Computing M&C Protocol Trade Study
ngVLA Computing Memo #2

P. Brandt, R. Hiriart, R. Amestica

August 4, 2021

1 Introduction

This memo explores a selection of technologies in a trade study for the ngVLA online control software. We examine the design approach that the software team will use, the technology selection criteria, and how the criteria were applied to choose between the identified technology candidates.

The technologies described in this memo are fitted into the five layered general architecture described in the monitor and control system reference design concept [9]. Specifically, we focus on the interface between the Hardware Interface Layer and the Supervisor Layer shown in figure 1. These two layers are present in the antenna and correlator systems.

This choice of technologies represents a small, though important, part of the online system. It may serve as the basis for further architectural decisions, and will have dramatic effects on the evolution of the system for at least a decade, if experience from ALMA and JVLA is any indication.

An important assumption that this memo makes is that the interface between all of the layers[1] will be Ethernet-based. Many industrial control systems have adopted Ethernet, and given its ubiquity and low cost it is a natural choice for the ngVLA. Even in the event that the standard internet interface evolves

[1] Excluding the interface between the Hardware Controller Layer and the Hardware Device Layer. The interface between those is the responsibility of the hardware team(s), and also tends to be rather specialized to the custom hardware needed for various components of the antennas and correlator.

or changes in the years to come, choosing software and hardware that is in the same upgrade path should provide the ngVLA with an opportunity to avoid getting locked into legacy technologies.

2 Control System Architecture

Control systems for radio astronomy and other physics instruments have traditionally been built as custom one-off systems. Even single observatories with multiple telescopes often create new control systems for each project. However, similar architectures can be applied to any control system, ranging from radio telescopes to factories. One example is ANSI/ISA-95[2], an international standard describing how to create automated interfaces between enterprise systems and control systems. It defines common terminology, information models, and operations models that can be applied to any industry.

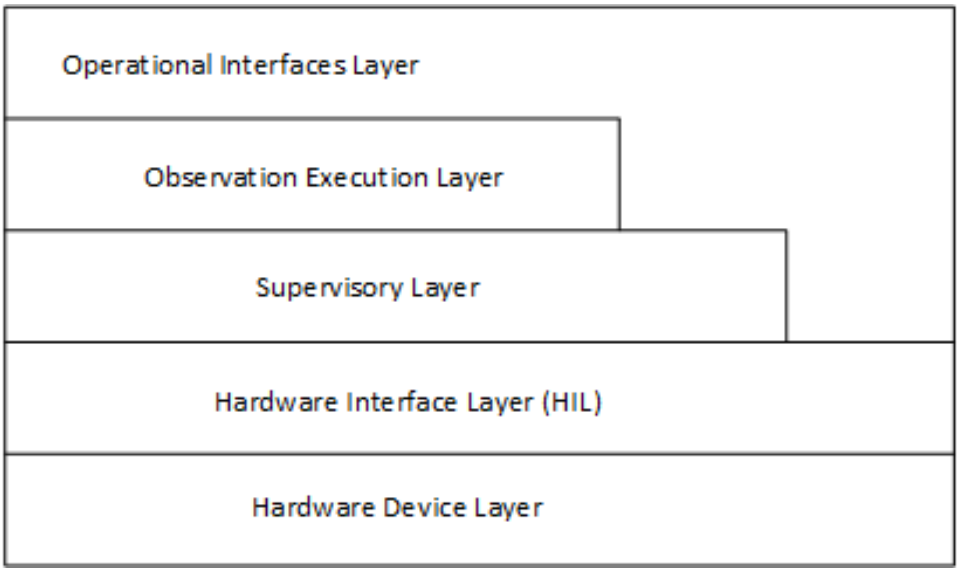

Figure 1: ngVLA Reference M&C Architecture

The ISA-95 "automation pyramid" is shown in figure 2. Each of the five layers can be directly mapped to the ngVLA's five-layered reference design architecture (and indeed to the ALMA and VLA control systems).

