Quality Assurance Subsystem Design Document

Contents

Signatures

Roc Cutri WSDC Manager

Tim Conrow WSDC Architect

J. Davy Kirkpatrick WSDC Lead Quality Assurance Scientist

Heidi Brandenburg WSDC Quality Assurance Cognizant Engineer

Revision history

- 2008-08-01: **Initial Draft** (version 0.5)

Document number

WSDC D-D015

Introduction

Description

WISE science data quality assurance seeks to characterize data during pipeline processing and identify data which do not meet WISE science requirements. This document describes the design of the software subsystem which supports this activity and the processes by which this subsystem interacts with team members and other subsystems.

Supporting Documentation

Detailed information about WISE QA can be found in the Quality Assurance Plan. This plan gives a functional overview of all QA activities and maps activities to high level mission requirements. It is available from the WSDC Document Tree (http://web.ipac.caltech.edu/staff/roc/wise/docs/wsdc_document_tree.html) . Additional background information about the design and operation of the WISE Science Data Center is given in the WSDC Functional Design Document.

Requirements

The QA Subsystem is designed to support the activities and create the products specified by the Quality Assurance Plan.

Jargon

- WISE Science Data Center: WSDC
- WISE Science Data System: WSDS
- Science Operations Center: SOC
- Mission Operations System: MOS

Institutional Roles and Responsibilities in the QA Subsystem

The WISE Science Data Center (WSDC) will generate quality assurance assessments of spacecraft and instrument health. WSDC team members will be responsible for reviewing and acting on QA reports and notifications generated by the automated processing system.

The Science Operations Center (SOC) at UCLA will receive notifications about processing progress and have read-only access to the web-based QA reporting system. The SOC will have full access to the anomaly tracking system and is expected to provide input to the anomaly response process.

The Mission Operations System (MOS) at JPL will receive notifications on progress and have read-only access to the web-based QA reporting system. As with the SOC, MOS will have full access to the anomaly tracking system and is expected to provide input to the anomaly response process.

Products of the QA Subsystem

The QA Subsystem generates products for project team members as well as some products which are used programmatically by other subsystems during processing. Some QA products are generated by application programs delivered as part of other subsystems.

Products Used by the EXEC Subsystem

A few QA products will be used to choose data for submission to the cluster for processing. An example of this is automatically determining which scans to send for Scanframe processing based on Ingest or Quicklook QA results. Scanframe QA products will be used to choose data groupings for follow-on processing in the Multiframe pipeline. The details of these products and processes are yet to be determined.

Products for QA Scientists and Other Team Members

The QA Subsystem generates reports and notification emails for both internal and external use. Reports are generated on

user demand from the most recent pipeline products available in the file system. Notifications, most likely in the form of email, are generated on a regular basis by processes outside the automated pipeline system.

Reports

There are two types of reports. The web-accessible reports are generated on user demand and will be used for QA Scientist review during regular operations. The second type of report is a PDF file which contains the same content as the web-page report, but is more suitable for archiving. This PDF is generated at the time that the QA scientist completes his or her review and signs off on a report. We anticipate the following types of reports: an ingest report per Delivery, a quicklook report per Scan, a scanframe report per Scan, a coadd report per Coadd.

Dynamic Web Page Reports	
Format	XHTML 1.1
Generated	On user demand
Availability	Via Apache server in DMZ. A user account (MOS, SOC, Science Team, or Internal) and password required for access.
How	QA web-application which summarizes and presents various pipeline products. Takes inputs from operations archive.
Updates	QA Scientists are able to review & update information on deliveries, scans, and co-adds. If pipelines are re-run (either full or partially) updates are immediately available.

Archival PDF Reports	
Format	PDF
Generated	After QA Scientist Review
Availability	Via filesystem to internal WISE users. By request to outside users. Potentially may be delivered to IRSA for persistent archive.
How	Upon sign-off by QA Scientist, web-application passes the complete XHTML report to an external job. Job runs the report through Apache FOP to create PDF and places the output in the pipeline directory.
Updates	If a pipeline is re-run and re-reviewed by a QA Scientist, a new archival report is generated.

