Exceptional service in the national interest

Sandia National Laboratories

SAND2015-4822PE

IDC Reengineering Phase 2

Prototyping Status

Ryan Prescott

22 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

Sandia National Laboratories

- Prototyping Overview
- Timeline
- Executable Architecture Prototype
  - Definition
  - Conceptual Overview
  - Assumptions
  - Current Technologies
  - Key Features
  - User Scenario
- Current Prototype Status
  - Overview
  - User Interface Prototyping Status
    - Netbeans Display Prototype
    - OWF Display Prototype
  - Application Control Prototyping Status
  - Data Access Prototyping Status
- Next Iteration Goals
- Backup

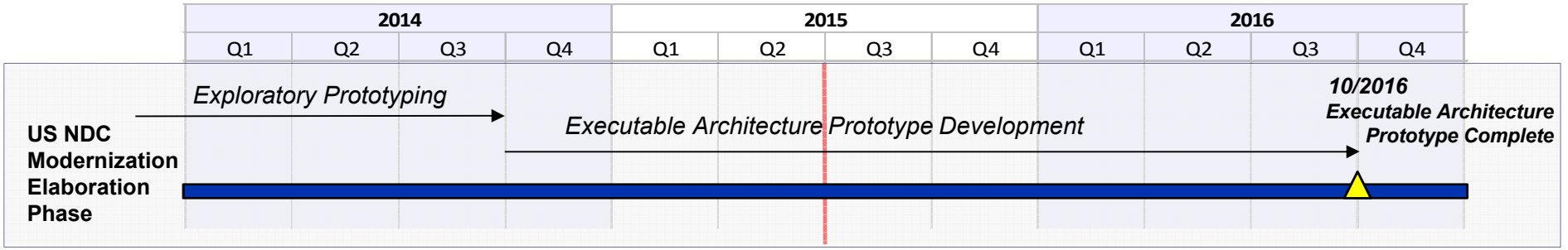
Prototyping Overview (1 of 2)

Sandia National Laboratories

- 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

- The project plan includes two prototyping phases:
  - Exploratory Prototyping, FY2014
    - Focused on technology evaluation
    - Software language selection and third-party software evaluations for key software mechanisms
    - See backup slides for technology evaluation summary
  - Executable Architecture Prototyping, FY2015-FY2016
    - Focused on demonstration of key system features and mechanisms
    - Following a SCRUM process with 3-week sprints



Sandia National Laboratories

EXECUTABLE ARCHITECTURE PROTOTYPE

Agenda

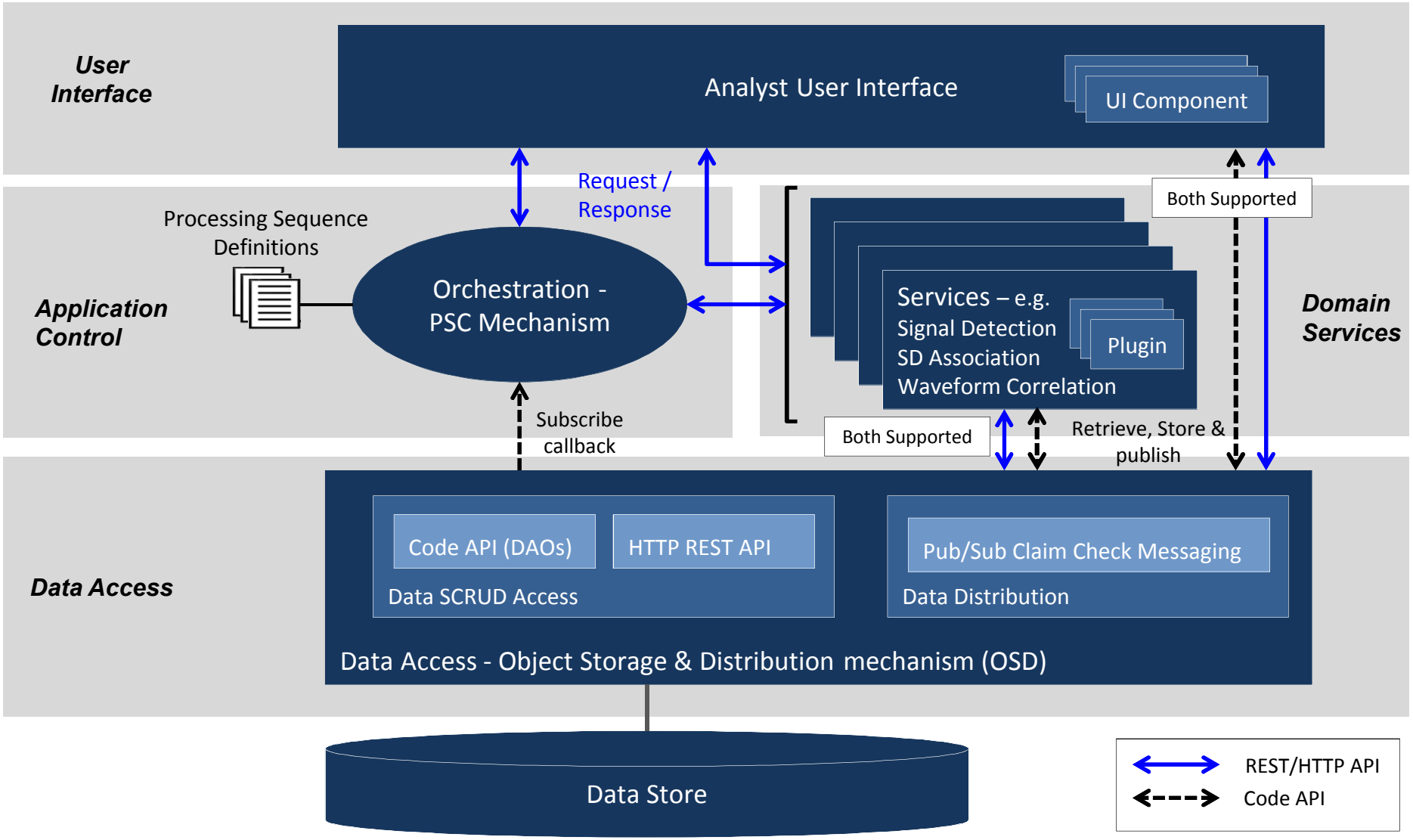
- Prototyping Overview
- Timeline
- Executable Architecture Prototype
  - Definition
  - Conceptual Overview
  - Assumptions
  - Current Technologies
  - Key Features
  - User Scenario
- Current Prototype Status
  - Overview
  - User Interface Prototyping Status
    - Netbeans Display Prototype
    - OWF Display Prototype
  - Application Control Prototyping Status
  - Data Access Prototyping Status
- Next Iteration Goals
- Backup

Executable Architecture Prototype – Definition

Sandia National Laboratories

- Elaboration-phase activity to implement a portion of the system architecture as defined in the Architecture Document and Analysis Model
- Determine if it is feasible to implement the architecture
  - Must satisfy system requirements
  - Feedback loop to update architecture when needed
- Executable Architecture is a prototype
  - The intent is to validate key features & mechanisms of the architecture rather than to develop an early version of the system

Executable Architecture Prototype – Conceptual Overview



Executable Architecture Prototype – Assumptions (1 of 3: platform and languages)

Sandia National Laboratories

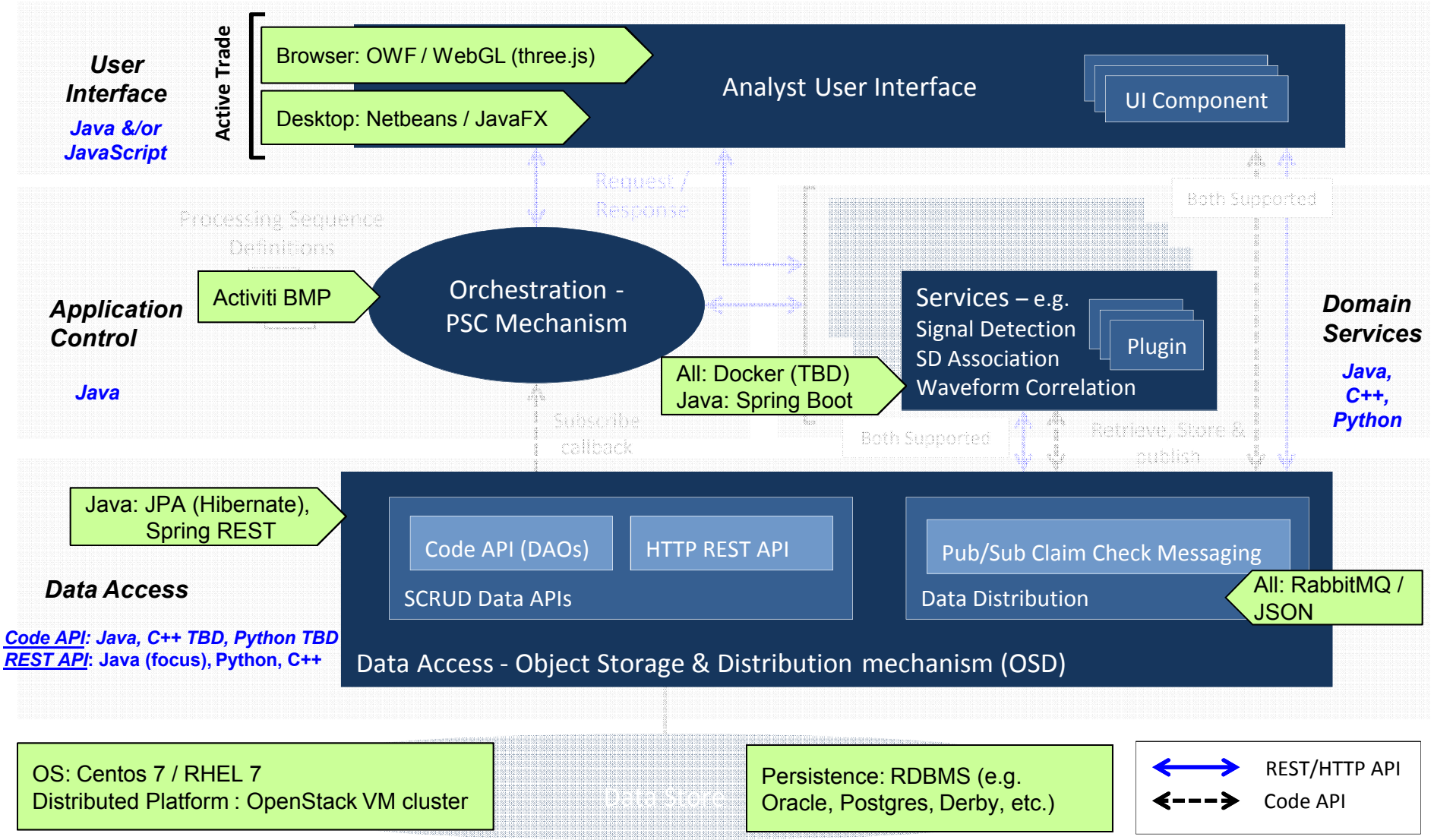
- **Platform**:
  - Infrastructure:
    - Distributed deployment: bare metal, private cloud (e.g. OpenStack), possibly Cloud
  - OS: Centos 7+ (current) / RHEL 7+ (future)
- **Storage architecture**: RDBMS (Oracle, Postgres, etc.)
- **Software Development Languages**:
  - User Interface: Java &/or JavaScript (open trade re: desktop vs. browser-based UI)
  - Application Control & Orchestration: Java
  - Domain Services:
    - Multiple supported (C++, Java, Python)
    - Only Java demonstrated
  - Data Access:
    - Code API: Java (C++ TBD)
    - REST API: Multiple supported (C++, Python, Java, etc.), Java included in prototype

