Exceptional service in the national interest

Sandia National Laboratories

SAND2016-5759PE

IDC Reengineering Phase 2

E2 Prototyping Overview

Ryan Prescott

20 June 2015

SAND Number:

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Agenda

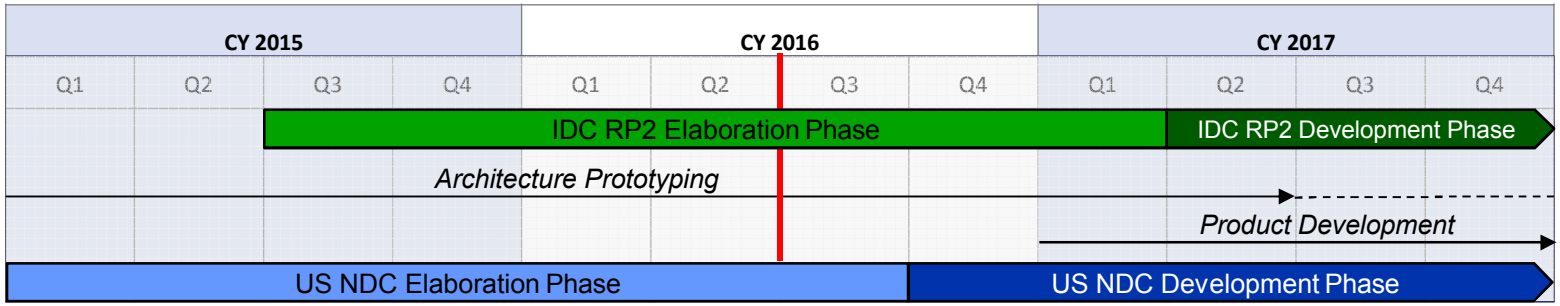
- Prototyping Overview
- Timeline
- Current Prototyping Assumptions
- Conceptual Overview
- E2 Focus Areas
  - Web UI
  - Data Services
  - Data Provenance
  - Development Tooling
- E3 Path Forward

Prototyping Overview

- The US NDC Modernization project plan includes a software prototyping component supporting definition of the system architecture
- Prototyping is intended to facilitate:
  - Definition of high-level design patterns
  - Demonstration of key architecture concepts & features
  - Selection of representative technologies
    - System platform
    - Software languages
    - Third-party software

Timeline

- Architecture Prototyping
  - Focused on core design patterns and technologies
  - Software language selection and third-party software evaluations for key software mechanisms
  - Reduced emphasis starting Q3 FY 2017 with transition to production development
  - Ongoing architecture prototyping as needed throughout the Development phase
- Production Development
  - Beginning Q1 2017
  - Early focus:
    - Establish production development envt./tools, including CI, testbeds, etc.
    - Core SW infrastructure development
    - Initial domain application development