Level 0 components would be devices inside of the correlator or antenna which utilize a field bus to communicate with Level 1, the Hardware Controller

[2] https://en.wikipedia.org/wiki/ANSI/ISA-95

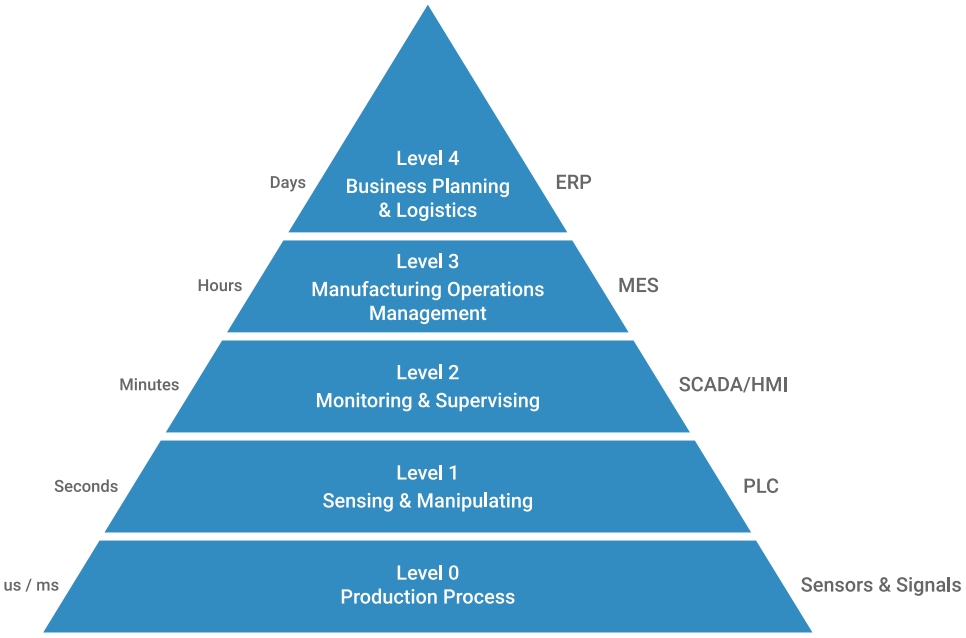

Figure 2: The ISA-95 automation pyramid. The time scales listed on the left side show the representative deadline for decision making. For example, Level 0 devices operate in the hard real-time domain where events must be handled as fast as possible, often with deterministic behavior. Components at Level 2, usually the SCADA system or humans interacting with a console, can take minutes to deal with events and make decisions.

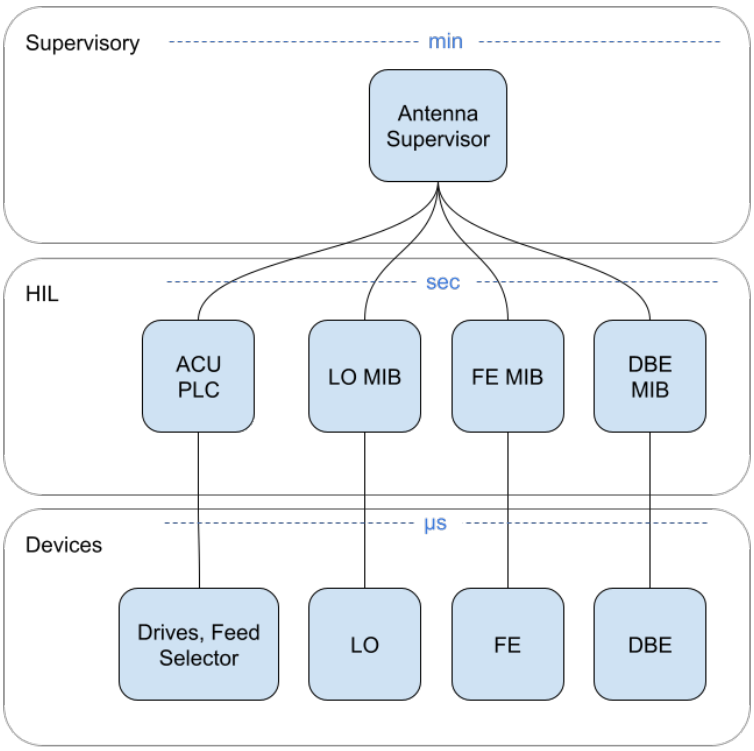

Figure 3: Recontextualizing the automation pyramid in terms of how the M&C architecture might look. The time scale has been moved to show the communication deadline between the layers.

