Software Architecture and System Design of Rubin Observatory

William O'Mullane,[1] Frossie Economou,[1] Kian-Tat Lim,[2] Fritz Mueller,[2]
Tim Jenness,[1] K. Simon Krughoff,[1] Leanne P. Guy,[1]
Gregory P. Dubois-Felsmann,[3] and Ian S. Sullivan[4]

[1]*Vera C. Rubin Observatory, 950 N. Cherry Ave., Tucson, AZ 85719*

[2]*SLAC National Accelerator Laboratory, 2575 Sand Hill Rd., Menlo Park, CA 94025*

[3]*IPAC, California Institute of Technology, MS 100-22, Pasadena, CA 91125*

[4]*University of Washington, Dept. of Astronomy, Box 351580, Seattle, WA 98195*

Abstract. This paper covers some astronomy design patterns and perhaps some anti-patterns in astronomy. We will use our experience on several long projects such as Rubin Observatory, Gaia, SDSS, UKIRT, and JCMT to highlight some of the the things which worked and a few things that did not work so well.

1. Introduction

The Legacy Survey of Space and Time (Ivezić et al. 2019) is a "deep fast wide" optical/near-IR survey of half the sky in *ugrizy* bands to *r* 27.5 (36 nJy) based on 825 visits over a 10-year period. Carried out by the Vera C. Rubin Observatory on Cerro Pachón (with an altitude of 2647 m) in Chile, the survey will produce around 100 PB of data consisting of about a billion 16 Mpix images, enabling measurements for 40 billion objects.

The telescope mount assembly is due to be handed over by the end of 2022, and in 2023 the mirrors should be mounted with the commissioning camera (a single raft, about 5% of the full camera size) to perform verification of the system. The observatory is due to go into full operations in 2024. We routinely operate the Rubin Auxiliary Telescope with the LATISS instrument as both an imager and spectrograph (Ingraham et al. 2020), the latter being its purpose in operations as a calibration aid. For regularly updated key milestones see O'Mullane (2022).

2. System Vision

The mission statement for Rubin Data Management (DM) is to "Stand up operable, maintainable, quality services to deliver high-quality LSST data products for science and education, all on time and within reasonable cost."

Rubin Observatory will take an image approximately every 40 s (slew and settle time plus 30 s exposure time) which leads to around 20 TB of images streaming off the mountain from Chile each night. The 100 Gbps network from Chile to the USA was

an early investment of the project which is now finally in place (though the redundant link still has some pending work). Using this network Rubin Data Management must get the images to SLAC within seven seconds and once at SLAC prompt processing commences (see Section 3.1). After 80 hours images will be available to the data rights holders (see Marshall (2020) for more on data releases). On a roughly annual cadence DM will reprocess all the images taken since the start of operations and release new catalogs and other products as defined in Jurić et al. (2021) (see Section 3.2).

Rubin's LSST is not the first wide-field imaging survey but the combination of depth, area, and throughput make it uniquely challenging. Everything is blended and many measurements are systematics limited, but we are already testing on precursors like Hyper Suprime-Cam (HSC; Bosch et al. 2018).

2.1. Democratizing research in astronomy

We feel open-source software is key to open science: not just for traditional reproducibility arguments, but because it is a step toward inclusivity as open data alone is not enough. We must also find ways to support researchers who are resource-poor (do not have the compute resources associated with major research universities), time-poor (have a high teaching load, few/no grad students or postdocs), or who work in liberal arts colleges, historically black colleges, or other places that lack a large peer network for technical and research support.

Lowering the barrier to entry requires minimizing the investment (time, money, experience) necessary to meaningfully engage with the scientific questions that can be resolved with the data. As we move to operations we wish to provide a quality experience to all of our users and the Rubin Science Platform provides a level playing field for interacting with Rubin data. You do need an internet connection and a browser but the load is all on our servers, hence your institute does not need to have a super computer to allow sophisticated experimentation.

3. Architecture

Figure 1 shows a simplified view of the system architecture, the full details are publicly available in (Lim et al. 2018). All the DM code is available on GitHub at `https://github.com/lsst`.

