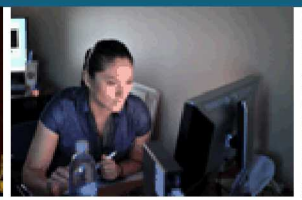
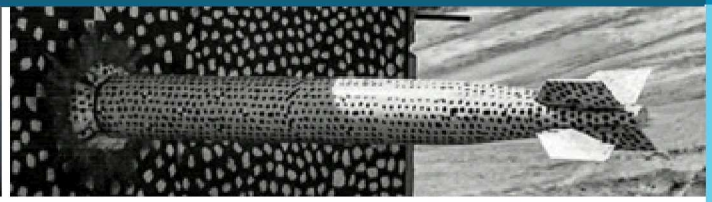
SAND2020-3471PE

GMS Architecture Overview





PRESENTED BY

Ben Hamlet

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

Architecture Approach

Architecture Overview

- Software
- Platform

Key Architecture Updates

- PSC
- Container Platform

GMS Project Pivot

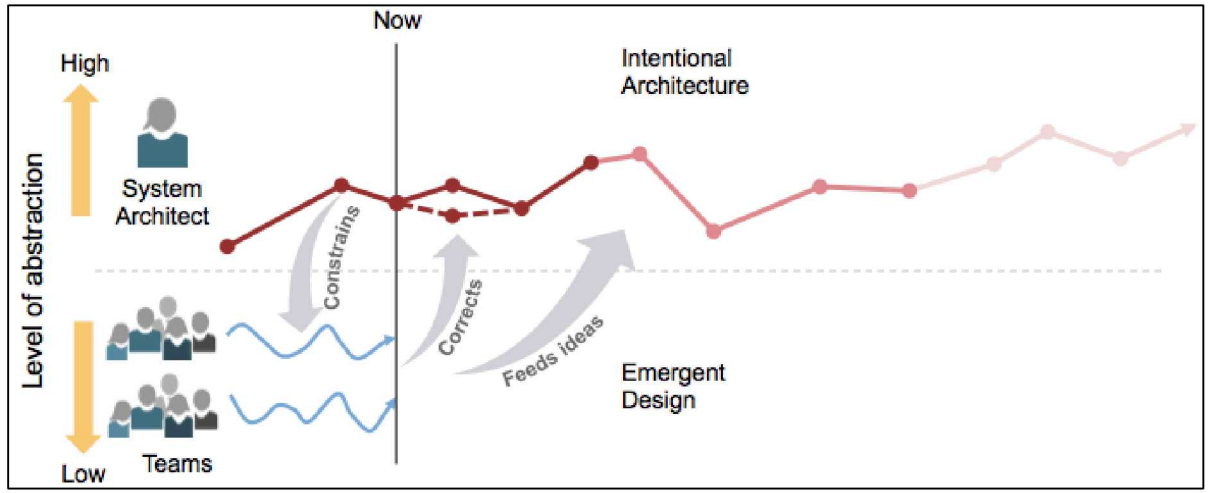
- Pivot Context
- PI 11 SOH Software Architecture
- Bridged Architecture

Architecture Approach

The GMS project is using a tailored process based on the Scaled Agile Framework (SAFe)

GMS Architecture development follows the SAFe approach

- **Intentional Architecture** – Planned initiatives to guide the overall System design across teams & features
  - **Architecture Runway**: components, infrastructure, guidance, etc. needed to develop near-term features
- **Emergent Design** – Development teams drive the design of solutions as part of feature development



The architecture is documented using a small set of views adapted from the *4+1 View Model*[1]

