# QA Working Group

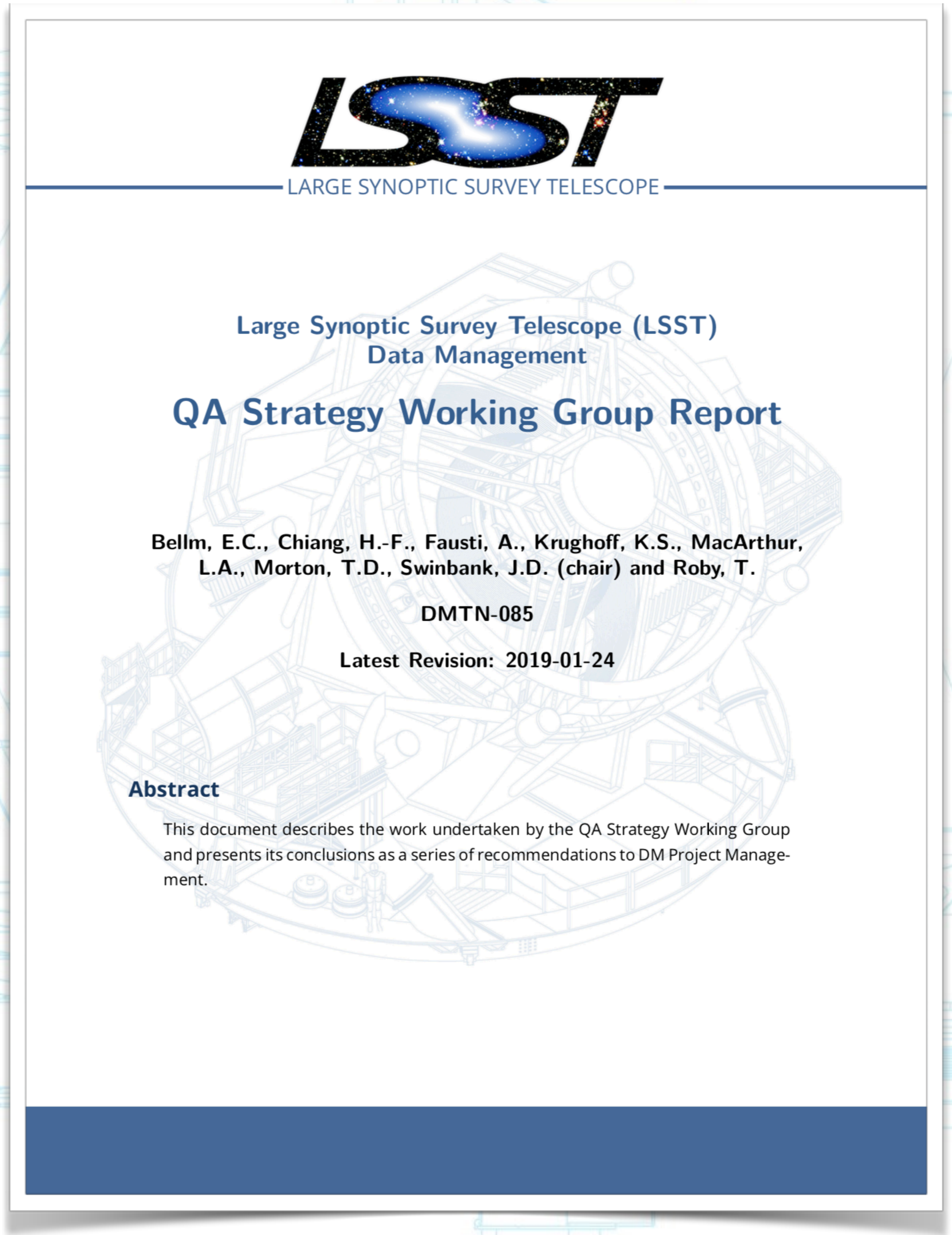
John Swinbank

Large Synoptic Survey Telescope

QA Working Group

## Overview and administration

- Charge: LDM-622
- Report: DMTN-085
  - 43 pages; 40 recommendations – sorry!
- Members:
  - Eric Bellm (AP)
  - Hsing-Fang Chiang (LDF)
  - Angelo Fausti (SQuaRE)
  - Simon Krughoff (SQuaRE)
  - Lauren MacArthur (DRP)
  - Tim Morton (DRP)
  - John Swinbank (AP)
  - Trey Roby (SUIT)



LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)
Data Management

QA Strategy Working Group Report

Bellm, E.C., Chiang, H.-F., Fausti, A., Krughoff, K.S., MacArthur, L.A., Morton, T.D., Swinbank, J.D. (chair) and Roby, T.