DM's work commences once the image is read out of the Camera. DM already gathers some information which the Camera software puts in the image header to make a minimally meaningful image. As the data is written the Camera software also provides a quick look of the image. Once available the image is written to the Observatory Operations Data Service (OODS) and simultaneously transferred to the US Data Facility (USDF) at SLAC via the Prompt Transfer System. Though we use Rucio for transfers between facilities we could not make it fast enough for the Prompt Transfer which is custom code. We will say a little more about prompt processing in Section 3.1

On the summit there is a restricted access Science Platform which allows staff to interact with the images in the OODS directly. A cluster of about 400 cores is available for quick adhoc processing in situ, however, we expect most processing to be done at the USDF.

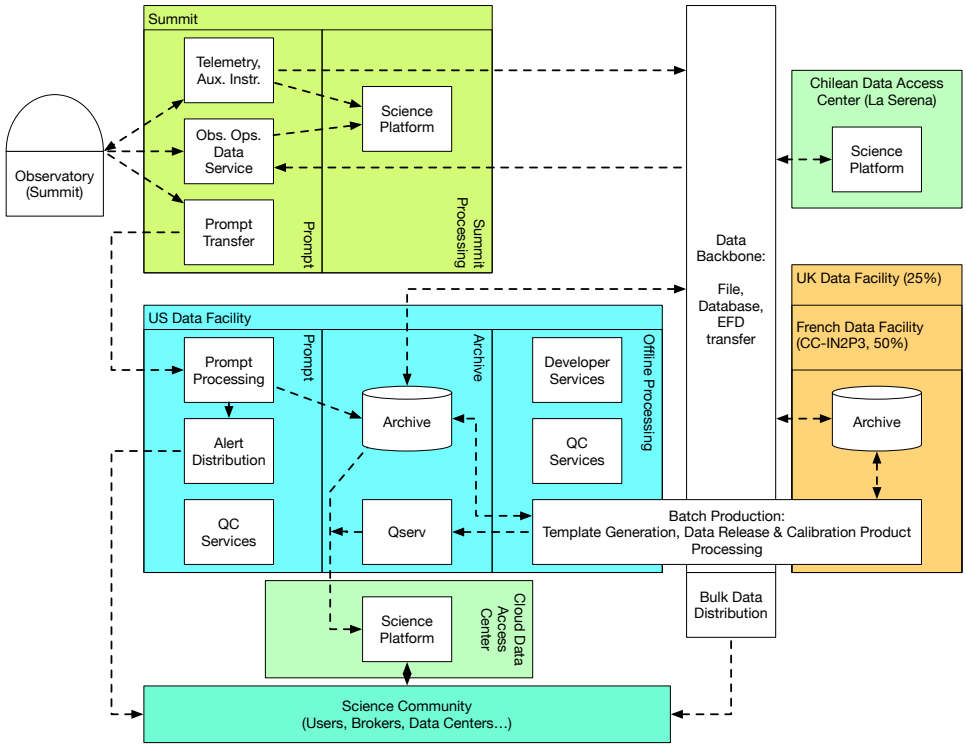

Figure 1. Simplified Vera C. Rubin Architecture diagram from Magic Draw

3.1. Prompt Processing

The Prompt Processing framework runs at the USDF, though many of the components of the framework will be reused to drive rapid analysis and quick-look functionality at the Summit and test stand facilities. The design of Prompt Processing is driven be the requirement that alerts be distributed within 60 seconds of completion of readout of the last exposure of the visit. To enable as much I/O and computation as possible to be done in advance, we instantiate one process per CCD when the summit sends a next_visit Kafka event. These next_visit events provide notice of the telescope pointing, exposure duration, filter selection, and other metadata at least 20 seconds in advance of the first exposure of a visit. Upon receiving the next_visit event, we use `knative` in a Kubernetes environment to prepare a new container where we connect to the Butler to pre-load reference catalogs, calibration products, templates, solar system ephemerides, and prior alert history. Once the raw images corresponding to an earlier next_visit event for a given detector finish downloading to a local Ceph object store, the images are ingested to the container-local Butler and the Alert Production pipeline payload begins processing. Other payloads, such as the Commissioning pipelines or Calibration Products pipelines, can also be run using the framework. The Alert Production pipeline produces packaged alerts that are streamed to the Community Brokers, writes all data products to the repository at the USDF with the Butler, and updates the Alert Pipeline Database (APDB) with new measurements. Detailed information on the initial design and prototype in the Google Cloud environment can be found in Lim (2022b). Figure 2 provides flow chart for prompt processing.

