Exceptional service in the national interest

Sandia National Laboratories

IDC Reengineering Phase 2

Inception Iteration I2 Architectural Prototyping Review

Jonathen Kwok

January 27, 2015

SAND2015-XXXXXXX

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. SAND NO. 2011-XXXXP

Architectural Prototype Overview

Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- Service Oriented Architecture (SOA) Proof of Concept
- E1 Prototyping: Common Object Interface
- E1 Prototyping: Processing Control Framework
- E2 Prototyping: OSD & PC Software Infrastructure
- E1 Prototyping: User Interface Framework
- E2 Prototyping: User Interface Framework

Architectural Prototype Overview

Sandia National Laboratories

- **Service Oriented Architecture (SOA) Study Status**
- Service Oriented Architecture (SOA) Proof of Concept
- E1 Prototyping: Common Object Interface
- E1 Prototyping: Processing Control Framework
- E2 Prototyping: OSD & PC Software Infrastructure
- E1 Prototyping: User Interface Framework
- E2 Prototyping: User Interface Framework

Study: Service Identification

Sandia National Laboratories

Goal: identify if the USNDC can be built from services

- **Granularity** – ratio of how much computation is performed in a single call to a service to its invocation overhead.
  Higher ratio is better
- **Autonomy** – likeliness of a component's results independently meeting a need versus always used as an intermediate step in a larger process.
  Higher is better
- **Modularity** – component is described by a general interface
  Higher is better
- **Volume** – indicates how often a component is used

Sample Service Identifications

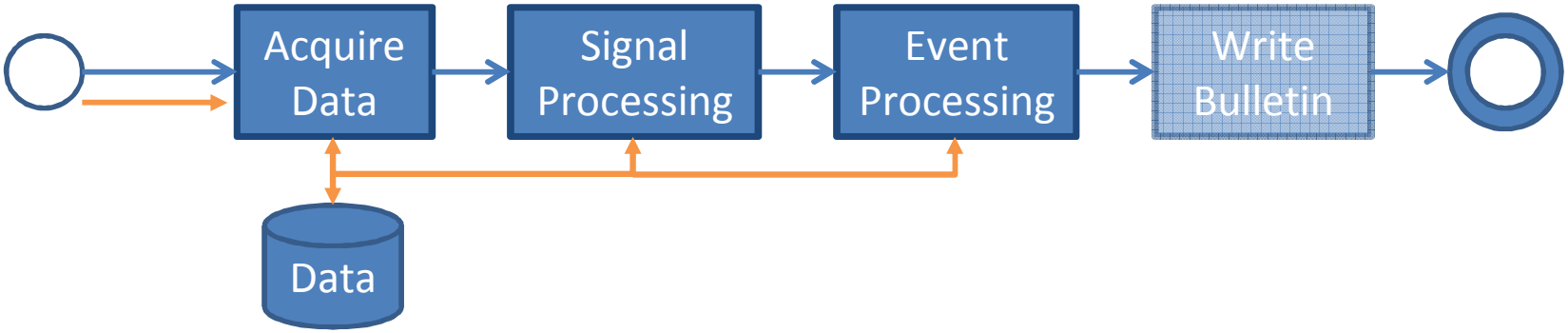
Sandia National Laboratories

Potential Service	Granularity	Autonomy	Modularity	Volume	Service / Library / Application
Event location	C	H	H	H	S
Signal association	C	H	H	M	S
Event identification	C	H	H	M	S
Discriminant calculations	F	L	M	M	L/S
Individual signal processing / feature extraction operations (filter, beam, rotate, onset time, …)	F	L	H	M	L/S
Combined signal detection & feature extraction	M	M	H	M	S
Waveform correlation based signal analysis	C	H	H	M	S
Analyst work assignment creation	F	H	H	L	A
Analyst work assignment distribution	F	H	H	L	A