DMTN-085

Latest Revision: 2019-01-24

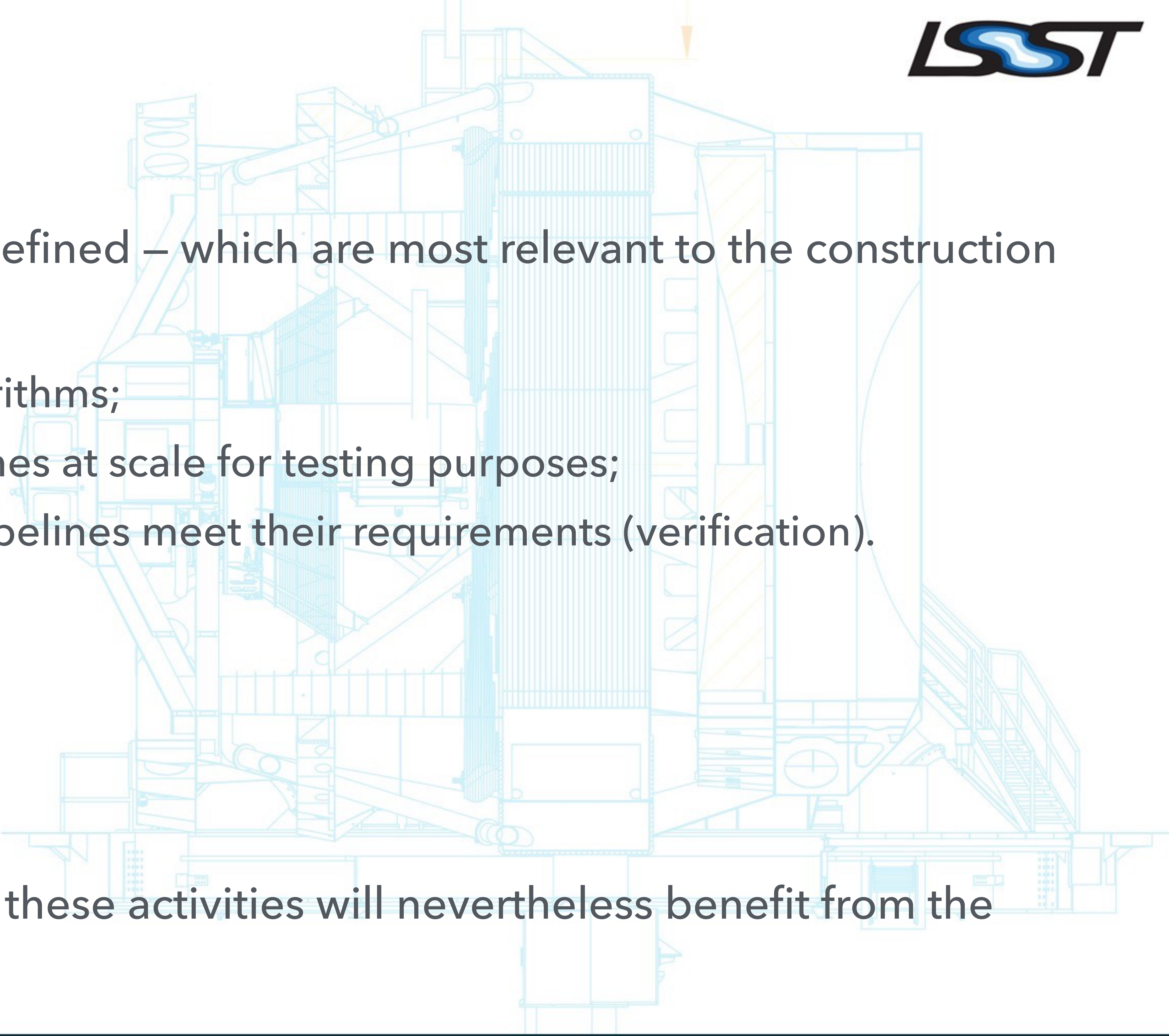
**Abstract**

This document describes the work undertaken by the QA Strategy Working Group and presents its conclusions as a series of recommendations to DM Project Management.

## QA ??

- All things to all people.
- LDM-522 (SDQA Conceptual Design) attempts to draw a distinction between (manual) QA activities and (automated) QC processes.
  - LDM-522 has never been baselined; it exists only as a nearly-two-years-old draft.
    - (This should scare you… it does me.)
  - Some members of the WG regarded LDM-522 as normative; others had never read it.
- We also learned – after the group had been established for some time! – that some of our number were referring to Quality *Assurance*, and some to Quality *Analysis*.
  - LDM-522 prefers the latter.

## Scope



- "Those aspects of QA – however defined – which are most relevant to the construction of the LSST Science Pipelines."
  - Tools needed for debugging algorithms;
  - Enabling developers to run pipelines at scale for testing purposes;
  - Demonstrating that the Science Pipelines meet their requirements (verification).
- Note that this **excludes**:
  - Commissioning;
  - Science Validation;
  - SDQA.
- It is our hope and expectation that these activities will nevertheless benefit from the outputs of this WG.

## Approach to the problem

- WG split into three sub-groups to explore the following areas:
  - Addressing the needs of developers writing and debugging algorithms on the small scale [Lauren, Simon, Trey];
  - Developing tooling to address the drill down use case [Angelo, Eric, Tim];
  - Providing the infrastructure needed to support automated testing and verification [Hsin-Fang, John].
- Subgroups conducted independent brainstorming and met regularly to compare notes.
- Aim to distill common "components" that were identified as having maximum impact across all groups.

## Pipeline debugging

- What tools do we need to help pipeline developers with their every-day work? For example:
  - How do you go about debugging a Task that is crashing?
  - Is `lsstDebug` adequate?
  - Do we need an `afwFigure`, for generating plots, to go alongside `afwDisplay`, for showing images?
  - What additional capabilities are needed for developers running and debugging at scale, e.g. log collection, identification of failed jobs, etc.
  - What's needed from an image viewer for pipeline developers? Is DS9 or Firefly adequate?
  - Is there value to the `afwDisplay` abstraction layer, or does it simply make it harder for us to use Firefly's advanced features?
  - How do we view images that don't fit in memory on a single node?
  - How do we handle fake sources? Is this a provenance issue?

## Drill down

- What types of metric should be extracted from running pipelines?
- How can those metrics be displayed on a dashboard? Is a simple time-series adequate, or do we need other types of plotting?
- By what mechanism can the user drill-down from those aggregated metrics to identify sources of problems? Do they click through pre-generated plots, or jump straight into a notebook environment?
- Assuming the user ends up in an interactive environment, what are its capabilities?
- What do the above tell us about the data products that pipelines need to persist (both in terms of metrics that are posted to SQuaSH, and regular pipeline outputs, Parquet tables, HDF5 files, etc)?

## Datasets and test infrastructure

- Are any changes needed to the way that DM currently handles unit testing? How are datasets made available to developers? Git LFS repositories? GPFS?
- What is the appropriate cadence for running small/medium/large scale integration tests and reprocessing of fixed datasets?
- How is the system for tracking metrics managed? – how are the metric calculation jobs run? By whom? How often?
- How should run-time performance of the science algorithms be tracked?