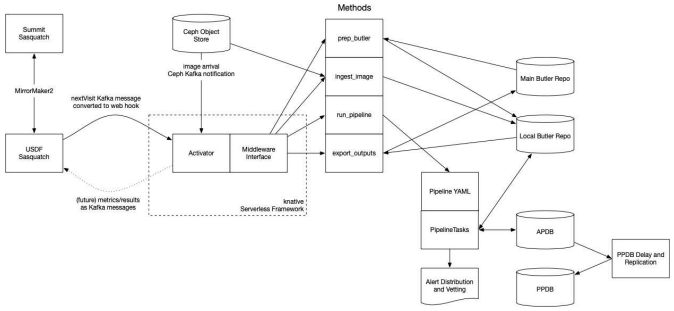

Figure 2. Prompt Processing flow diagram.

3.2. Data Release Processing

About once per year we will reprocess all data from the start of the survey. We will spend a few months in precursor runs and QA before starting the nine month processing ordeal. The jobs for this are distributed between France, US and UK using PanDA (Lim 2022a) Figure 3 provides an event chart for this.

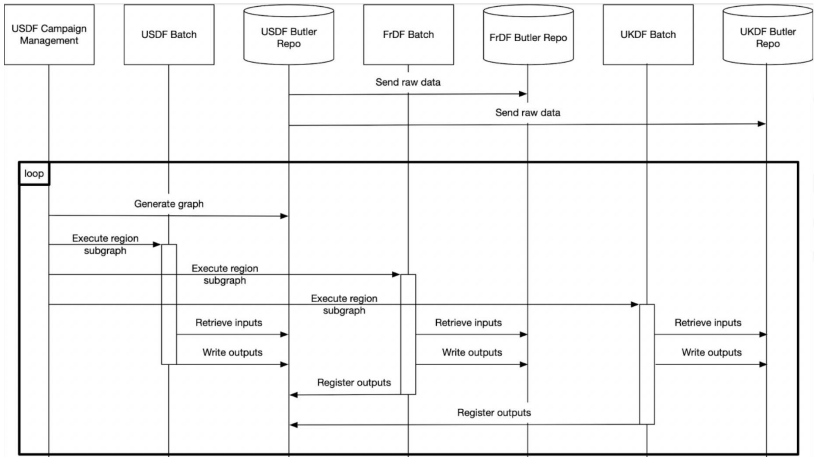

Figure 3. DRP event chart from Magic Draw

3.3. Data Access

Data rights holders will have access to the catalogs and images via the Rubin Science platform. The catalogs are large for relational databases; while the object catalog at 4×10^{10} rows may be manageable, the source catalog (of all observations) will run to 10^{12} rows, which is beyond conventional relational technology. Qserv (Mueller et al. 2023) was built specifically to answer astronomu queries quickly for large numbers of users. We have also tried non relational DBs including Google's BigQuery Thomson (e.g., 2019) but Qserv provides the best value for catalog queries. However, we see the advantage of non-relational technology for some *unpredictable* and *complex* access which Qserv may or may not handle for user-defined functions, pattern matching, or unusual iteration schemes. In fact many cloud tools work best with cloud formats like

parquet and we are considering having parquet files along side Qserv for use with Spark or DASK.

A subset of data will be public via Education and Public Outreach, including for citizen science projects. The Alert stream itself will be fully public.

We will hold most data at SLAC but run the science platform on Google (O'Mullane et al. 2021). We choose this Hybrid model since Compute is cheap on the cloud but storage is expensive. The ingress is free and the user egress is manageable.

3.4. Cloud Native

Like many projects (O'Mullane et al. 2017) Rubin leaned heavily on containers early on. We also quickly understood the need for sophisticated container orchestration and settled on Kubernetes (K8S). This decision drove service architectures that are well isolated from the underlying infrastructure. This approach has already paid massive dividends:

- When funding lines suddenly shifted we were able to painlessly transition from an on-premises facility to an Interim Data Facility on Google Cloud.
- The Rubin Science Platform (RSP) became a generic data services platform that is currently deployed on eight distinct (and distinctly managed) infrastructures (on-prem and cloud).
- Cloud can now be freely leveraged for services, like the RSP, which benefit from its advantages such as elasticity, scalability, and isolation.