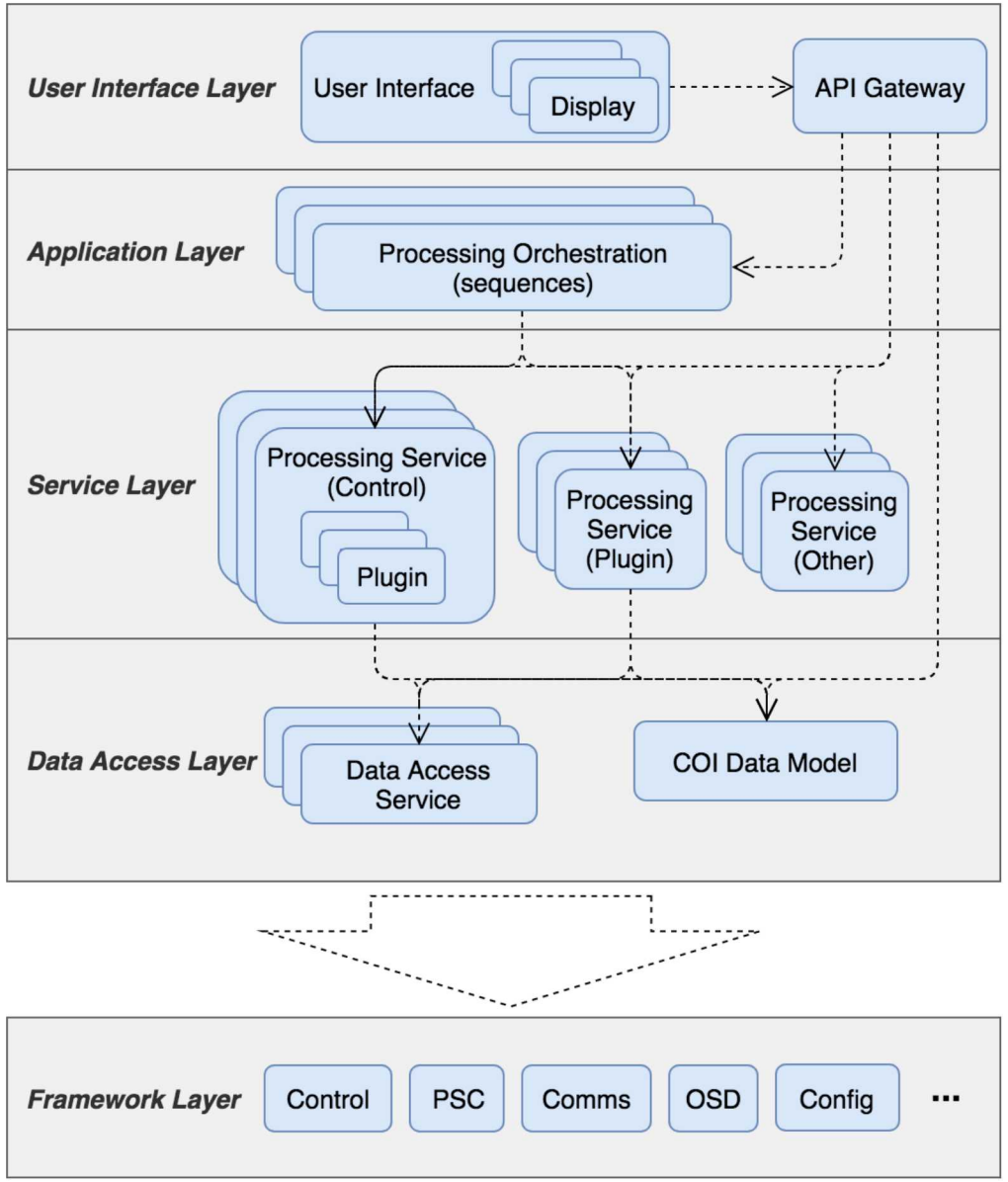
- **Logical View** focused on the high-level functionality provided by the System
- **Deployment View** focused on the design of the physical/virtual platform and deployment of software components
- **Scenario View** consisting of the set of GMS use cases

Architecture Overview - Software

The GMS software architecture can be organized into a set of layers

- **User Interface** – The User Displays, organized into User Interfaces, together with supporting components
- **Application** – The orchestration of services into processing workflows that implement the core mission logic
- **Service** – Software functions supporting automatic processing workflows and User Interface interactions
- **Data Access** – Interface software providing access to persistent data while encapsulating the underlying storage implementation
- **Framework** – Shared project software providing common support functions and implementing standard patterns