Executable Architecture Prototype – Assumptions (2 of 3: third-party software)

Sandia National Laboratories

- **User Interface**:
  - Browser-Based
    - Framework: Ozone Widget Framework
    - Waveform displays: OpenGL (three.js)
    - Map displays: TBD (Worldwind, Google Maps)
  - Desktop
    - Framework: Netbeans RCP
    - Waveform displays: JavaFX charts, OpenGL (TBD)
    - Map displays: TBD (Worldwind, Google Maps)
- **Processing Sequence Controller mechanism (PSC)**: Activiti BPMN Engine
- **Domain Service Encapsulation**: docker (TBD - see backup for docker overview)
- **Inter-process Communications:**
  - Data Distribution (pub/sub): RabbitMQ / JSON
  - Service Invocation (request/response): HTTP(S) / JSON (REST)
    - Java: Spring REST
- **Data Access (SCRUD)**:
  - Code API - Java: JPA (Hibernate), C++ (support TBD)
  - REST API: Multiple clients supported Spring REST included in prototype
- **Data Access (Distribution): RabbitMQ / JSON**

Executable Architecture Prototype – Assumptions (3 of 3: Technology Summary)



Executable Architecture Prototype – Key Features

Sandia National Laboratories

- Implemented features demonstrate fundamental concepts
  - PSC: processing sequence definition, execution, control
  - OSD/COI: data models, persistence, data distribution, data provenance
  - UI: modern frameworks, extensible, undo/redo, OSD/PSC integration
- Features demonstrate high risk architectural aspects and non-functional ("-ility") SRDs
  - Usability
    - Low-latency, high-performance analysis user interfaces
    - Show undo/redo, data synchronization, user customizable displays
  - Data Provenance
    - Tracking and preserving data availability, processing parameters, processing histories, and the users/processes who worked on all results
  - Configurability
    - Processing parameters configurable by station, phase, etc.
    - Processing sequences initiated based on configurable criteria
  - Maintainability / Extensibility
    - Creating data abstraction layers and pluggable algorithm implementation patterns
  - Deployment
    - A variety of deployments are required. Will demonstrate a data center deployment supporting local and remote interactive analysis.
- **Approach: Demonstrate requirements through the implementation of select user scenarios**

User Scenario (1/3) – Interactive Analysis

Sandia National Laboratories

- Analyst selects data (time interval or selected event)
- Views waveforms, signal detections, and events
- Interacts with waveforms
  - Scroll, pan, zoom, scale, and filter
- Works with signal detections and events
  - Create signal detections, create events, modify certain signal detection/event parameters
- Undo / redo certain operations
- Marks data analysis complete
  - Mark processing stage complete
- Receives notifications about other Analyst or System activity
  - Option to update display to show changes
- Views Data provenance for events & signal detections
- Alternate scripting interface to access waveform, signal detection, event hypotheses, etc.

User Scenario (2/3) – OSD support

- Implement database access abstraction
  - SCRUD Java Code APIs implemented in Java DAOs
  - Partial entity classes developed for waveforms, signal detections, events, processing sequences, and processing stages
  - REST-ful HTTP / JSON SCRUD APIs partially implemented to evaluate performance
- Implement data distribution (pub/sub) both with serialized Entity classes & using the *claim check* pattern
  - Publications may trigger
    - Processing sequence execution
    - Notifications to other Analysts
- Implement data model and persistence updates in order to record and display provenance and event history

User Scenario (3/3) – PSC support

- Define and execute mock processing sequences for automated processing of signal detections and event hypotheses
  - Processing Sequences predefined and persisted in textual format (*BPMN 2.0 XML standard*)
  - OSD loads Processing Sequences into Entity classes
  - Use stubbed domain services accessing stubbed Plugins
  - Processing sequences executed based on pub/sub data distribution and direct REST-based invocation
    - Implement interfaces based on Entity classes
    - Geophysics algorithm implementations possible, not required
- Demonstrate Processing Sequences triggered by OSD callbacks, e.g.
  - Analyst modifying Signal Detection attribute
  - Analyst modifying an Event attribute
  - Analyst marking processing stage complete

Sandia National Laboratories

CURRENT PROTOTYPE STATUS

Agenda

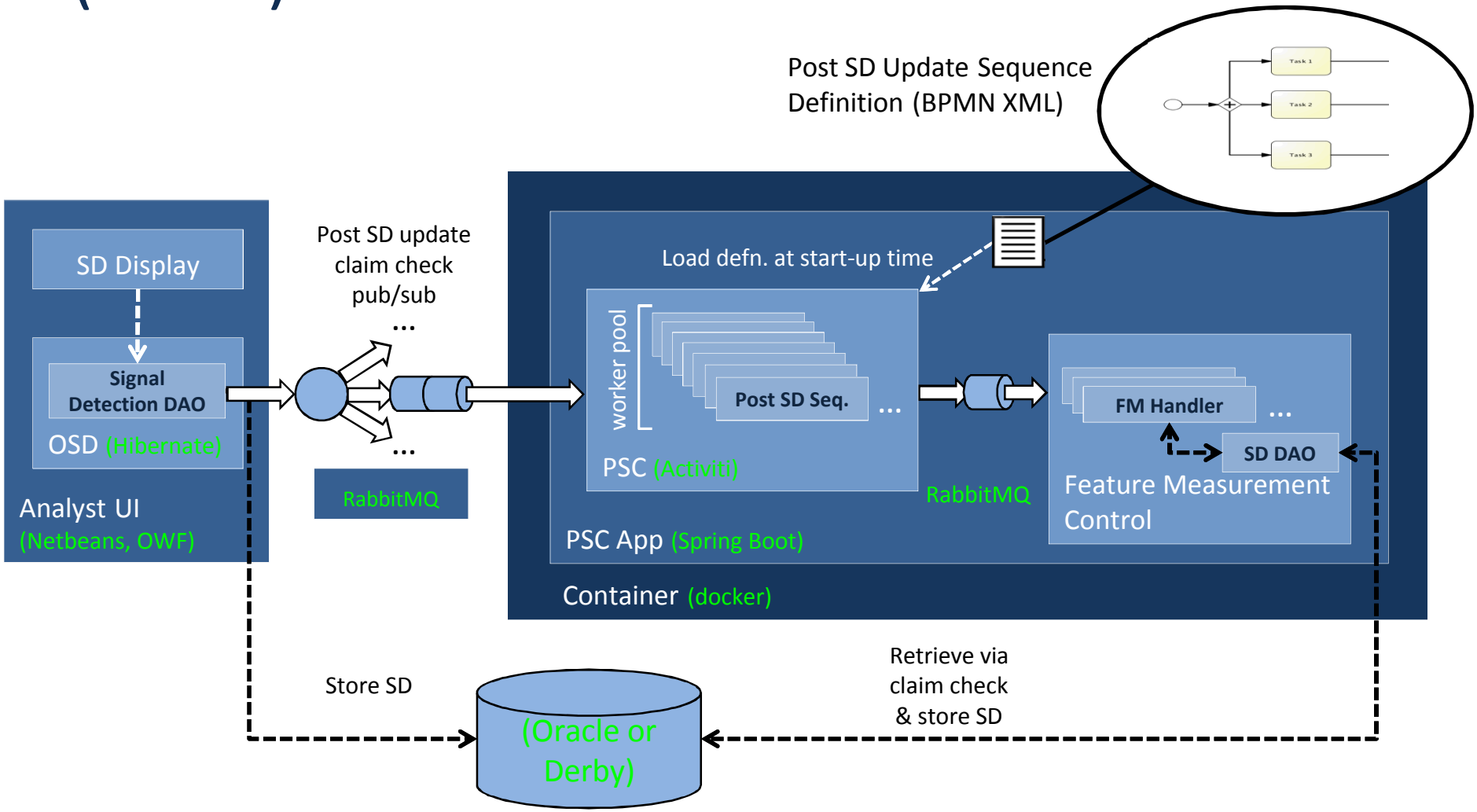
- Prototyping Overview
- Timeline
- Executable Architecture Prototype
  - Definition
  - Conceptual Overview
  - Assumptions
  - Current Technologies
  - Key Features
  - User Scenario
- Current Prototype Status
  - Overview
  - User Interface Prototyping Status
    - Netbeans Display Prototype
    - OWF Display Prototype
  - Application Control Prototyping Status
  - Data Access Prototyping Status
- Next Iteration Goals
- Backup

Current Prototype Status - Overview (1 of 2)