Our architectural approach is geared towards lowering the cost for developing and deploying a new data service. Services utilize a common infrastructure (Phalanx[1]) providing services such as authentication and authorization, secrets management, Transport Layer Security (TLS) certificates, and templates to speed up creation of new services in the FastAPI framework. The GitOps infrastructure for K8S deployment using ArgoCD takes care of easy per-infrastructure configuration and deployment.

On the summit in Cerro Pachón, where we have on-prem machines, the Chile DevOps team use Foreman and Puppet to bring up a full K8S infrastructure. Both DM and Telescope and Site software then deploy services on top of this infrastructure. Deploying control components on K8S allows for better resilience. There are of course some devices for which this approach does not work.

4. Lessons learned

We would like to share some observations from this project in a number of areas.

4.1. Standards

Standards are excellent, they minimize the learning curve for new hires which is a major problem on all software projects. Gaia used the European Cooperation for Space

[1] `http://phalanx.lsst.io/`

Standardization (ECSS; O'Mullane et al. 2008), and Rubin used Model-Based Systems Engineering (MBSE; Selvy et al. 2018).

For astronomy we have also have the International Virtual Observatory (IVOA) standards. Gaia archive is fully IVOA based (Salgado et al. 2019; Gonzalez-Nunez 2015), and the Sloan Digital Sky Survey (SDSS) implemented and helped define many of the original protocols (Thakar et al. 2005). Rubin is IVOA first, with implementations of TAP, HiPS, and the SODA cutout service. Rubin uses DataLink to abstract image access from ObsTAP results and it is further utilized within the system such that we do not just expose IVOA services but we use them internally also.

One upside of picking IVOA standards is that now there are many implementations available. Rubin uses the CADC's TAP implementation with our own Qserv plugin; users of the TAP service have no idea they are using Qserv . That allows us to use Firefly for visualization fairly easily as it is fully VO based.

4.2. Architecture

There is a lot of analysis and design to come up with an architecture – standards help with that. Undoubtedly tools to support your chosen standards are extremely useful in the beginning, later they may become cumbersome. Both Rubin and Gaia started out with Rational Architect and switched at some point to Magic Draw. Rubin still maintains a full set of requirements and design elements in Magic Draw while Gaia switched to code as prime and reverse engineers some diagrams for documentation purposes.

For verification Rubin uses Jira Test Manager (Selvy et al. 2018) while Gaia used an in-house system based on open software (Comoretto et al. 2012). A systematic and largely automated approach to verification is needed from the outset.

One catch on both projects was to have clearly written and testable requirements, this is extremely difficult and neither project was stellar. We can easily fall in the trap of assuming that all agree on vague statements or requirements when usually a little delving will show quite the opposite. It is worth putting effort in early to write down in detail and as precisely as possible how we think things will work. This requires systems engineering which is often underestimated.

How you do your initial breakdown is possibly immaterial, they all work and you will inevitably end up with 7 ± 2 subsystems. You will end up with a a range of client server, model view controller, tiers, pipes and filters etc. Frequently large projects start all subsystems together but a slower ramp up is better in some cases to start subsystems when needed; admittedly it is hard to keep politics out of that sort of decision. For Gaia, Coordination Unit 1 (CU1 Architecture) and CU3 (Astrometry) were concentrated on initially with others trailing by some months – people from other CUs formed part of CU1. CU9 (Gaia Archive) was purposefully delayed to many years after the other parts of the project were almost built and launch was close. On Rubin all DM WBS elements started together; some teams were ready but others did not need their components yet. It did lead to good involvement in the developer guide and project management approach.

4.3. Documentation

It is highly recommended to have a good document publishing and indexing system. Provide templates for standard documents early on to make it easy for people to follow the standards. Texmf is a good way to do this for LaTeX. Most Gaia docs used the provided Latex templates and were in SVN while Livelink held the published PDFs and

the Livelink search system worked to some extent. Metadata in Livelink was curated and only documentalists were allowed to upload documents.

Rubin developed a documentation infrastructure that further lowers the barrier to documentation by providing templated creation via Slack and uses the same IDE/toolchain developers use for coding, supports Restructured Text and LaTeX, and is published via Github. Single page documents (tech notes) and site-based documentation (e.g., `https://pipelines.lsst.io`) share the same infrastructure (Sick 2015) and search indexing hub (`https://www.lsst.io`). This has also minimized useful information being consigned in hard to find (e.g., Google docs) and easy to forget or hard to edit in one's preferred IDE (e.g. wikis) locations. Rubin Docushare holds change controlled documents (PDFs, and Word), it has a search function though everyone has difficulty finding documents in it. This may be because the metadata is often incomplete or incorrect.