Layer (aka the "MIB" in the case of antennas). This document seeks to examine technologies to connect levels 1 and 2.

This parallel speaks to a lack of unique requirements for astronomical control systems, and opens the door to consider many other options than have traditionally been evaluated. It also places the ngVLA in a much wider context of control systems, which can inform how we approach building something orders of magnitude larger than previous projects at NRAO. Importantly, the ngVLA is **not** significantly larger than existing systems in other industries, and may in fact be smaller.

Specific technologies tend to fill niches within each of these layers. There is a trend towards some technologies spanning multiple layers, but they tend to introduce interoperability problems due to using non-standard or proprietary protocols. One example is TANGO, which falls into layer 2; it can expand into layers 1 and 3, but that forces CORBA (or perhaps ZeroMQ in the future) as the protocol between layers.

3 Methodology

The ngVLA software team will use attribute-driven design[3] (ADD) for the evaluation and selection of the monitor and control system. ADD combines functional requirements with a list of quality attributes that the system must have in order to generate an architecture.

Quality attributes are measurable or testable properties of a system which are used to indicate how well the system satisfies the needs of its stakeholders. That is, non-functional requirements which represent what makes a system good for a specific stakeholder. Quality attributes can be expressed through scenarios which describe their impact on the system, either generic or concrete. Concrete scenarios in particular can be useful for turning nebulous attributes desirable for a system, for example "performance" or "reliability", into well-defined tangible requirements. However, in this memo we will cover quality attributes qualitatively, as concrete scenarios are mainly useful for refining subsystem L2 requirements.

Quality attributes goals are achieved by utilizing *tactics*. These tactics are analogous to design patterns (recipes or designs which have proven effective in solving a particular problem) but applied at an architectural level. For example, an obvious tactic that could be applied to the problem of controlling an array with hundreds of antennas is to introduce concurrency in sending commands.

[3] https://en.wikipedia.org/wiki/Attribute-driven_design

A number of quality attributes have been identified for the ngVLA control system: cost, interoperability, scalability, reliability, and security. These will assist in selecting an appropriate architecture and technology stack. The quality attributes are discussed below, along with a brief summary of what tactics could be applied to achieve them.

This list of quality attributes notably omits Maintainability, because as a standalone term it lacks specificity and is all too often included as a catch-all without any tangible meaning for the project requirements and goals [8]. Therefore we will attempt to elaborate on the ways in which other quality attributes contribute concretely to this goal.

3.1 Cost

One of the biggest drivers of the design for the ngVLA project is to minimize costs throughout the lifecycle of the project, as described in the concept document [2]. This attribute will feed into many other attributes, as well as the design of the monitor and control system. For example, one strategy being considered is using code generation based on the system model to lower development and maintenance costs for the monitor and control software systems. The ability to purchase and integrate COTS software can also be a large cost savings.

The cost of software maintenance can be difficult to quantify, especially early on in a project, but most estimates put it between 40% and 80% of software lifecycle costs, or 60% on average [7]. This means that maintenance is the most important part of the complete software lifecycle to consider when looking to minimize overall cost. When looking at modern software development practices of major software companies (such as Netflix, Google, Microsoft, and Amazon), sophisticated testing and deployment pipelines are created to handle the majority of maintenance tasks and improve developer productivity [10, 11, 12].

Automation in general will be important to minimize the number of human hours required to maintain and service the telescope during operations. Subarrays will be critical to the operation of the telescope, which will require careful planning to successfully schedule, operate, and manage. This will necessitate upfront investment in telescope support infrastructure, as well as careful planning in the design to increase maintainability. One example is ensuring that subarrays capabilities such as running multiple software versions are considered in the architecture and design of the monitor and control system.

Cost (and maintainability) relate to the protocol selection by determining how operators, maintenance personnel, and their associated software systems will interact with each other. Condition-based maintenance, predictive mainte-

nance, and engineering tools could be purchased if we select an industry standard protocol.