Notifications

Regular notifications provide summary information about processing events. We expect notifications to be sent directly to interested parties by email. Types of emails:

- **Ingest** notification email. Sent to internal and MOS users after the ingestion of a delivery. Gives information on all delivery contents. Auto generated.
- **Quicklook** notification email. Sent to internal and MOS users after the quicklook pipeline runs for a delivery. Gives information on scans completed in the delivery. Auto generated.
- **Scanframe** notification email. Sent to internal and SOC users daily. Gives information on scans processed in that day. Generated after initial scientist review.

Notification Emails	
Format	Plain text email

Generated	Automatically by pipelines, or via cron job.
Availability	To configured recipients upon pipeline completion or at the time of the cron job (TDB, but likely daily).
How	Assembled and sent via perl script.
Updates	If a pipeline is re-run, the updated results will be reported in the next scheduled notification.

Public Release Products

QA Products released as part of the final image atlas and source catalog are TBD. We anticipate that information gathered during anomaly response activities will be used in preparing the explanatory supplement.

Quality Assurance Processes

QA Processes in the Pipeline

Each major pipeline (scan, frame, coadd) executes a *qa* step toward the end, which runs a Perl wrapper script. This wrapper is an execution wrapper as described in the Functional Design Document. Its inputs are products created during the pipeline and its outputs are the required QA products for that pipeline. To generate these products the wrapper may run subprograms, filter tables, summarize tables, move and rename files, and create plots.

frameqa

The frameqa execution wrapper

- Runs jpgmaker to generate greyscale jpegs of each band and a 3-color jpeg
- Filters and reprocesses the frameset source list for later input to the scan synchronization monitor

scanqa

The scanqa execution wrapper

- Runs the scan synchronization monitor
- Summarizes frame data to generate scan level plots

scansync

Scansync wraps together the programs which make up the scan synchronization monitor. It prepares the inputs for these programs and post processes the output to create jpegs of the derived PSFs. These jpegs and the table containing the moments of the PSFs are used in the QA report. Scansync is run in the quicklook and scanframe pipelines.

Notifications

Notifications are generated by scripts run asynchronously to pipeline operations. A script uses the WISE::QA::Events module to find recent processing events. The script constructs summaries of the events and emails them to a pre-configured list of users.

Reporting and the QA Web Application

The QA web application is the primary method of user interaction with the QA subsystem. It is an application which takes requests from the users for a report on a delivery or scan, and returns a web page displaying detailed information about the

processing, including tables, plots, images, and links to the underlying ASCII files.

The QA web application is built using the Catalyst (http://catalyst.orgl) web application framework for Perl. Catalyst leverages modern Perl amenities like DBIx::Class and enforces a strict Model-View-Controller separation of application components.

When a request comes in from a user, the URL indicates which pipeline the user wants a report on (Ingest, Quicklook, Scanframe, Multiframe) and the ID of the data (Delivery, Scan, or Coadd ID) the user wants to see. This information flows into the controller, which assembles a collection of objects containing content from operations filesystem. The application doesn't blindly pick up all contents of the target directory in the filesystem; desired files are specified in the application's sqlite configuration database. The collection of content objects created from these files is passed to the View component which executes a Template Toolkit (TT) template with the objects and returns the output XHTML to the user's browser.

All team members, including those at outside institutions, will have access to the reports. Only users inside the IPAC network with privileged logins will have update capabilities. Updates will come from QA scientists reviewing and signing off on reports. Upon sign-off the QA application will initiate another request-response cycle to produce an archival PDF of the report.

Since the web application uses the operations file system, it is a risk to both the resources and security of the WISE network. The web application will run as a standalone FastCGI server which has resource usage limits and a built in monitor. Internal and External Apache servers will forward requests to this FastCGI server. In this scenario access from the outside proceeds as follows:

- Web server in the IPAC DMZ authenticates users and takes report requests. It returns any reports already in its cache, otherwise, it runs a shim application which validates the user's URL and forwards the request through the IPAC perimeter to a FastCGI application inside the WISE network.
- The FastCGI application revalidates the request and executes it by accessing the operations filesystem. XHTML is returned to the originating Apache server.

