GNAOI Data Reduction Software

Software Non-Functional Requirements

Kathleen Labrie

Science User Support Department

V1.1 – 16 September 2019

Revision History

V1.0 – 29 April 2019 Kathleen Labrie
V1.1 – 16 September 2019 Kathleen Labrie

Document ID: GNAOI-SRS-101_DRSoftwareNonFunctionalReqs

Table of Contents

1. Introduction

1.1 Purpose

This document serves as a guide to designers, developers, and testers who are responsible for the engineering of the GNAOI Data Reduction Software project. It contains the non-functional requirements for the design, the development, and the testing the software.

1.2 Scope

This document is limited to the nonfunctional requirements. A companion document presents the top-level requirements which are mostly functional requirements.

1.3 System Overview

Gemini North Adaptive Optics Imager (GNAOI) is a planned imager that will be used with both the planned Gemini North Multi-Conjugate Adaptive Optics system (GNAO), and a planned Ground Layer Adaptive Optics system (GLAO). GNAOI will use a single HAWAII-4RG detector.

GNAO will provide an f/32 beam to the instrument. GLAO will provide and f/16 beam. A single set of camera optics in GNAOI will give a field of view of 85 arcseconds square with GNAO (which will correct a 2-arcminute diameter circular field) and 170 arcseconds square with GLAO (or indeed in natural seeing).

The imager will be provided with a suite of broad and narrow band filters that will support a broad range of science applications. The core wavelength regime is 0.9 - 2.5um, though a strong consideration is to expand this to 0.6 - 5um.

Gemini already has data reduction primitives and recipes for near-infrared imaging. Those will be available to GNAOI. The GNAOI team is requested to reuse as many existing tools as possible to avoid duplication and to avoid unnecessarily increasing the size of the code base, and as the result the maintenance burden. Improvements to existing routines will be welcomed.

The GNAOI Data Reduction Software will:

- Generate automatically or semi-automatically scientific quality calibrated products;
- Generate automatically “quicklook” / ”fast reduction” products for target-of-opportunity follow-up assessment;
- Generate automatically data quality assessment products.

The scope of the project does not include scientific analysis tools.

The GNAOI data reduction software will use Gemini’s DRAGONS pipeline infrastructure. It will use Astrodata and be built to work with DRAGONS’ Recipe System.

1.4 References

- Internet Engineering Task Force RFC 2119, https://www.ietf.org/rfc/rfc2119.txt
- GNAOI-SRS-102_DRSoftwareTopLevelRequirements.docx
- DPSG-STD-102_CodingStandards.docx
- DPSG-STD-104_VarianceDQPixelUnits.docx
- DRAGONS repository: https://github.com/GeminiDRSoftware/DRAGONS

2. Definitions

Glossary, definitions of inputs, and any other organizational or workflow-related information that is needed to understand the software requirement specification.

2.1 Language

Adapted from *Internet Engineering Task Force RFC 2119*.

- MUST: This word, or the term “REQUIRED”, mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase means that the definition is an absolute prohibition of the specification.

- SHOULD: This word, or the adjective “RECOMMENDED”, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED” mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word means that an item is truly optional.

2.2 Acronyms

- DR: Data Reduction
- GOA: Gemini Observatory Archive
- ICD: Interface Control Document
- QA : Quality Assessment
- QL : Quick-Look
- SQ : Science Quality
- SUSD: Science User Support Department
- SUSD-DR: Science User Support Department Data Reduction Staff

3. Nonfunctional Requirements

The nonfunctional requirements listed here should apply more or less system-wide and generally are those requirements that cannot obviously be associated with a use case.

When a requirement refers to an “approval”, the process is essentially to consult with the SUSD data reduction team to ensure that the request is justified and necessary. The team will evaluate the request and discuss possible other solutions. It should be possible to approve or deny the request within a day or two.

3.1 Nonfunctional Requirements from DRAGONS

The software written for this project must follow the nonfunctional requirements already established for DRAGONS.