Both Rubin and Gaia have "bibfile" generation for all recorded docs: Gaia using Livelink is complete, Rubin using `https://lsst.io` is incomplete (since not all docs use that infrastructure). The Livelink API allowed the construction of the bibfile easily while we have failed to crack the Docushare API for Rubin, plus the Rubin metadata is less than perfect.

Glossaries are good and it is great to start them early and get everyone using them. Both Rubin and Gaia also have tools to generate acronym lists from documents (text, or tex – not Word).[2]. Gaia goes one step further with all constants used in docs and code stored in a single parameter database (de Bruijne et al. 2005), that really requires work from the outset.

4.4. Interfaces

One of our core tenets is to *separate the data model from the persistence mechanism*. Rubin Butler makes sure algorithms never access data directly (Jenness et al. 2022, 2019; Lust et al. 2023; Gower et al. 2023). Felis holds the model in YAML, Butler passes Python Objects to clients with algorithm code not knowing about data location or file formats. Similarly Gaia had data trains and the Main DataBase (MDB) dictionary which insulated algorithms from data access since the outset (O'Mullane & Lindegren 1999).

Then there are the interfaces between systems and to others; these are controlled by Interface Requirements Documents (IRD) and Interface Control Documents (ICD). These need system engineering and test plans from the outset. What Rubin calls ICDs are really only IRDs. The Gaia MDB dictionary holds all data models (Hernandez & Hutton 2015; O'Mullane et al. 2011a) and is the basis of the ICD between the subsystems. This was insufficient and we had to do work later to make data transfers and processing work well.

4.5. Products, repositories and technology stacks

Not all projects use product trees but they can be very useful. The Rubin product tree identified all software products and who is responsible for them in construction. This was good but infrequently updated (partially perhaps since it was in MagicDraw). It

[2] `http://gaia.esac.esa.int/gpdb/glossary.txt`, `https://www.lsst.org/scientists/glossary-acronyms`

should help group packages into products and help dependencies but on Rubin we have 100s of repos on GitHub, the dependencies are not straight forward, and few are usable standalone. We pretty much need to build a complete set from source giving us a sort of mono-build but we have a package-based set of GitHub repos. Clearly the latter is the correct pattern but getting cleanly defined buildable packages is hard. The use of Conda environments has allowed improvements but it started late and we have not been able to turn the tide. We did stop having patched versions for third-party packages by ensuring that corresponding packages existed in conda-forge. The middleware has been made independent and put on PyPI which allowed SPHEREx to adopt it. Some software best practices are given in Jenness et al. (2018).

Gaia has a huge SVN repo of everything based roughly on product tree. Builds are done on parts of the SVN tree – dependencies strictly managed at Jar level via Nexus. Printed out full product trees are impressive to see the amount of work to do and are useful at early reviews.

4.6. Deployment

As mentioned in §3.4 we are cloud-native on Rubin. Abstracting infrastructure effectively (Kubernetes / container orchestration, middleware etc) facilitates wider adoption of software and services by others, reducing context switching penalty and supporting continuing expertise. At a low level DM uses Puppet but SLAC were already using Chef and continue to do so – having both of these is unfortunate on one project.

Gaia chose Java for portability and ease of coding. There were no containers but Jars were always deployed from Nexus. All configurations for various machines were in SVN deployment scripts pulled correct versions to a specific machine.

4.7. Databases

Some people love them and some people hate them, some of us love them and hate them but databases are a part of any big system and choosing the correct one is hard. We think databases are great for persistence, the ability to query in different ways and relational support for Atomicity, Consistency, Isolation, and Durability (ACID). But there are problems with centralization/replication, schema evolution, and performance cliffs. We have difficulties with multi-user/multi-tenant systems but REST APIs in front help a lot.

A mistake often made is trying to use one single database – more are better and per-application databases, sometimes specialized (Redis, InfluxDB), add resilience and are more manageable nowadays. On Rubin we have InfluxDB for summit Engineering, Postgres for observing logs and ancillary info and AlertsDB. We use Cassandra for Prompt Products and of course we have in house developed Qserv for catalog access (Mueller et al. 2023). Gaia had at least Intersystems Cache for processing (O'Mullane et al. 2011b), Postgres for archive and the dictionary.