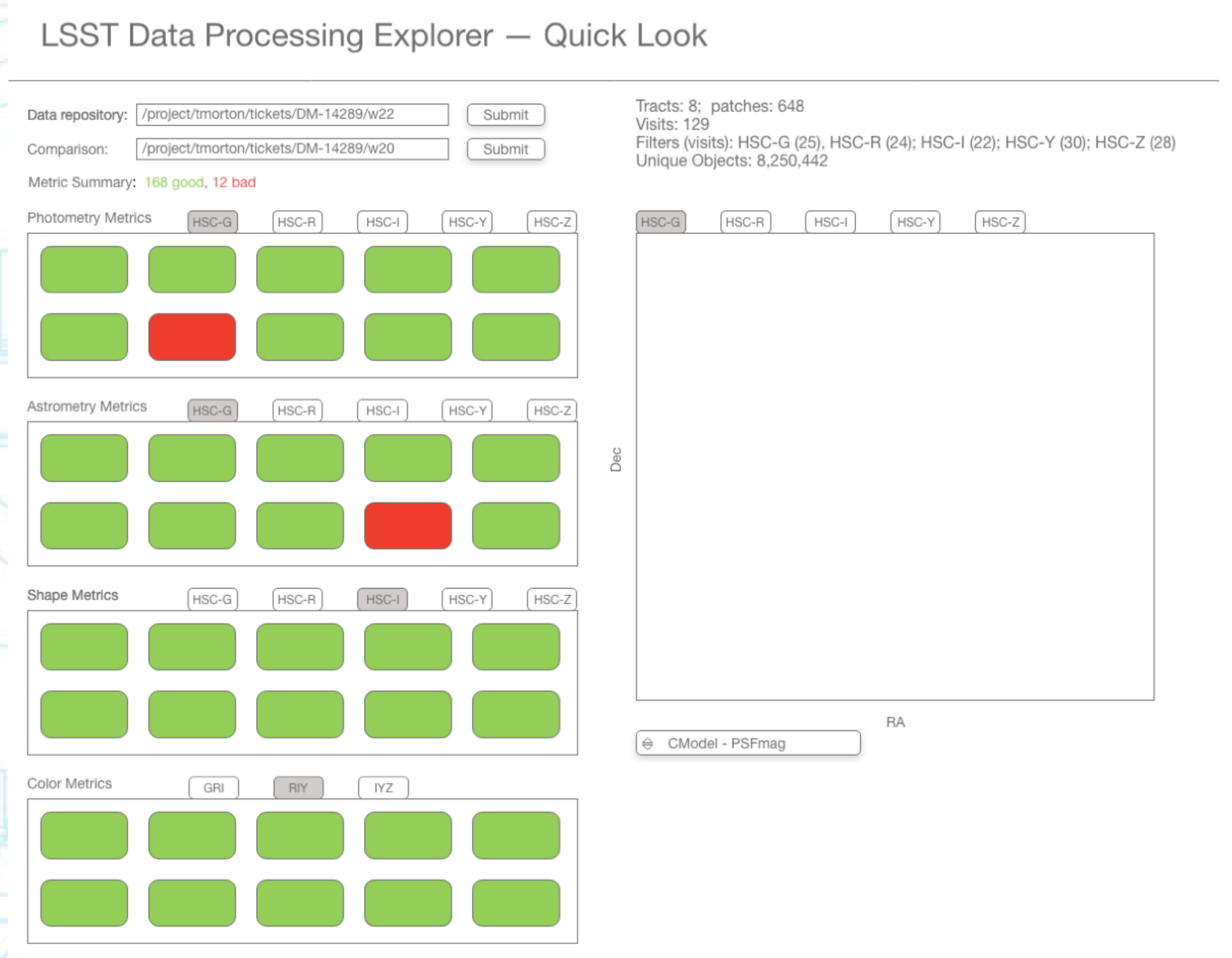
## Conclusions: Pipeline debugging

- There is no over-arching tool or concept which will address all concerns.
- However, a few pain-points were identified which we suggest could be attacked at relatively low cost. These include:
  - A revised version of `lsstDebug`;
  - A revised set of tooling for generating, aggregating and analyzing logs;
  - Revised documentation on interacting with workload management;
  - Guidelines for the maintenance of data repositories.
  - A development roadmap for, and guidance on the use of, visualization tools.
  - Prioritize developer-accessible systems for tracking and understanding provenance.

QA Working Group

## Conclusions: Drill down

- Here we suggest a browser-based tool which provide drill down capabilities from high-level summary metrics to input data.
- This is complementary to the existing SQuaSH service (and enhancements thereof): SQuaSH can track metrics over long time periods, but does not have access to output data repositories to enable drill down.
- Adopting a common model to describe metrics will enable users to switch between these tools effectively & efficiently.



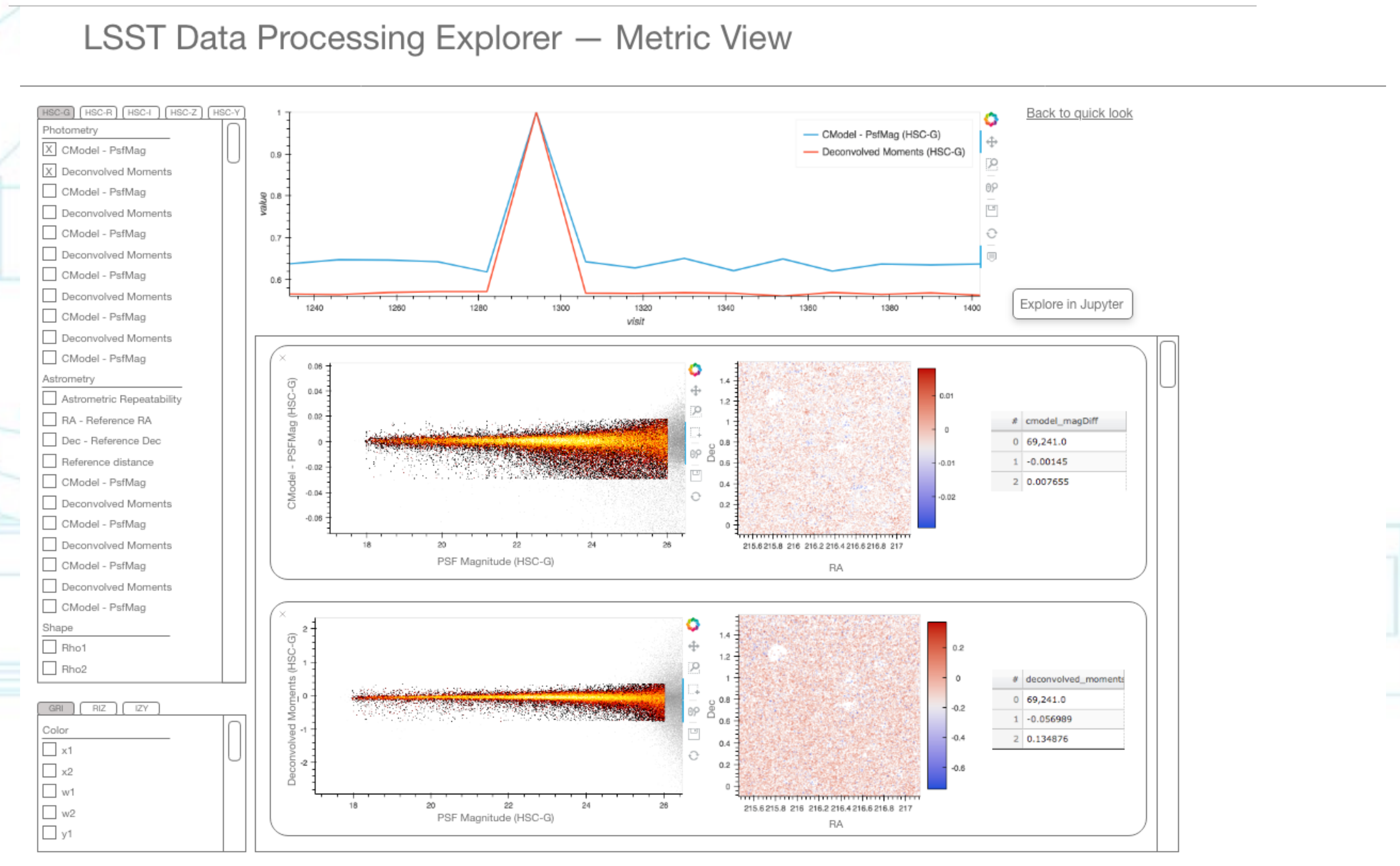
LSST Data Processing Explorer — Quick Look