3.2 Interoperability

Interoperability refers to the ability for a system to provide services to or receive services from another system to enable them to operate together, even when they are produced by different vendors. Typically this is accomplished by developing well-defined, standardized interfaces between the components that must be interoperable, or using standardized interfaces (e.g. Modbus or OPC UA). Interoperability can be useful for simplifying automation, integrating COTS hardware and software, improving efficiency between developers, and allowing for a cleaner final integration.

For example, the antenna and correlator control systems would need to communicate with operator consoles, science observation executor(s), and engineering and maintenance tools. This necessitates a well defined interface that can serve the needs of all the systems involved.[4]

We should avoid reinventing the wheel wherever possible as developing a system from scratch will be either more expensive than commercial solutions, or it will lack the quality and support of commercial solutions. With one-off development we miss the economies of scale involved in developing software for many customers. An exception to this rule happens when software is developed for very specific industries to a limited number of customers that are willing to pay because these systems are critical infrastructure items for their operations and revenue. An example of this are SCADA solutions developed specifically for the oil, electric or gas industries. They tend to be prohibitively expensive. However, it is possible to find much affordable generic SCADA systems that could be appropriate for our requirements. The needs of a monitor and control system are not so different from other industrial control applications. Therefore, it should be possible to use largely off-the-shelf solutions, rather than writing a custom system.

An industrial protocol would be best to interact with the lower layers of hardware and software. It is especially important to consider how the system will evolve over time and deal with obsolescence.

More desirable is a protocol specification at the application level (i.e. monitor and control), rather than one only at the transport level. This is also important if we want to look at buying commercial products (e.g. an Human-

[4] Ideally without exposing unnecessary functionality; an operator should not be presented with engineering interface functionality, unless it is also required for their interfaces.

Machine Interface or Historian if using OPC UA). However, any solution must also take into account what field bus is being used in the low level hardware.

3.3 Scalability

Any solution we choose must be able to scale to the expected size of the ngVLA.

With this attribute, we will favor architectures that can be scaled horizontally. Scaling horizontally means that processing can be distributed across multiple cores, CPUs, or computers. In contrast, scaling vertically involves purchasing more powerful hardware to run the software faster.

3.4 Reliability

One of the most important attributes that derives from keeping lifecycle costs at a minimum is reliability. With a significant number of antennas at extremely remote sites, it will be important to minimize the number of maintenance visits to each antenna. It will also be important to maximize the uptime of the system, as a small failure rate can translate into a large maintenance footprint when scaled up to thousands of components within the system. Downtime also reduces the effective observing time available, representing a large time (and therefore monetary) loss to the observatory.

3.5 Security

At least some parts of the ngVLA system will be in remote locations, with communication running through public networks. Therefore security is an important consideration for the monitor and control system to ensure safe operation of these remote telescopes.

Security is not an attribute that can be ignored in the early phases of a project. There are a variety of resources we plan to utilize, primarily the approach laid out in the National Science Foundation Major Facilities Guide[6]. Other resources we could use include the NSF Cybersecurity Center of Excellence Trusted CI guide[1], NIST cybersecurity risk framework[13], and the Purdue Model for Industrial Control Systems Security. Creating a comprehensive cybersecurity plan is a broad topic and will be the subject of a future memo or project document.

Security concerns for the ngVLA control system revolve around preventing unauthorized control of the telescopes and correlator hardware, since intrusions into industrial SCADA systems have increased dramatically in the last two decades[4]. However, there are also data security and data privacy concerns to

consider with the data acquired by the telescopes being transmitted to the correlator, and later to the data center. These are largely new challenges, though there is some experience with transmitting data between ALMA Regional Centers (ARCs) that the ngVLA project team can benefit from.

4 Field Survey of Telescope Control Systems

In order to select candidate technologies, we examined the quality attributes detailed above, used experience with prior projects (ALMA and VLA), and a survey of what other telescopes and wider industry are using for control software.

There are a few technologies developed specifically for scientific instruments, primarily Tango[5] and EPICS[6]. EPICS uses its own custom protocol layered over TCP or UDP. Tango currently uses CORBA, though there has been efforts to transition to ZeroMQ.

