# DAQ/Control Software Framework Trade Study

## L3 Lead - Cosmin Deaconu

CQ3

# Outline

- Presenter Introduction
- Key L3 contributors
- Overview / Scope
- Key Criteria for Trade Study
- Summary of Each Package
- Result of Trade Study
- Conclusion

# Presenter Introduction

Name:  Cosmin Deaconu

Institution: University of Chicago

Discipline: Particle Astrophysics

Previous experience:

- Postdoc (2015-2020)/Research Scientist (2020-): Development and integration of data acquisition / control / data management software for the ANITA long-duration balloon payload and prototype radio neutrino detectors at the South Pole, the White Mountains of California and Greenland.  Also key contributor to analysis/simulation software and data analysis.
- PhD (2009-2015, MIT) on directional dark matter detection instrumentation.

# Key Contributors in this L3

All members of this L2 contributed to the trade study.

Abby Crites  (Toronto)

Cosmin Deaconu (Chicago)

Brian Koopman (Yale)

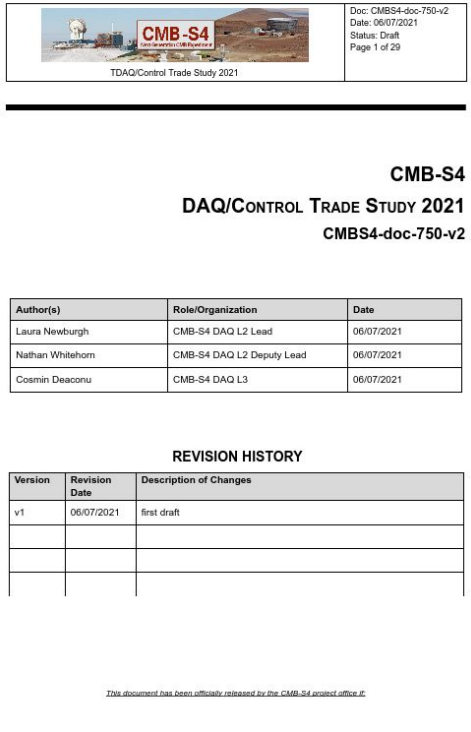
Laura Newburgh (Yale)

Sasha Rahlin (Fermilab)

Nathan Whitehorn (Michigan State)

# Overview/Scope

- To select the software framework (or identify that none exists) for S4 DAQ and Control, a trade study of various packages used in the community was commissioned to see if any offer advantages over or can help shape the baseline design.
- The output of this trade study is a detailed document describing the findings of the trade study, to be distributed to the collaboration for feedback.
- You should have received a draft, expected to be complete “soon.”



CMB-S4

TDAQ/Control Trade Study 2021

Doc: CMBS4-doc-750-v2
Date: 06/07/2021
Status: Draft
Page 1 of 29

**CMB-S4**

**DAQ/Control Trade Study 2021**

**CMBS4-doc-750-v2**

| Author(s) | Role/Organization | Date |
|---|---|---|
| Laura Newburgh | CMB-S4 DAQ L2 Lead | 06/07/2021 |
| Nathan Whitehorn | CMB-S4 DAQ L2 Deputy Lead | 06/07/2021 |
| Cosmin Deaconu | CMB-S4 DAQ L3 | 06/07/2021 |

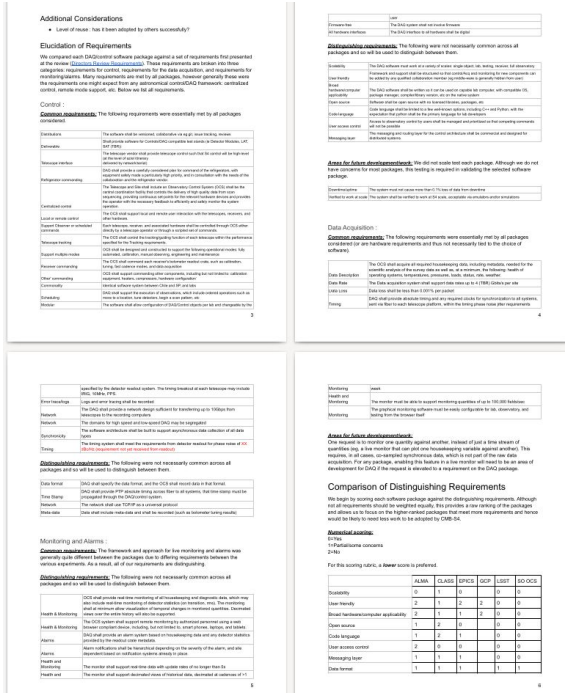
**REVISION HISTORY**

| Version | Revision Date | Description of Changes |
|---|---|---|
| v1 | 06/07/2021 | first draft |
| | | |
| | | |
| | | |