Data repository: /project/tmorton/tickets/DM-14289/w22 Submit

Comparison: /project/tmorton/tickets/DM-14289/w20 Submit

Metric Summary: 168 good, 12 bad

Tracts: 8; patches: 648
Visits: 129
Filters (visits): HSC-G (25), HSC-R (24); HSC-I (22); HSC-Y (30); HSC-Z (28)
Unique Objects: 8,250,442



LSST Data Processing Explorer — Metric View

| # | cmodel_magDiff |
|---|---|
| 0 | 69,241.0 |
| 1 | -0.00145 |
| 2 | 0.007655 |

| # | deconvolved_moments |
|---|---|
| 0 | 69,241.0 |
| 1 | -0.056989 |
| 2 | 0.134876 |

## Conclusions: Datasets and test infrastructure

- Again, there is no one-size-fits-all approach adopted here.
- We suggest improvements to:
  - The Continuous Integration system;
  - Integration testing;
  - Management of test datasets;
  - SQuaSH.

QA Working Group

## Recommendations

- DMTN-085 makes 40 specific recommendations for consideration by the DMLT.
- Each of those recommendations is based upon the reasoning presented in the document – they are designed to be "stand alone", but fully understanding them will require reading (and not just the 11 slides that precede this one!)
- Those recommendations:
  - May already be rendered obsolete or put into practice by ongoing work (the Gen 3 middleware is the obvious example);
  - Are not zero-cost: the DMLT must decide whether they are worth the investment.
  - Are not without schedule impact: the DMLT must decide if they should be prioritized.
- Let's review them…

QA Working Group: Recommendations

## QAWG-REC-1

| | |
|---|---|
| Recommendation | Adopt the definitions of QA-related terms in the DMTN-085 glossary subsystem-wide. |
| Explanation | For example, by inclusion in a subsystem-level glossary; refer to DM-9807, DM-14877, and DM-14911. |
| JDS Commentary | An easy one to get started. |
| Responsible team | DM-SST? |
| Cost | Negligible? |
| Priority | Let's do it! |

# QAWG-REC-2

| | |
|---|---|
| Recommendation | Develop a new pipeline instrumentation and debugging system, replacing `lsstDebug`. |
| Explanation | The total effort expended here should be modest: we expect that design, implementation and documentation of a new system should take no more than one full time equivalent month. Converting existing code make take somewhat longer. |
| JDS Commentary | - |
| Responsible team | Pipelines? (Architecture could also do it, but unlikely…) |
| Cost | 1 FTE month (as above) for implementation; more to sweep up old test cases. |
| Priority | ? |

QA Working Group: Recommendations

## QAWG-REC-3

| | |
|---|---|
| Recommendation | Guidelines for the effective use of the pipeline debugging system should be supplied to developers. |
| Explanation | These guidelines should be supplied by the Science Pipelines Product Owners and the LSST Software Architect, and made available through the DM Developer Guide. |
| JDS commentary | This effectively means establishing a baseline for how developers should instrument their tasks in a way that keeps e.g. Robert happy. |
| Responsible team | Pipelines? (Architecture could also do it, but unlikely…) |
| Cost | 1 FTE month (as above) for implementation; more to sweep up old test cases. |
| Priority | ? |

## QAWG-REC-4

| | |
|---|---|
| Recommendation | Debugging mode should be binary: it is either enabled or disabled, with no further configuration. |
| Explanation | For example, within a Task, one might check if debugging mode is enabled with a statement like `if self.config.debug: ....` |
| JDS commentary | This is really a suggestion for a simplified API for debugging. It is (arguably?) a suggestion to the implementor of QAWG-REC-3. |
| Responsible team | Pipelines? (Architecture could also do it, but unlikely…) |
| Cost | Nothing additional to QAWG-REC-3. |
| Priority | ? |

## QA Working Group: Recommendations

## QAWG-REC-5

| | |
|---|---|
| Recommendation | A log aggregation and monitoring service should be provided for large-scale processing jobs at the Data Facility. |
| Explanation | Such a service should not be a requirement for jobs to execute (in particu- lar, when running in non-Data Facility environments, logs should continue to be generated and collected as at present). |
| JDS commentary | Commonly heard complaint from devs is that simply finding out when something isn't working in a big job is too hard. This includes e.g. a facility to highlight errors/exit status, track particular threads/processes/CCDs/etc, search with regexp, etc. |
| Responsible team | Combination of Architecture & LDF? |
| Cost | ? |
| Priority | ? |

QA Working Group: Recommendations

## QAWG-REC-6

| | |
|---|---|
| Recommendation | Tutorial and reference documentation for developers attempting to run jobs at scale should be refreshed. |
| Explanation | In particular, revised documentation should focus on identifying and resolving common failure modes, and understanding how best to use existing resources, such as the dashboard at https://monitor-ncsa.lsst.org/, to rapidly diagnose and escalate issues with underlying infrastructure. |
| JDS commentary | Related to logging; commonly heard complaint that keeping track of jobs (and particularly failures or disappearance of jobs) at scale is too hard and a major drain on velocity. Perhaps some of this goes away with new middleware? |
| Responsible team | Combination of Architecture & LDF? |
| Cost | ? |
| Priority | Low? Clearly current systems work for some people, and new ones are coming. |

QA Working Group: Recommendations

## QAWG-REC-7

| | |
|---|---|
| Recommendation | DM should formally adopt the PyViz ecosystem |
| Explanation | This adoption would include, for example, including PyViz tools in a regular installation of the LSST Stack; providing training and documentation for developers and – crucially – developing interfaces which enable LSST conventions (afw tables, the Data Butler) to be used in the PyViz context. |
| JDS commentary | This provides a convenient "getting started" point for developers and baseline which can be assumed in documentation with minimal effort on our part. Presumably these tools are useful to Commissioning, etc. |
| Responsible team | Arch, SQuaRE, Pipelines? |
| Cost | Low? |
| Priority | Medium-high? |

