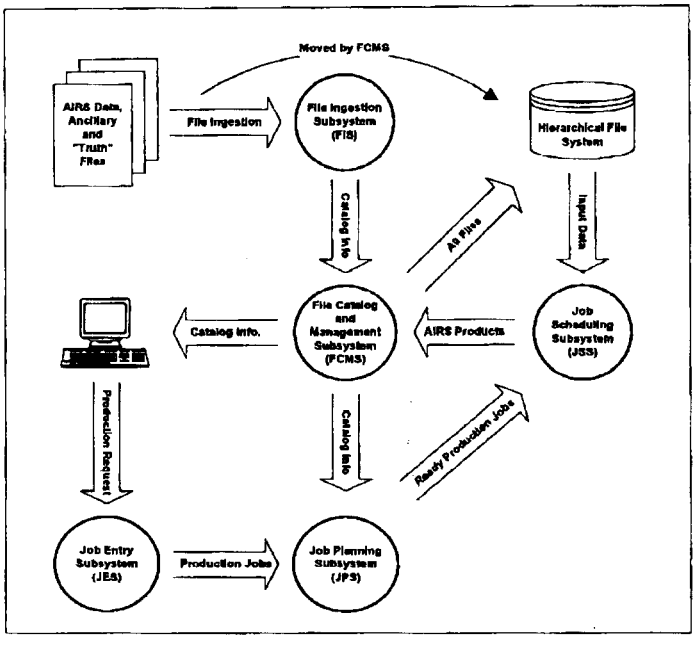

Figure 4. TDS Architecture

Both the external data file and the TDS-created metadata file are submitted to the File Catalog and Management Subsystem. The File Catalog and Management Subsystem subsequently parses the metadata file to create catalog information and stores the data file in its file repository.

Job Entry Subsystem (JES). The Job Entry Subsystem enables the TLSCF operator to enter a production request. It then decomposes the production request into one or more PGE jobs. For each PGE job request, the JES creates a Job Description File (JDF) that is "submitted" to the requested jobs spooling area for processing by the Job Planning Subsystem.

Job Planning Subsystem (JPS). The Job Planning Subsystem periodically retrieves JDF for the newly requested jobs from the requested jobs spooling area. Once a requested job is successfully inserted into the internal pending queue, the JPS removes the corresponding JDF from the spooling area. The JPS uses the predefined PGE production rules to create a dependency list regarding the required and optional input files for each input PGE job that it received. Periodically, the JPS checks the latest available file catalog information to determine if the input dependencies for a given PGE job request have been satisfied. When all the dependencies for a given PGE job are satisfied, the JPS creates a JDF for the *ready* job that contains information needed to initialize the PGE job execution. Then, the JPS submits the *ready* job to the Job Scheduling Subsystem for execution.

Job Scheduling Subsystem (JSS). The Job Scheduling Subsystem periodically retrieves new *ready* JDF from the ready jobs spooling area. Once a ready PGE job is successfully inserted into the internal working queue, the JSS deletes the *ready* JDF from the spooling area.

The JSS allocates a work area (i.e., computer disk space) and creates a Processing Control File (PCF) for each *ready* PGE job with information provided in the corresponding JDF. The PCF is an ASCII text file that contains all the runtime parameters required to execute the applicable PGE. As the system resource becomes available, the JSS executes the PGE job and archives the generated products in the File Catalog and Management Subsystem.

File Catalog and Management Subsystem (FCMS). The File Catalog and Management Subsystem is built on top of the Distributed Object Manager (DOM) developed at the Jet Propulsion Laboratory and has been in operation since 1997 to support various Deep Space Missions. The Distributed Object Manager includes a homegrown on-line file catalog system. The Distributed Object Manager provides command-line programs that allow user to store and catalog a new data file, search for an archived file, and to retrieve an archived file. In addition, DOM also provides application level interface libraries (in C++ and Java) for other software applications to store, search and retrieve files.

5. AIRS SCIENCE PROCESSING SOFTWARE – LESSONS LEARNED

All AIRS SPS PGEs performed well during the first year after launch. During that period, three major software updates were produced and delivered on schedule to the GES DAAC and placed into operations. The success of the AIRS SPS can be attributed to an iterative development and test approach, where all PGEs evolved over a period of time. Each PGE was incrementally developed and tested utilizing simulated data streams, both at JPL and the GES DAAC.