Current Prototyping Assumptions – Platform & Languages

Sandia National Laboratories

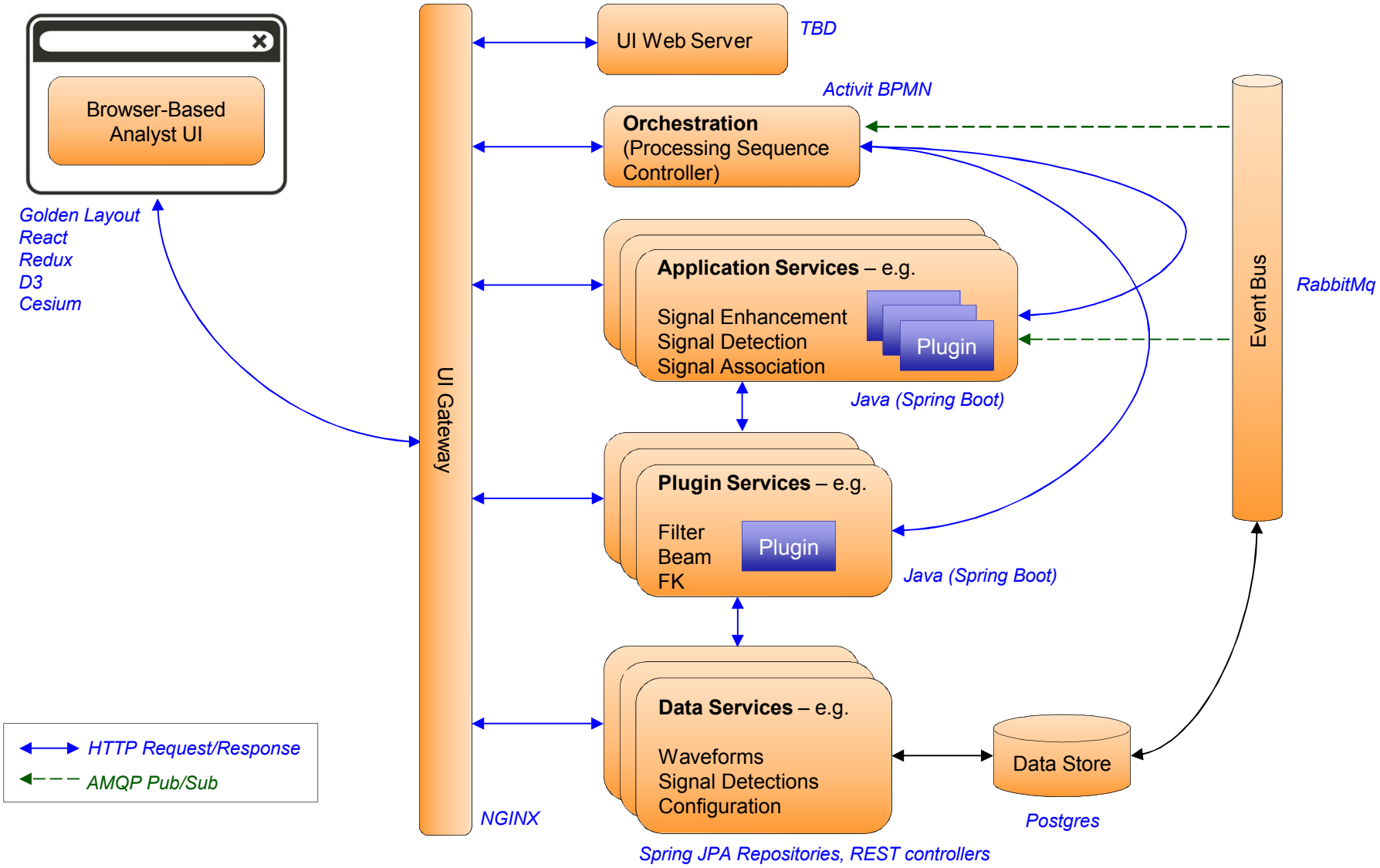
- **Platform**:
  - Infrastructure:
    - Deployment: VMs + containers on dedicated physical hosts, custom cloud (e.g. OpenStack), or possibly commodity cloud (e.g. AWS)
  - OS: Centos 7+ (current) / RHEL 7+ (future)
- **Storage architecture**:
  - Current: RDBMS (Postgres)
  - Future: Likely will include other technologies for waveform storage, provenance support
- **Software Development Languages**:
  - User Interface: Web UI stack (.js, css, html), Java RCP as fallback
  - Application Control & Orchestration: Java
  - Domain Services:
    - Multiple supported (C++, Java, Python)
    - Only Java demonstrated to date
  - Data Access:
    - Code API: Java (C++ TBD)
    - REST API: Multiple supported (C++, Python, Java, etc.), Java included in prototype

Current Prototyping Assumptions - COTS

Sandia National Laboratories

- **Web-Based User Interface**
  - Framework: Golden Layout, React, Redux, D3, …
  - Map displays: TBD (currently Cesium)
- **Fallback UI approach: Java rich client**
    - Netbeans RCP, JavaFX charts, OpenGL
- **Processing Sequence Controller mechanism (PSC)**
  - Current: Activiti BPMN Engine
- **Inter-process Communications**
  - Data Distribution (pub/sub): RabbitMQ / JSON
  - Service Invocation (request/response): HTTP(S) / JSON (REST)
- **Data Access (SCRUD)**
  - Code API - Java: Spring JPA (Hibernate), C++ (support TBD)
  - REST API: Spring Data REST, Spring Web + JPA

Prototype Conceptual Overview



E2 Focus Areas Overview

Sandia National Laboratories

- Web UI
  - Golden Layout
  - Continuous FK
  - Undo/Redo
- Data Services
  - REST Repositories
  - REST Controllers
- Data Provenance
  - Event Sourcing + CQRS
- Development Tooling
  - Continuous Integration
  - VM Packaging & Deployment