QA Working Group: Recommendations

## QAWG-REC-8

| | |
|---|---|
| Recommendation | DM should adopt Dask to enable users to work with larger than memory data. |
| Explanation | This might be achieved by providing users with the ability to spin up Dask clusters on demand using (say) Kubernetes, or by providing a Dask cluster at the LSST Data Facility to which users can connect. If ongoing middleware development renders this obsolete, then it can be retired. |
| JDS commentary | I know Simon/SQuaRE were working on this, but I don't know the current status. Note the final sentence re middleware: this addresses a need we have now, and standing up this service would be valuable even if it is later retired. |
| Responsible team | SQuaRE, LDF |
| Cost | Medium? |
| Priority | ? |

## QA Working Group: Recommendations

# QAWG-REC-9

| | |
|---|---|
| Recommendation | DM should provide clear, written guidance to developers about the availability, status and expected usage of image display tools. |
| Explanation | - |
| JDS commentary | There was a move to push Firefly, but that had the rug pulled from under it by SUIT descopes. The WG failed to find a compelling image display story that satisfies all requirements. Even this vestigial recommendation is really to quiet the perennial “will we all have to use Firefly someday?” question. |
| Responsible team | Proj Mgmt? Pipelines? |
| Cost | Low. |
| Priority | ? |

QA Working Group: Recommendations

## QAWG-REC-10

| | |
|---|---|
| Recommendation | The design and implementation of the provenance system should have high priority in the project scheduling. |
| Explanation | - |
| JDS commentary | ***So*** many discussions came down to “this sounds like a provenance issue”. This is now addressed by middleware work. There may be no more to be done here? Is there any concern about making provenance information available to developers? |
| Responsible team | DAX (with Architecture, LDF, Pipelines input) |
| Cost | High (but included in G3 effort) |
| Priority | High |

QA Working Group: Recommendations

## QAWG-REC-11

| | |
|---|---|
| Recommendation | Obsolete and unclear sections of the Developer Guide should be rewritten to provide clearer guidance on unit tests. |
| Explanation | This should include at least:<br>• Guidance for unit vs. integration tests, and in which packages it is appropriate to write tests (e.g. is it adequate for certain functionality to be only tested through packages like ci_hsc?);<br>• Requirements for code coverage;<br>• Appropriate languages for writing tests (should C++ code be tested in C++, or is it acceptable–or encouraged–to test only the Python-wrapped version?);<br>• Are there certain types of code that it is appropriate not to test (e.g. boilerplate accessor methods)? How exhaustive should tests be? |
| JDS commentary | "Everybody knows" that we test in certain ways. Except they don't, and the Dev Guide has more cruft than useful advice. Guidance and rules from the Architecture team could help avoid squabbling in Pipelines…! |
| Responsible team | Architecture |
| Cost | Low |
| Priority | Medium |

## QA Working Group: Recommendations

# QAWG-REC-12

| | |
|---|---|
| Recommendation | The Developer Guide should be expanded to provide checklist-style documentation for code reviewers making clear what is expected from them during the review. |
| Explanation | - |
| JDS commentary | Another "everybody knows"; closely related to QAWG-REC-11. |
| Responsible team | Architecture |
| Cost | Low |
| Priority | Low-Medium |

QA Working Group: Recommendations

## QAWG-REC-13

| | |
|---|---|
| Recommendation | Provide a central location where examples, scripts and utilities which are not fundamental to pipeline execution are indexed and made discoverable. |
| Explanation | See also DM-15807. |
| JDS commentary | DM-15807 is "Documentation conventions for utilities and scripts", a SQuaRE ticket assigned to Jonathan. |
| Responsible team | SQuaRE |
| Cost | Low |
| Priority | Medium |

QA Working Group: Recommendations

## QAWG-REC-14

| | |
|---|---|
| Recommendation | The Project should adopt a documented (in the Developer Guide) policy on the maintenance of example code. |
| Explanation | - |
| JDS commentary | Broken examples are a longstanding hurdle for newcomers (and veterans) trying to work with the codebase. The WG would be bullish about simply deleting them if we can't CI or otherwise test them. |
| Responsible team | SQuaRE |
| Cost | Low |
| Priority | Medium |

QA Working Group: Recommendations

## QAWG-REC-15

| | |
|---|---|
| Recommendation | The Project should prioritize the development of a documentation system which makes it convenient to include code examples and that tests those examples as part of a documentation build. |
| Explanation | - |
| JDS commentary | I interpret this as fundamentally meaning either “CI for Jupyter notebooks” or (less likely) adopt something like Sphinx Doctests. |
| Responsible team | SQuaRE |
| Cost | Medium |
| Priority | High |

QA Working Group: Recommendations

## QAWG-REC-16

| | |
|---|---|
| Recommendation | When running regularly scheduled (timer) jobs on the master branch of any releasable product, any build failure should be announced prominently to key stakeholders. |
| Explanation | Those stakeholders should include:<br>• Senior DM management (DM Project Manager, DM Software Architect, Science Pipelines T/CAMs and Science Leads);<br>• The developer who caused the failure, if it is possible to identify them.<br>The term "prominent" should be taken to indicate a personalised message (e.g. e-mail, direct Slack message), not a general posting in a Slack channel which regularly sees traffic. |
| JDS commentary | The gist of this is that existing Slack notifications are inadequate, and relying on K-T to catch build failures doesn't scale. |
| Responsible team | SQuaRE |
| Cost | Low-medium? |
| Priority | Medium? |

QA Working Group: Recommendations

## QAWG-REC-17

| | |
|---|---|
| Recommendation | The Developer Guide should provide guidance about expected responses to Jenkins failures. |
| Explanation | For example, the policy for handling integration test (e.g. ci_hsc) failures following a merge must be documented: who is responsible for checking for failure? Should the merge be reverted? Who is responsible for doing so? |
| JDS commentary | - |
| Responsible team | SQuaRE / Architecture / Pipelines |
| Cost | Low |
| Priority | Low-medium? |

## QA Working Group: Recommendations

## QAWG-REC-18

| | |
|---|---|
| Recommendation | The versions of external packages used in the Jenkins system must always correspond to the minimum versions specified in stub packages and/or in the document list of prerequisites. |
| Explanation | - |
| JDS commentary | This has already been RFCed and implemented as stated. It's probably also (being) obsoleted by ongoing work in the Architecture team. |
| Responsible team | SQuaRE / Architecture |
| Cost | Low-medium |
| Priority | Medium, but it's ~done! |