Other implementation and design decisions, many made early on during the evolution of AIRS SPS, have affected the effectiveness of AIRS SPS. They are outlined in the following sections.

5.1. Separation of Instrument Data

Keeping data separate by instrument throughout Level-1 processing and the further separations of AIRS IR from V/NIR and engineering from scene and calibration data have proven highly successful. For product users, this separation minimizes the size and complexity of data sets needed to perform specific tasks. For example, an analysis of AIRS instrument temperatures requires only the AIRS engineering product, and any AMSU analysis requires only the very small AMSU products. The concern with this approach was that granules from separate instruments might not correspond to the same regions when that data is finally combined in Level-2 processing.

AIRS's original inter-instrument synchronization schemes would have required Level-1A to make products containing information from multiple instruments or to compare data streams to ensure that they are synchronized in time and location. This would have led to a combinatorial explosion in special cases to be handled where different instruments are in different operational modes. Instead this concern was addressed by carefully defining granule boundaries in a way that corresponds closely with the mechanism used on the spacecraft to synchronize the instruments, and by adding checks in Level-2 to confirm that data processed together corresponds to the same time and place. No problems have been encountered with this approach. In fact we easily handled pre-launch instrument test data, which typically had data from only one instrument in each data set. Similarly, after launch, when instruments were turned on one-by-one the data could be processed and analyzed in a natural manner.

5.2. Geolocation in Level 1A

Geolocation algorithms are usually included in Level 1B PGEs. However, for AIRS, a decision was made to include these algorithms as part of Level 1A processing instead. The advantage to performing geolocation earlier in the processing stream is that Level 1A products will include location information, making them more useful to diagnose certain data and instrument related problems. For example, it becomes easier to check if a particular anomalous event coincides with the passage of the spacecraft through the South Atlantic Anomaly. But disadvantages include increased volume of Level 1A products and increased complexity in the otherwise lightweight Level 1A process. Thus, a tradeoff between size of the Level 1A data products versus their potential utility was made.

For AIRS, Level 1A product size was not significantly impacted because AIRS's 2378 channels of IR data dwarf the volume of geolocation data. AIRS chose to make geolocation part of Level 1A so that geolocation information would be available to users of Level 1A data.

Many years after this decision to include geolocation within Level 1A data processing was made, the AIRS Project decided not to release AIRS Level 1A data to the public. Consequently, the only users of Level 1A data are AIRS Team members. But these team members have benefited from having geolocation information with Level 1A data both for investigating anomalies in instrument engineering data and for making sense of the first look at calibration and scene data. The part of geolocation information that is most useful for these purposes is not the scene latitude & longitude, which is the core geolocation product for Level 1B users, but other ancillary information such as spacecraft position data, moon-in-view indicators, land/water fraction, and solar glint distance. These geolocation parameters enable the investigation to determine if events such as cooler resets and observation overflow were associated with the South Atlantic Anomaly, polar horns, glints, etc.

A few unexpected drawbacks were also found to result from this decision to include geolocation in Level 1A processing. AIRS geolocation includes land fraction and surface elevation as beam-weighted averages over 15 kilometer or larger spots. These calculations are time-consuming, making the Level 1A PGEs slower than they would otherwise be. However, the overall end-to-end system execution time is not changed, only the duration of Level 1A processing. Finally, this decision also impacts the AIRS Project's delivery schedules. Typically at one year after launch, the Level 1A PGE would be considered in its final form. However, residual uncertainties in instrument pointing have delayed final geolocation adjustments, and updates to the Level 1A PGE are still expected during the second year after launch.

5.3. HDF-EOS Swath Format

Use of HDF (Hierarchical Data Format, a portable binary format) for production of all product files was mandated by the EOS project. In addition they strongly encouraged us to use the HDF-EOS dialect, specialized to structures most often seen in remote sensing data. AIRS uses HDF-EOS Swath for all Level 1A, Level 1B, and Level 2 products. The HDF-EOS Swath structure maps very intuitively to the structure of these basic AIRS products, and no problems have been found using HDF-EOS across architectures. However, HDF-EOS swath does not support all features of HDF, in particular it lacks a facility for annotating individual fields. It is possible to directly access the underlying HDF facility, but AIRS chose instead to conform precisely to the HDF-EOS Swath format and omit annotations.