Sandia National Laboratories

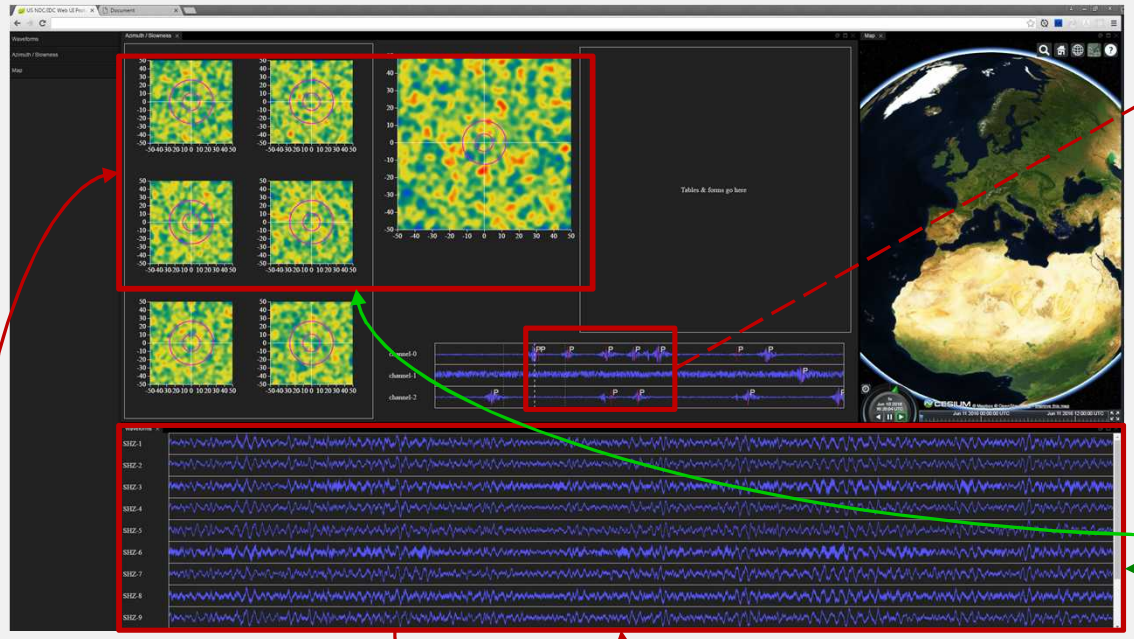
WEB UI

Web UI Prototype Overview

Sandia National Laboratories

- Three primary efforts in E2:
  1. Golden Layout
     - Goal: Evaluate Golden Layout as an alternative to OWF for display window management
     - Status: Replaced OWF with Golden Layout in current prototype
  2. Continuous FK
     - Goal: Demonstrate responsive displays for a computationally-intensive processing function as part of the Web vs. Java thick client UI trade
     - Status:
       - Developed an initial prototype for the 'Continuous FK' feature
       - Performance assessment and optimization is ongoing
  3. Basic Undo/Redo
     - Goal: Demonstrate an initial, global-history undo/redo capability to better understand the web UI solution space
     - Status: Developed a browser-side prototype using Redux undo

Web UI Prototype Design



- User slides a detection window through time, showing change in FK heat map
- User can resize the detection window

Undo/Redo Stack Display (Not Shown)

Publish state changes

Global UI State Cache

Waveforms
Signal Detections
FK Results
Undo/Redo actions

Populate Waveforms from Data Service

Browser (Workstation)

Generate FK Via Service

Store Signal Detections Via Data Service

Waveform Data Services

FK Plugin Service

FK Plugin

Signal Detection Data Services

Data Store

Postgres

Java (Spring Boot)

Spring JPA Repositories, REST controllers

Application (Server)

Golden Layout

Sandia National Laboratories

- Golden Layout is a popular OSS alternative to Ozone Widget Framework (OWF), providing similar features
  - User customizable browser display layouts
    - Save/load configurations
    - Manage display components
  - Data bus for synchronizing display data across components
- Impressions
  - Easier to integrate than OWF - does not require components to use specific patterns & SW
  - Supports display component ‘tear off’ into separate browser instance, with continuity of data synchronization
    - Greater flexibility for display configuration across multiple screens