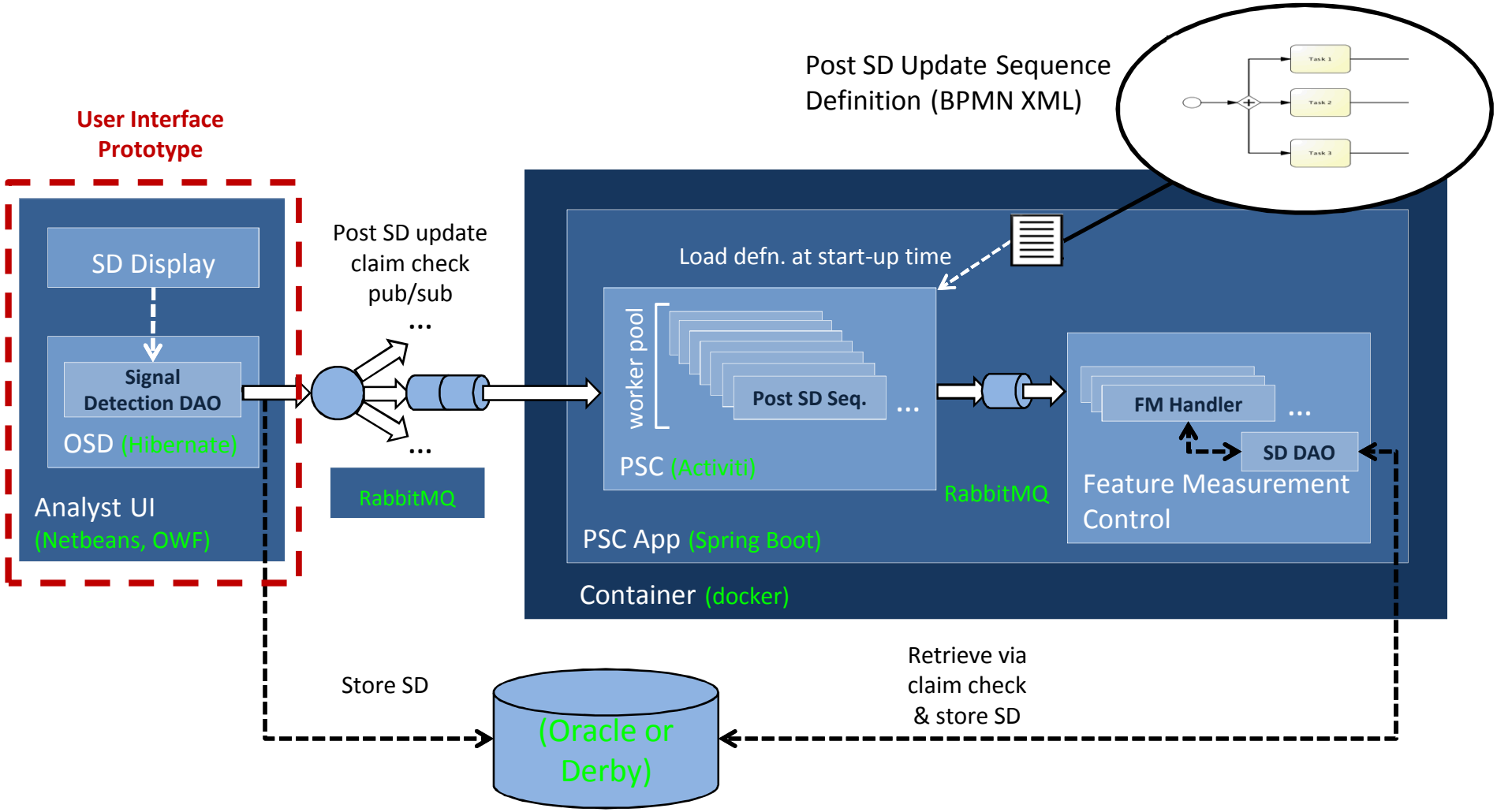
Sandia National Laboratories

- Focus: Signal detection automated processing and analysis
- User Interface: Waveform display and signal detection table
  - Scroll, pan, zoom and scale waveforms
  - Create & re-time phase-labeled signal detections, storing and publishing new/updated signal detection entities via the OSD API
  - Competing desktop and browser-based implementations (Netbeans and OWF, respectively)
- Application Control: Automated execution of mock signal detection processing sequences via the PSC mechanism
  - Selected COTS BMPN engine (Activiti)
  - Developed an initial PSC prototype
- Object Storage & Distribution: Storage/retrieval and distribution of signal detection & waveform entities
  - Load waveform data into the display via the OSD
  - Store and retrieve signal detection entities via the OSD
  - Notify subscribers via pub/sub (RabbitMQ) whenever signal detections are created or modified

Current Prototype Status - Overview (2 of 2)



User Interface Prototyping Status (1 of 2)

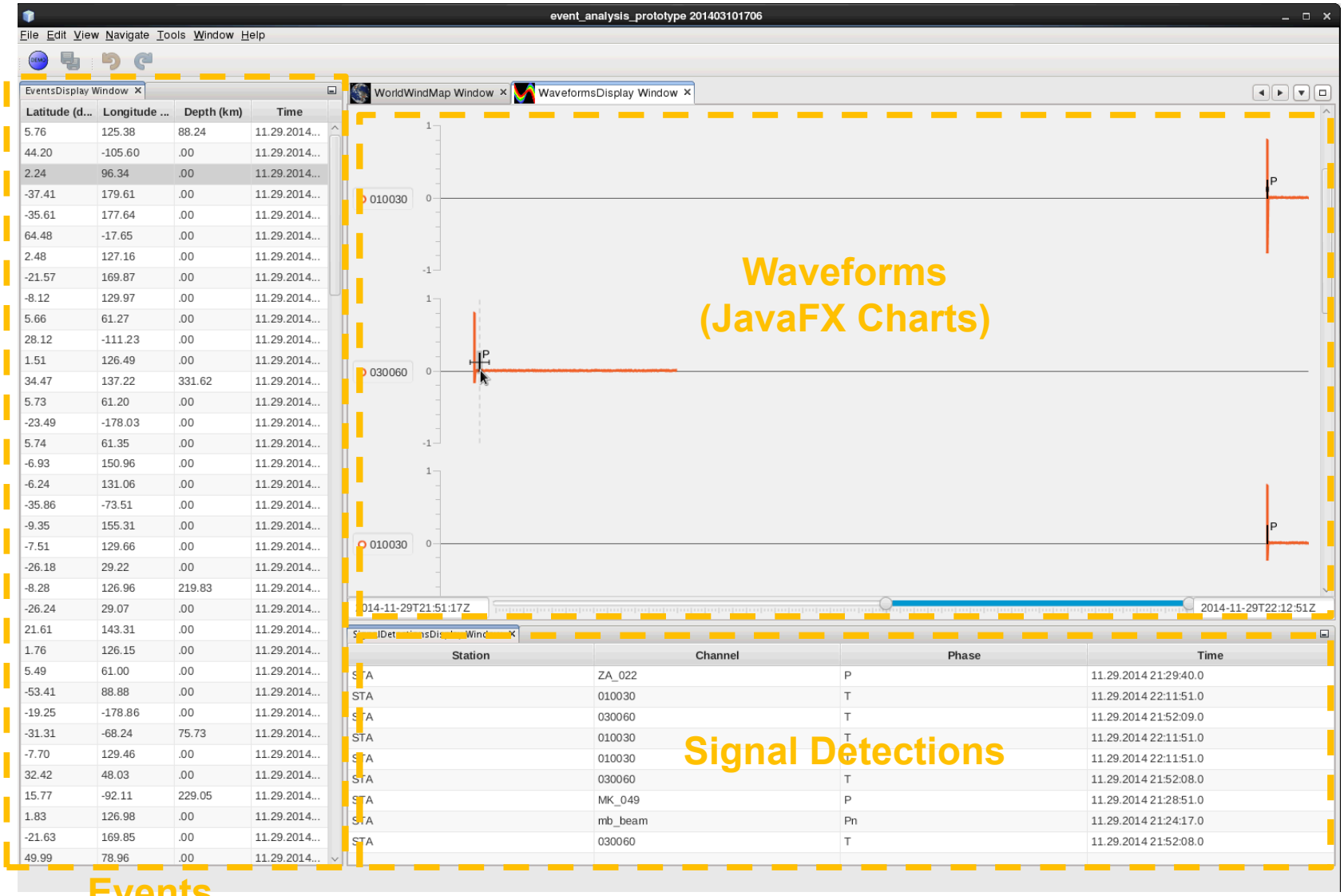


User Interface Prototyping Status (2 of 2)

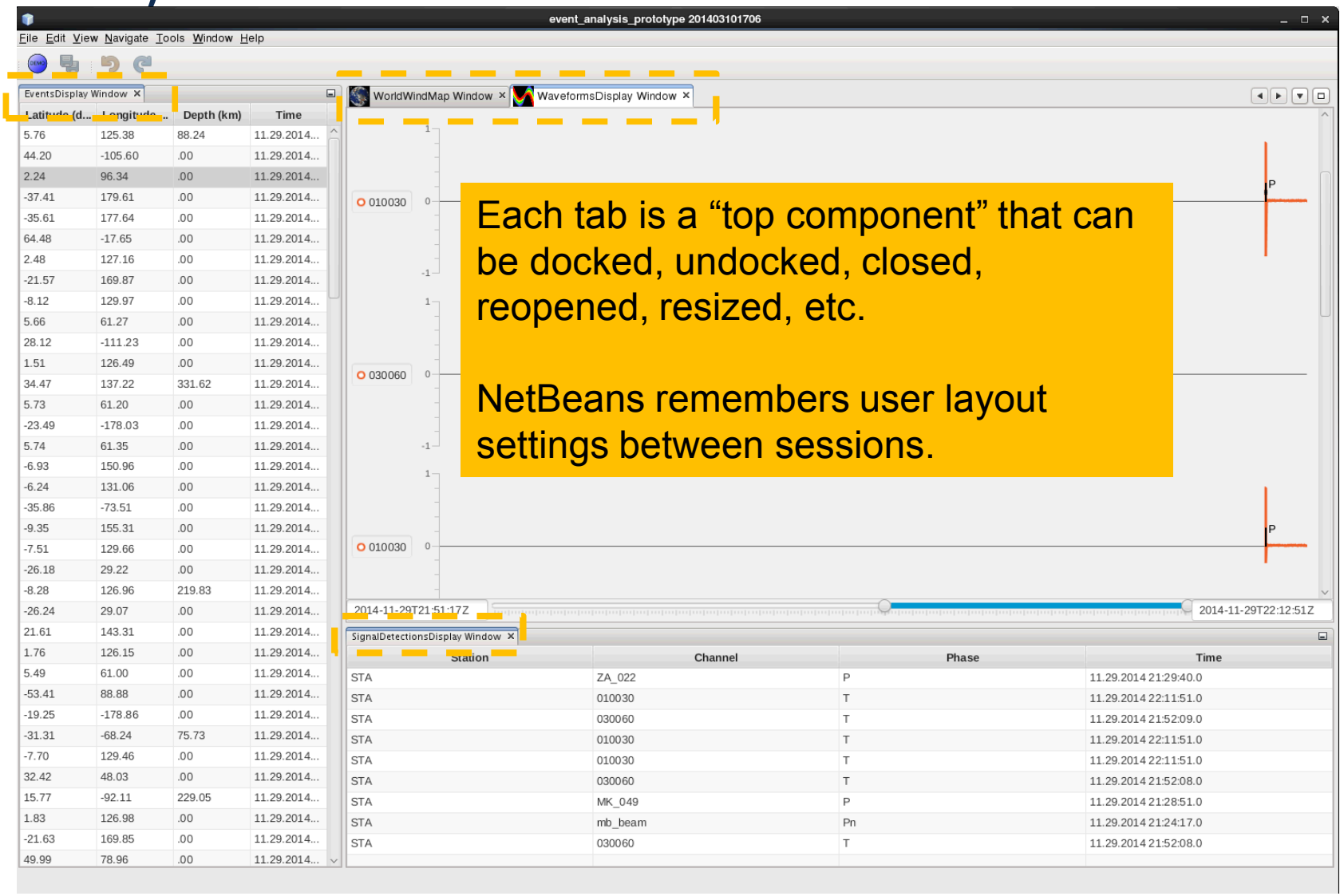
Sandia National Laboratories

- Developed initial Netbeans/JavaFX signal detection displays
  - Implemented a waveform plotting display
  - Implemented a map display
  - Implemented event and signal detection table displays
- Developed initial OWF/WebGL signal detection displays
  - Implemented a waveform plotting display
  - Implemented a map display

Netbeans Display Prototype – Signal Detection (1 of 2)



Netbeans Display Prototype – Signal Detection (2 of 2)