4.8. Open software project management

It seems appropriate to mention management here though that could be a topic for a full paper or book of its own (many insights may be found in O'Mullane (2005)). First we should be clear leadership is required for complex astronomy/software projects, not just management. Finding good leaders is very hard and that requires domain expertise and management training. We try to help by spreading some management across several

people, exposing them to the issues in the large project and ways to deal with them. Most importantly, provide support to potential managers/leaders. One must acknowledge this route is not for everyone – experimenting is good but give people a route back in a short timescale if they decide it is not what they wanted.

You can do Agile but you probably want earned value (Gill et al. 2014; Kantor et al. 2016) to help understand what is being delivered and since NSF and other agencies require it. You need to find good managers who understand both technical and managerial needs which is super hard. Of course the aim is to build a techno/scientific culture in leadership and breed more managers of the required ilk and create community and collaboration around a codebase nurtured by those mangers. To help we should be offering opportunities for getting career credit for supporting the mission and its community, not just first-to-publish.

There are lots of parties and institutes in big projects. If you want to do open source put it in the contracts/agreements from the outset. Licensing is important, Rubin picked GPL but would now prefer a less restrictive license – it's hard to change so do not make it too explicit in the contracts.

Acknowledgments. This material or work is supported in part by the National Science Foundation through Cooperative Agreement AST-1258333 and Cooperative Support Agreement AST1836783 managed by the Association of Universities for Research in Astronomy (AURA), and the Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory managed by Stanford University.

References

Bosch, J., et al. 2018, PASJ, 70, S5. arXiv:1705.06766

Comoretto, G., Gallegos, J., Els, S., Gracia, G., Lock, T., Mercier, E., & O'Mullane, W. 2012, in Modeling, Systems Engineering, and Project Management for Astronomy V, vol. 8449 of Proc. SPIE, 84490G

de Bruijne, J. H. J., Lammers, U., & Perryman, M. A. C. 2005, in The Three-Dimensional Universe with Gaia, edited by C. Turon, K. S. O'Flaherty, & M. A. C. Perryman, vol. 576 of ESA Special Publication, 67

Gill, R., Gracia, G., Lupton, R. H., & O'Mullane, W. 2014, in Modeling, Systems Engineering, and Project Management for Astronomy VI, vol. 9150 of Proc. SPIE, 91501E

Gonzalez-Nunez, J. 2015, in Science Operations 2015: Science Data Management, 8

Gower, M., et al. 2023, in ADASS XXXII, edited by S. Gaudet, S. Gwyn, P. Dowler, D. Bohlender, & A. Hincks (San Francisco: ASP), vol. TBD of ASP Conf. Ser., TBD

Hernandez, J., & Hutton, A. 2015, in ADASS XXIV, edited by A. R. Taylor, & E. Rosolowsky, vol. 495 of ASP Conf. Ser., 47

Ingraham, P., et al. 2020, in Software and Cyberinfrastructure for Astronomy VI, vol. 11452 of Proc. SPIE, 114520U

Ivezić, Ž., et al. 2019, ApJ, 873, 111. arXiv:0805.2366

Jenness, T., Bosch, J. F., Salnikov, A., Lust, N. B., Pease, N. M., Gower, M., Kowalik, M., Dubois-Felsmann, G. P., Mueller, F., & Schellart, P. 2022, in Software and Cyberinfrastructure for Astronomy VII, vol. 12189 of Proc. SPIE, 1218911. arXiv:2206.14941

Jenness, T., et al. 2018, in Software and Cyberinfrastructure for Astronomy V, vol. 10707 of Proc. SPIE, 1070709

Jenness, T., et al. 2019, in ADASS XXVII, edited by P. J. Teuben, M. W. Pound, B. A. Thomas, & E. M. Warner, vol. 523 of ASP Conf. Ser., 653. arXiv:1812.08085

Jurić, M., et al. 2021, Data Products Definition Document. URL https://lse-163.lsst.io/

Kantor, J., Long, K., Becla, J., Economou, F., Gelman, M., Juric, M., Lambert, R., Krughoff, S., Swinbank, J. D., & Wu, X. 2016, in Modeling, Systems Engineering, and Project Management for Astronomy VI, edited by G. Z. Angeli, & P. Dierickx, vol. 9911 of Proc. SPIE, 99110N