QA Working Group: Recommendations

## QAWG-REC-19

| | |
|---|---|
| Recommendation | The project should adopt a single source of dependency information and versions. |
| Explanation | This might consist of "stub" EUPS packages, or of a Conda environment, or of a list of packages on a website, but there must be one unambiguously authoritative source of information. |
| JDS commentary | As QAWG-REC-18, I think we can regard this as taken care of. |
| Responsible team | SQuaRE / Architecture |
| Cost | Low-medium |
| Priority | Medium, but it's ~on track! |

## QA Working Group: Recommendations

## QAWG-REC-20

| | |
|---|---|
| Recommendation | A standardized format for dataset repositories should be adopted across DM. |
| Explanation | Obviously, not all repositories will have exactly the same contents: in some cases, it may be necessary for a repository to contain (say) calibration products, while in others they may be inappropriate. However, it should be possible to establish at a glance what the contents of each dataset is; if calibration products are included, it should be immediately obvious what they are and how to apply them. |
| JDS commentary | Currently, they all look different, some are on GitHub, some are on GPFS, some have documentation, some don't, some are maintained, some aren't... |
| Responsible team | Effectively any team could take this on. |
| Cost | Low to define a format; maybe medium to enforce it. |
| Priority | Medium? |

QA Working Group: Recommendations

## QAWG-REC-21

| | |
|---|---|
| Recommendation | Each dataset should have an explicitly named product owner. |
| Explanation | Product owners are responsible for ensuring that the content and use cases of the datasets are well described and compatible with recent stack versions. The owner of a dataset could often be a member of the team with immediate use cases and knowledge of the relevant camera package. |
| JDS commentary | This is not really "product owner" in the sense we use it for development work. This is somebody who is responsible when I try to look at a dataset and find it was persisted with a decades-old version of the stack and no longer does anything useful. |
| Responsible team | Anybody and everybody. Project Management to lead? or LDF? |
| Cost | Low to assign product owners; higher to make them do anything! |
| Priority | Medium? |

QA Working Group: Recommendations

## QAWG-REC-22

| | |
|---|---|
| Recommendation | Datasets may be stored on either shared filesystems or Git LFS as appropriate, depending on the total size of the dataset. |
| Explanation | - |
| JDS commentary | This is really just implementation guidance for QAWG-REC-20. |
| Responsible team | Anybody and everybody |
| Cost | Nothing. |
| Priority | - |

## QA Working Group: Recommendations

## QAWG-REC-23

| | |
|---|---|
| Recommendation | A standardized test package design should be developed which addresses all existing use cases. |
| Explanation | Existing test packages (lsst_dm_stack_demo, ci_hsc, validate_drp, ap_verify, etc) should be adapted to the new design, and, where possible, merged with each other. |
| JDS commentary | A companion to QAWG-REC-20. This is a nice-to-have for developers, but perhaps we should also think of it in terms of future verification and large-scale-test needs (see QAWG-REC-24). |
| Responsible team | Pipelines? SQuaRE? |
| Cost | Medium-high? |
| Priority | ? |

## QA Working Group: Recommendations

## QAWG-REC-24

| | |
|---|---|
| Recommendation | A coherent plan for integration testing at all scales should be developed and published. |
| Explanation | Such a plan should then drive the development of the test package standard discussed above. Note DM-15348 in this context. |
| JDS commentary | DM-15348 begat DMTN-091, but it's not really clear what the audience of that document is or if it's supposed to be normative. |
| Responsible team | Pipelines? SQuaRE? DM-SST likely very interested from the verification point of view. |
| Cost | Low-medium to develop a plan. Maybe high to actually implement it. |
| Priority | Medium-high?? |

## QAWG-REC-25

| | |
|---|---|
| Recommendation | Formalise the `lsst.verify.metrics` system as the source of truth for metric definitions, by e.g. describing it in LDM-503 and LDM-639. |
| Explanation | This should not be taken as a blanket endorsement of the current implementation of this system; §4.11.1 provides a number of recommendations as to the way that metrics are defined. |
| JDS commentary | It's implicit that work needs to be done here to develop and build consensus around this system: simply editing the document is trivial. See also QAWG-REC-26. |
| Responsible team | Pipelines & SQuaRE? |
| Cost | Medium? |
| Priority | High? |

QA Working Group: Recommendations

## QAWG-REC-26

| | |
|---|---|
| Recommendation | Provide a high-level overview and data-model describing the metric definition system. |
| Explanation | At present, the lack of material providing an overview of the system is a significant barrier to entry. Although excellent API documentation is available, it is accompanied by references to technical notes describing relatively vaguely-specified design goals and referring to concepts like “provenance” and “specification” which require further elucidation. A revision of the documentation which expunges mention of old, obsolete packages and provides a clear set of getting-started guidelines should be untertaken. We suggest that that the SQuaRE group engage with one or more stakeholders in the Pipelines while working on this material. |
| JDS commentary | A companion to QAWG-REC-25. |
| Responsible team | Pipelines & SQuaRE? |
| Cost | Medium? |
| Priority | High? |

## QAWG-REC-27

| | |
|---|---|
| Recommendation | The computation, selection, and aggregation steps that define a metric should be well compartmentalized. |
| Explanation | - |
| JDS commentary | For metrics defined as aggregates over sources, the idea here is that you calculate and store a value for all sources, then the selection of sources you are interested in and the calculation of the aggregate is done on the fly. This is arguably implementation guidance for QAWG-REC-025/026. |
| Responsible team | Pipelines & SQuaRE (needs implementation and incorporation into existing systems) |
| Cost | Medium? |
| Priority | Medium? |

QA Working Group: Recommendations

## QAWG-REC-28

| | |
|---|---|
| Recommendation | Metric values should be stored with full granularity (source, CCD, patch, dataset). |
| Explanation | - |
| JDS commentary | Broadly the same intention as QAWG-REC-27, but for metrics which cannot be calculated on a per source basis (e.g. they require fitting a model to many sources at once). |
| Responsible team | Pipelines & SQuaRE (needs implementation and incorporation into existing systems) |
| Cost | Medium? |
| Priority | Medium? |

QA Working Group: Recommendations

## QAWG-REC-29

| | |
|---|---|
| Recommendation | Metric values should be stored as “tidy data” in columnar data stores (e.g., Apache Parquet) as part an output data repository. |
| Explanation | In particular, this should make it possible to load the data quickly enough for interactive work on hundreds of tracts or an equivalent number of visits. |
| JDS commentary | This is a natural consequence of RFC-465 and then implementation work on other recommendations. Not clear that any specific action is required here. |
| Responsible team | - |
| Cost | - |
| Priority | - |