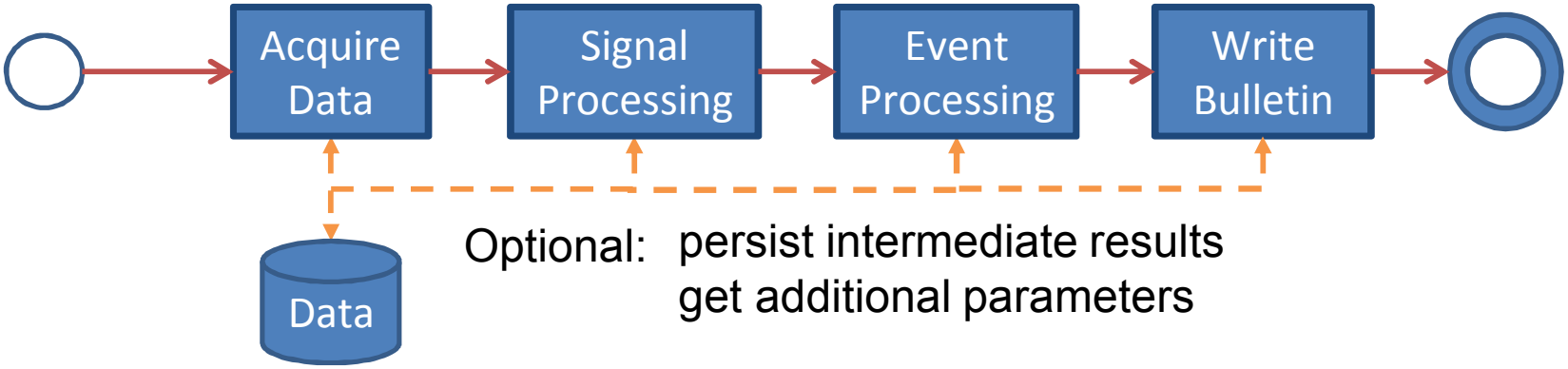
SOA Study Status

Sandia National Laboratories

Light interfaces: control flows between services; data flows through DB



Rich interfaces: control and data flow together between services

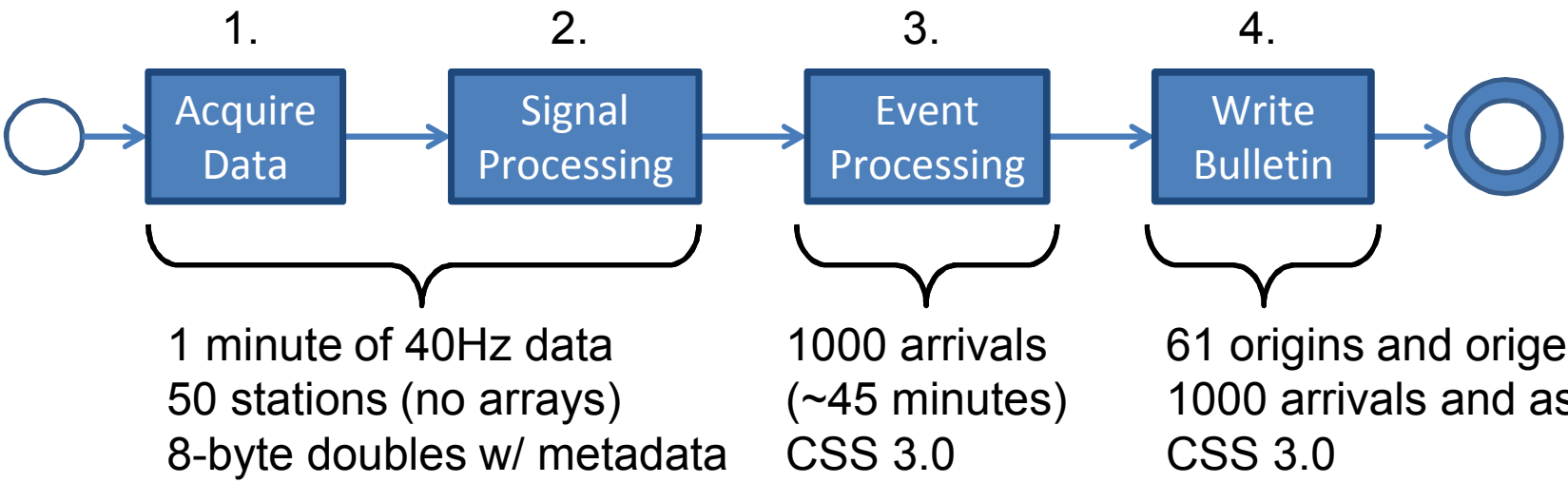


Context: XML, direct COI access, no central pipeline controller

Results using other configurations are available

SOA Study Status

1. Receive 1 min intervals of waveform data from a network
2. Immediately signal process waveforms to produce arrivals
3. Collect 45 minutes of arrivals then form network events
4. Immediately store events and associated detections to DB