Lim, K.-T. 2022a, DMTN-213: Multi-Site Data Release Processing Using PanDA and Rucio. Vera C. Rubin Observatory, URL https://dmtn-213.lsst.io/

— 2022b, DMTN-219: Proposal and Prototype for Prompt Processing. Vera C. Rubin Observatory, URL https://dmtn-219.lsst.io/

Lim, K.-T., Bosch, J., Dubois-Felsmann, G., Jenness, T., Kantor, J., O'Mullane, W., & Petravick, D. 2018, LDM-148: Data Management System Design. URL https://LDM-148.lsst.io/

Lust, N., et al. 2023, in ADASS XXXII, edited by S. Gaudet, S. Gwyn, P. Dowler, D. Bohlender, & A. Hincks (San Francisco: ASP), vol. TBD of ASP Conf. Ser., TBD

Marshall, P. 2020, Release Scenarios for LSST Data. URL https://ls.st/RDO-11

Mueller, F., et al. 2023, in ADASS XXXII, edited by S. Gaudet, S. Gwyn, P. Dowler, D. Bohlender, & A. Hincks (San Francisco: ASP), vol. TBD of ASP Conf. Ser., TBD

O'Mullane, W. 2005, Large Scientific Data Systems: analysis of some existing projects and their applicability to Gaia, Tech. rep., University of Barcelona. Treball GAIA-C1-ESAC-HA-WOM-003, URL https://dms.cosmos.esa.int/COSMOS/doc_fetch.php?id=497678

O'Mullane, W. 2022, Celebratory Milestones. URL https://dmtn-232.lsst.io/

O'Mullane, W., Economou, F., Huang, F., Speck, D., Chiang, H.-F., Graham, M. L., Allbery, R., Banek, C., Sick, J., Thornton, A. J., Masciarelli, J., Lim, K.-T., Mueller, F., Padolski, S., Jenness, T., Krughoff, K. S., Gower, M., Guy, L. P., & Dubois-Felsmann, G. P. 2021, in ADASS XXI, ASP Conf. Ser., in press. arXiv:2111.15030

O'Mullane, W., Hoar, J., & Lammers, U. 2008, in ADASS XVII, edited by R. W. Argyle, P. S. Bunclark, & J. R. Lewis, vol. 394 of ASP Conf. Ser., 191. arXiv:0712.0249

O'Mullane, W., Lammers, U., & Hernandez, J. 2011a, in ADASS XX, edited by I. N. Evans, A. Accomazzi, D. J. Mink, & A. H. Rots, vol. 442 of ASP Conf. Ser., 351

O'Mullane, W., Lammers, U., Lindegren, L., Hernandez, J., & Hobbs, D. 2011b, Experimental Astronomy, 31, 215. arXiv:1108.2206

O'Mullane, W., & Lindegren, L. 1999, Baltic Astronomy, 8, 57

O'Mullane, W., Morris, D., & Hoar, J. 2017, in ADASS XXV, edited by N. P. F. Lorente, K. Shortridge, & R. Wayth, vol. 512 of ASP Conf. Ser., 33

Salgado, J., González-Nuñez, et al. 2019, in ADASS XXVII, edited by P. J. Teuben, M. W. Pound, B. A. Thomas, & E. M. Warner, vol. 523 of ASP Conf. Ser., 445

Selvy, B. M., Roberts, A., Reuter, M., et al. 2018, in Modeling, Systems Engineering, and Project Management for Astronomy VIII, edited by G. Z. Angeli, & P. Dierickx, vol. 10705 of Proc. SPIE, 107050U

Sick, J. 2015, SQR-000: The LSST DM Technical Note Publishing Platform. Vera C. Rubin Observatory SQuaRE Technical Note, URL https://sqr-000.lsst.io/

Thakar, A. R., Szalay, A. S., O'Mullane, W., Budavári, T., Nieto-Santisteban, M. A., Fekete, G., Li, N., Carliles, S., Gray, J., & Lupton, R. 2005, in ADASS XIV, edited by P. Shopbell, M. Britton, & R. Ebert, vol. 347 of ASP Conf. Ser., 684

Thomson, J. R. 2019, LSST Benchmarking of Qserv and BigQuery. Vera C. Rubin Observatory, URL http://ls.st/Document-31100