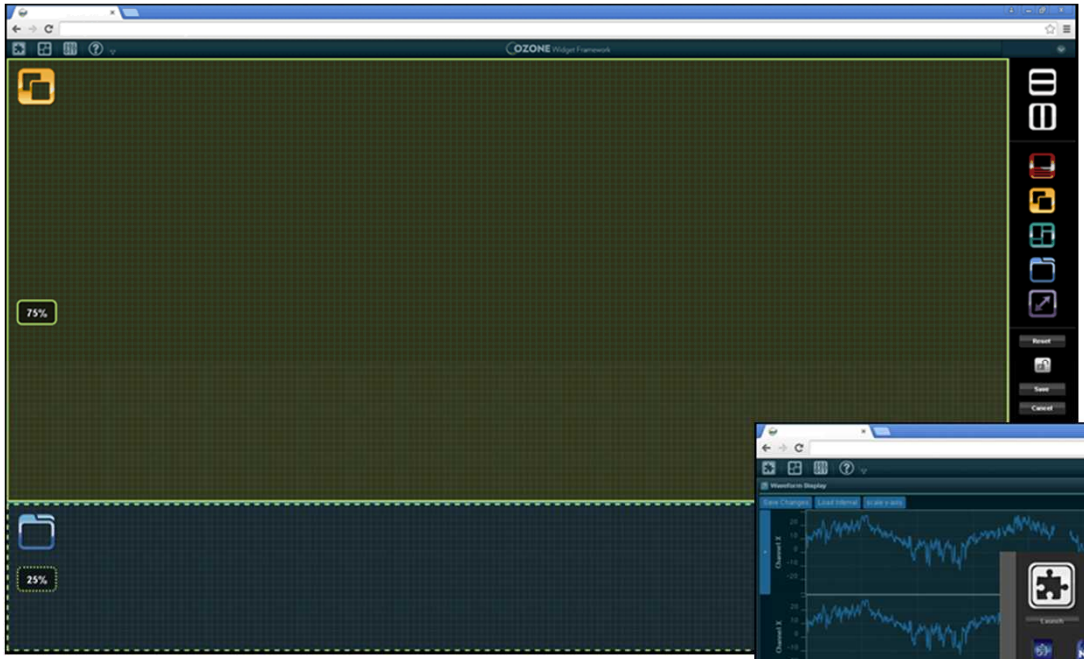


Each tab is a “top component” that can be docked, undocked, closed, reopened, resized, etc.

NetBeans remembers user layout settings between sessions.

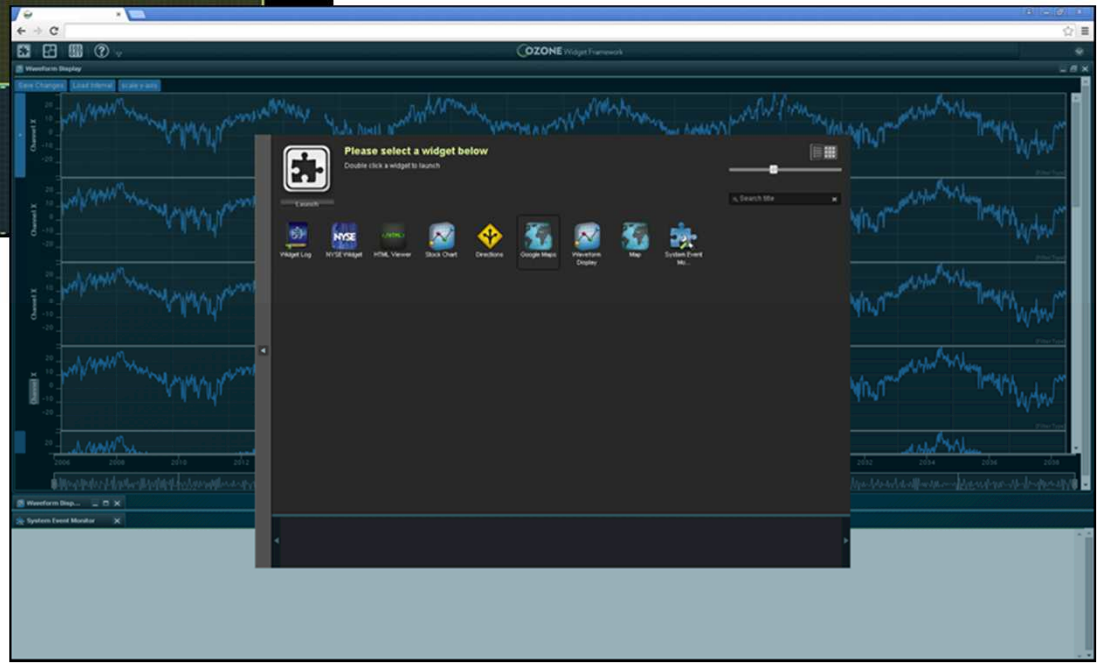
OWF Display Prototype – Default and Custom Dashboards (2 of 3)

Sandia National Laboratories



Design and save Analyst display layouts (dashboards):

- Partition the display space
- Select *desktop, accordion, tabbed, and portal layouts for partitions*
- *Save display layouts & select upon login or configure as default*

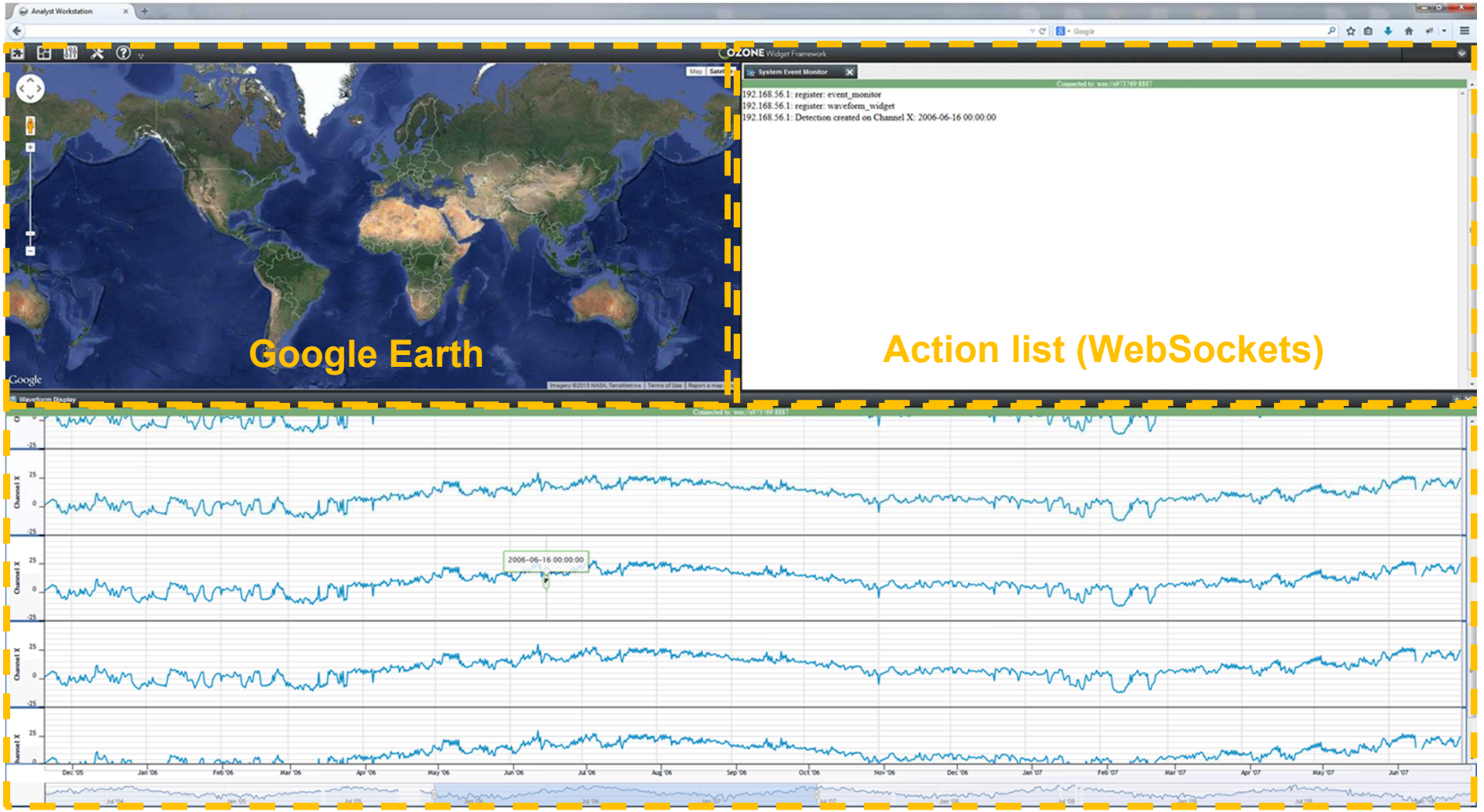


Design and save Analyst display layouts (dashboards):

- Select Widgets to display in each partition of the dashboard (drag and drop)
- E.g. waveform display, signal detection list, event detection list, map, etc.

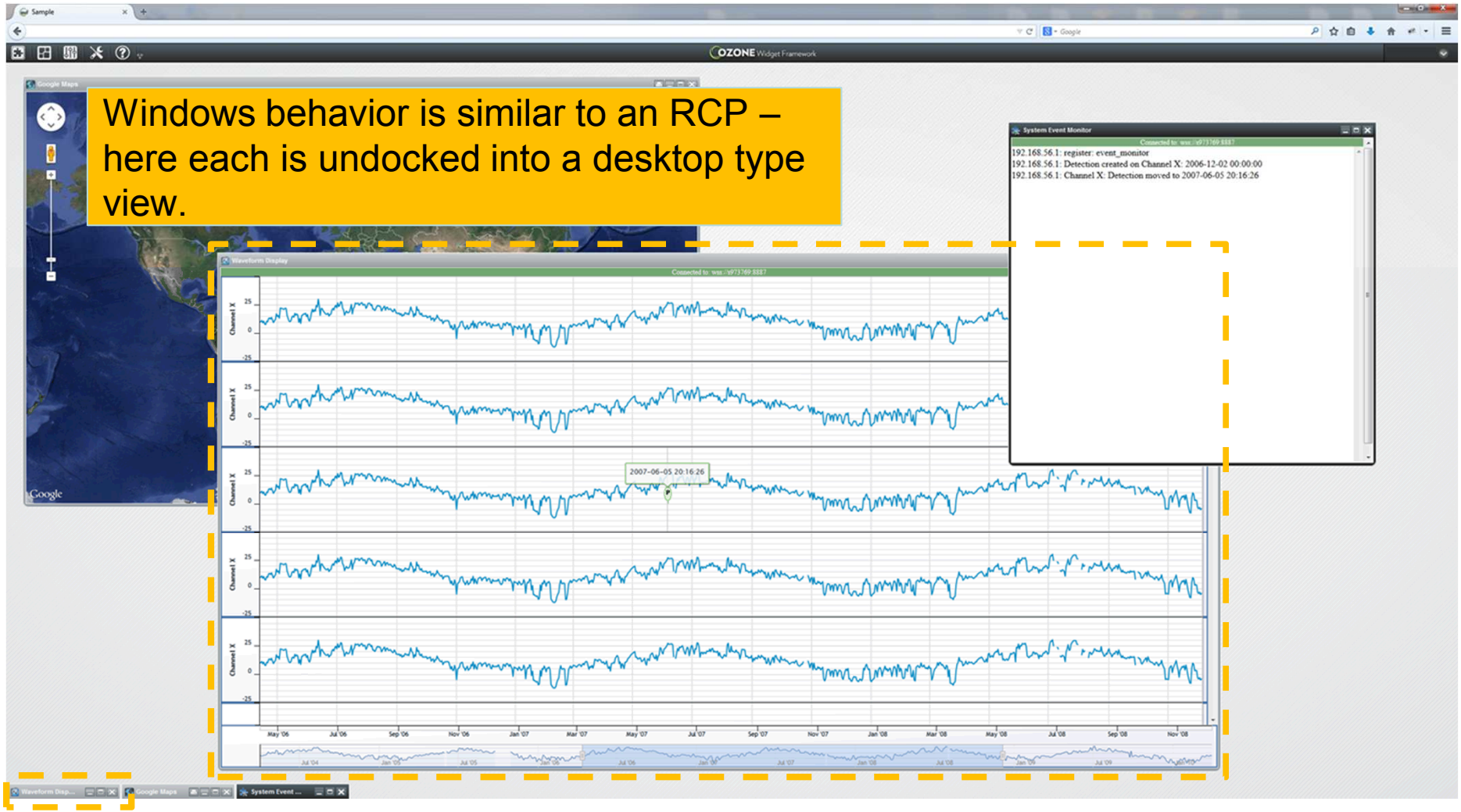
OWF Display Prototype – Map & Signal Detection (1 of 3)