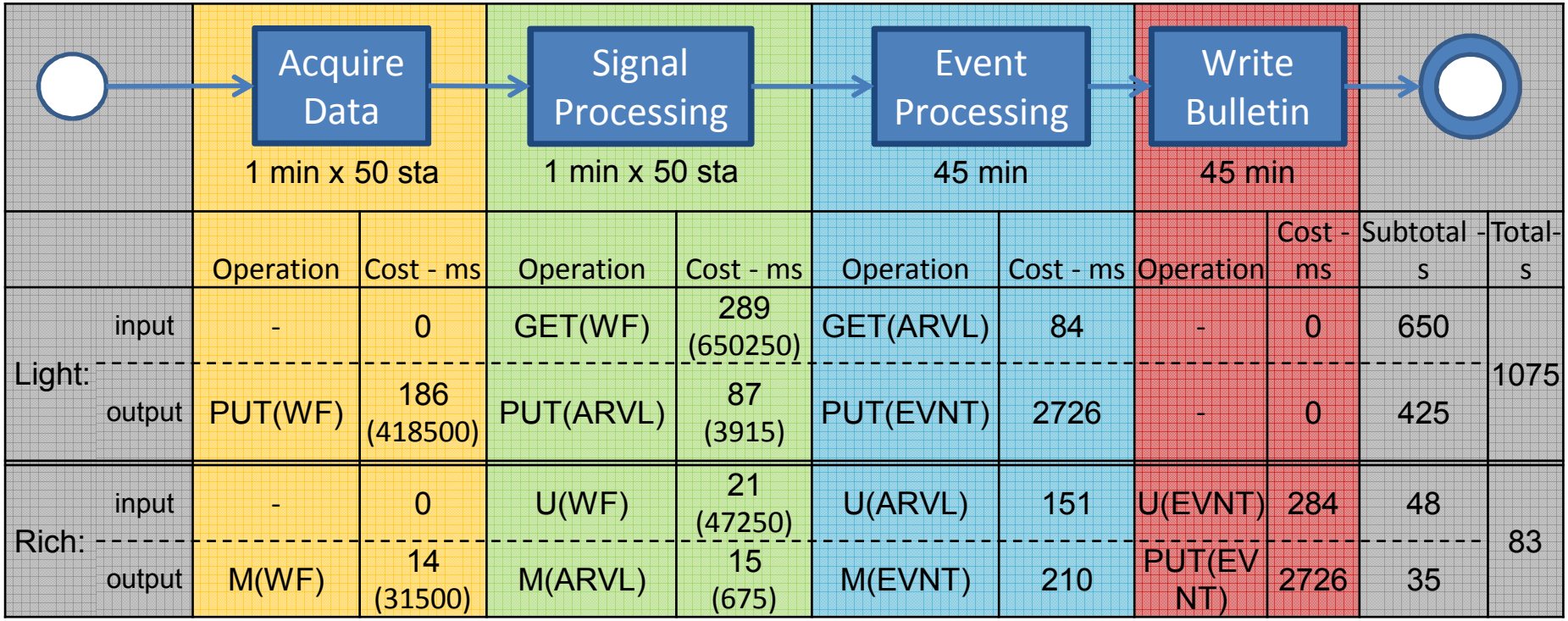


- IDCX data has on the order of 1000 arrivals in a 45 minute time interval
- A sample time period with 1000 arrivals produced 61 automatic events

SOA Study Status: Results

Sandia National Laboratories

Human readable XML; no central pipeline controller; direct access to COI



		Acquire Data: Operation	Acquire Data: Cost - ms	Signal Processing: Operation	Signal Processing: Cost - ms	Event Processing: Operation	Event Processing: Cost - ms	Write Bulletin: Operation	Write Bulletin: Cost - ms	Subtotal - s	Total- s
Light:	input	-	0	GET(WF)	289 (650250)	GET(ARVL)	84	-	0	650	1075
	output	PUT(WF)	186 (418500)	PUT(ARVL)	87 (3915)	PUT(EVNT)	2726	-	0	425	
Rich:	input	-	0	U(WF)	21 (47250)	U(ARVL)	151	U(EVNT)	284	48	83
	output	M(WF)	14 (31500)	M(ARVL)	15 (675)	M(EVNT)	210	PUT(EVNT)	2726	35	

M(X): Marshal X U(X): Unmarshal X GET(X): Query DB for X PUT(X): Store X to DB

Machine Configurations

- Marshaling and unmarshaling: RHEL 6 server; Nehalem Xeon xx5570 processors
- Database: Solaris SunOS 5.10 server; SPARC processor
- Waveform NAS : NetApp FAS3240; 256GB cache

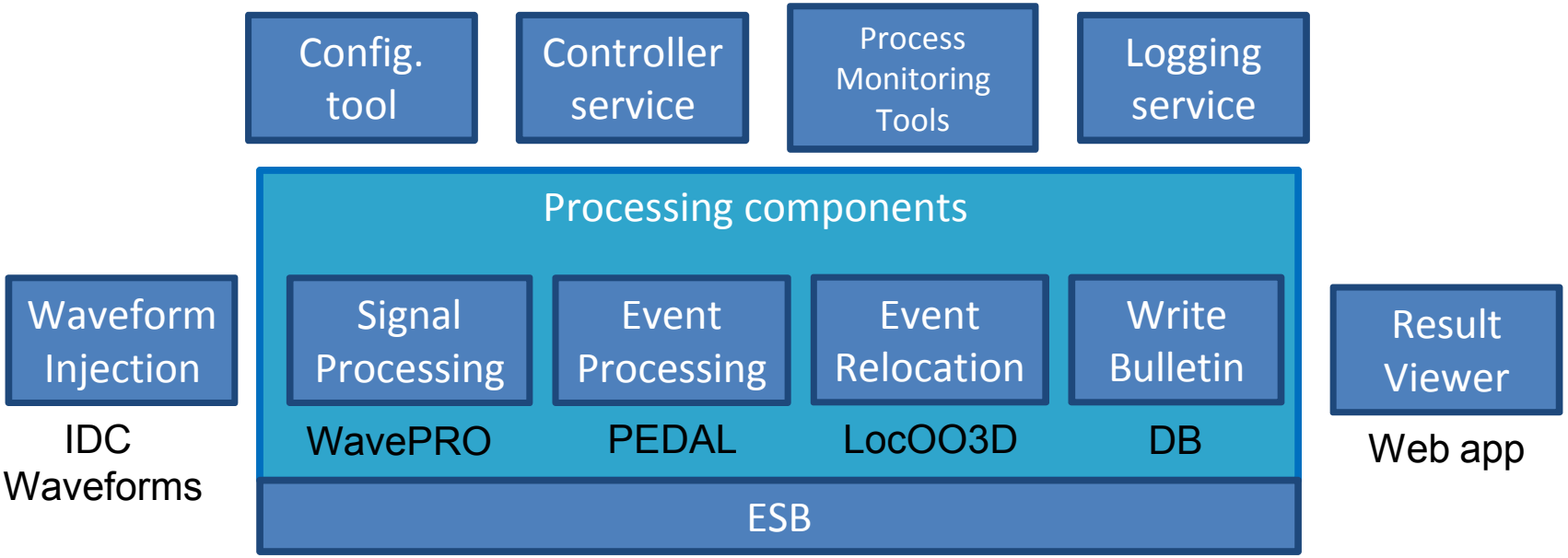
Architectural Prototype Overview

Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- **Service Oriented Architecture (SOA) Proof of Concept**
- E1 Prototyping: Common Object Interface
- E1 Prototyping: Processing Control Framework
- E2 Prototyping: OSD & PC Software Infrastructure
- E1 Prototyping: User Interface Framework
- E2 Prototyping: User Interface Framework