*This document has been officially released by the CMB-S4 project office it*

# Key Criteria for Trade Study (Summary)

- Capability
  - Meets the throughput, monitoring, latency requirements and supports the variety of devices required by CMB S4 without too much additional development (reduce risk, control costs)
- Availability
  - Either open source or without severe licensing restrictions. Must be future-proof, including dependencies for lifetime of experiment.
- Simplicity for hardware integrators
  - Adding devices must be user friendly to non-experts on DAQ. Use standard protocols, programming languages known to collaborators (Python/C++), etc..
- Flexibility / Ease of Configuration
  - Same system should be usable both in Chile and South Pole and at various labs where hardware integration will happen.
- User-friendly, yet powerful, control and monitoring




More details in trade study doc

# Evaluation Rubric

- We developed a rubric for our key criteria (below), scoring each package with a numerical score as a rough means of quantitative comparison
  - 0: Meets requirements
  - 1: Partially meets requirements or some concerns
  - 2: Does not seem to meet requirements

- Scalability
- User-friendly
- Broad hardware/computer support
- Open Source
- Familiar Coding Language
- Access Control
- Messaging Layer
- Data Format
- Time Stamp
- Network Transparency
- Metadata support

- Realtime + historical monitor
- Monitoring via browser
- Alarms on detector stats
- Hierarchical alarms
- <5 s monitor latency
- Decimated monitoring
- Monitor supports 100 k fields/s
- Monitoring easily configurable

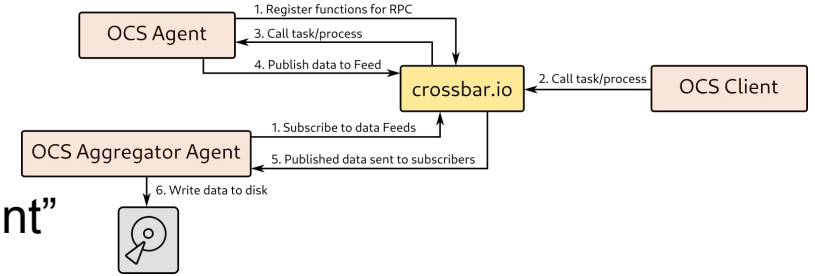
# Packages Considered

- We considered a number of packages from the community we thought might be up to the task (or we could at least learn from), as well as one commercial package.
  - Simons Observatory (SO) Observatory Control System (OCS) (*baseline design*)
  - ALMA Control Software (ACS)
  - EPICS
  - Generic Control Program (SPT )
  - CLASS Control Software
  - LSST / Vera Rubin Observatory Control Software
  - Ignition (Inductive Automation) (*Commercial*)
- In nearly all cases, we were able to obtain the software and test it out, in some cases doing stress testing, etc.
- What follows is a one slide summary of the key points of each package

# Simons Observatory OCS (baseline)

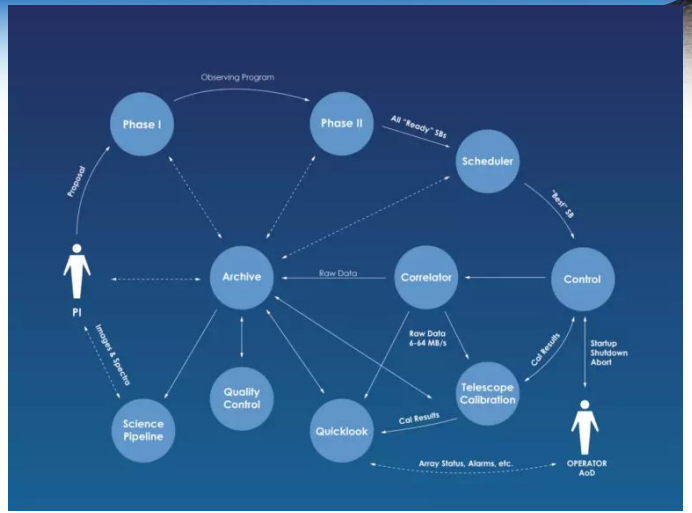
- Focus on modular design using commercially-supported OSS
- Crossbar.io is the message/pubsub layer
  - Seems capable, some concerns about packaging.
- Largely Python3 with simple “hardware agent” API. Clients can also be JS.
- Preferred deployment uses docker
- SPT-3G data format (boost + cereal)
- Stress testing found bottlenecks, since fixed.
- InfluxDB + Grafana for monitoring.
  - Grafana flexible, pretty, easy to use, and supports many back ends.
  - InfluxDB may need to be swapped out to meet all monitoring cases.
- Impression: No showstoppers, high familiarity.