Sandia National Laboratories



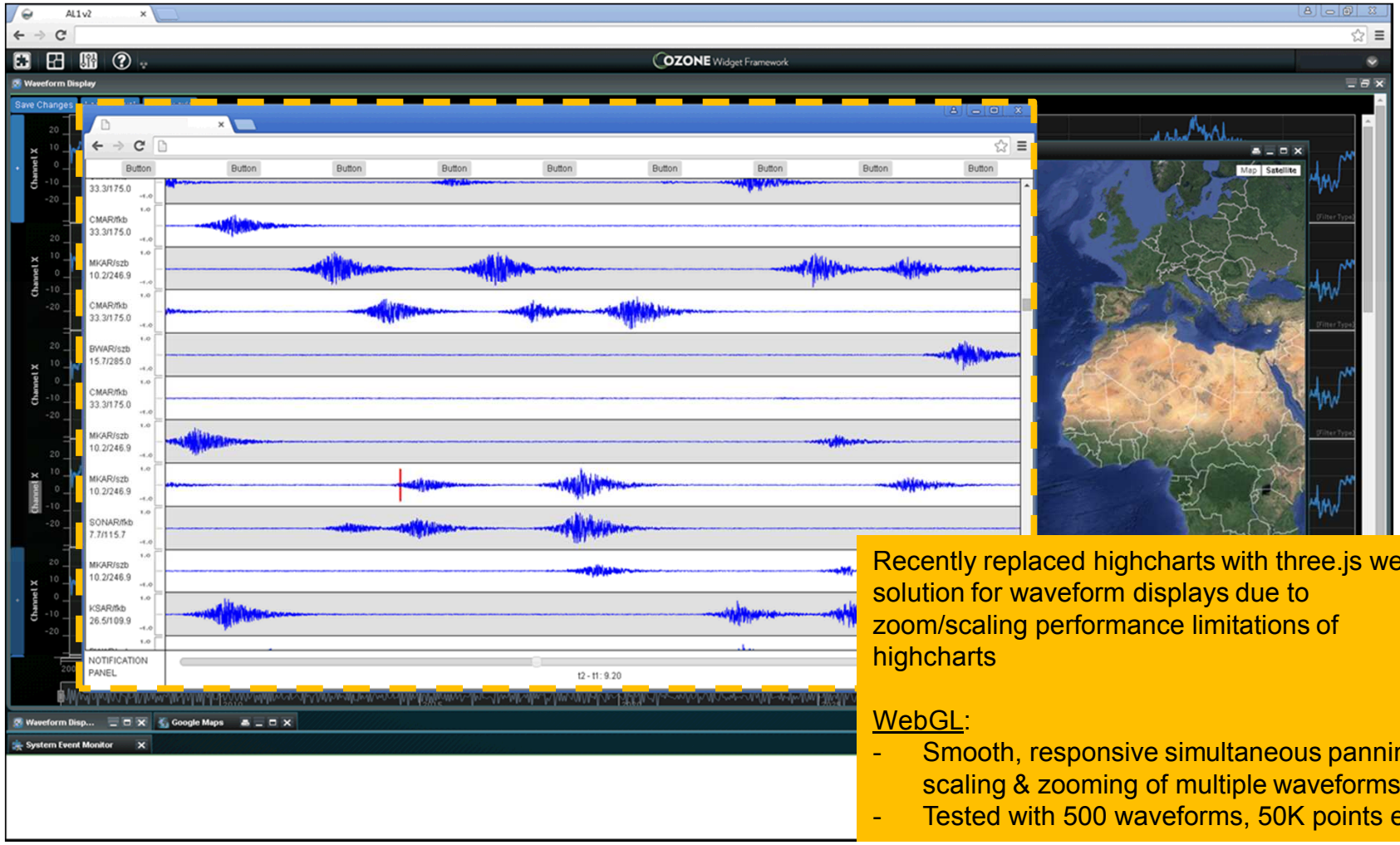
Highstocks Waveforms

OWF Display Prototype – Map & Signal Detection (2 of 3)



OWF Display Prototype – WebGL Waveform Display

Sandia National Laboratories



Recently replaced highcharts with three.js webGL solution for waveform displays due to zoom/scaling performance limitations of highcharts

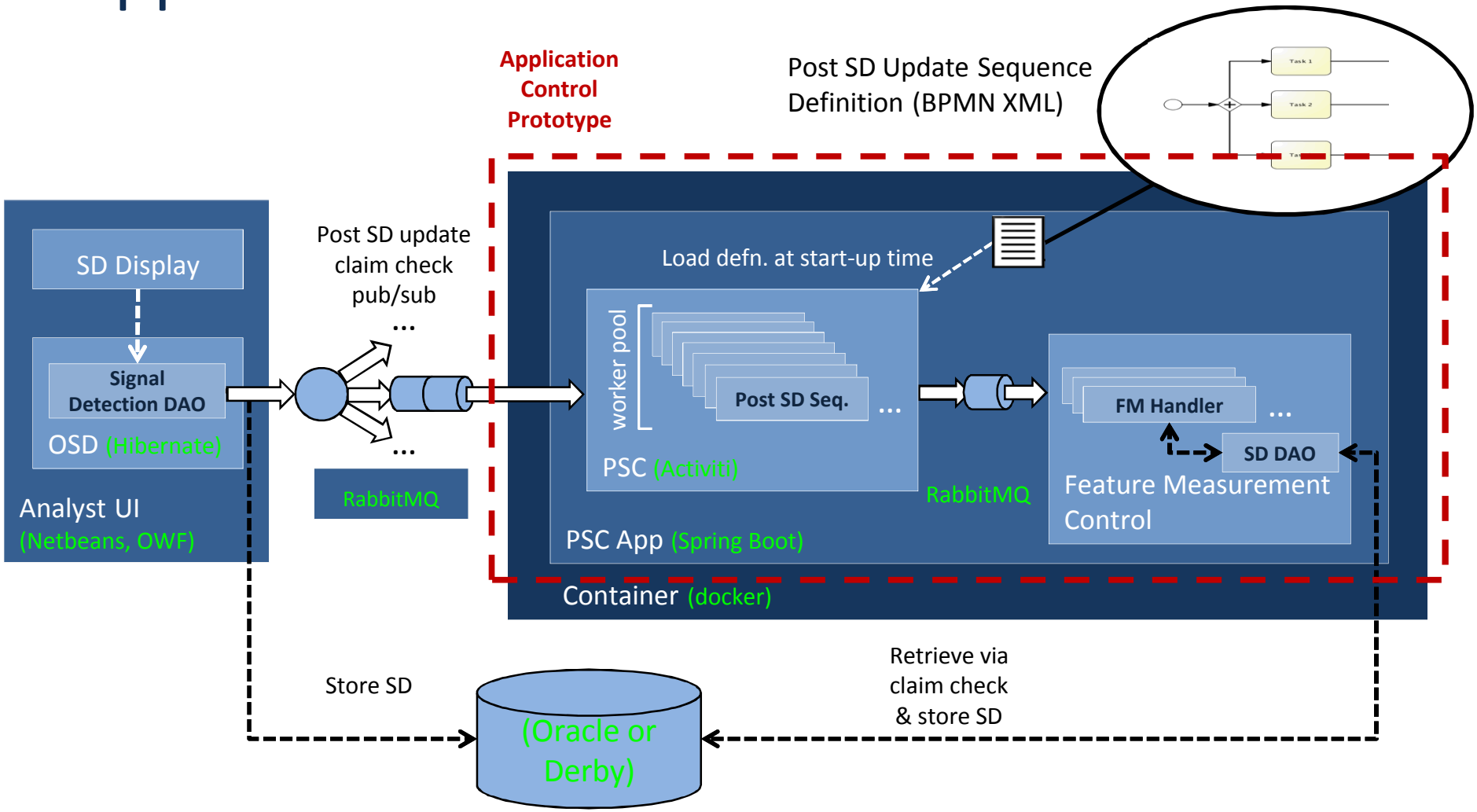
WebGL:

- Smooth, responsive simultaneous panning, scaling & zooming of multiple waveforms
- Tested with 500 waveforms, 50K points each

Agenda

- Prototyping Overview
- Timeline
- Executable Architecture Prototype
  - Definition
  - Conceptual Overview
  - Assumptions
  - Current Technologies
  - Key Features
  - User Scenario
- Current Prototype Status
  - Overview
  - User Interface Prototyping Status
    - Netbeans Display Prototype
    - OWF Display Prototype
  - Application Control Prototyping Status
  - Data Access Prototyping Status
- Next Iteration Goals
- Backup

Current Prototype Status – Application Control



Application Control Prototyping Status (1/4)

Sandia National Laboratories

- Selected COTS BMPN engine (Activiti) as the basis for the PSC mechanism
  - Traded Activiti, JBoss BPMN, Spring Batch (see backup for summary comparison)
- Developed an initial PSC prototype, including:
  - Execution of mock signal detection post-processing sequences based on Analyst actions (create & update)
  - Parallel execution of mock automated signal detection processing sequences
  - Processing sequence steps delegated to domain services via RabbitMQ & REST request/response messaging interfaces

Application Control Prototyping Status (2/4)