Study: Proof-of-concept

- Configure SOA technologies using SNL tools
- Gain working understanding of SOA for a simple system
- Look at:
  - Configuring service directories
  - Configuring messaging
  - Implications of mixing control flow and data flow



Architectural Prototype Overview

Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- Service Oriented Architecture (SOA) Proof of Concept
- **E1 Prototyping: Common Object Interface**
- E1 Prototyping: Processing Control Framework
- E2 Prototyping: OSD & PC Software Infrastructure
- E1 Prototyping: User Interface Framework
- E2 Prototyping: User Interface Framework

E1: Common Object Interface

- E1 focus
  - Means for persisting data
  - Abstraction of underlying data storage
- The COI includes:
  - Application data model: class model representation of data
  - Application Programming Interface (API): provides SCRUD[1] functionality via the application data model

[1] *Search/Create/Read/Update/Delete*

E1: Common Object Interface

Sandia National Laboratories

- COI Goals
  - Minimize dependencies between applications and data storage solution
  - Decouple logical data model (e.g. database schema) from application data model
  - Provide a query language independent of data storage solution
  - Provide optimizations to support performance requirements
  - Support storage solutions and application languages defined for the system

[1] *Search/Create/Read/Update/Delete*

E1: Common Object Interface

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

E1: Common Object Interface

Sandia National Laboratories

- E1 COI Conclusions
  - Hibernate and ODB are OSS solutions that meet many of the goals outlined for the COI in this prototyping effort including:
    - Minimizing dependencies between applications and data storage solution
    - Decoupling the logical data model (e.g. database schema) from application data model
    - Providing a query language independent of the data storage solution
    - Providing optimizations to support performance requirements
    - Supporting storage solutions defined for the system

Architectural Prototype Overview

Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- Service Oriented Architecture (SOA) Proof of Concept
- E1 Prototyping: Common Object Interface
- **E1 Prototyping: Processing Control Framework**
- E2 Prototyping: OSD & PC Software Infrastructure
- E1 Prototyping: User Interface Framework
- E2 Prototyping: User Interface Framework

E1: Processing Control Framework

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 US NDC modernized architecture (US NDC is not a heterogeneous system of systems).
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 US NDC processing needs particularly well. Not standards-based. Does not support multiple languages (Java only).

E1: Processing Control Framework

Sandia National Laboratories

- **Apache Storm**
  - The E1 Storm prototype demonstrates a flexible, robust, fault-tolerant distributed processing architecture for the automated pipeline
  - Storm is a recent offering (2011), but has generated significant interest in the open source community and has seen significant commercial adoptions since its initial release
  - Storm natively supports processing components built in multiple languages via the JSON *multilang* protocol, including Java & C++
    - JVM languages were easier to work with
    - It is not clear whether the *multilang* protocol provides a significant advantage over JNI or JNA

- **Java EE / Wildfly**
  - The E1 Java EE prototype demonstrates a flexible, robust, fault-tolerant distributed processing architecture for the automated pipeline built on widely-used open standards
  - Wildfly server provides a stable, secure runtime environment for highly-available Java EE applications
  - Together the Java EE APIs and Wildfly server administration tools, documentation and examples enable highly efficient application development
  - Lack of support for direct invocation of non-JVM language software components is a significant limitation
    - Further investigation of options is necessary if Java EE is to be considered further