Internally the AIRS team has adapted relatively easily to HDF-EOS swath formatting, using the HDF-EOS capabilities of MatLab and IDL to help build tools. As yet there are still relatively few external users, and no complaints have been received.

The AIRS Project experienced less success using HDF-EOS data formatting where data is not structured as a continuous series of satellite observations. JPL worked with staff at GSFC's Earth Science Data and Information System (ESDIS) to develop a format for the NOAA PREPQC quality-controlled radiosonde product, which EOS did not want to store only in its specialized BUFR binary format. We selected HDF-EOS Point format, which is designed to facilitate storage of data sets consisting of isolated observations. However, in this case we found that HDF-EOS data-mapping to be less satisfactory. Point formatting imposes arbitrary limits on the numbers of fields per "level," forcing us to add an extra nonintuitive layer of structure. And because HDF-EOS does not pack data fields, we ended up with much larger products containing a small subset of the contents of the original.

AIRS also developed a highly specialized data product to support validation activities. This product, termed a "Match-up" Product, contains AIRS instruments suite data for footprints near selected radiosonde locations. Although matchup data has the same underlying geographically "scattered" structure that led to the selection of HDF-EOS Point for PREPQC data, AIRS utilized HDF-EOS Swath formatting for storage of Match-up data. HDF-EOS swath freed us from the limitations of HDF-EOS Point data formatting and also facilitated the employment of some tools and PGEs developed for our typical 6-minute granule science data files. However, there is a trade-off here as well. The mapping of HDF-EOS structures to this data is not as intuitive. JPL defined the "along satellite track" dimension to skip around the globe in the order of observation, instead of forming a continuous set. Still, in the end, the use of Swath formatting for Match-up files proved to be effective.

5.4. Granule Definition

An important trade-off from the standpoint of usability is granule size. AIRS originally considered granule sizes of one orbit, one-half orbit or one-quarter orbit. However, the AIRS Project realized that users who wanted just polar data, or just tropical just South Pacific data etc., would all end up with large files from which they must subset. Another concern was that for such large granules, granule metadata would be of little use. Metadata are the per-granule information that can be used to select data to order. They include for AIRS such information as percent cloud cover, percent land, bounding latitude/longitude rectangle. For large granules these would all tend toward global means.

Instead, the AIRS Project decided to utilize a granule definition that makes a granule as compact as it can be without creating too many granules per day. A natural size is suggested by the observed swath width, which is about 1700 km.

This is roughly the length covered in 4 1/2 minutes of observation. A whole number of minutes per granule is convenient for users, but AIRS has an 8-second scan cycle, so only multiples of 2 minutes would have a whole number of scan cycles. We considered 4-minute, 6-minute, and 8-minute granules, settling on 6 minutes as the best compromise.

The six-minute size has proven very convenient. Product files are all of manageable size (121 kilobytes for the largest, L1B infrared radiances), 240 granules per day is a manageable number, and having ten granules per hour works out well. One can tell at a glance that granule 120 will correspond to a time near 12:00 UT.

The details of when granules start are more confusing for users. These rules are designed to mimic the inter-instrument synchronization process on the Aqua platform (described in section 5.1 of this paper). By the current rule the last granule of a day starts 34 seconds before the end of its nominal day and contains data mostly from the next day. A minor adjustment to the rules could have made this part more intuitive.

5.5. Target Environment Scope

The original target environment for AIRS SPS software is the ECS (EOSDIS Core System) software environment as run at the GES DAAC. The ECS environment placed many constraints on PGEs to run there, including: restricting the number of PGE instantiations per day, limits on the amount of execution time per each PGE's instantiation, restricting all I/O to utilizing the SDP Toolkit, and the prohibition from using environment variables or command line directives.

AIRS designed its PGEs to conform to these constraints, and in general, they have served us well. JPL's own development and test systems need to run the software while the software is being developed. Additionally, JPL processes instrument test data to support instrument and spacecraft testing. The TDS processes significant amounts of real downlinked data. The TDS was developed to conform to ECS specifications for PGE invocation, data staging, and product generation. The TDS has handled a myriad of data files over the past year, and our localized ECS simulation has performed adequately.