Continuous FK

Sandia National Laboratories

- Previously investigated options for translating C/C++ & Java into JavaScript for browser-local invocation of e.g. FK functions
  - Looked at GWT & Emscripten
  - Impression: complex solution better suited to standalone applications than individual components of a system
- Prototyped service-based FK computation as an alternative
  - Multi-threaded FK service for a given window; parallel HTTP service invocations from UI
  - Multi-threaded batch FK service for a given window, time range and step size; single service call from UI
  - Pre-fetch optimization where possible
- Follow-on work
  - Ongoing performance assessment, including simulation of remote performance
  - Further prototyping of optimization options

Undo/Redo

Sandia National Laboratories

- Integrated the Redux framework to provide a shared, global state cache across components of the browser-based UI
- Implemented a global undo/redo history with associated display component using the Redux Undo framework
  - User support for undo & redo of most recent action as well as those farther back in the history
  - Limited action set included in the prototype: signal detection creation, phase assignment, re-timing, FK-based azimuth/slowness feature measurement
- Follow-on work
  - Extend the prototype to support multiple undo/redo histories, e.g. per event and for unassociated signal detections
  - Investigate design tradeoffs for state restoration vs. re-execution
  - Investigate server-side state management use cases

Sandia National Laboratories

DATA SERVICES

Data Services Prototype Overview

Sandia National Laboratories

- Prototyped two approaches for providing HTTP/JSON SCRUD interfaces to persistent data:
  1. REST repositories (Spring Data REST + JPA)
     - Supports the HATEOAS REST constraint (Hypertext As The Engine Of Application State)
       - Associations between data model entities are discoverable and traversable via Hypertext Application Language (HAL) links embedded in data response
     - Automatic generation of REST interfaces for all data entities
     - Does not support cascade of updates to associated data entities
       - Separate update service calls required for all associated entities that are modified
  2. REST controllers exposing JPA generated APIs (Spring Web + JPA)
     - Enables server-side cascade of updates to associated data entities
     - Does not provide built-in support for the HATEOAS
     - Requires custom controller software; no auto-generated interfaces

Data Services Prototype Overview

Sandia National Laboratories

- The two approaches offer advantages for different use cases
  1. REST repositories are well-suited for data access (query, read)
     - HAL link traversal is an appealing solution for data associations
  2. JPA REST controllers well-suited for data updates (create, update, delete)
     - Server-side cascade of updates to associated entities is efficient and provides a simpler interface without the need for aggregated service interfaces
- Follow-on work
  - Select a design for further prototyping
  - A hybrid design combing both approaches may be viable
    - Under an event sourcing/CQRS design, the System separate read and write interfaces (and data models)
    - See the Data Provenance section for information

Sandia National Laboratories

DATA PROVENANCE

Motivation

- The IDC System Specification includes requirements to preserve the history of the System state
- Motivated by the desire to track the provenance of information
  - *SSD 1926: The System shall provide the Analyst the capability to view the complete history of an event*
  - *SSD 5736: The System shall maintain a history of the system configurations*
  - *SSD 2134: The System shall store raw waveform data availabilities for specific points in the processing history*

Event Sourcing

Sandia National Laboratories

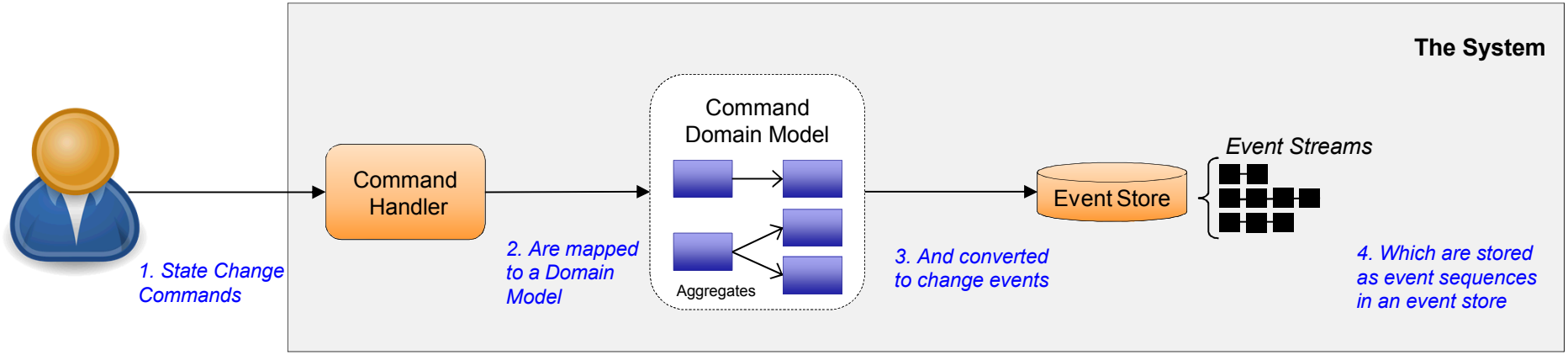
- Event Sourcing (ES) is an architectural approach that seeks to address this type of requirement
  - Generalized solution to historical state tracking
- Typically seen in industries where:
  - Rigorous audit logging is a critical requirement &/or
  - State is most naturally defined as the sum of a sequence of events
- E.g. Banking, stock trading