Name	**SRS-NFR-001 – Python**
Summary	The code base must be written in Python
Rationale	Gemini has adopted Python for data reduction software. The staff who will maintain the code have expertise in Python. The *DRAGONS* infrastructure is written in Python.
Requirements	The code base must be written in Python and use Python modules. (A handful of exception might be considered, see NFR in References box.)
References	SRS-NFR-003, SRS-NFR-010, SRS-NFR-011, SRS-NFR-012, SRS-NFR-018

Name	**SRS-NFR-002 – Python version**
Summary	The software must run on Python 3.6 and above
Rationale	Python 2.7 is being phased out.
Requirements	The software must run on Python 3.6 or above as distributed by Anaconda. The software may also be compatible with Python 2.7.x, but this is optional.
References	SRS-NFR-003

Name	SRS-NFR-003 – AstroConda and external dependencies
Summary	Non-Gemini dependencies should be limited to software with 3-clause BSD license and to software included in AstroConda, the Anaconda astronomy distribution.
Rationale	The software will be distributed to third-parties as Open-Source software. Therefore, licensing conflicts are to be avoided. Also, it is important to ensure easy installation of the software and of all its dependencies. AstroConda takes care of that.
Requirements	All the dependencies should be contained within the AstroConda channel, DRAGONS, and the dependencies of those software. If other modules are believed to be required, approval from SUSD-DR must be obtained. DRAGONS is distributed under the 3-clause BSD license. SUSD-DR reserves the right to restrict the use of a dependency if it is found to be incompatible with our license or other conflicts arise. Finally, the software must not use or depend on proprietary software like IDL, matlab, etc.
References	SRS-NFR-010

Name	SRS-NFR-004 – AstroData
Summary	All MEF dataset access must be done with DRAGONS' *Astrodata*
Rationale	The DRAGONS' *recipe_system* package requires the data to be passed around as *AstroData* objects. *Astrodata* allows more instrument agnostic code to be written, encouraging code re-use.
Requirements	All GNAOI datasets must be accessed with the *AstroData* class. This includes pixel access, binary table access, and header access. The header access in particular must be done through a *descriptor* when the info is reflected in one of the standard, already defined *descriptors*. The AstroData class is built around astropy's NDData. Within that structure, the pixels are stored as NumPy ndarray and should be manipulated with numpy. The binary tables are represented with astropy.table Table class. Consult the Astrodata documentation for more information.
References	SRS-NFR-005

Name	SRS-NFR-005 – recipe_system
Summary	The software must use the RecipeSystem to process the data
Rationale	The RecipeSystem is the Python infrastructure that automates the processing of the data. This is Gemini's data reduction platform. The package is named recipe_system.
Requirements	The data reduction must be done with the RecipeSystem. The algorithms must be wrapped in primitives and the reduction controlled by recipes.
References	

Name	SRS-NFR-006 – gnaoi_instrument
Summary	The Astrodata instrument definition software must be developed in a package named gnaoi_instrument.
Rationale	For Astrodata to recognize the GNAOI data, a configuration layer needs to be written. This includes the tags and the descriptors, and any necessary lookup tables. No data reduction software belongs in this package. During development of new instruments by third-party teams the work will be done in a xxx_instrument package that will follow the same structure as the main gemini_instruments package for easy integration later on.
Requirements	The GNAOI Astrodata instrument definition software must be developed in an Astrodata support package named gnaoi_instrument located in the same directory as gnaoidr (SRS-NFR-007) in the GNAOIDR git repository hosted in the GeminiDRSoftware github organization (SRS-NFR-015).
References	SRS-NFR-007, SRS-NFR-015

Name	SRS-NFR-007 – gnaoidr
Summary	The data reduction algorithms, the primitives and recipes must be developed in a package named gnaoidr
Rationale	No tags or descriptor definition should be found in this package. This package is for reduction algorithms. This package can import modules with core algorithms. During development of new instruments by third-party teams the work will be done in a xxxdr package that will follow the same structure as the main gemini_instruments package for easy integration later on.
Requirements	The GNAOI data reduction software must be developed in a recipe_system support package named gnaoidr located in the same directory as gnaoi_instrument (SRS-NFR-006) in the GNAOIDR github repository (SRS-NFR-015).
References	SRS-NFR-006, SRS-NFS-015