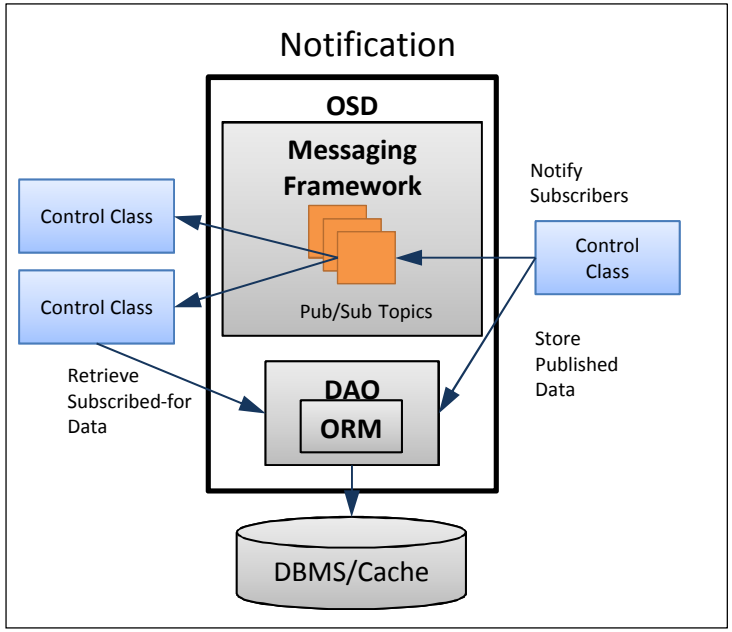
Architectural Prototype Overview

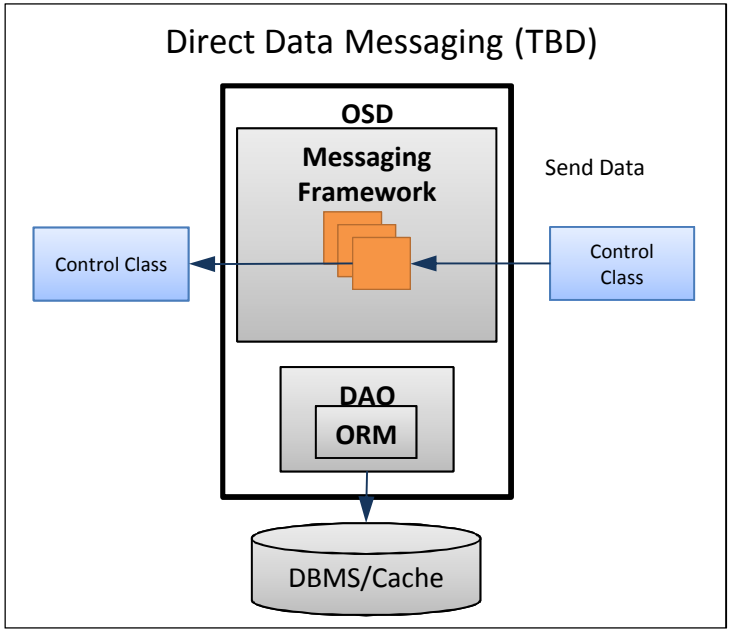
Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- Service Oriented Architecture (SOA) Proof of Concept
- E1 Prototyping: Common Object Interface
- E1 Prototyping: Processing Control Framework
- **E2 Prototyping: OSD & PC Software Infrastructure**
- E1 Prototyping: User Interface Framework
- E2 Prototyping: User Interface Framework

E2: OSD & PC Software Infrastructure

- Messaging COTS Investigation
  - Design Assumption: Two basic data distribution models considered:
    1. Notification: publish/subscribe distribution of data notifications where clients retrieve the indicated data either from the database or distributed cache (TBD)
    2. Direct Data Messaging (TBD): distribution of serialized data where clients receive the data directly
  - E2 Activities: Surveyed open-source messaging frameworks supporting both distribution models
    - Focused on standards-based solutions: AMQP & DDS
    - Selected RabbitMQ as preferred AMQP messaging solution
      - Feature set, cross-language support, performance, popularity





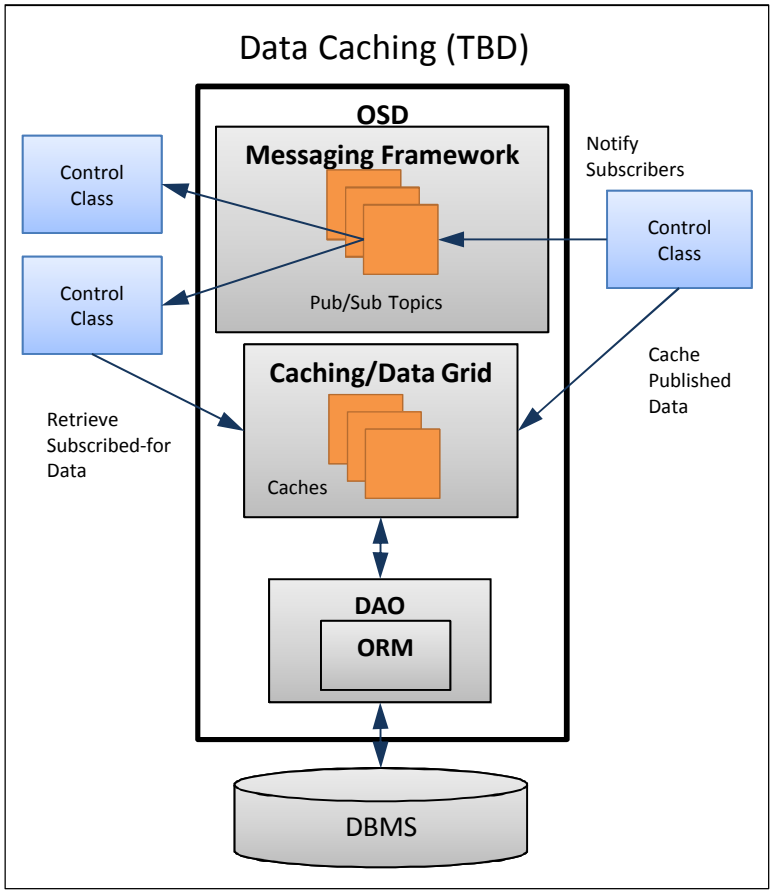
E2: OSD & PC Software Infrastructure

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)

E2: OSD & PC Software Infrastructure

- Design Assumption: The OSD will include data caching facilities to support temporary storage of non-persistent application data and optimized access to data stored in the database
- E2 Activities: Surveyed open-source data caching & data grid frameworks
  - Focused on cross-language, distributed caching solutions
- E3 Path Forward: TBD – The need for a caching solution will be assessed during development of the executable architecture prototype in E3.