Event Sourcing

Sandia National Laboratories

- How it works
  - Every change in System state is captured as an event
  - Events are persisted as sequences in an event store (“streams”)
  - Current state is accessed by applying the sequence of events
  - Snapshots provide an optimization, reducing the number of events that must be applied to produce the System’s current state
- There are many good resources describing Event Sourcing
  - Martin Fowler
  - Greg Young
  - More Fowler

Event Sourcing Conceptual Design



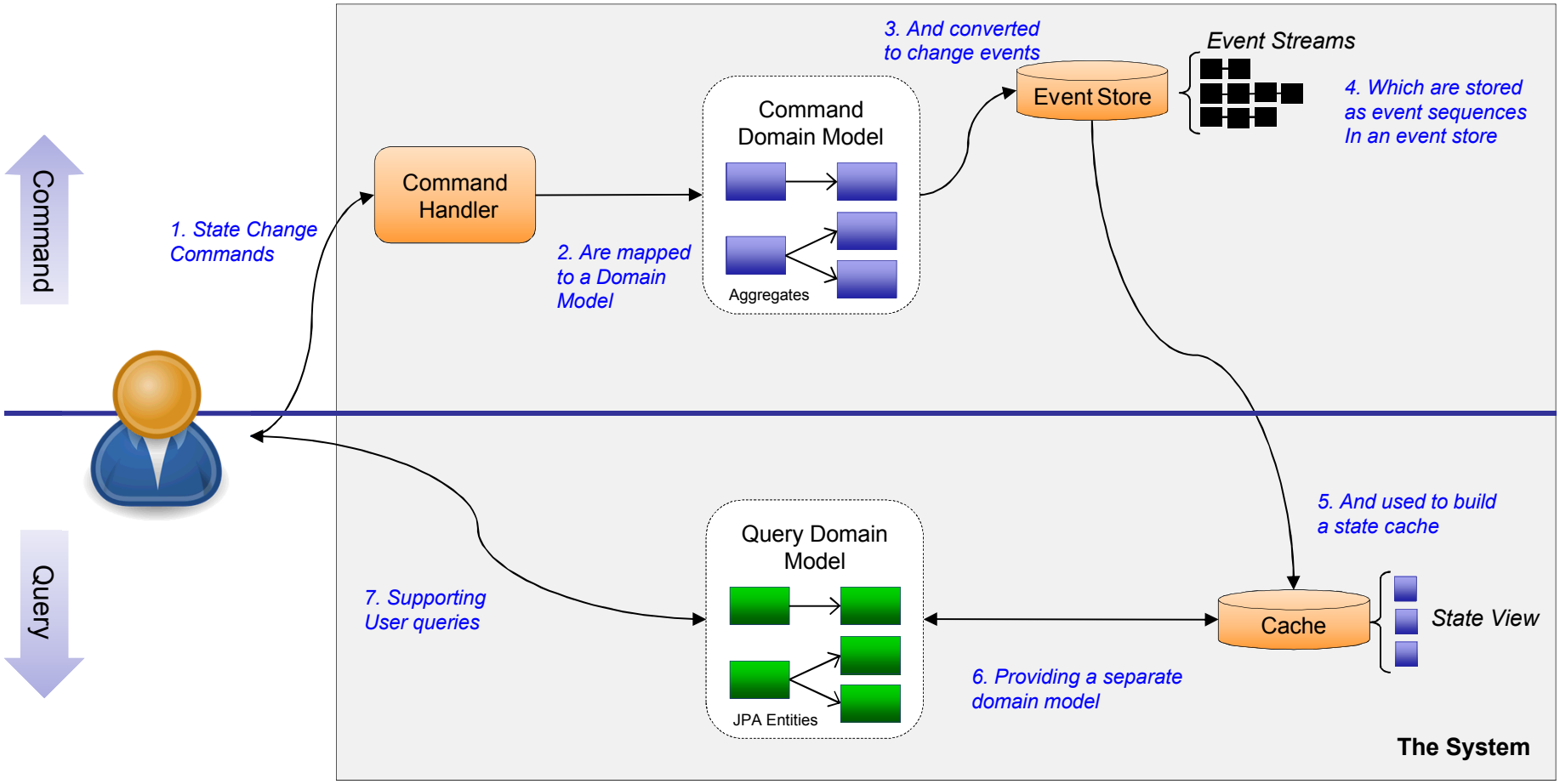
Command Query Responsibility Segregation (CQRS)