Name	SRS-NFR-008 – NumPy
Summary	The use of NumPy is recommended for computational algorithms
Rationale	NumPy is a mature package that offers a lot of optimized computational functions and methods. Also, the pixel arrays in an AstroData object are numpy.ndarrays.
Requirements	The use of NumPy is recommended and strongly encouraged for computational algorithms.
References	http://www.numpy.org

Name	SRS-NFR-009 – Astropy
Summary	The use of Astropy is recommended, whenever appropriate
Rationale	The astropy package is well supported and growing fast. It already contains a lot of very useful functionalities developed for astronomy by astronomers. There is no point re-inventing the wheel if a functionality is already in Astropy.
Requirements	The use of Astropy is recommended and strongly encouraged.
References	http://www.astropy.org

Name	SRS-NFR-010 – IRAF

Summary	The software must not depend on IRAF
Rationale	IRAF is reaching its end-of-life. Long-term support by NOAO has ended. Also, Gemini is no longer hiring IRAF programmers, hence long-term maintenance of IRAF dependencies would become difficult.
Requirements	There must be no calls to IRAF tasks or import of PyRAF.
References	

Name	**SRS-NFR-011 – C-extensions**
Summary	C-extensions may be allowed, if required for performance
Rationale	It might happen that a computationally intensive step is unacceptably slow in Python. If this happen, writing that part of the software in C should help improve performance.
Requirements	If required for performance, C-extensions may be considered but require justification followed by approval from SUSD-DR. C-extensions are to be considered only in extreme cases where the performance benefits are truly significant.
References	SRS-NFR-001, SRS-NFR-012

Name	**SRS-NFR-012 – Cython**
Summary	C-extensions must be implemented with Cython
Rationale	Cython is favored for its simplicity of usage, and simplicity of installation since it comes with AstroConda.
Requirements	C-extensions, if approved by SUSD-DR, must be implemented with Cython.
References	SRS-NFR-011

Name	**SRS-NFR-013 – Installation**
Summary	The installation of the software must be possible using the “python setup.py <command>” utility, and it must be possible to package and distribute the software through conda.
Rationale	The installation of the software must be easy, straightforward, and not prone to mistakes. For release, the software will be wrapped into the DRAGONS conda package. But during development and for internal deployment, the standard setuptools utility will be used. Gemini will be responsible for the building of conda packages.
Requirements	The GNAOIDR package requires its own setup.py. The standard installation of the package must not require the user to type anything more than “python setup.py install” (or “pip install” if bdist_wheel is used to package).
References	

Name	**SRS-NFR-014 – Open-source**
Summary	The code base must be freely available and distributable under the AURA license used by DRAGONS.
Rationale	The Gemini data reduction code has always been and must always be distributed freely. The Gemini data reduction code base uses an AURA license. Any new software will be distributed under that open-source license.
Requirements	The new software must be open-source and distributed under the AURA license.
References	

Name	**SRS-NFR-015 – Revision control**

Summary	The code must be under revision control and stored in a Gemini Observatory Data Reduction Software github repository named GNAOIDR.
Rationale	Gemini data reduction is kept in the github collection: https://github.com/GeminiDRSoftware. A GNAOIDR repository will be created in that collection. GNAOI DR team and SUSD-DR members are given full access to the repository. Configuration management and revision control is required for Gemini DR software.
Requirements	The software must be under revision control and stored in the GNAOIDR repository located in the Gemini DR Software github collection, https://github.com/GeminiDRSoftware. The GNAOI DR team and SUSD-DR members are given full access to the GNAOIDR repository.
References	