E2: OSD & PC Software Infrastructure

Sandia National Laboratories

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)

E2: OSD & PC Software Infrastructure

Sandia National Laboratories

- E2 Goal: Identify OSD & Processing Control COTS for use in executable architecture prototyping starting in E3
- E2 Results:
  - Identified AMQP standard-based messaging COTS for internal OSD data distribution and processing control
    - Evaluation of REST/HTTP for external COI data access underway
  - Continuing with COTS ORM (Hibernate) to support the Java DAO development, based on E2 prototyping results and pending performance evaluation in E3
  - Surveyed data caching solutions for OSD data distribution
    - Selection TBD pending COI design decisions during executable architecture prototyping
  - Completed initial investigation of serialization COTS for cross-language direct data distribution (student project)
  - Demonstrated Python access to Java APIs for scripting language access to DAOs
  - Completed initial survey of Batch processing frameworks for processing control

Architectural Prototype Overview

Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- Service Oriented Architecture (SOA) Proof of Concept
- E1 Prototyping: Common Object Interface
- E1 Prototyping: Processing Control Framework
- E2 Prototyping: OSD & PC Software Infrastructure
- **E1 Prototyping: User Interface Framework**
- E2 Prototyping: User Interface Framework

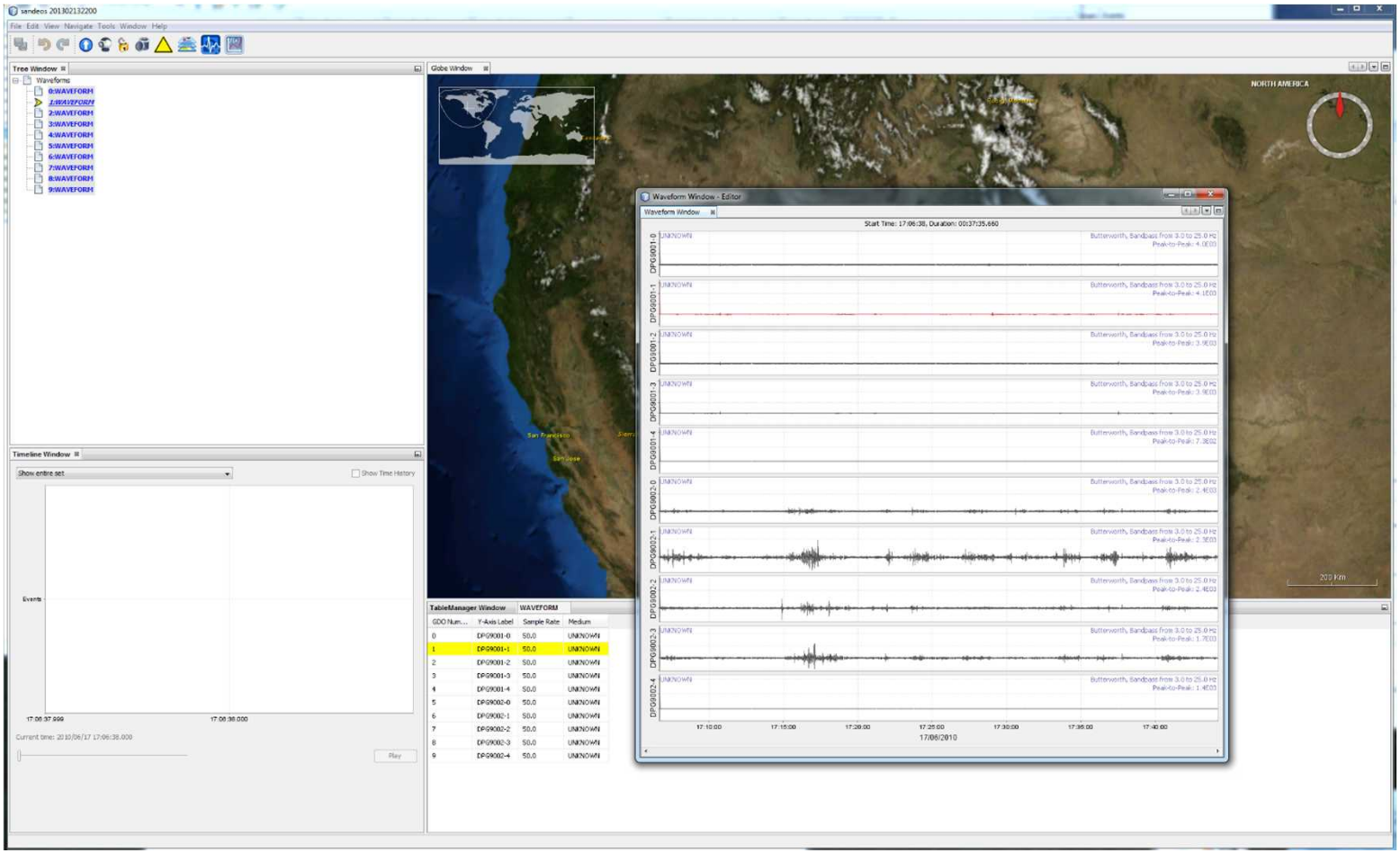
E1: User Interface Framework

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.

E1: User Interface Framework

- Netbeans: Dockable and Floating Displays



E1: User Interface Framework

Sandia National Laboratories

- Conclusions:
  - Netbeans and the Swing widget kit proved to be very strong candidates for feature breadth, customization, plugin support, with very efficient code integration, reuse, and development
    - Prototype goals in these areas were not only met but **exceeded**
  - Exercised on multiple platforms
    - Some difficulties were encountered when the mapping viewer required some platform dependent OpenGL Java libraries – these issues are deemed to be both resolvable and independent of the prototyped Netbeans UIF and the Swing widget toolkit