Sandia National Laboratories

- Supporting queries is a problem for Event Sourcing
  - The Event Log does not provide a view into current state
  - Replaying the Event Log in for each query likely is not practical

- CQRS addresses this problem
  - An architecture pattern where the System uses separate models for modifying and querying state
  - In an event sourcing system, a separate cache or DB instance provides a view of System state for query purposes that is populated from the stream of events published via an event bus
  - There are many good resources describing CQRS
    - Martin Fowler

Event Sourcing + CQRS Conceptual Design

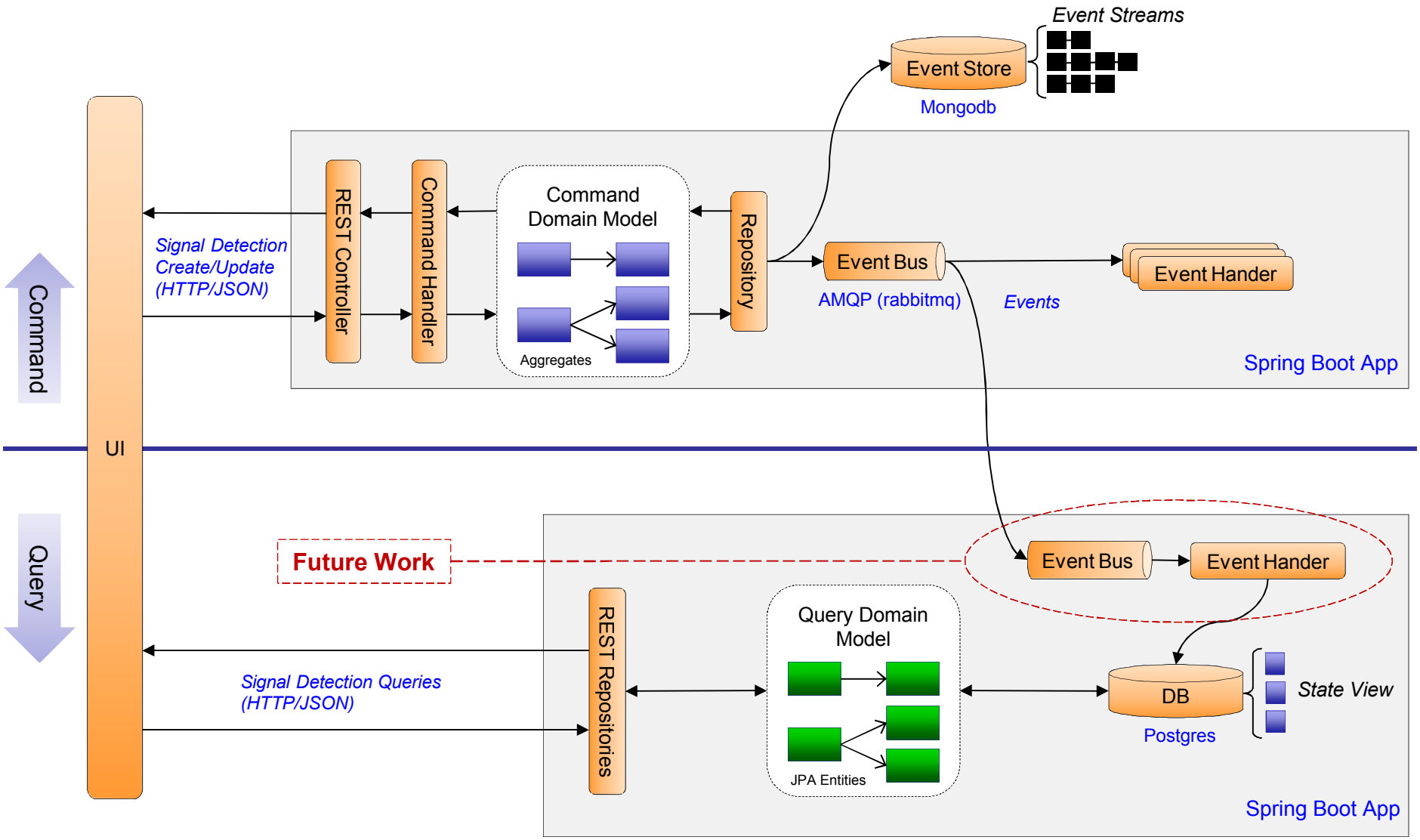


Event Sourcing + CQRS Prototype

Sandia National Laboratories

- Developed an initial ES + CQRS prototype in E2
- Goals:
  - Investigate technologies
  - Investigate data model implications
  - Demonstrate ES design for signal detections
- Selected Technologies
  - Axon framework (Java ES + CQRS framework)
  - Spring Boot (Java application development framework)
  - RabbitMQ (event bus implementation supporting distributed, polyglot system)
  - MongoDB (event store implementation)

Event Sourcing + CQRS Prototype Design



Findings

- ES + CQRS
  - Appears to be a viable approach to recording state history in the reengineered system
  - May support some or all replay requirements
  - Adds complexity to the architecture
    - Separate data models, separate event store and state DB, snapshotting, playback)
- There are few OSS frameworks providing ES/CQRS support
  - Does not appear to be a widely-used pattern
- Axon
  - Is fairly easy to work with, although there are few code examples to draw on
  - Is under active development
    - Redis and Cassandra event store implementations
    - Kafka event bus implementation (TBD)

Follow-On Work

- Elaborate the command-side data model to identify the events and aggregates
- Estimate scaling needs of the event store based on predicted network size
- Prototype a query-side database populated from the event bus, using the existing data model as a starting point
- Characterize performance for more realistic signal detection loads
  - Command processing & event dispatching

Sandia National Laboratories

DEVELOPMENT TOOLING

Development Tooling Prototype Overview