Name	**SRS-NFR-016 – Unit tests**
Summary	Each function should be unit tested. The tests must pass.
Rationale	Unit tests are useful during development and during maintenance to ensure robustness and stability.
Requirements	Each function and method should be unit tested. The tests must pass for the software to be accepted. The tests should pass on committed code.
References	SRS-NFR-017 Test coverage wisdom: http://www.artima.com/forums/flat.jsp?forum=106&thread=204677

Name	**SRS-NFR-017 – pytest**
Summary	Unit tests must use pytest.
Rationale	pytest is the test utility adopted by Astropy and now fully adopted by Gemini.
Requirements	Unit tests must use pytest.
References	SRS-NFR-016

Name	**SRS-NFR-018 – Coding standards**
Summary	The code must adhere to the Gemini data reduction software coding standards
Rationale	The maintainability of software is greatly simplified when the code and the style is uniform across the code base. Adhering to a coding standard also help minimize mistakes. SUSD has already established coding standards for its Python software; it is based mostly on PEP8 with a few adaptions.
Requirements	The code must adhere to the Gemini data reduction software coding standards. When not specified in the Gemini document, PEP8 must be followed.
References	DPSG-STD-102_CodingStandards.docx, https://www.python.org/dev/peps/pep-0008/, SRS-NFR-019

Name	SRS-NFR-019 – pylint
Summary	A pylint score greater than 7 out of 10 and the absence of specific pylint error or warnings must be achieved
Rationale	pylint is a good tool to catch violation of coding standards and to create better, cleaner code. The Python community recommends a score of at least 7 to deem the code "good".
Requirements	A pylint score > 7 out 10 should be achieved. In particular, the code must be free of variable errors/warnings (eg. unused variables). Other such rules might be added, if necessary, as we learn about common issues. The pylintrc in DRAGONS/gempy/support_files must be used to define the scoring rules specific to the DRAGONS software (a better match to our coding standards than the default definitions).
References	SRS-NFR-018

Name	SRS-NFR-020 – Primitives ICD
Summary	The primitives ICD must be respected (primitive name and output suffix)
Rationale	A primitive of a given name is expected to apply a specific transformation regardless of instrument. For example, flatCorrect will apply the flat field correction, nothing less, nothing more, and any primitives applying the flat field correction must be named "flatCorrect". This helps users understand what a recipe is doing and what is really happening to the data. Similarly, a primitive of a given name writing to disk must be appending a specific suffix to the file name to help identify the products on disk. For example, flatCorrect must append the suffix _flatCorrected.
Requirements	The primitive ICD defines the processes associated with a given primitive name and its associated output suffix. Eg. The primitive flatCorrect must always applies and only applies the flat correction, and must append to the output's filename the suffix _flatCorrected, if writing to disk.
References	Primitives ICD (not yet available, though the primitives exist. Look at code for primitives names and the parameter files for suffix setting in the meantime.)

Name	SRS-NFR-021 – Descriptors ICD
Summary	The Descriptor ICD must be respected.
Rationale	The Descriptors are critical to the functioning of Astrodata and the RecipeSystem, and essential to instrument agnostic code.
Requirements	A specific set of descriptors (see reference below) must be implemented for the GNAOI AstroData types. The meaning of the descriptors must be respected.
References	AstroData User's Manual Appendix A – List of descriptors (http://astrodata-user-manual.readthedocs.io/en/latest/)

Name	SRS-NFR-022 – Documentation
Summary	Both user and programmer documentation must be delivered
Rationale	Users must know how to use the software and what it does. The programmers must know how to maintain the software, how it works, and how to add to it.
Requirements	Both a User Manual and a Programmer Manual must be delivered with the software.
References	SRS-NFR-023.