Higher layers depend on lower layers



Architecture Overview - Platform

Docker containers provide the standard deployment unit for GMS software components

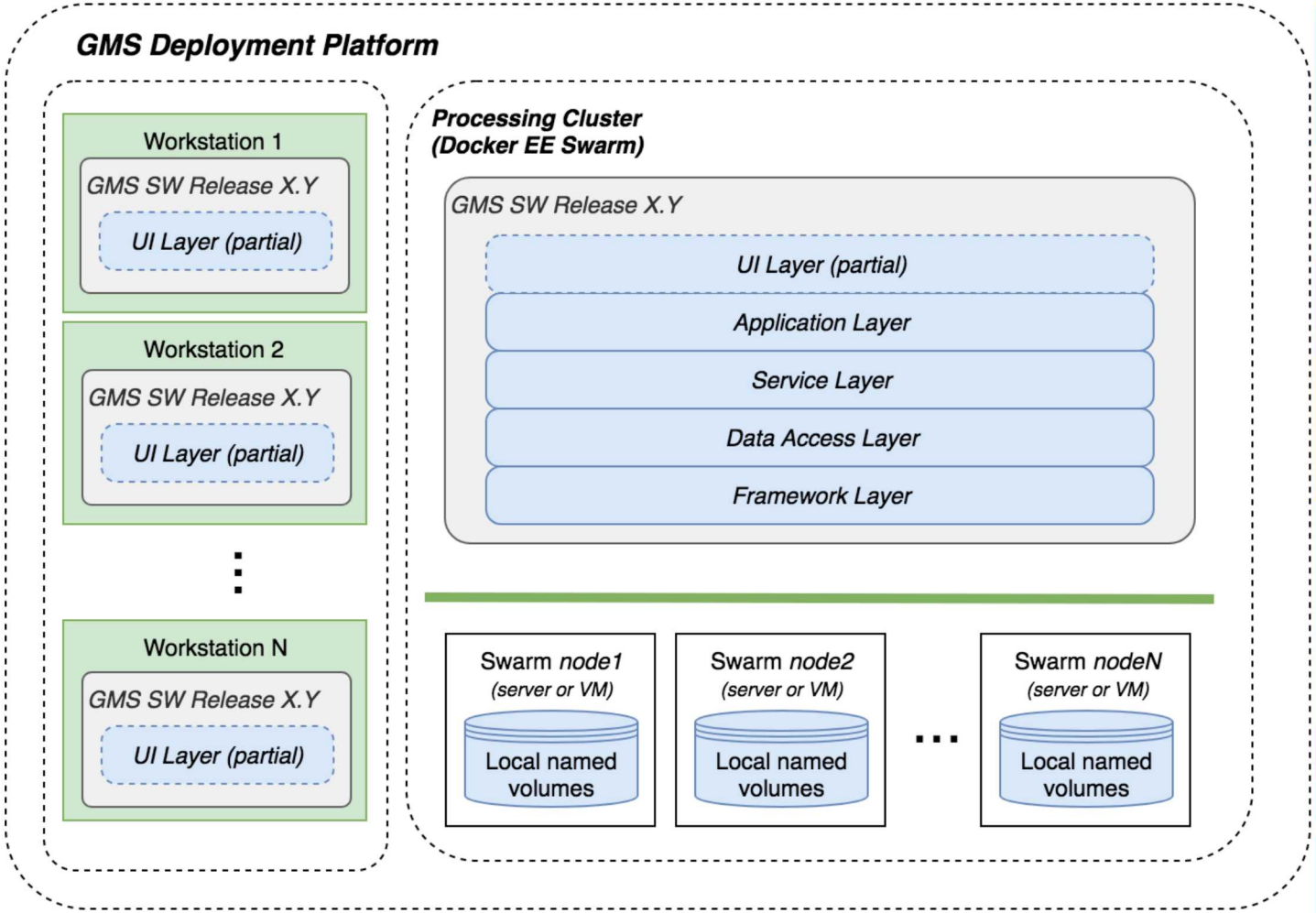
- Docker is a framework for packaging and deploying application components