QA Working Group: Recommendations

## QAWG-REC-30

| | |
|---|---|
| Recommendation | Metric values should have Butler dataIds. |
| Explanation | - |
| JDS commentary | See also QAWG-REC-31.<br>Facilitates joining and filtering metric values by other metadata.<br>I believe this is consistent with ongoing Butler work; not clear if other effort is required here. |
| Responsible team | - |
| Cost | - |
| Priority | - |

## QAWG-REC-31

| | |
|---|---|
| Recommendation | It should be possible to use the Data Butler to persist and retrieve metric values. |
| Explanation | - |
| JDS commentary | See also QAWG-REC-30. |
| Responsible team | - |
| Cost | - |
| Priority | - |

## QAWG-REC-32

| | |
|---|---|
| Recommendation | Develop clear guidelines for integrating metric collection with pipeline code. |
| Explanation | DMTN-057 suggested a number of ways in which this might be done; DMTN-098 and related work has begun development of a specific approach, which may evolve into an accepted standard. |
| JDS commentary | The "MetricTask" system described on DMTN-098 appears to be working well for AP so far. What buy in is necessary from DRP? Do SQuaRE care? |
| Responsible team | ? |
| Cost | ? |
| Priority | ? |

QA Working Group: Recommendations

## QAWG-REC-33

| | |
|---|---|
| Recommendation | Pipelines leadership should start using the metric definition and collection system. |
| Explanation | As the above recommendations are met, this system will be usable. However, driving adoption will require proactive measures from pipelines Product Owners and T/CAMs. |
| JDS commentary | The AP team have started to try and regularly follow AP metrics in SQuaSH. This would be easier given a more mature data model (see QAWG-REC-025, 026, etc). Is this useful for DRP? See comments re TE1/2 on DM-17881 which were scary. |
| Responsible team | Pipelines & SQuaRE |
| Cost | ? |
| Priority | ? |

## QA Working Group: Recommendations

# QAWG-REC-34

| | |
|---|---|
| Recommendation | SQuaSH should issue alerts to developers and key stakeholders on regressions in important metric values. |
| Explanation | Key stakeholders should include:<br>• Senior DM management (DM Project Manager, DM Subsystem Scientist, Pipelines Scientist, Science Pipelines T/CAMs and Science Leads);<br>• The developer who caused the regression, if it is possible to identify them (e.g. through commit logs).<br>This will require careful design, as it may be in tension with the desire to enable developers to define arbitrary metrics for their own use: clearly, key stakeholders will not wish to be informed of every developer-defined metric which suffers a regression. We suggest that, for example, a "subscription list" for each metric be defined, and the key stakeholders automatically be added to it for all metrics deriving directly from high-level requirements. |
| JDS commentary | I think this is likely already achieved by Chronograf (not sure how stable that is). |
| Responsible team | SQuaRE |
| Cost | Low, from where we are now |
| Priority | Medium-high |

## QAWG-REC-35

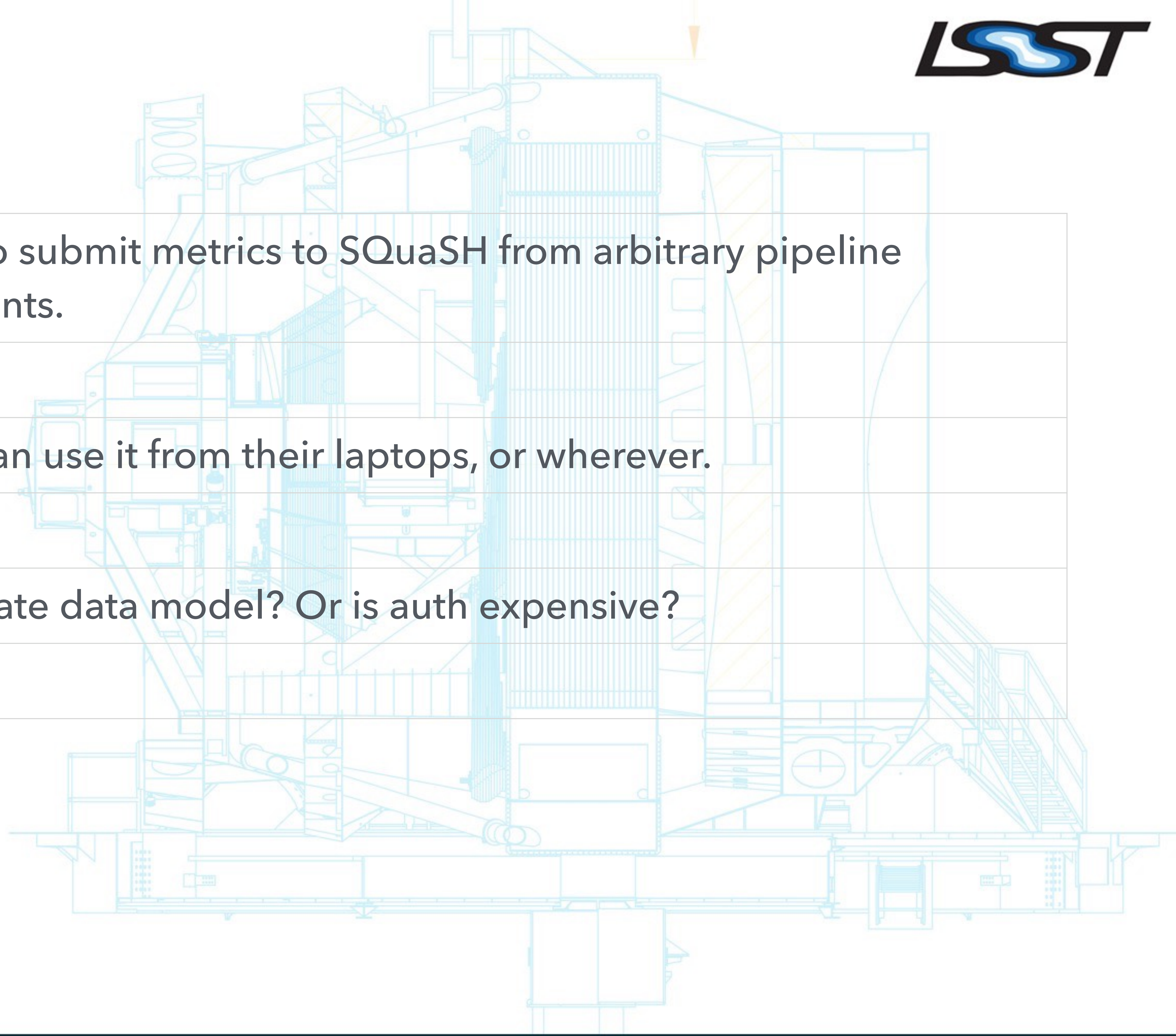
| | |
|---|---|
| Recommendation | Provide a single, reliable source of documentation describing the SQuaSH system and a vision for its use in DM-wide metric tracking. |
| Explanation | - |
| JDS commentary | I think this is effectively a repeat of QAWG-REC-25 & 26. |
| Responsible team | SQuaRE |
| Cost | None (?) additional if we address recs 25 & 26. |
| Priority | High? |