Many industrial applications, as well as some instruments at CERN [5], are using OPC UA. The ELT project is also using OPC UA as part of their middleware system.[7] OPC and OPC UA are also widely used for control systems in both motion controls and static sensors, such as plants and field monitoring devices in the petroleum business [3]. OPC UA is a standardized service oriented architecture which provides a number of low level mechanisms like service discovery, security, and platform independence. It also allows for modeling systems within it's servers, allowing clients to handle common components in a consistent manner.

A newer trend is the use of protocols related to the Industrial Internet of Things[8] (IIoT). IIoT is a distributed system of sensors and devices which can communicate directly with one another, as well as higher level orchestration and visualization software. IIoT also provides integration of control system data into cloud services via edge computing, allowing existing cloud services to operate on the data for tasks like predictive maintenance.

There is also the work done for current NRAO telescopes, such as the ALMA monitor and control system utilizing CORBA in the ALMA Common Software (ACS), and the current VLA control system which uses a custom protocol.

[5] https://www.tango-controls.org

[6] https://epics-controls.org

[7] https://medium.com/control-sheet/middleware-abstraction-layer-in-the-elt-core-integration-infrastructure-28b91e705136

[8] https://en.wikipedia.org/wiki/Industrial_internet_of_things

5 Protocol Selection

In order to simplify the many factors that go into selecting a protocol, we can break the problem down into a series of questions in a decision tree. In each of the following subsections, we present a question in the decision tree, the option picked by the software team, and a justification for the choice.

5.1 Implicit Decisions

Security is a non-negotiable attribute for the ngVLA. Parts of the instrument will be separated by public internet infrastructure, and therefore we must consider how we will protect them from external threats. All technological candidates have some consideration for security.

The same can be said of scalability and reliability. The ngVLA necessitates an approach that can meet the needs of a facility almost ten times larger than the VLA, and roughly four times larger than the ALMA array with almost three orders of magnitude more data.

5.2 Standardized or Custom?

Given the cost constraints of the project, as well as the lack of novelty in our requirements, we choose to use an existing standardized technology.

The time allocated to go from concept to working ngVLA prototype does not give the software team much time to invent, refine, and maintain a custom solution. What we are trying to achieve is also not so different from already existing systems in astronomy or other industrial control systems.

Cost is also a concern. Commercial solutions can require paid licenses in some cases. However, custom solutions are not free; they require effort to design, develop, and most importantly maintain for the life of the telescope.

Fortunately, there are a number of existing technologies that are open source or have been recently open sourced which could serve as a technological basis for the ngVLA's monitor and control system. It should be noted that these may or may not be cost-free. There are a number of companies which provide commercial tools built on top of open source technologies. These could prove beneficial in saving us development time, as well as offloading some of the maintenance burden in the future.

Using pre-existing solutions allows us to interoperate and integrate with other third party applications such as human-machine interfaces, development tools, CMMS, and SCADA systems. This could save time beyond the realm of only a monitor and control system, as monitor data and other metrics will need to

be integrated into systems responsible for predictive and reactive maintenance, RFI mitigation, observation planning, etc.

Lastly, a major concern for commissioning and long term maintenance is the ability to support multiple versions of the software at once. In terms of communication protocols, this involves versioning interfaces and messages or adding capabilities to update interfaces through discovery so that they can be properly handled by both new and old consumers and producers. Creating this type of versioning or discovery in custom solutions is often difficult to implement, and can continue to present new difficulties as new edge cases are encountered through real world use. Many mainstream industrial protocols have already solved this problem over many years of development. Simple formats such as HTML and XML support version fields that must be handled by clients, while full libraries like Google Protocol Buffers[9] support extending interfaces in ways that preserves backward compatibility for existing clients.

Typically, the reason that is cited to prefer a custom solution is that in this way we are "shielded" of obsolescence. However, this ignores the high cost of maintain such a solution in house for the lifetime of the instrument. For well adopted protocols the risk is minimal for the 10 year timescale, and for longer timescales we should plan to review and potentially replace system technologies.