Docker Enterprise Edition (EE) Compose / Swarm supports definition, deployment, scaling and monitoring of distributed, container-based applications

- Migrating to alternative Kubernetes solution (see Key Architecture Updates)

Each GMS deployment includes two primary elements:

- A Processing Cluster (VM or server-based) built using Docker orchestration platform
- A set of UI workstations



Key Architecture Updates - PSC

The Processing Sequence Controller (PSC) supports orchestration of services into processing flows

- E.g. data acquisition, SOH processing, station processing, network processing

Implemented using Apache NiFi prior to PI 10

- Open-source orchestration framework for automated data processing flows

GMS experience: NiFi complexity has hindered development productivity

- Deployment & scaling via Docker platform proved complex, brittle
- Awkward integration with processing configuration
- Custom version control difficult to integrate

Decision: replace NiFi orchestration with event-based Apache Kafka processing model for SOH delivery

- Processing Services in the *Service Layer* are directly connected into an event-driven pipeline via Kafka messaging middleware (see SOH Architecture slide)

Upcoming architecture runway work will evaluate alternatives addressing PSC requirements (including custom solution)

Key Architecture Updates – UI Performance

Under the current architecture, the GMS user interface software interacts closely with a UI backend component to support user interactions

- The UI backend provides a number of functions, including backend service routing & API consolidation, protocol conversion, data caching, session management, undo/redo state management, etc.

This approach introduces latency due to network communication between the UI client and backend, impacting GMS display responsiveness

There is an active architecture runway effort investigating UI architecture updates to address this issue

Updates under consideration:

- Migration of UI backend functions to the UI client
  - Data caching & session management, undo/redo state management
- Migration of select processing logic from backend services to the UI client
  - E.g. Filtering, beaming, FK
- Transition of the UI backend to a lightweight proxy providing backend service routing and protocol conversion only

Key Architecture Updates – Container Platform

GMS currently uses Docker EE for container deployment and orchestration

Security concerns with Docker EE are driving the project to an alternative Kubernetes-based solution

- Aligns with AFTAC enterprise platform selection

Near-term architecture runway work required to select a Kubernetes distribution

Future platform effort required to migrate the GMS CI/CD container build/deployment from Docker EE to the selected Kubernetes distribution

GMS Project Pivot

Pivot Context

Starting in PI 10, the GMS project pivoted to a new approach emphasizing operational deployment of capabilities working with the legacy system

Initial roadmap priorities:

- Station SOH monitoring capability
  - Initial capability delivery: PI 11 release
- Incremental replacement of interactive tools
  - Development starting PI 12

Architecture Effects:

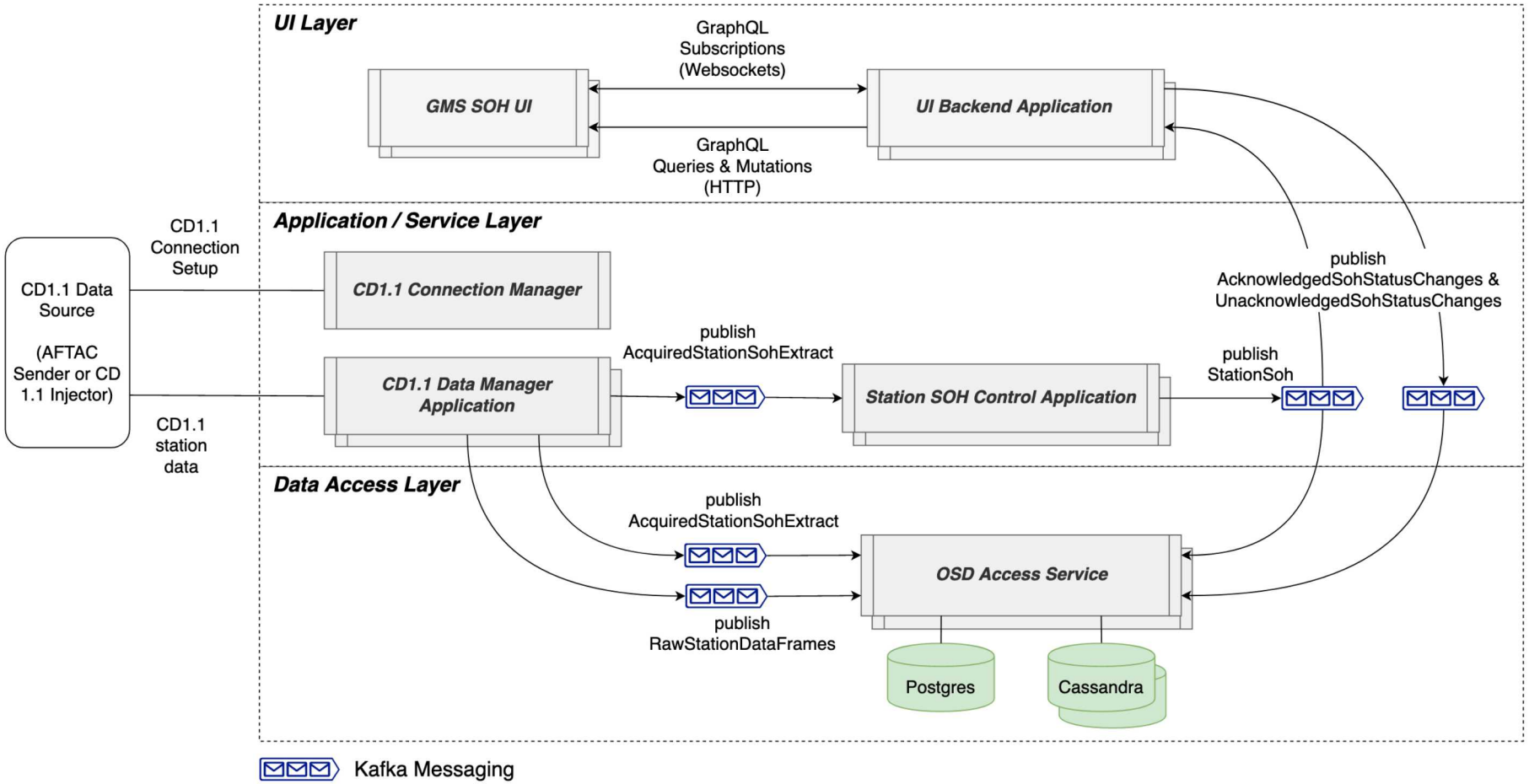
- Increased architecture runway focus on robustness, performance, and scalability, driven by near-term use in operations
- Introduction of "bridge" middleware integrating GMS components with the existing US NDC System

PI 11 SOH Software Architecture

Initial SOH delivery supports real-time SOH monitoring for stations received via CD-1.1

Acquisition/SOH processing pipeline composed of services connected via Kafka messaging middleware

UI updates and OSD storage of SOH data via Kafka subscriptions



Bridged Architecture (1 of 4)

Bridge middleware will provide interoperability between the existing US NDC System and GMS

Two bridge components:

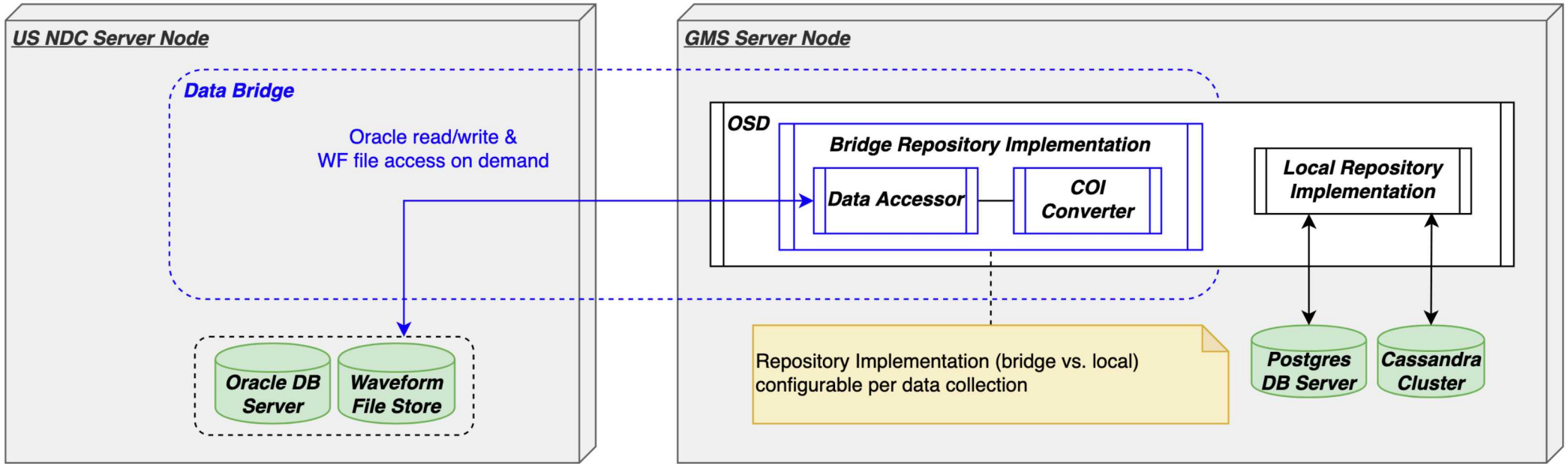
1. *Data Bridge* – provides GMS with read and write access to data stored in the existing US NDC databases (Oracle and waveform file store) via the OSD framework
2. *UI Bridge* – provides the existing US NDC interactive tools with access to GMS displays via ICD-controlled messaging interface

IDC interoperability feasible via custom implementation of the bridge design

Bridged Architecture (2 of 4)

Data Bridge - Integration of existing US NDC databases with GMS Object Storage & Distribution (OSD) framework