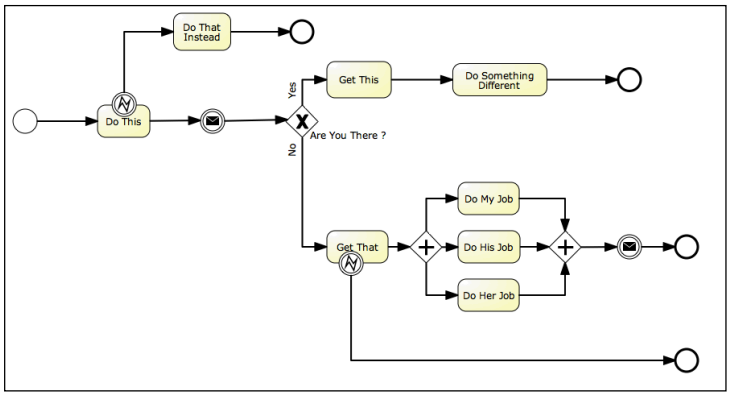
Sandia National Laboratories

- PSC COTS evaluation focused on two primary technologies:
  - Business Process Management (BPM) Engines
    - Workflow engines providing for the definition and execution of Business Process Model and Notation (BPMN) 2.0 standard processes (BPMN standard: www.bpmn.org)
    - Prototyped Activiti BPM
    - Selected for executable architecture development
  - Java Batch Processing Engines (JSR 352)
    - "Comprehensive framework designed to enable the development of robust batch applications"
      - "Provides reusable functions that are essential in processing large volumes of records, including logging/tracing, transaction management, job processing statistics, job restart, skip, and resource management"
    - Prototyped Spring Batch

Application Control Prototyping Status (3/4): Activiti/BPMN

Sandia National Laboratories

- Activiti provides sequence definition and execution
  - Visual sequence definition (Activiti Designer Eclipse plugin) produces BPMN 2.0 XML processing definitions
  - Engine executes standard BPMN 2.0 XML definitions
  - Designer and engine are separable – engine works with standard BPMN 2.0 XML
- Activiti Engine is a multi-threaded runtime engine
  - Built on Spring
  - Can be embedded in any Java application (Spring or not), standalone or as a web application
- BPMN supports
  - Conditional, looped, and parallel tasks
  - Nested sequence (invoke one sequence from another sequence)
  - Timer, message driven, and rule based execution
  - REST, Camel & Mule ESB integration (e.g. initiate Activiti process from Mule, invoke Mule service from Activiti task)
  - Tasks can execute scripts and shell operations
  - Transactional sequences
  - Supports user tasks (require human intervention)