Name	**SRS-NFR-023 – Sphinx**
Summary	Documentation must be written for Sphinx
Rationale	Sphinx is widely used in the Python community and has become the go-to tool for Python software documentation. One source can produce HTML pages and PDF (via latex).
Requirements	The user manual and the programmer manual must be written for Sphinx using the ReST markdown language.
References	http://sphinx-doc.org

Name	**SRS-NFR-024 – Docstrings**
Summary	The docstrings must follow the NumPy format and standards
Rationale	NumPy is a reference in the Python community. The Astropy project has also adopted the NumPy format and standards for its in-code documentation.
Requirements	The docstrings must follow the NumPy format and standards. Part of this standard is that each function or method, each class, each module should have a properly written docstring.
References	https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt

Name	**SRS-NFR-025 – Scientific flow charts**
Summary	The flow of the data, in scientific terms, must be documented
Rationale	It is important to identify how the data needs to be processed, scientifically, before we can identify the necessary Recipes and Primitives to be used by the RecipeSystem. Also, it is important to document the data reduction in a format understandable by the users to avoid the “black box” effect.
Requirements	The flow of the data, in scientific terms, must be documented. This includes the science observation and all calibrations, for all the modes and type of data offered by the instrument. Diagrams and flow charts are strongly recommended.
References	SRS-NFR-026

Name	**SRS-NFR-026 – *recipe_system* flow charts**
Summary	The flow of the data in the RecipeSystem must be documented
Rationale	A RecipeSystem flow chart shows how the scientific flow charts are translated into a flow that is compatible with the RecipeSystem. This helps draw the list of primitives required and how the recipes need to stream the data to produce the desired products in an automated way, even if the data is coming in one dataset at a time (which is the case at night). The RecipeSystem flow charts are expected to differ somewhat from the Scientific flow charts.
Requirements	A RecipeSystem flow chart must use the name of the primitives and recipes it uses, and must identify the streams being used. RecipeSystem flow charts must be created for the processing of the science observations and for the processing of the calibrations, for all the modes and type of data offered by the instrument.
References	SRS-NFR-025

Name	SRS-NFR-027 – QA mode performance
Summary	The QA processing must be fast enough to be useful for nighttime operations
Rationale	The purpose of the QA is to help the observer assess the data and assess the sky condition in order to make the best, informed decisions at night. Therefore, near-real time feedback from the pipeline is necessary.
Requirements	The QA processing must be fast enough to be useful for nighttime operations. Specifically, this means being able to keep up with a typical science observing sequence on a quad core 2.8GH machine with 16 GB of RAM, locally, with a standard 7200 rpm disk.
References	The raw data will be on a remote NFS disk. The first step, the prepare primitive, copies the file locally. Delays associated to the initial NFS transfer are not GNAOI team's responsibility.

Name	SRS-NFR-028 – Hardware x86
Summary	The software must run x86-style hardware
Rationale	Gemini Operations machine are x86-style hardware. The Gemini community uses, in a large majority, x86-style hardware (Linux PCs and Macs).
Requirements	The software must run on standard 64 bit x86-style hardware and must not require GPUs, or any other special hardware.
References	

Name	SRS-NFR-029 – Hardware configuration
Summary	The software must require only one machine
Rationale	Reduce unnecessary complexity. Also, the typical user does not have access to clusters and is normally equipped with one desktop and/or a laptop.
Requirements	The software must require one machine: no clusters or client-server systems, for example.
References	

Name	SRS-NFR-030 – Multiple instances
Summary	It must be possible to run multiple instances without conflict
Rationale	Independent datasets do not need to be processed sequentially. Therefore, the system must allow the users to launch multiple reduction without worry that they will interfere with each other.
Requirements	It must be possible to run multiple instances on the same machine with the same user ID without those instances interfering with each other. Typically, the independent data sets are stored in different directories.
References	

Name	SRS-NFR-031 – Platform
Summary	The software must support Linux (CentOS 7) and Mac OS X (10.12+)
Rationale	The large majority of the Gemini community and its staff are Linux and/or Mac OS X users. Gemini Operations is Linux-based, with the current officially support OS being CentOS 7. (This is a moving target that needs updating once in a while as new OS's are released.)
Requirements	The software must compile, run, and generally be fully compatible with the Linux and Mac OS X platform. For Linux, the base OS is CentOS 7. For Mac OS X, circa April 2019, the base OS version is set to 10.12, Sierra. It likely to change by the time the software is delivered, however. Yet, it is expected that those two platforms will serve as common lowest denominator for several years.
References	