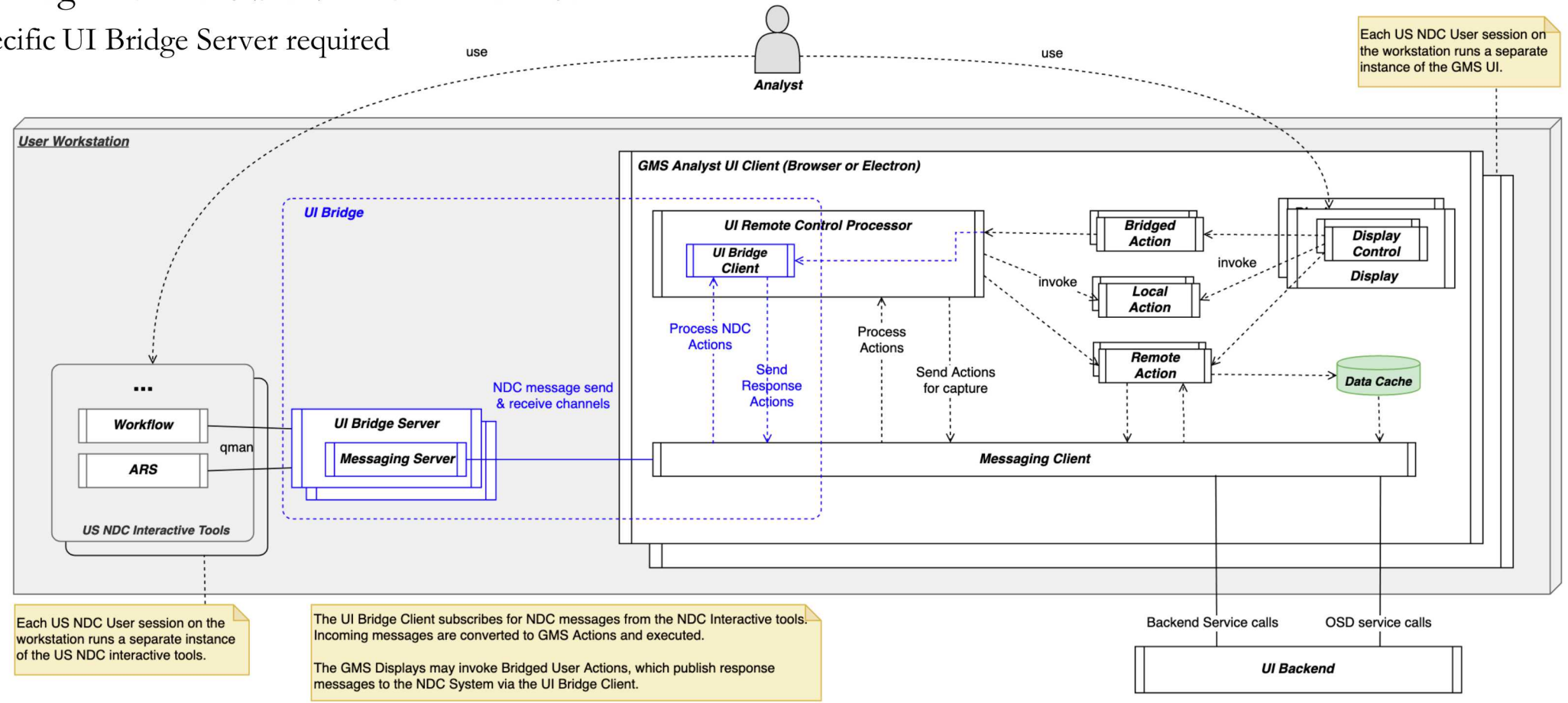
- Enables interactive analysis via GMS displays based on raw data and processing results stored in US NDC databases
- Concept: US NDC Oracle DB and waveform file store serve as an alternative storage backend for the GMS OSD
  - Alternative to local Postgres and Cassandra storage backends configurable by data collection (e.g. waveform data, signal detection data)
  - *GMS Service*, *Application* and *User Interface* Layers access data persistence via the existing OSD Repository interface
- IDC database integration feasible via custom Bridge Repository Implementation



Bridged Architecture (3 of 4)

User Interface Bridge - Integration of existing US NDC interactive tools with GMS Analyst UI displays

- Enables interactive analysis via existing tools with select functions delegated to GMS displays (e.g. FK, amplitude measurement)
- Concept: GMS Analyst UI displays managed from existing interactive tools via ICD-controlled messaging interface
- IDC UI integration feasible via ICD interface
  - IDC-specific UI Bridge Server required



Bridged Architecture (4 of 4)