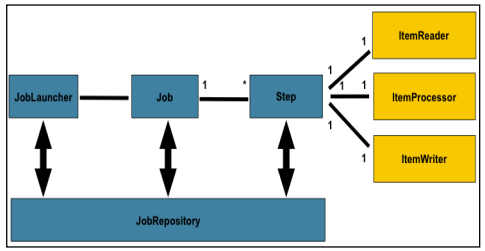

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODE
  <process id="process1" name="process1">
    <startEvent id="startevent1" name="Start"></startEvent>
    <userTask id="usertask1" name="User Task" activiti:assigne
      <extensionElements>
        <activiti:formProperty id="name" name="Name" type="str
      </extensionElements>
    </userTask>
    <endEvent id="endevent1" name="End"></endEvent>
    <sequenceFlow id="flow1" name="to usertask" sourceRef="sta
    <sequenceFlow id="flow2" name="ending" sourceRef="usertask
  </process>
```

Application Control Prototyping Status (4/4): Spring Batch

Sandia National Laboratories

- Runtime provides multi-threaded execution of batch job definitions
  - Built on Spring
  - Can be deployed standalone or as a web application
  - Transactional tasks
  - Supports remote partition-based process execution
- Job definitions support
  - Conditional, looped, and parallel tasks
  - Nested sequence (invoke one sequence from another sequence)
  - Message driven execution
  - Camel & Mule ESB integration (e.g. initiate Activiti process from Mule, invoke Mule service from Activiti task)
  - Tasks can execute scripts and shell operations
  - Supports user tasks (require human intervention)
- Limitations
  - No visual modeling support - team experienced difficulties developing complex sequences
  - No rule engine integration for rule-based job execution
  - No timer-based flow job execution
  - Limited practical documentation

Job Definitions

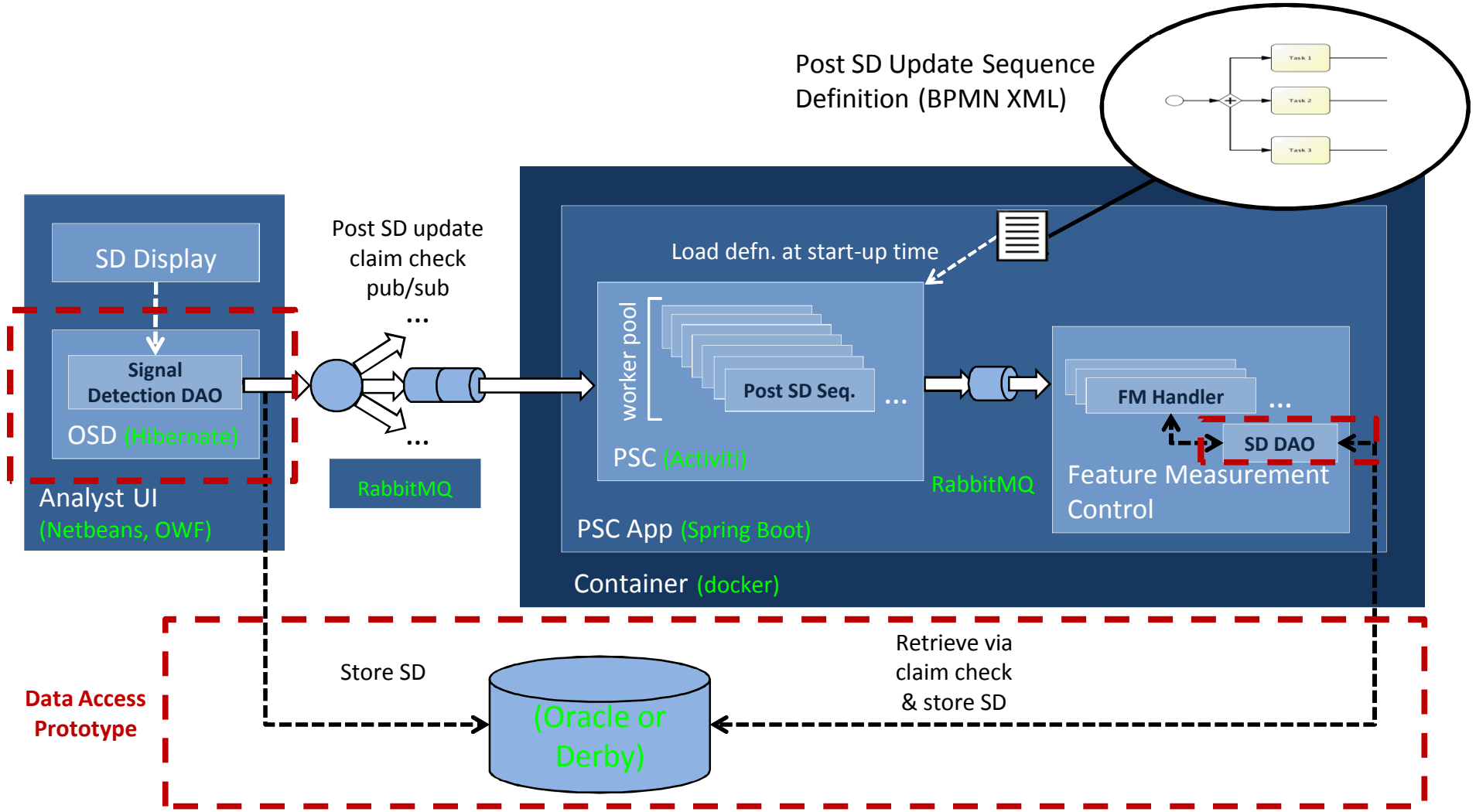


Runtime

Agenda

- Prototyping Overview
- Timeline
- Executable Architecture Prototype
  - Definition
  - Conceptual Overview
  - Assumptions
  - Current Technologies
  - Key Features
  - User Scenario
- Current Prototype Status
  - Overview
  - User Interface Prototyping Status
    - Netbeans Display Prototype
    - OWF Display Prototype
  - Application Control Prototyping Status
  - Data Access Prototyping Status
- Next Iteration Goals
- Backup

Data Access Prototyping Status (1 of 2)



Data Access Prototyping Status (2 of 2)

Sandia National Laboratories

- Developed initial Java COI entity classes
- Developed Waveform and Signal Detection Data Accessor Objects (DAOs) providing SCRUD access to Waveform and Signal Detection entities stored in the DB
- Developed an initial OSD data distribution prototype
  - Pub/sub distribution of Signal Detection entities via RabbitMQ
    - Claim check pattern (DB reference provided via messaging, used by consumer to retrieve the entity from the DB)
    - Direct JSON serialization of entity
- Demonstrated subscription-based distribution of newly created/modified Signal Detections entities from the Analyst UI to the PSC for mock post processing

Agenda

- Prototyping Overview
- Timeline
- Executable Architecture Prototype
  - Definition
  - Conceptual Overview
  - Assumptions
  - Current Technologies
  - Key Features
  - User Scenario
- Current Prototype Status
  - Overview
  - User Interface Prototyping Status
    - Netbeans Display Prototype
    - OWF Display Prototype
  - Application Control Prototyping Status
  - Data Access Prototyping Status
- Next Iteration Goals
- Backup

Next Iteration Goals

- Design and develop initial prototypes for:
  - Plugin deployment and binding
  - Event history and processing parameter provenance
  - Undo/redo
- Select browser vs. RCP user interfaces
- Evaluate performance of REST-based data access
- Begin development of event analysis scenario

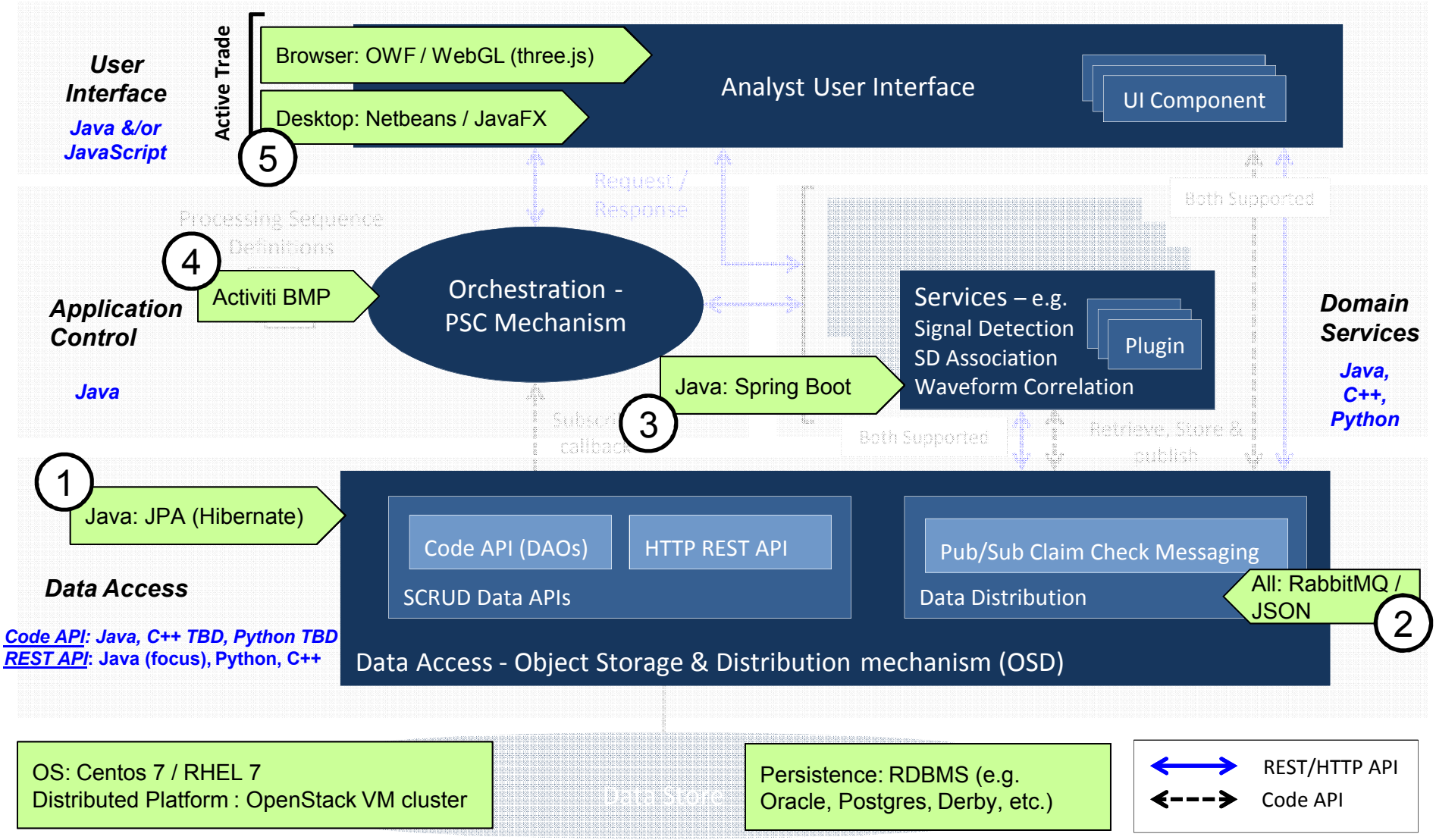
Sandia National Laboratories

BACKUP

Sandia National Laboratories

TECHNOLOGY EVALUATION SUMMARY

Technology Evaluation - Overview



1. Technology Evaluation – Object Relational Mapping (data access code API)

Sandia National Laboratories

Candidate Solution	Solution Type	Summary Assessment	
Java			
Hibernate	Java Object Relational Mapping (ORM) OSS	Advantages: Leading ORM candidate for Java. Hibernate Query Language (HQL) could provide both application and researcher level access to underlying COI objects. JPA provider. Disadvantages: A dependence on HQL could introduce a tight coupling to Hibernate.	Lower database solution coupling
Open JPA	Java ORM OSS	Advantages: JPA provider. Disadvantages: ORM features supported through embedded SQL. Not a prevalent software solution.	
Apache Cayenne	Java ORM OSS	Advantages: Supports Remote Object Persistence Disadvantages: CayenneModeler required for mapping. Not a prevalent software solution.	
Apache Empire-DB	Java RDBMS Abstraction OSS	Advantages: Database interactions more easily optimized since interactions are at such a low level. Disadvantages: Database abstraction layer (not an ORM). SQL-centric. Not a prevalent software solution.	
Apache Torque	Java ORM OSS	Advantages: Uses XML that describes the database schema, which avoids reliance on reflection. Disadvantages: Requires that domain model extend Torque specific classes. Not a prevalent software solution.	Higher database solution coupling
C++			
ODB	C++ ORM OSS	Advantages: Leading ORM candidate for C++. Does not require manual entry of mapping code. Disadvantages: Developed by Code Synthesis, located in South Africa. Does not provide C++ object to relational database mapping for existing DB tables.	Lower coupling
QxORM	C++ ORM OSS	Advantages: Supports object relational mapping with MySQL, SQLite, PostgreSQL, Oracle, and SQL Server databases. Disadvantages: Market usage is unknown and documentation is limited.	Higher coupling

2. Technology Evaluation – Inter-process Communication (pub/sub data distribution)

Sandia National Laboratories

Name	Standards	Language Support	Advantages	Disadvantages
RTI DDS	DDS JMS REST SOAP	C, C++ C# Java Ada	• Standards-Based • Cross-Language Support • Designed for low-latency, high-throughput with configurable QoS • Flexible communication patterns & configurable transports • Open-source version available with commercial support from RTI • Generally considered to be higher performance than brokered solutions	• Open-source license is more restrictive than for other solutions • Many features are only available in the commercial edition • Appears to be less popular than other solutions (based on Google Trends) • Configurable QoS introduces complexity relative to other solutions • Past prototyping efforts have struggled with product complexity
Qpid	AMQP JMS	Java C, C++ C# Ruby Perl Python	• Standards-Based • Cross-Language Support • Free OSS with community support	• Appears to be less popular than other solutions (based on Google Trends)
ActiveMQ / Apollo	AMQP STOMP REST XMPP JMS 1.1	Java C, C++ C# Ruby Perl Python\	• Standards-Based • Cross-Language Support • Free OSS with community support • Mature & highly stable (widely used since early 2000s) • Highly popular	• Performance limitations at scale (Apollo subproject attempts to address these, but is not yet a full-featured product) • Interest in ActiveMQ appears to be declining in recent years (based on Google trends)
RabbitMQ	AMQP STOMP	Java C++ .NET Ruby Perl Python	• Standards-Based • Cross-Language Support • Free OSS with community support • Commercial support available from Pivotal • Highly popular (highest search term frequency on Google Trends) • Favorable performance on a number of benchmarks	• Broker is implemented in Erlang (not necessarily a disadvantage)
ZeroMQ	None	Java C, C++ C# Ruby Perl Python	• Cross-Language Support • Free OSS with community support • Generally considered to be higher performance than brokered solutions	• Not standards-based • Appears to be less popular than other solutions (based on Google Trends)

3. Technology Evaluation – Application Control

Sandia National Laboratories

Category	Candidate Solution	Summary Assessment
Enterprise Java Application Frameworks	Java EE	Advantages: Widely-used open standards with large development community. Provides a robust platform for development of scalable, fault-tolerant, distributed processing architectures. Disadvantages: EJB standard prohibits use of native libraries and direct thread creation, limiting design options supporting non-JVM languages.
	Spring Framework	Advantages: Widely-used open-source solution with large development community. Provides a robust platform for development of scalable, fault-tolerant, distributed processing architectures. Disadvantages: Not standards-based.
Stream Processors	Apache Storm	Advantages: Open-source solution with significant industry interest. Provides a robust platform for development of scalable, fault-tolerant, distributed processing architectures. Supports multiple development languages. Disadvantages: New offering. Not standards-based.
	Apache Samza	Advantages: Provides a robust platform for development of scalable, fault-tolerant, distributed processing architectures. Disadvantages: New offering that has yet to establish significant industry interest. Not standards-based. Does not support multiple languages (Java only).
	Apache S4	Advantages: Provides a robust platform for development of scalable, fault-tolerant, distributed processing architectures. Supports multiple development languages. Disadvantages: Little industry interest and development activity. Not standards-based.
Enterprise Service Bus	WS02 ESB	Advantages: Provides a robust platform for integration of heterogeneous systems via standardized messaging as part of a service-oriented architecture. Disadvantages: Design strengths not well aligned to the end-state modernized architecture.
Complex Event Processor	Esper	Advantages: Provides a robust platform for development of scalable, fault-tolerant, distributed processing architectures. Disadvantages: Specialized, query-based architecture does not fit processing needs particularly well. Not standards-based. Does not support multiple languages (Java only).

4. Technology Evaluation – Processing Sequence Control

Sandia National Laboratories

Name	Standards	Advantages	Disadvantages
Activiti BPMN	BPMN 2.0	• Standards-Based • Free OSS with community support • Strong community, active feature development • High-quality documentation • Eclipse plugin integration for visual modeling	• Less mature than JBoss BPMN • Commercial support by a smaller, less well known company