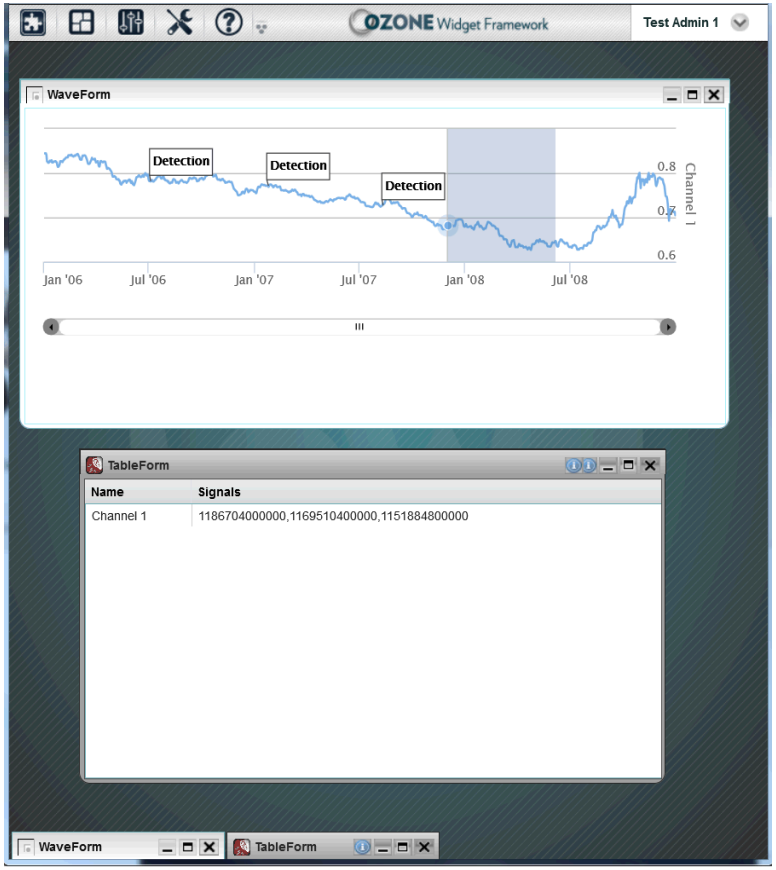
Architectural Prototype Overview

Sandia National Laboratories

- Service Oriented Architecture (SOA) Study Status
- Service Oriented Architecture (SOA) Proof of Concept
- E1 Prototyping: Common Object Interface
- E1 Prototyping: Processing Control Framework
- E2 Prototyping: OSD & PC Software Infrastructure
- E1 Prototyping: User Interface Framework
- **E2 Prototyping: User Interface Framework**