While the ECS centric implementation paradigm has suited JPL and the GES DAAC well, a number of new customers have emerged for AIRS software over time, and they have not benefited in the same manner. The National Oceanic and Atmospheric Administration's National Environmental Satellite, Data, and Information Service (NOAA-NESDIS) developed a rear-real-time AIRS data processing system to process as much AIRS data as possible for redistribution to numerical weather prediction centers within 3 hours of observation. Also, Direct Broadcast downlink stations will receive AIRS data whenever the Aqua satellite is over their horizon. These local data processing centers will process data for local information content. JPL wrapped the AIRS software in scripts that emulate the ECS environment. As we've added customers these scripts have become more and more elaborate. While these scripts do work adequately, they were not designed with knowledge or intent to support our growing pool of end-users. Finally, as new projects begin developing code, they projects may want to use parts of the AIRS software as well. In retrospect, if the full scope of these applications had been anticipated from the start, SDP Toolkit functionality could have been wrapped and isolated from AIRS algorithmic code.

5.6. Testing

SPS testing started early, using simulated data based on weather models. This data was successively converted to simulated Level 2, Level 1B, Level 1A, and Level 0 by simulators written before the corresponding PGE was developed. Level 2 simulation consists of sampling and extrapolation; Level 1B simulation models atmospheric radiative transfer; Level 1A simulation models signal propagation through the AIRS suite of instruments; Level 0 simulation models instrument and spacecraft packaging of data for downlink. These models produced mostly ideal data, though a limited number of expected variations in input were also tested.

In the last year before launch two additional sources of test data were added to the mix: instrument test data and (simulated) test data routed through real upstream ground system. These data sets allowed us to identify and correct

most Level 1 issues before launch and even allowed us to help instrument and ground system engineers identify issues in their areas.

Frequent end-to-end testing, successively bringing in new and/or improved PGEs, adding DAAC to JPL data flows, and finally receiving simulated data from Aqua satellite receiving stations proved to be effective and adequate for mission success. In the first year after launch, no major failure of any operational version of AIRS PGEs has been encountered. The system continues to perform well, both at JPL, the GES DAAC and at NOAA-NESDIS. (Delivery of Direct Broadcast code was not scheduled during the first year after launch.) Still, the testing regime could always be improved. All AIRS testing is based on "success oriented" scripts, and there has not been enough testing of many failure states and conditions.

5.7. Science Team Interface

The AIRS Project directly incorporates Level 2 algorithmic code provided by Science Team participants at a variety of universities and government centers. The AIRS team at JPL performs the role of software integration. Also, when need, JPL modifies the Science Team developed software to conform to AIRS and ECS standards. JPL has not reengineered or redesigned any of the Level 2 code to date, favoring an approach of keeping it recognizable to its originators. Developers at remote institutions, under AIRS overall cognizance, continue to perform algorithm enhancements and bug fixes.

While this approach has succeeded overall, it has not been without its problems. First, it is important to foster a team-oriented involvement. We have been most successful with Science Team organizations where their staff takes an active interest in the success of the overall product. We have found that the best code is developed by those individuals, although at remote location, take ownership and responsibility for delivering quality Level 2 components and communicate frequently with the JPL development staff. The largest negative impact to this development approach has been to restrict JPL's ability to reengineer the code to operate more efficiently in the DAAC environment. While this has not been a problem to date, the system would certainly perform better if Science Team members would have delivered prototype code to JPL, and JPL would have been responsible for the overall PGE's development. Again, the Level 2 PGE does perform up to performance specification at the GES DAAC, and the overall benefit of keeping the Science Team members directly engaged in the code outweighs any negative externalities associated with the development paradigm.