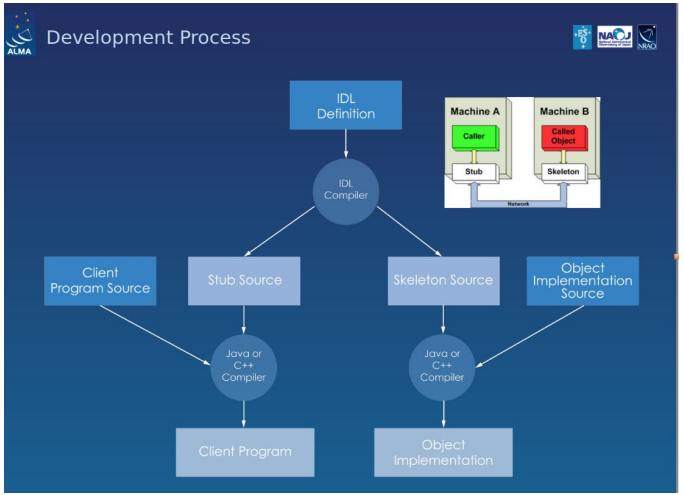
Full description: arXiv:2012.10345




# ALMA Control Software

- Based on a CORBA-based IPC framework
  - CORBA has lost a lot of mindshare since the 90's and feels outdated.
  - Other messaging systems  also partially supported for some pieces, though some proprietary/$ (RTI DDS)
- Supports C++/Java/Python components
  - Tools are heavy on Java.
  - Python3 migration in progress
  - All modules must be defined using the CORBA interface DSL, with lots of boilerplate.
- Installation tricky, mostly supports EL7 (VM is often best choice).
- 6 FTEs working on supporting it, also chosen for CTA.
- Impression: Not a good starting point in 2021

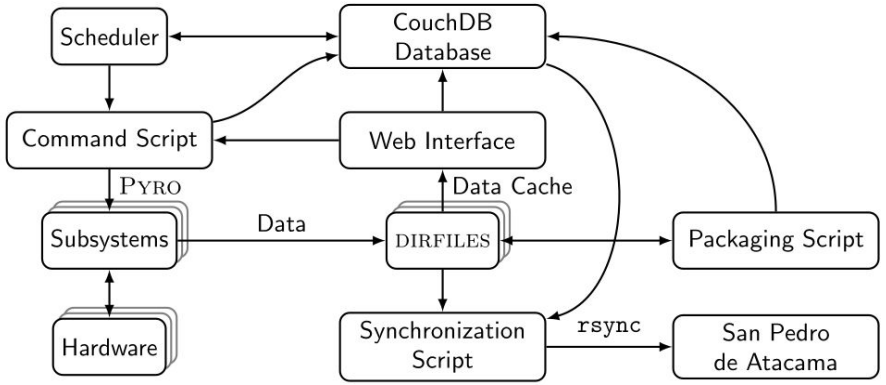


# CLASS control and monitoring

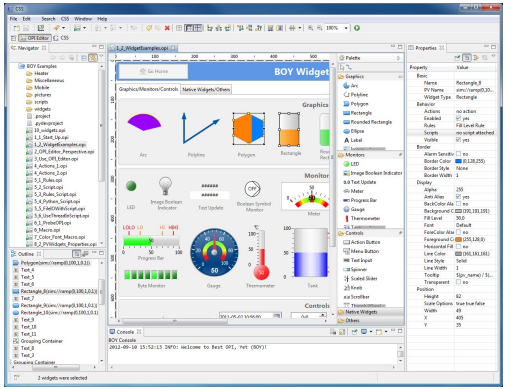
- Software catered to CLASS requirements, would require significant effort to adapt to other experiments (hardcoded paths, etc).
- Not open-source or easily available.
- PYRO4 used for IPC layer (potentially doesn't scale)
- Data cache is DIRFILES (i.e file system as database) with KST for monitoring (unlikely to be scalable, poor long-term prospects)
- Impression: Not flexible or scalable enough to serve as a base for S4.

SPIE Proceedings: doi: 10.1117/12.2561609



# EPICS


- Comprehensive control/monitoring ecosystem, going strong for 30 years and used for operation at many accelerators/national labs.
- The “new” messaging/pubsub layer (pvAccess) flexible and high-throughput
- A ton of mindshare and HW support.
- However, steep learning curve for “outsiders” (us) and paralysis of choice
  - e.g. a plethora of GUI frameworks to choose from, from motif to QT-based to REACT. Some legacy, some with different capabilities than others.
- Many domain-specific jargon, tools would require training
  - e.g. DSL for defining each client
- Impression: We lack enough familiarity with ecosystem in our community to recommend, though obviously capable.