Anomaly Tracking

The WSDC provides anomaly tracking facilities which can be used by all WISE Project members. Detailed information about the Anomaly Response Plan is given in the Quality Assurance Plan. To support tracking we will deploy an off-the-shelf issue tracking system in the IPAC perimeter network. All team members who require access will be given individual accounts on this system.

Anomalies are found by software (see the Notification process above) or team member. In either scenario the issue is brought to the attention of QA scientists who will investigate the problem and if needed initiate a new issue in the tracking system. All parties with interest in the issue will then be able to update the ticket in the tracking system.

To support these activities the issue tracking system must

- Support SSL encryption and password protected login
- Support configurable categories, priorities, and status settings
- Allow attachment of files
- Allow an issue to progress through any status change
- Automatically map a category setting to an email notification list
- Be backed up via simple file system backup

Information and analysis gathered during the anomaly tracking process will be used in the explanatory supplement.

Interfaces between QA and Other Subsystems

QA interacts with other software subsystems through command lines and files. The majority of interaction occurs through files. In normal usage, application programs from other subsystems will output QA data into IPAC and Metadata tables; programs in the QA subsystem will read these tables and use the contents to create plots.

Filesystem

QA processing is dependent on the filesystem specified in the WSDC Functional Design. The root of this filesystem is configurable but subdirectories, file layout, and file naming must conform to the project standard.

Files

The QA subsystem handles several types of file; some formats are unique to IPAC/WSDC and others conform to public standards.

- **IPAC Tables** IPAC Tables are output ASCII tables created by modules and subsystems. They may be created by application programs or by upstream Perl wrappers or pipelines. All tables used directly by the QA to create products, either directly or indirectly by plotting, must have an associated SIS.
- **Metadata Tables** Metadata tables are a specialized form of IPAC Table. As with regular IPAC Tables they may be created by application programs or by Perl wrappers. All metadata tables used directly by the QA subsystem to create products must have an associated SIS.
- **Scalable Vector Graphics (SVG)** SVG is an XML specification for describing two-dimensional vector graphics. In the WSDS plots are written as SVG files for inline display in webpages and PDF files. SVG files are created by Perl code within the automated processing system using the gnuplot SVG engine. Most of these files are made by wrapper scripts which are considered part of the QA subsystem, but they can be output from Perl programs delivered by other subsystems. SVG files will contain keywords for basic identification and copyright information.
- **JPEG Images** Some FITS images are translated to JPEG for display in webpages and PDF files. A subset of the FITS keywords will be included in the JPEGs as EXIF data. These keywords will provide basic identification and positional information, as well as an appropriate copyright. JPEGs are not intended for public release.

Frame Index

The Frame Index is database storing meta-data on the characteristics, location and processing status of all framesets received at the WSDC. At this time the database is a single binary file that is accessed using sqlite. The QA subsystem both reads and updates the Frame Index; these operations are done through a custom Perl API which abstracts the database using the Perl module DBIx::Class. Database reads are done throughout the QA subsystem, including by the parts of the web-application accessible to outside users. Updates are constrained to a few places in the system; in the QA subsystem updates occur only when a QA scientist reviews a delivery or scan.

Testing

Unit Testing

The web application and Perl Frameworks delivered as part of the QA System will provide a minimal level of basic functional testing to demonstrate that code compiles and runs, libraries can be imported, and major subroutines respond correctly to inputs and respond with correct output formats. These tests are intended to demonstrate that delivered code works syntactically.

Additionally, when a test case is provided to isolate a bug in a framework, that test case will be added to the unit test suite. These tests are intended to verify that released code contains all patches.

Integration Testing

At a higher level, the Perl wrappers which execute several applications in series will be delivered with test commands and data. These tests asses that the functional interfaces between the programs and may or may not address the issues of data correctness. Some wrappers may have products which are more easily used for qualitative assessment than others.

Regression Testing

QA reports from the web application will be used in the verification process for major deliveries of the WISE Science

Data System. The reports will be tested against reports generated from the same dataset in previous releases and changes will be matched to tickets in the issue tracking system.