6. TDS LESSONS LEARNED

The TDS, our primary environment for testing new software releases and for producing validation data sets, has performed well over the past year, has met all our basic needs. However, the development of a hybrid data processing system always has its positive and negative points. The following are findings related to TDS:

6.1. Keep As-built Documents Up-to-Date

The TDS code was created over a 1.5-year period by a small team using a spiral development cycle. Each cycle of requirements, design, implementation, and integration and test lasted approximately 4 months. A set of requirements and design documents were written for the initial version. For subsequent versions, only delta-documentation was written. These described only the new requirements and design changes that were implemented in that cycle. Using delta-documentation seemed to make sense, because it was easy to review upcoming changes and traceability, and to track progress during each cycle.

After launch it was requested that a set of as-built documentation be created. This turned out to be a time consuming task even though the process throughout had been documented. The lesson-learned is that delta-documentation is useful for executing each cycle, but the as-built documents should also be brought up to date at the end of each cycle.

6.2. Data Catalog File Catalog Should Have Open File Access for Users

The file cataloging system selected for use by the TDS is a JPL-built system, the Distributed Object Manager (DOM), which has been adopted by many of JPL's deep-space missions. All data products produced at the TDS are ingested into DOM, which catalogs the metadata and moves the product file to an online storage location. DOM provides APIs, command-line and graphical interfaces to allow TDS data users to search the catalog for files based on metadata.

One aspect of DOM is that the data files themselves are stored in an open UNIX file system. DOM uses a logical directory structure based on data-date and type (collection / year/ month/ file-type or collection / year/ month/ day/ file-type, depending on type). Users querying the catalog are returned a UNIX link or fully resolved path to the files that match their search criteria.

The open file system and simple directory structure has the advantage that users knowing only the data date and file type of interest can by-pass the catalog search altogether, provided the filenames are readily interpretable. This is certainly the case for all TDS-generated L1-L2 product files; but not necessarily the case for Level 0 S/C or instrument files from the DAAC, which have names that encode their creation time, not the data time.

We had expected that once the science team and PGE development team realized the utility searching the catalog metadata, they would adopt that interface. However, even with more than 1 year of data in the system (2 million files) this has never happened; users near-universally resort to navigating the file directory. In the end, only the TDS software and those involved in TDS operations use the data catalog searching capability.

6.3. Flexibility to Handle Multiple PGE Versions is Crucial

During the initial requirements stage, the AIRS Project anticipated that it would only be necessary for two 2 PGE versions to be in use within the TDS concurrently, the *baseline* and latest *test* versions. Nevertheless, we designed the flexibility to handle any number of PGE versions. Today we routinely are processing 3 or more versions in a single week (baseline, n*test, GES DAAC validation, etc.).

This capability is facilitated by having high definition in the AIRS PGE Version specification itself, e.g. V2.3.6.1. The first 2 numbers have been prearranged with the Goddard DAAC based on scheduled PGE deliveries (e.g., version 2.7 to be delivered on such-and-such date) and effectively offer only one digit of resolution. The third number is incremented every time there is a delivery to the TDS. The final number is for nightly developer-builds out of configuration management. This ensures TDS has unique version numbers different from that of developers executing their local code.

The full PGE Version is captured in the AIRS filename to minimize confusion of where a file originated. Also for test-runs, the Local Version ID string is placed in the file name, as a further version discriminant. This is used to encode special options that were exercised in the test run (e.g., "RegressionOff").

6.4. Job Dependency Implementation Based on Files, not Jobs has Plusses and Minuses

TDS implements the AIRS PGE production rules based on file dependencies. For each Job, the data-time of each input file is known. Further specifications allow the choice of the input collection (e.g., test-type, baseline) and constraints for minimum and maximum PGE Version (for AIRS files) or for minimum and maximum creation dates (for externally generated files). The production rules are enforced by having Jobs block in the JPS until it finds all their input files in the catalog (or until the Job times-out).

The alternate approach, which is that provided by commercial Job schedulers, is based on Job dependencies. In this approach, Jobs are strung together in a dependency graph (e.g., a particular L1b job starts running when a set of L1a Jobs complete). In this scheme, simply defining a Job that waits for a file would accommodate the arrival of an external input file.

The use of file dependencies works well in TDS because this is the natural way PGEs are defined. Each new Job can be specified independently of other Jobs in the system (however, the Operator must have some knowledge of what files and product versions will be produced by these other Jobs). If a Job dependency scheme had been implemented, the Jobs would have had to be defined to include the archival of products that were generated, since this condition is assumed for downstream Jobs. The result would have looked very much like a file-dependency implementation.