## QAWG-REC-36

| | |
|---|---|
| Recommendation | The SQuaSH system should be closely coupled to the drill-down environment; in particular, the former should use the latter to enable drill-down functionality into particular metric values. |
| Explanation | - |
| JDS commentary | See QAWG-REC-39 & 40. |
| Responsible team | SQuaRE |
| Cost | Low, if we have both SQuaSH & drill down. |
| Priority | High |

QA Working Group: Recommendations

## QAWG-REC-37



| | |
|---|---|
| Recommendation | It must be possible to submit metrics to SQuaSH from arbitrary pipeline execution environments. |
| Explanation | - |
| JDS commentary | That is, developers can use it from their laptops, or wherever. |
| Responsible team | SQuaRE |
| Cost | Low, given an adequate data model? Or is auth expensive? |
| Priority | Low? |

QA Working Group: Recommendations

## QAWG-REC-38

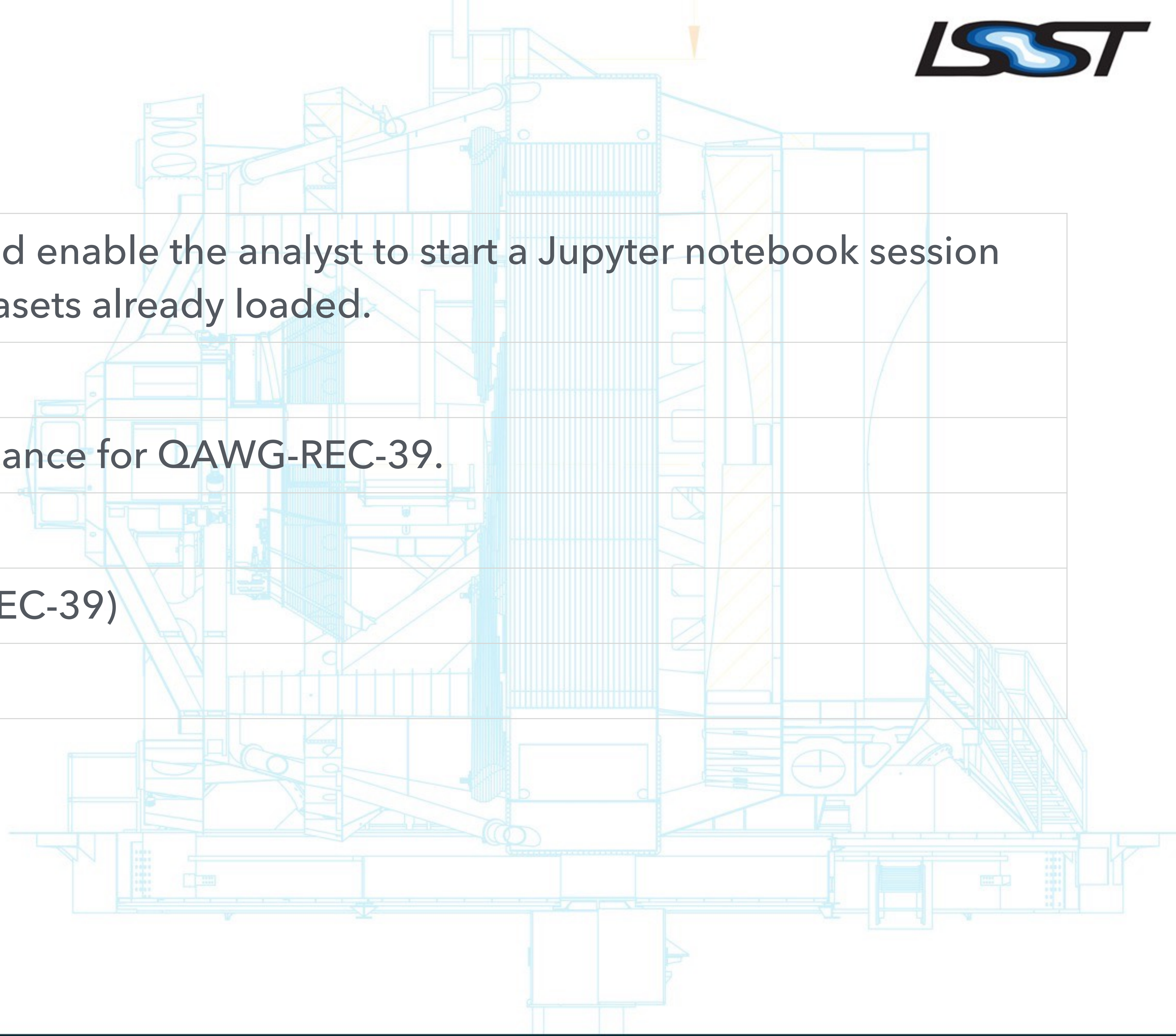
| | |
|---|---|
| Recommendation | SQuaSH should be able to store and display appropriate metric values per DataId. |
| Explanation | For example, CCD, visit, patch, tract, filter. |
| JDS commentary | Not sure how much of this is already possible – Simon? |
| Responsible team | SQuaRE |
| Cost | Low-medium, given an adequate data model? |
| Priority | High? |

## QAWG-REC-39

| | |
|---|---|
| Recommendation | DM should develop a browser-based interactive dashboard that can run on any pipeline output repository (or comparison of two repositories) to quickly diagnose the quality of the data processing. |
| Explanation | This dashboard should have two levels of detail: a high-level summary of top-level global metrics, and and a metric view showing more information on a selected metric (or set of metrics. The more detailed metric dashboard should be able to explore both coadds and individual visits. |
| JDS commentary | We're already addressing this through an external contractor (Quansight). |
| Responsible team | Pipelines |
| Cost | Medium-high |
| Priority | High |

QA Working Group: Recommendations

## QAWG-REC-40



| | |
|---|---|
| Recommendation | The dashboard should enable the analyst to start a Jupyter notebook session with the relevant datasets already loaded. |
| Explanation | - |
| JDS commentary | Implementation guidance for QAWG-REC-39. |
| Responsible team | Pipelines |
| Cost | Low (given QAWG-REC-39) |
| Priority | High |