Name	SRS-NFR-032 – MEF
Summary	The inputs, outputs, and intermediate pixel or table datasets must be MEF files
Rationale	All of Gemini's facility instrument data produce MEF files. Gemini's data reduction software therefore is developed to expect and produce MEF files.
Requirements	All the inputs, outputs, and intermediate pixel or table datasets written to disk must be formatted as MEF files. Any outputs, final or intermediates, must have named and versioned extensions (see SRS-NFR-033, 034, 035).
References	SRS-NFR-033, SRS-NFR-034, SRS-NFR-035

Name	SRS-NFR-033 – VARDQ
Summary	The software must calculate and propagate the variance and data quality planes.
Rationale	Variance: a scientific result is meaningless without an error estimate. Data quality: Bad pixel values should not be used in calculations. Astrodata will take care of most variance and data quality plane calculation and propagation through the NDData class.
Requirements	The software must calculate and propagate, at every step, the variance plane and the data quality plane as described in the Gemini Data Reduction "Guide to Variance and Data Quality extensions and to Pixel Data Units". Astrodata will take care of most variance and data quality plane calculation and propagation through the NDData class.
References	DPSG-STD-104_VarianceDQPixelUnits.docx, SRS-NFR-034.

Name	SRS-NFR-034 – Extension names
Summary	The FITS extension names must follow the coding standards, eg. SCI, VAR, DQ
Rationale	Astrodata expects certain extensions to be named a specific way for loading and writing MEF files. Gemini's FITS standard is to use SCI, VAR, DQ.
Requirements	The extension naming must follow the coding standards. Science pixel extensions are SCI, variance planes are VAR, data quality planes are DQ, detected object catalog are OBJCAT, etc. New names for new types must be approved by SUSD-DR.
References	SRS-NFR-018, DPSG-STD-102_CodingStandards.docs (GP-Py-Name-32)

Name	SRS-NFR-035 – FITS standard
Summary	The software and the data must be compliant with the latest FITS standard specifications.
Rationale	Interoperability with other software.
Requirements	Inputs are assumed to be compliant, if not, the software must fix them. Outputs (final or intermediate) must be compliant.
References	http://fits.gsfc.nasa.gov/fits_standard.html

Name	SRS-NFR-036 – WCS
Summary	The software must provide FITS standard World Coordinate Systems even when storing and using more complex format.
Rationale	Interoperability with other software. This does not preclude the use of more complex WCS representations but some WCS keywords must be found in the FITS headers to provide an approximate representation that third-party software can recognize.
Requirements	Inputs are assumed to be compliant, if not, the software must fix them. Outputs (final or intermediate) must be compliant.
References	http://fits.gsfc.nasa.gov/fits_wcs.html

Name	SRS-NFR-037 – Metadata Housekeeping
Summary	The software must maintain the Gemini housekeeping metadata
Rationale	Gemini housekeeping metadata is used for archival purposes.
Requirements	Housekeeping metadata from the raw frames, like program and observation ID, must be propagated at each step and be present in any outputs. The minimum list is: • DATALAB • DATE-OBS • GEMPRGID • INSTRUME • OBSCLASS • OBSID • OBSTYPE • (TIME-OBS) [optional if DATE-OBS already contains the time]
References	

Name	SRS-NFR-038 – Metadata History
Summary	Outputs must contain metadata listing the processing history and the version numbers of the software used.
Rationale	Scientific reproducibility
Requirements	All outputs must contain processing information in the form of metadata listing the processing history and the version numbers of the software used. Each primitive must timestamp the PHU of the output file.
References	The function mark_history in DRAGONS' gempy/gemini/gemini_tools.py is the format currently in use.