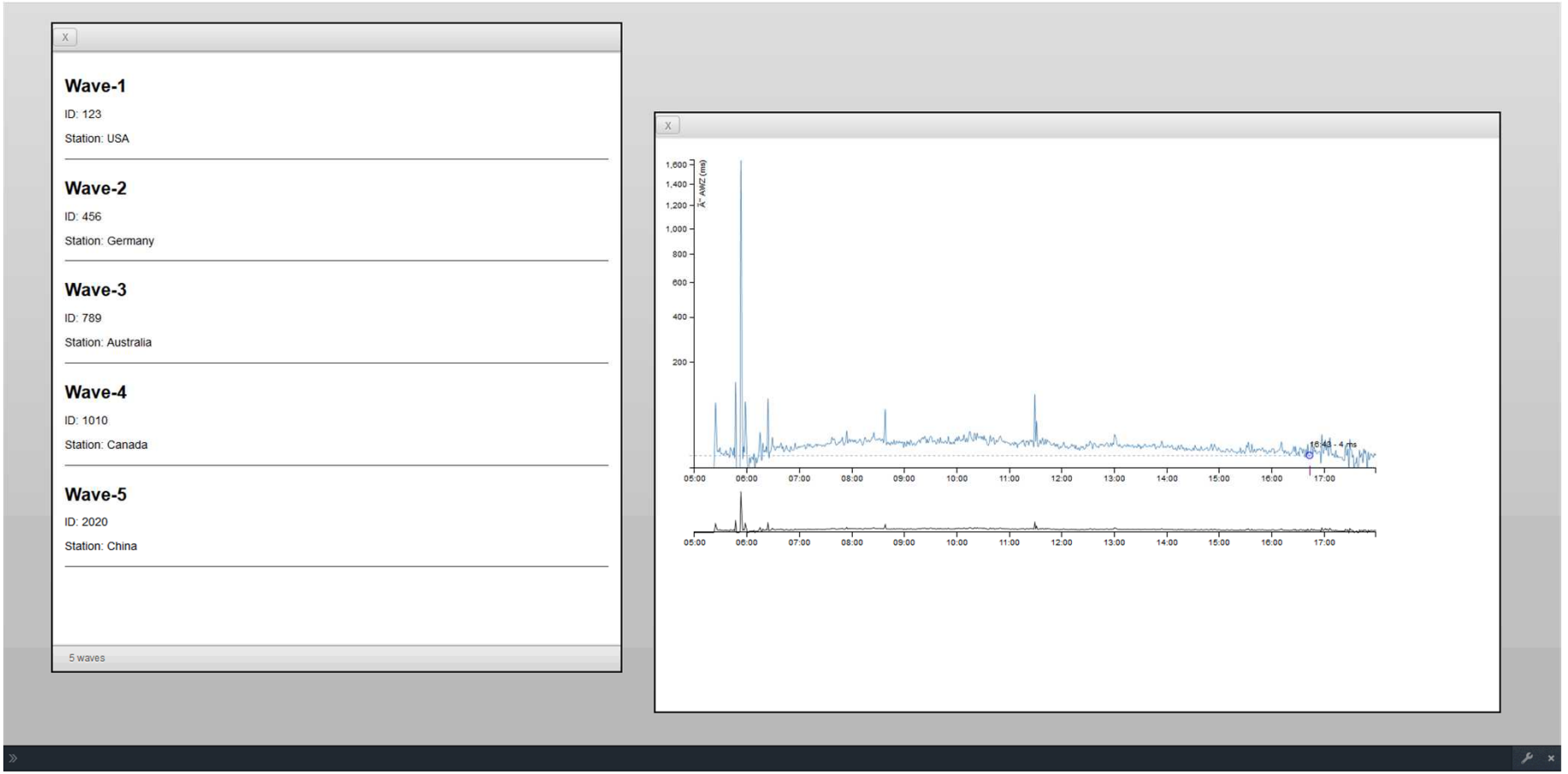
E2: User Interface Framework

- Browser-based UI: OWF with Highstock Plots



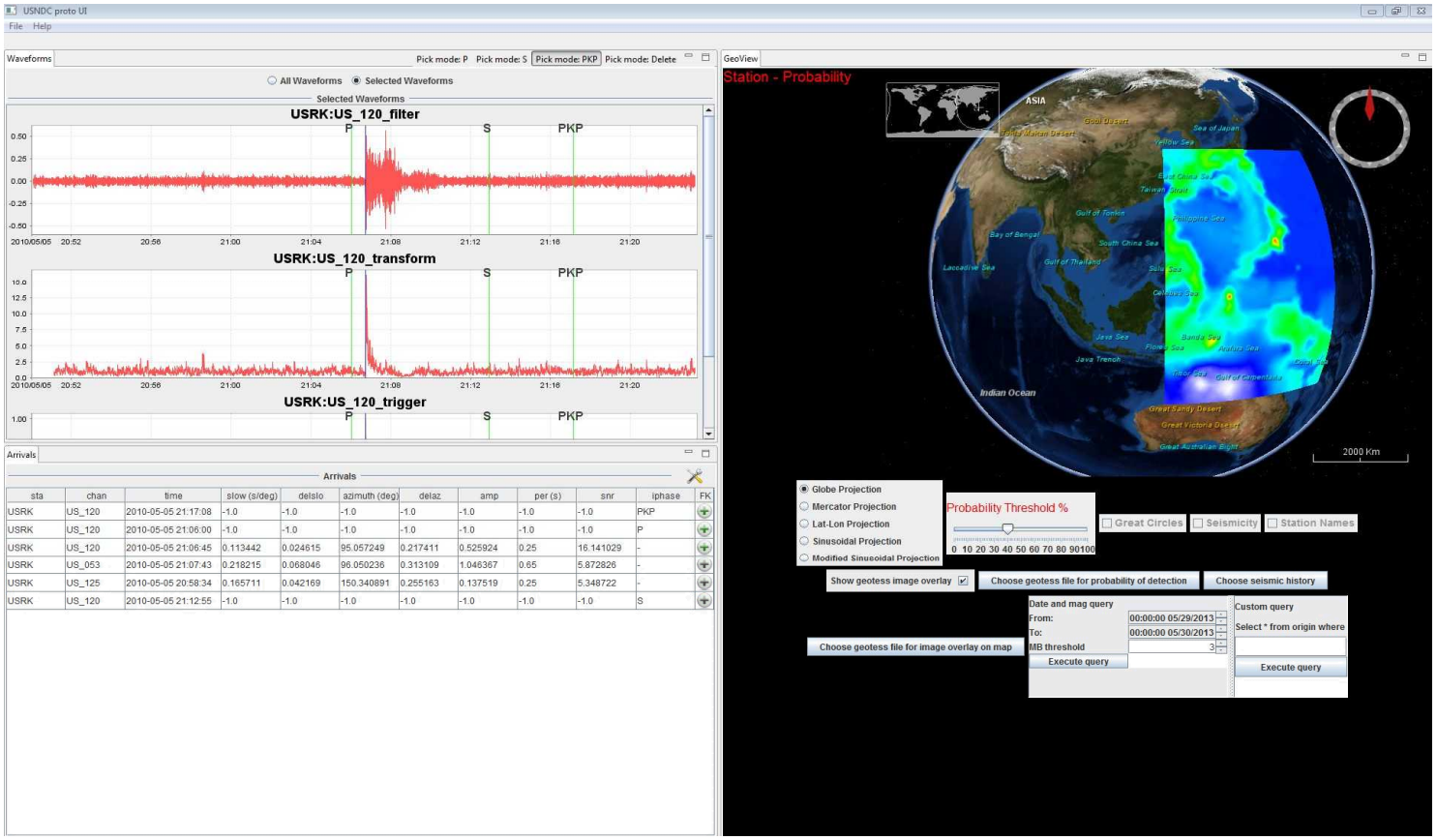
E2: User Interface Framework

- Browser-based UI: SproutCore with D3 Plots



E2: User Interface Framework

- Eclipse 4.x RCP



E2: User Interface Framework

Sandia National Laboratories

- Conclusions
  - OWF, SproutCore, and Eclipse RCP all support window management and workspace customization
  - Because OWF and SproutCore are browser based frameworks, there were concerns related to how they would integrate with the underlying system infrastructure. They also require expertise in web development.
  - While Eclipse 4 RCP is powerful, the lack of support and the dependence on SWT/JFace made the Eclipse 4 RCP a less attractive prototyping candidate than the NetBeans RCP.