The major downside of the direct file dependency approach is that executing Jobs don't always produce all their expected outputs. Each AIRS PGE has a complex set of rules to determine when to suppress creation of an archivable output file (e.g., all calibration coefficients are bad for a data granule). Each AIRS PGE, depending on type, has 20-2000 input files. If any one of these files is missing, a Job blocks despite all upstream Jobs having completing normally. These blocked Jobs have to be manually evaluated and cleared by the TDS Operator.

6.5. Simple-Minded Job Priority Naming Scheme Works Surprisingly Well

The specification of a Job in TDS is captured in an XML format Job Description File (JDF). By the time a Job can be executed, each JDF contains the full resolved path-filename of all input files and identification of the PGE type (e.g., L1b-VIS) and PGE Version (e.g., 2.6.1.1) to be run. A collection of Jobs is simply a directory full of JDF files. Prioritization of these Jobs in terms of the Job scheduler must either use some file property (e.g. creation time), some entry within the file, or the filename. Within these constraints, a natural parameter to build a priority scheme around is the filename, since this can be decoded without having to read the file contents, and can be changed by Operator after the file is created. Adopting natural alphabetical ordering of the filenames as the priority ordering extends this simplicity.

The form of a workable JDF naming convention was derived through some careful thought and experimentation. In the end, the convention adopted was based on a priority tag + Job descriptor. The Job descriptor consists of data-time (in hierarchical order), Job type and PGE version. This is a property of the Job and can be captured within the JDF file. The priority tag is Operator defined, though the Job Entry Subsystem suggests a default that is normally used.

An example Job Id is: *bl20020418.1600_02.099.S020_L1a_AIRS_2.22.3.32.* Here *bl20020418.1600* is the priority tag, and the rest is the Job descriptor. The Job descriptor starts with a hierarchical encoding of the data time: *02* is the data year, *099* is the day-of-year, and *S020* denotes the standard granule number (1-240) of the first granule of the Job (AIRS granule-level PGEs normally operate on 20 granules at a time). For AIRS quarter-day PGEs, "Sxxx" would be replaced by "Qx", and for daily jobs "Sxxx" would be replaced by "D". Finally, *L1a_AIRS* is the PGE name and *2.22.3.32* the PGE Version.

The priority tag consists of a 2-character prefix encoding the Job priority-class, and a submission-time-tag (year-month-day-hour-min). In the example, *bl* is the priority-class denoting a "baseline" job. Different priority-classes would be used to denote particular test-runs; these are normally capitalized to give the test Jobs higher priority than baseline Jobs. Depending on the situation, the Operator can select the appropriate priority-class to give a set of Jobs any relative priority with respect to other Jobs in the system.

The submission-time-tag is normally not unique to a particular Job, but to a large collection of Jobs. This is because Job Entry Subsystem takes limited Operator input on the data-start and end-time and expands on these to create all the JDFs necessary to fill a specified interval. It keeps these proto-JDFs in a buffer (actually a file directory) until the Operator has created all Jobs to be submitted at once. The Operator then chooses a priority tag (or takes the default based on the current time) and submits the Jobs. Only at this step do the JDFs get renamed and pushed to the Job Planner, all with the same priority-tag. If additional Jobs entered later need to be prioritized within a set previously entered, these merely have to be submitted with the same priority-tag as before (overriding the default current time).

In practice, this simple scheme works surprisingly well. Jobs are processed first according to priority-class; Jobs of the same priority-class are processed in order of submission; Job with the same class and submission time are processed in data-time order, regardless of PGE type. The result is there is smooth processing flow of all PGE levels. Also, the adjustable priority-tag prefix allows easy changing or adjusting Job priorities.

6.6. Off-line Tape Archive System Awkward, But Can be Workable

Because of limited capital resources, TDS data products are stored in a near-line tape archive jukebox, fronted by a raid disk cache. This system is controlled by commercially available hierarchical file system (HFS) software that simulates a UNIX file system that contains all the data files. Except for the several-minute time delay accessing a near-line file, the HFS software manages this virtual file system transparently to the user. The TDS tape jukebox is configured to hold approximately 680 DLT-7000 tapes, each with a data capacity of 35 gigaytes. As of this writing, the disk cache is 6 TB, soon to be expanded to 11 TB.