Name	**SRS-NFR-039 – Metadata Provenance**
Summary	Outputs should contain provenance metadata
Rationale	Scientific reproducibility
Requirements	All outputs should contain provenance metadata, including but not limited to the unique identification of all input data. (This is an objective to be met at some point. As of April 2019, Gemini has no mechanisms to do that in a complete and absolute way.)
References	Gemini has currently no specifications for this and is opened to proposals.

Name	**SRS-NFR-040 – Display**
Summary	Image displays must work with DS9
Rationale	DS9 is the image display used by a large fraction of the astronomy community.
Requirements	Image displays must work with DS9. Other displays can be supported, but DS9 is a requirement.
References	

Name	**SRS-NFR-041 – Graphics**
Summary	Graphics must use X11 and allow for remote display
Rationale	Compatibility with current Gemini Operations procedures and hardware. Compatibility with most users' display tools.
Requirements	Any display of graphics must work correctly over X11 to a remote display. VNC do not count, it must be simple X.
References	

Name	**SRS-NFR-042 – Graphics to disk**
Summary	There must be an option to output to files on disk rather than display on screen
Rationale	When running the reduction in the background, it can be useful to keep a record of graphics, for example for use in the archive.
Requirements	There must be an option to generate standard graphics files, eg. png, and send them to disk rather than display to screen. Eg. of filename format: datasetname_primitive_timestamp.png
References	

Name	**SRS-NFR-043 – Quality Assessment (QA) mode definition**
Summary	QA must be automatic and must not require calibrations to complete
Rationale	The purpose of the QA mode is to assist the nighttime observer in assessing 1) the sky conditions (seeing, sky brightness, cloud coverage) as measured from the data, 2) the quality of the data (eg. making sure it "looks" okay, that nothing is saturated, etc.). This information is used by the observer in the decision making required to navigate the queue plan. The observer does **not** have time to operate the pipeline. Scientific accuracy is not needed.
Requirements	The Quality Assessment (QA) mode must run automatically, without requiring observer interaction. Calibrations are sought but if none is found, the software must carry on with the processing without applying the correction. Scientific accuracy is not required.
References	

Name	**SRS-NFR-044 – Fast Science / Quick-Look (QL) mode definition**
Summary	QL must be automatic and requires calibrations to complete
Rationale	The purpose of the QL mode is to assist the astronomer in assessing 1) the quality of the data, 2) the scientific content of the data, 3) the scientific potential of the data, with the minimal amount of effort and quickly. The processing does not need to be optimized but should be close to scientifically valid in most cases. If expediency is required, canned calibrations can be used. Use case: LSST follow-up.
Requirements	The Quick-Look (QL) mode must run automatically, without requiring user interactions. Specifying the data and launching the reduction can be done by a user or an automatic dispatcher listening to the OCS. Processed calibrations are required. The absence of calibration will cause the software to abort and exit. Scientific accuracy could be achievable for some cases (if best calibrations are available.)
References	

Name	**SRS-NFR-045 – Science Quality (SQ) mode definition**
Summary	SQ should be automatic, require calibrations to complete, and the products must scientifically correct
Rationale	The purpose of the SQ mode is to generate scientifically accurate and optimized products. Automation is highly preferred but the ability to optimize and customize the reduction is also important. The processing must lead to scientifically valid products.
Requirements	The Science Quality (SQ) mode should run automatically, without requiring user interactions other than 1) launching the reduction, 2) specifying which data to run the software on, 3) optimizing critical steps too difficult to optimize automatically, or too critically dependent on the scientific objectives of the program. Interactivity must be kept to a minimum. Processed calibrations are required. The absence of calibration will cause the software to abort and exit. Scientific accuracy is unconditionally required.
References	

3.2 Nonfunctional Requirements Specific to GNAOI Data Reduction

At this time, there are no nonfunctional requirements defined specifically for the GNAOI Data Reduction Project.

4. Detailed Revision History

v1.0	29 April, 2019	Kathleen Labrie	
		Initial revision.	
v1.1	16 September, 2019	Kathleen Labrie	
		Minor revision to SRS-NRF-003 to emphasis the 3-clause BSD license requirement.	