5.3 Supervisory or Fieldbus?

What level should the protocol operate at? Or put differently, does the protocol need to be able to support real-time control, or can we "zoom out" to a higher level?

Here, we choose a protocol that sits at a supervisory level.

At present, there are no requirements that dictate the use of a hard real-time protocol above the Hardware Interface Layers of the system in either the correlator or antennas. This is supported by experience from ALMA: while ALMA uses a real-time fieldbus (Controller Area Network), obsolescence projects have been proposed which could render the need for hard real-time communication between the Supervisor and Hardware Controller unnecessary. This puts the supervisory control system in the realm of soft real-time.

Choosing a "supervisory" level protocol also allows us to more easily abstract away the underlying transport mechanisms from the logic required to operate the observatory.

Many fieldbuses require specialized hardware, while most standard higher level protocols ultimately use Ethernet (typically TCP/IP) under the hood. This

[9] https://developers.google.com/protocol-buffers/

allows for a greater range of debugging tools and provides easier debugging for both software and hardware engineers. The lack of custom hardware can also lower the cost of the system.

5.4 Transport protocol (CORBA, REST, MQTT) or Domain-level (for industrial control) Protocol?

Here we choose a domain-level protocol, OPC UA.

At this stage we have narrowed our choices to a much smaller pool of options. The primary choices we have identified fit the criteria discussed thus far: "web" protocols (SOAP, REST, etc) and OPC UA.

This decision centers around the proper level of abstraction. Protocols such as REST provide a basic set of verbs: for example GET, POST, PUT, and DELETE. While these are sufficient for the majority of interactions required in a monitor and control system, they leave for the implementation to decide how to apply these basic operations to the domain problem, something which is otherwise already specified in a protocol at the proper domain level. The key requirement is the seamless integration components from different vendors. For example, combining hardware of a specific vendor and an HMI of a different vendor without incurring in any software development cost. Using a protocol at the transport level of abstraction would require building infrastructure to handle modeling device's properties, operations, and attributes.

On the other hand, OPC UA was designed for industrial control system supporting capabilities like alarms, monitor variables, and state machines. It is extensible and has supported extensions for PLC communication, robotics, and motion controls. OPC UA has the additional benefit that its modeling language is quite similar to SysML. It should be possible to do XML transformations from our project models to OPC UA, providing an automated mechanism to update device parameters.

Another benefit is that OPC UA is independent of transport mechanisms. As of writing, it officially supports a few different transports, including its own binary protocol, SOAP, and REST.

5.5 Have the soft real-time requirements been met?

Use cases such as antenna-to-correlator or correlator-to-archive data streaming may require low latency or low overhead protocols. For these applications, options such as DDS may fill the gap. DDS has been used for many years with success in ALMA for transporting data from the correlator and total power

processors to the archive, and can also be integrated with OPC UA.[10]

5.6 Are there remaining unmet real-time requirements?

In the event that OPC UA and DDS still do not fulfill all of the needs of the system, another technology which could meet tighter timing constraints is Time-Sensitive Networking.[11] TSN provides a mechanism to ensure low latency and deterministic delivery for packets on a LAN between time-synchronized networking devices and endpoints. Using TSN would require special switches.

6 Other Concerns

6.1 Case for a publish/subscribe mechanism

We prefer technologies that support publish/subscribe mechanisms. The reasons for this preference are listed below.

In the industrial control system domain, there are use-cases that are better modeled and implemented based on distributed publishers and subscribers; which publish and subscribe specific data types that dictate their view of the system. For example, a number of GUIs displaying the same antenna position from different consoles, or total digital power measurements from different back-ends.

The concept of a data-bus and client applications tapping onto that bus is a sensible, data-centric architecture for those specific use-cases in the system. Adopting a publish/subscribe mechanism would be useful as much as it is able to simplify the way publishers and subscribers connect to the data-bus, and configuration mechanisms are made available to control the quality-of-service of individual data flows. DDS, an OMG standard introduced in 2004, is specific at facilitating an efficient data-bus and a fine-grained configuration interface based on quality-of-service parameters.