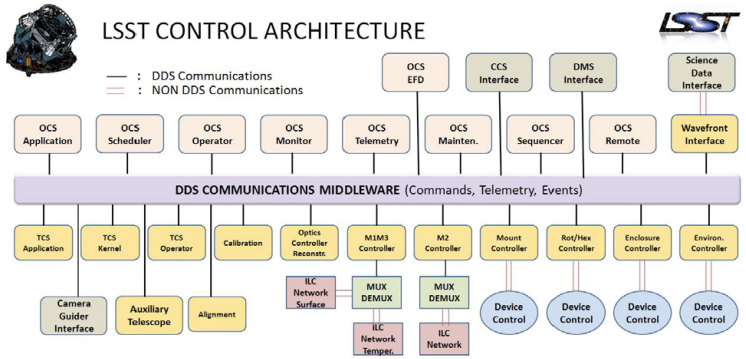

One of the many ways to design interfaces

# GCP (Generic Control Program)

- Used by SPT
- Monolithic giant process, hard to maintain.
- Somewhat clunky and fragile design using TCP sockets for IPC.
- Prior experience by collaborators not positive enough to warrant further investigation.
- C++ only
- Legacy code
- Not scored.

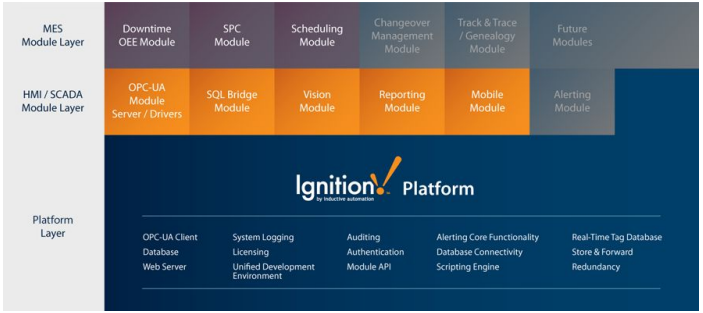
# LSST

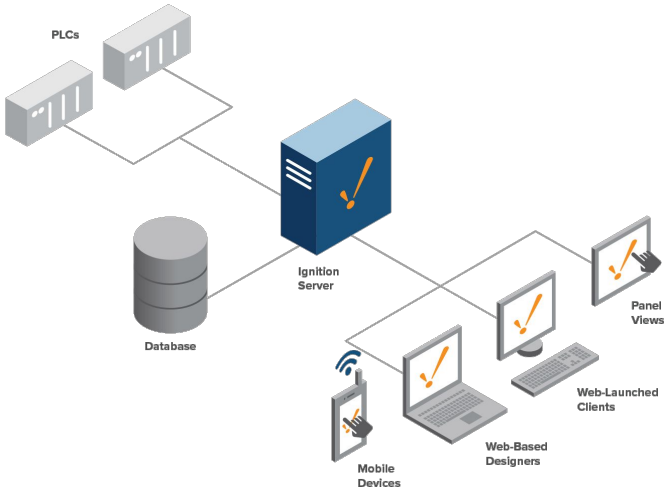
- Messaging built on top of uses OpenSplice by PrismTech
  - Potentially licensing issues. (??)
  - Only supports EL7 (??)
- Support for Python/C++/Java/Labview components
  - User-friendly Python API for hardware deployment
  - Many tools Java-based though
- Unclear how difficult a deployment setup is
  - May be difficult to setup as a testbench or multiple sites .
- Impression: Overall positive, but the middleware layer and potential difficulty of deployment are potential showstoppers.



# Ignition

- Proprietary industrial control tool by Inductive Automation
- Time-limited trial version evaluated
- Installation is easy, but licensing may be an issue for testbench setups.
  - Much functionality needs add-on modules.
  - Concern about future availability
- Simple point and click interface, language is from industry world (e.g. “Historian.”)
- Unclear to what extent it is scriptable/adaptable/ etc.
  - Scripting in Jython, which has uncertain future.
- Impression: Pretty, but is not clear if a realistic option and not open. Not scored.



# Trade Study Results

- Numbers for each criteria are summed. Lower score is better.
  - SO OCS: **1**
  - LSST: **6**
  - EPICS: **10**
  - CLASS: **19**
  - ALMA: **23**
  - GCP: **Not scored**, unlikely to be competitive.
  - Ignition: **Not scored**, unlikely to be competitive.
- Conclusion: No strong reason to deviate from baseline design (S4 OCS based on SO OCS).
  - There is no framework that will require less work / training to get up to speed for S4.
  - Reassuringly, most of the frameworks are pretty similar in architecture to SO OCS, although many burdened with legacy costs.
  - SO OCS is modular enough that components from other frameworks could be considered for adoption in the future if a problem is found with a given component.

# Conclusions

We did a survey of a variety of options on which to base the CMB-S4 DAQ, learning a lot about different packages.

In the end we found that the baseline design is very competitive and there is no alternative that is obviously a superior choice for the collaboration.

Trade Study draft will be finalized after review by the Project.