Sandia National Laboratories

- E2 included early work to establish a production-quality development tool set anticipating the transition to production development
- Two focus areas:
  1. Continuous Integration (CI)
  2. Virtual Machine (VM) packaging & deployment

Continuous Integration

Sandia National Laboratories

- Looked at a few well-known CI tools (Jenkins, Travis CI, Gitlab CI)
- Developed a CI pipeline prototype using Gitlab CI
  - Automatic build, unit test execution, service API documentation generation
    - Upon git push of any branch
    - Upon acceptance of merge request into the 'develop' branch
  - Executed on commodity Centos 7 VMs using docker containers
    - Minimizes host VM dependencies on third-party software
- Follow-on work:
  - Add code formatting, javadoc generation and static analysis (Sonar) to the pipeline
  - Add web UI third-party software libraries (NPM) to the build repository

VM Packaging & Deployment

Sandia National Laboratories

- Implemented RPM packaging of third-party development tools and initial scripts to build development environment VMs
  - Separate VM images targeting individual development environments (VirtualBox) and shared testbed environment (OpenStack)
- Investigated tools for automated configuration and deployment of VM images across environments (VirtualBox, OpenStack)
  - Evaluated Puppet and Ansible
- Follow-on work:
  - Prototype automated VM configuration/deployment (likely with Ansible)
  - Ultimate goal: automated, *dynamic* VM provisioning for CI pipeline, virtual testbeds

E3 Path Forward

Sandia National Laboratories

- Web UI
  - Further prototyping and performance assessment of key displays supported by services
    - Include remote UI performance assessment
  - Elaboration of the undo/redo prototype
    - Separate undo/redo histories e.g. per event and for unassociated signal detections
- Data Services
  - Select a design path forward reconciling the two approaches
  - Elaborate the prototype based on the design selection
- Data Provenance
  - Consider alternatives and select a path forward
    - Consider data scale implications of ES
- Development Tooling
  - Create VM configuration/deployment tooling to support development, CI and testbed deployment
- Other Focus Areas
  - Continue elaboration of the data model