JBoss BPMN	BPMN 2.0	• Standards-Based • Free OSS with community support • Mature solution	• Decline in community development activity • Poor documentation • Difficult to work with
Spring Batch	JSR 352	• Standards-Based • Free OSS with community support	• No visual modeling support - team experienced difficulties developing complex sequences • No rule engine integration for rule-based job execution • No timer-based flow job execution • Limited practical documentation

5. Technology Evaluation – Desktop User Interface

Sandia National Laboratories

Candidate Solution & Widget toolkit	Language	Summary Assessment
Netbeans / Swing	Java (RCP)	Advantages: Netbeans is a dominant Java UIF candidate. Swing widgets integrate alongside JavaFX code. OSGi open standard. Oracle supported. Large community. Disadvantages: Oracle (the company) dependence.
Eclipse / Jface (SWT)	Java (RCP)	Advantages: Eclipse is a dominant Java UIF candidate. OSGi open standard IBM supported. Very stable. Large community. Disadvantages: Eclipse learning curve is the most difficult. JFace/SWT is slightly dated compared to Swing and JavaFX2. IBM dependence.
Qt Creator / Qt	C++	Advantages: Qt is the leading C++ UIF candidate. GUI widgets are fast and native: strongest cross platform GUI behavior. Disadvantages: Not an RCP solution. Not OSGi. Smaller community than Java.
Netbeans / JavaFX2	Java (RCP)	Advantages: Netbeans is the leading Java UIF candidate. JavaFX2 has most modern Java GUI elements. OSGi open standard. Oracle supported. Large community. Disadvantages: JavaFX2 2D plotting package is beautiful but has serious scaling issues. Oracle dependence.
NA / wxWidgets	C++	Advantages: Native mode widget toolkit, also contains inter-process communication layer Disadvantages: Not an RCP solution or a UIF - mainly a standalone widget toolkit. Smaller community.
NA / XUL	XML & Java	Advantages: XML markup language for GUI construction. Quick study for web designers. Disadvantages: Not an RCP solution or a UIF - mainly a standalone widget toolkit. Not a prevalent solution.

Technology Evaluation – In-Memory Caching/Data Grid

Sandia National Laboratories

Selection pending determination of need for caching

Name	Client Language Support	Advantages	Disadvantages
JCS	Java	• Cross-Language Support • Free OSS with community support	• Java only (no cross-language support) • Appears to be less widely used/popular than other solutions (e.g. Redis, memcached) • It is not clear whether commercial support is available • Limited feature set relative to other solutions surveyed • Does not support partitioning (only replication)
memcached	C, C++ Java, Python Ruby Perl C#	• Well established and mature • Widely used highly popular • Cross-Language Support • Free OSS with community support • Commercial support available	• Popularity appears to be declining (based on Google Trends)
EHCache	Java C++ &C# (commercial version)	• Cross-Language Support • Free OSS version available • Commercial support available from Terracotta • Strong feature set, including partitioning, replication, transactions, security, etc.	• Many features are only available in the commercial edition • Limited cross-language support (and only in the commercial edition) • Appears to be less widely used/popular than other solutions (e.g. Redis, memcached) • Popularity appears to be declining (based on Google Trends)
Infinispan	C++ Java Python Ruby C#	• Cross-Language Support • Free OSS with community support • Commercial support available from JBoss • Strong feature set, including partitioning, replication, transactions, security, etc.	• Appears to be less widely used/popular than other solutions (e.g. Redis, memcached)
Redis	C, C++ Java Perl Python Ruby C# Closure Scala	• Widely used highly popular • Broad cross-Language Support • Free OSS with community support • Commercial support available from Pivotal • Strong feature set, including partitioning, replication, transactions, etc.	• Limited built-in security features
Hazelcast	Java C++ &C# (commercial version)	• Cross-Language Support • Commercial support available from Hazelcast • Strong feature set, including partitioning, replication, transactions, security, etc.	• Many features are only available in the commercial edition • Limited cross-language support (and only in the commercial edition) • Appears to be less widely used/popular than other solutions (e.g. Redis, memcached)

Sandia National Laboratories

DOCKER CONTAINER TECHNOLOGY OVERVIEW

Agenda

- What are Linux Containers?
- Containers vs. Virtual Machines
- Why Containers?
- Container Ecosystem
- What is Docker?
- Container Use Cases
- Development Lifecycle Concepts
- Integration Concepts
- Orchestration Concepts

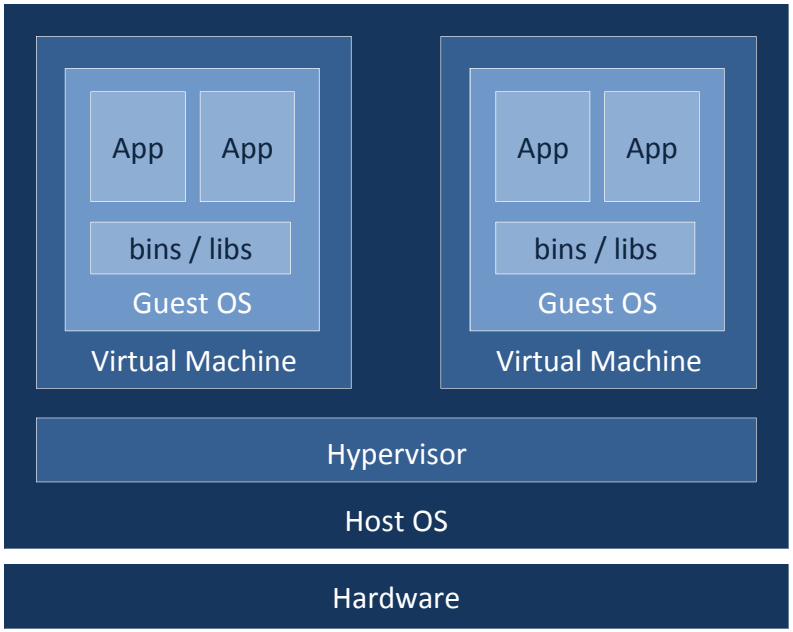
What Are Linux Containers?

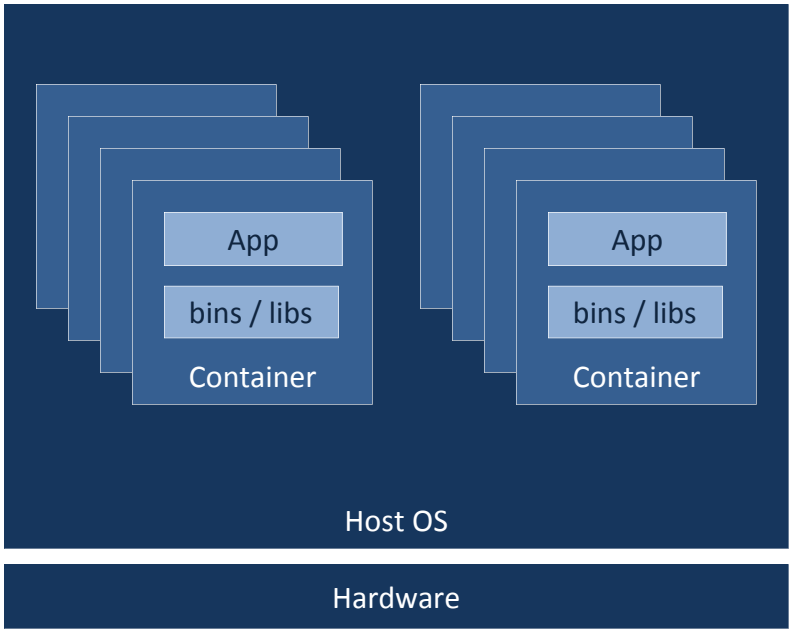
Sandia National Laboratories

- Lightweight OS-level virtualization technology
- Multiple, isolated systems (containers) run on a single Linux host, sharing the underlying kernel
  - Control Groups provide resource isolation (CPU, memory, block I/O, network, etc.)
  - Namespaces isolate applications' view of the OS environment (processes, networking, file system, etc.)
- Alternative paradigm to virtual machines
  - Containers offer greatly reduced start-up time & resource utilization, as well as near-native speed
    - CPU performance – native
    - Memory – Very small overhead
    - Network – Very small overhead (can be optimized to near zero)
  - Individual applications & services are typically deployed in separate containers (100s-1000s of containers per node is common)

Containers vs. Virtual Machines

- Unlike virtual machines, containers provide separate virtual OS environments that share the underlying host OS directly, without the need for separate guest OS instances or hypervisor software





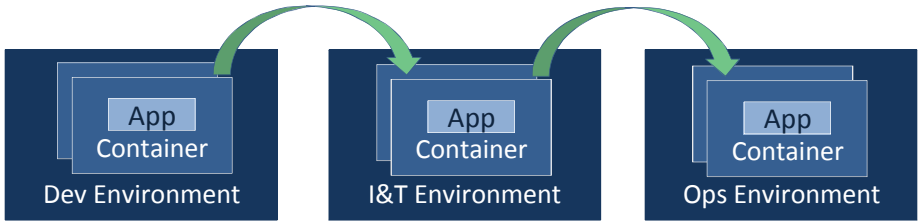
Why Containers?

Common Use Cases

Sandia National Laboratories

- Isolate individual *applications*
- Constrain application resource utilization
  - CPU, memory, network & storage
- Manage heterogeneous application dependencies across complex systems
  - Multiple software stacks, third party tools & versions
  - Multiple Linux flavors & versions
- Provide orchestration of applications across clustered system deployments
- Support application scalability and fault tolerance

- Provide a consistent application runtime for development through operational deployment



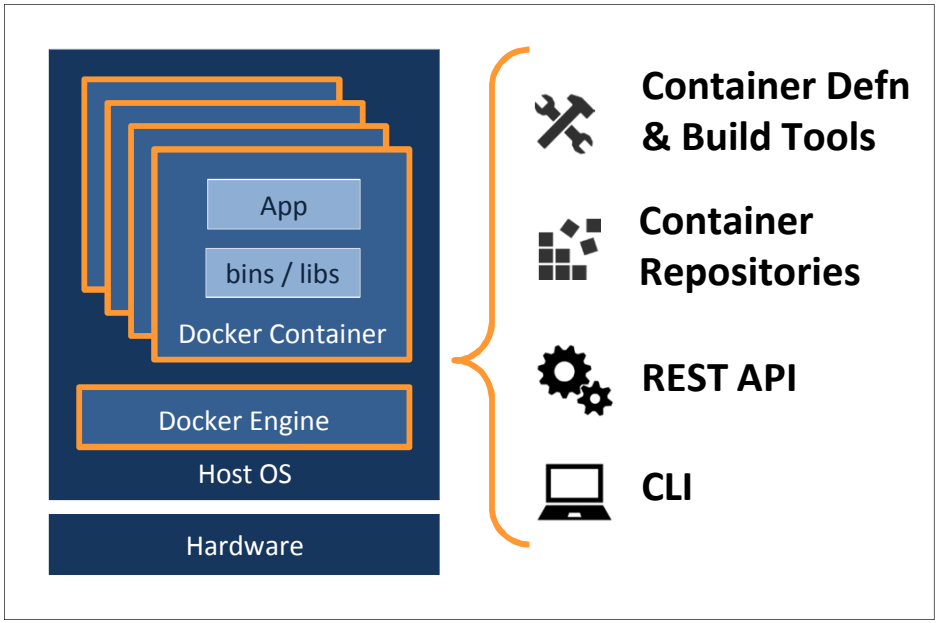
Container Ecosystem

- A thriving community of open-source container technologies has emerged in the last few years supporting the development and deployment of container-based systems
- Large impact on industry with significant backing and community interest
  - Widely used in modern PaaS solutions – e.g. Amazon EC2, DigitalOcean, Google Compute Engine, Microsoft Azure, OpenStack, QEMU/KVM, Vagrant and Vmware
  - Widely used in corporate IT infrastructure - **Google's infrastructure runs nearly entirely on containers (2 billion+ provisioned per week)**
- **Docker has emerged as the dominant container project around which the majority of these technologies is currently developed**

Orchestration	Swarm Compose Machine Fleet Kubernetes Mesos	Load-balancing, scalability & high availability for distributed, container-based systems
Container Management	Docker Rocket LXC	Definition, build, & deployment of containerized apps
Host OS	RedHat Linux CentOS CoreOS Ubuntu Project Atomic	OS shared across containers

What is Docker?

- Open-source project providing container abstraction and management tooling
- Container Definition & Build
  - Build containers from pre-defined images
  - Define images declaratively by specifying new layers on top of existing images, from the base OS up
- Container Repositories
  - Manage container images for projects & communities using shared repositories
  - Publish and consume public images with docker hub (the definitive public repository)
- REST API & CLI
  - Define, build, deploy & manage containers via CLI and RESTful service interfaces



Potential US NDC/IDC Container Use Cases

Sandia National Laboratories

- Development Lifecycle
  - Provide a common software platform for application developers across environments, from development, to I&T and operations
- Integration
  - Enable integration of heterogeneous applications developed by multiple organizations on a common platform
    - Isolation, resource constraints, support for multiple software languages & stacks, third-party COTS, versions, OS versions, etc. as needed
- Orchestration
  - Provide software stack-agnostic, scalable, fault-tolerant orchestration of applications and services in a distributed environment
- Deployment
  - Support for deployment to bare metal, VM and cloud platforms

Development Lifecycle Concepts

Sandia National Laboratories

- A common application platform is provided to US NDC/IDC contributors as a set of base container images
  - Includes OS, third-party tools & libraries, core framework software
  - Provided through a shared container registry
- Applications are developed using the base platform containers and are packaged as new container images layered onto the base platform
  - Application containers are delivered back to the shared container registry
  - Continuous build & integration includes automated build and test of containerized applications & services
- Application containers are deployed into the US NDC/IDC environments as part of the development lifecycle, including development, I&T and operations

Integration Concepts

- The system integrator organization maintain the configuration needed to deploy and manage the set of application containers comprising the system
  - System cluster definition
  - Application definitions
  - Application resource constraints
  - Network and persistent storage interface management
  - High-availability & scalability policies
- Multiple libraries, third-party tools, versions, etc. are encapsulated within individual application containers where they are required