At the same time, it is interesting to note that OPC UA has recently (2019) introduced a publish/subscribe communication scheme to the standard; which adds extra convenience to the adoption of OPC UA for ngVLA. Differences between OPC UA PubSub and DDS implementations that could empower one over the other, are to be investigated as part of the ngVLA system definition effort.

[10] https://www.rti.com/blog/two-approaches-to-integrate-dds-and-opc-ua-for-future-industrial-systems

[11] https://en.wikipedia.org/wiki/Time-Sensitive_Networking

6.2 But what about TANGO?

There is an important distinction between TANGO as a system that integrates transport-level protocols (CORBA, ZeroMQ) and as a SCADA system. At the protocol level, we have opted to use OPC UA. This choice does not exclude using TANGO as a SCADA system. By choosing OPC UA for our device-level protocol, we avoid vendor lock-in and use an industry standard between the device and supervisory layers.

It is also worth noting that at least one prototype antenna vendor used OPC UA in their antenna control unit without any input from NRAO. Their initial bid included adapting this OPC UA control to work with TANGO.

7 What is OPC UA?

OPC UA is not a singular technology, but a standardized service-oriented architecture for industrial control automation. It provides a number of low level services such as discovery, security, platform independence, and interchangeable transport protocols.

OPC UA also has the ability to model data with hierarchical and non-hierarchical relationships that can be navigated and accessed by clients. This model is expressed in a metamodel that resembles SysML and UML, and stored in an XML format. Given that the ngVLA project is using model based systems engineering, it should be possible to convert parts of the system model which express attributes and operations on hardware devices into an OPC UA model. If necessary, the SysML or OPC UA models could be used to automatically generate code stubs for both the supervisor and hardware controller layers, speeding up the rate at which system changes can be rolled out.

The OPC UA modeling capability also allows clients to handle common components in standard ways, for example a user interface element developed for displaying information from a temperature sensor. These common elements simplify developing user interfaces and maintaining consistency across applications. OPC UA also has specifications for interface introspection, allowing us to handle running new software versions in tandem with old versions.

As initially mentioned, OPC UA is not a specific technology, but rather a set of standards used in many industrial applications with facilities that allow extending the set of standards to new applications (robotics, motion control, PLCs). Therefore, many commercial tools can interoperate or integrate with OPC UA systems, opening the door to many COTS tools that could be used by the ngVLA teams, or even allow us to purchase industrial control systems that we could adapt to our needs.

8 Summary

In this memo we explained the decision making process used in selecting one of the core technologies that we will use in the ngVLA monitor and control system: OPC UA. In addition we have identified technologies which could fill potential gaps in the system's required capabilities, DDS and TSN. While this represents an important choice for the project, there are many follow up decisions that will branch out from this one. Having a justified technological basis for these later decisions will allow us to begin developing a prototype system in the next phases of the project.

References

[1] CTSC. Guide to developing cybersecurity programs for nsf science and engineering projects. 2014.

[2] E. Ford et al. Operations concept. 2019.

[3] I. González et al. A literature survey on open platform communications (opc) applied to advanced industrial environments. *Electronics*, 8(5):510, 2019.

[4] K. Hemsley et al. History of industrial control system cyber incidents. 2018.

[5] P. Bordalo et al. Control systems: an application to a high energy physics experiment (COMPASS). *CoRR*, abs/1206.3709, 2012.

[6] National Science Foundation. Major facilities guide. 2019.

[7] R. L. Glass. Frequently forgotten fundamental facts about software engineering. *IEEE Software*, 18(3):112–111, 2001.

[8] T. Glib. Maintainability in software engineering: a quantified approach. 2008.

[9] R. Hiriart. Monitor and control system: Reference design concept. 2019.

[10] M. Manseur. Going faster with continuous delivery. https://aws.amazon.com/builders-library/going-faster-with-continuous-delivery/, 2019.

[11] Netflix. Deploying the netflix api. https://netflixtechblog.com/deploying-the-netflix-api-79b6176cc3f0, 2013.

[12] Netflix. Preparing the netflix api for deployment. https://netflixtechblog.com/preparing-the-netflix-api-for-deployment-786d8f58090d, 2013.

[13] National Institute of Standards and Technology. Framework for improving critical infrastructure cybersecurity. 2018.