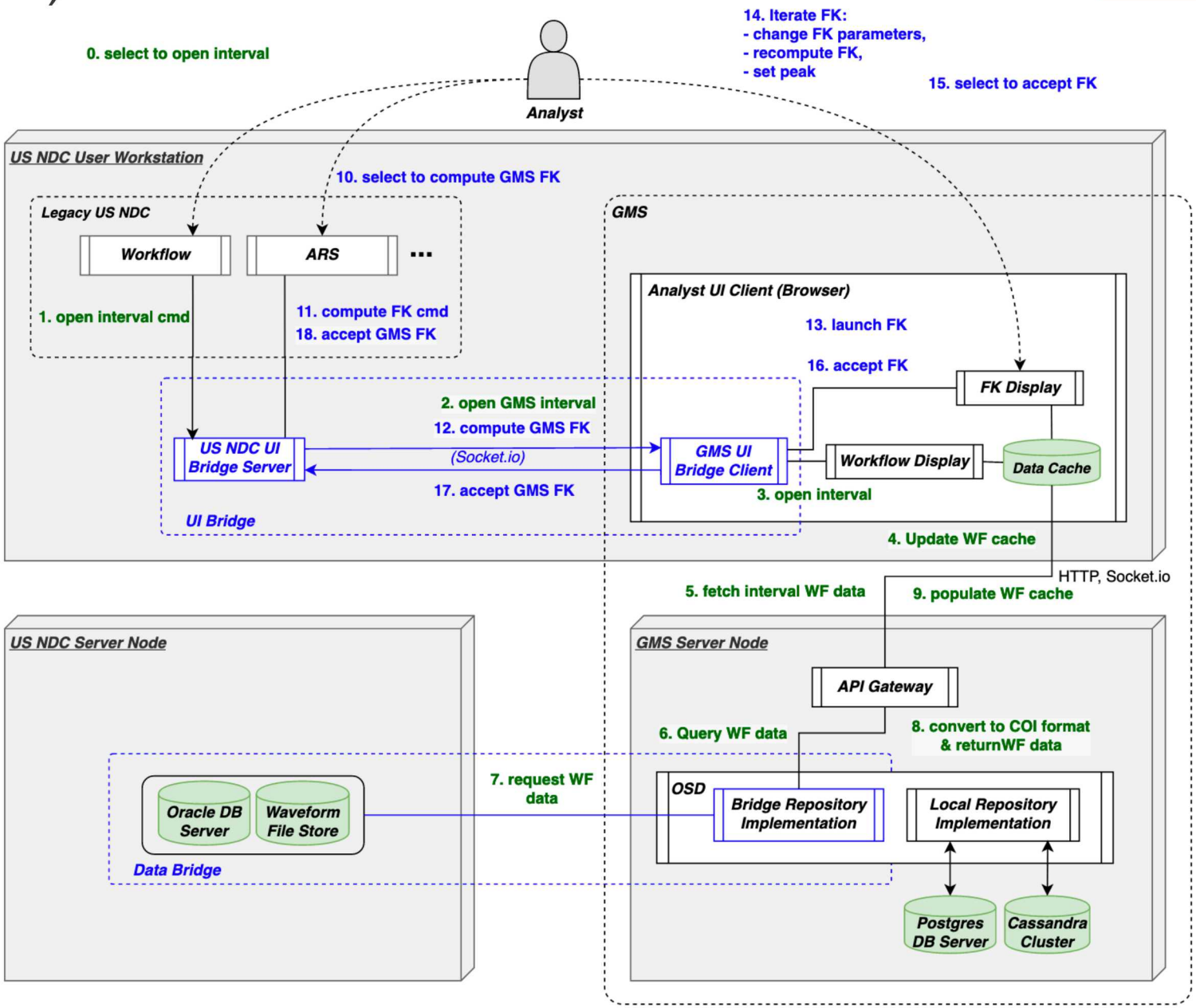
Bridge Example: FK

Open Interval

- **Steps 0-3** The Analyst selects to open an interval via the US NDC Workflow display, which sends a request to GMS via the UI bridge to open the interval
- **Steps 4-6** The GMS UI opens the interval, requesting waveform data from the OSD
- **Steps 7-8** The OSD retrieves the requested waveform data from the US NDC data stores via the data bridge & converts to COI format
- **Step 9** The UI caches the waveform data for the open interval

Compute FK

- **precondition** The interval containing the input raw waveform data is open (waveform data cached in the GMS UI client)
- **Steps 10-13** The Analyst selects to compute an FK via the ARS display, which sends a request (with waveform claimchecks) to GMS via the UI bridge to compute & display the FK
- **Step 14** The Analyst iterates on the FK, changing parameters & re-computing the FK via the GMS display
- **Step 15** The Analyst accepts the FK via the GMS display
- **Steps 16-18** The GMS UI sends the accepted FK results to ARS via the UI bridge



Questions

References

[1] Kruchten, Philippe (1995, November). Architectural Blueprints — The "4+1" View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50.