When the TDS requirement were originally specified, it was naively thought that (1) old data was not of interest and would routinely be deleted from the system, (2) the tape space holding these old files could be reclaimed, and (3) the TDS only had to work for the first year after launch [in fact developers were discouraged from thinking or planning how the TDS would function beyond that]. The first assumption proved untrue; there is always a data user interested in accessing old data. The second assumption, though technically true, was later deemed a high risk by the system administrator. The third presumption also proved false, as the TDS is now expected to be useful throughout the mission.

After 5 months of post-launch operation, it became apparent that the tape archive was on track to fill before the end of the first year. If this had happened, either no new data could be placed in the system (effectively shutting down the TDS), or random tapes would have to be swapped out for new ones, leaving it up to chance when someone would try to access a file that wasn't actually there. This situation would have been aggravated by the fact the HFS software isn't very informative to the user when this happens.

In response to this challenge, we were forced to devise a systematic way to organize the data archival so that there would be control on what data was removed from the system. The facility to map specific filename or directory matches to specific tape-groups is provided by the HFS software, but is not recommended as a means to manage an off-line tape archive system. There were however no obvious alternatives.

First, files were categorized according to top-level collection: test or baseline. All test files were organized into one tape-group per operational year. The baseline files were organized into 6 different overall classifications (Science, L0-Airs, L0-Support, L1b-IR-Rad, L1a-VIR-Counts, L1-Support) based on file type. Each of these was then organized by data month. Each month and classification formed a tape-group on which all matching files were placed. Each tape-group could then be added or removed from the jukebox as a unit. Fortunately, this was easy to configure because the file catalog's directories were organized by data month and file type.

The classifications were derived based on data-usage and collection size. Some data (e.g., L0) were of less interest to the general user and need not be in the system as long as other data types. Although there is a large user interest in L1b-IR files, these are so large (29 gigabytes per day) that only a few months could be placed near-line at a time. However L1B QA support files are much smaller and could be placed in a tape-group with a longer near-line residency.

To minimize human errors when removing or swapping tapes, the tape-group names and dates had to be encoded in the tape Id's and tape labels. This new labeling was complex enough that purchasing custom labels would have been prohibitively expensive. We had to purchase and try several tape-label printing packages before we found one that was suitable.

The transition to rearchive all data from a single tape-group to this new collection of individually managed tape-groups was quite painful, since by this time the system could not support both sets of tapes in the jukebox. Much overtime by the system administrator was necessary to complete this. This process took several months, but was completed before the system reached capacity.

Today, at 14 months post launch, there are 630 tapes near-line and 450 tapes off-line, with the off-line collection growing at 80 tapes per month. A web page instructs the user which data sets are available in near-line tapes. Accepting this inconvenience, the system runs smoothly, and should continue to do so for years to come.

REFERENCES

Navid Dehghani, Evan M. Manning, and Quentin Sun, "AIRS Science Processing System (ASPS): A Description of Architecture and Capabilities, in *Infrared Spaceborne Remote Sensing IX*, Marija Strojnik, Bjorn F Andersen, Editors, Proceedings of SPIE, Vol. 4486, pp. 104-110 (2002).

ACKNOWLEDGEMENT

The authors appreciate the efforts of the entire AIRS software development and test team at JPL including Robert Ando, Amy Braverman, Luke Chen, Christopher Cordell, Solomon De Picciotto, Jose Donhauser, Eric Fetzer, Evan Fishbein, Alex Foo, John Gieselman, Stephanie Granger, Mike Gunson, Mark Hofstadter, William Irion, Bjorn Lambrigtsen, Sung-Yung Lee, Vicky Myers, Quyen Nguyen, Robert Oliphant, Edward Olsen, Zi-Ping Sun, Vivian Tang, Yuan-Ti. Ting. Additionally several members of the AIRS Science Team have supported our efforts. Finally, the authors appreciate the efforts of Navid Dehghani and Quentin Sun, former members of the AIRS software development staff. Sections 2-4 of this document were, in part, based on an earlier publication (see reference) by Deghani, Manning and Sun.

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.