Exceptional service in the national interest

SAND2014-0082P

Sandia National Laboratories

IDC / US NDC Modernization

Architecture Exploratory Prototyping

Ryan Prescott

13 January 2014

U.S. DEPARTMENT OF ENERGY | NNSA

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

- IDC/US NDC Common Architecture Definition
  - Overview
  - Timeline
- Exploratory Prototyping
  - Overview
  - Current Activities
  - Current Areas of Focus
  - Status
    - Common Object Interface
    - Automated Processing Control Framework
    - GUI Framework
- Summary

IDC / US NDC Common Architecture Definition - Overview

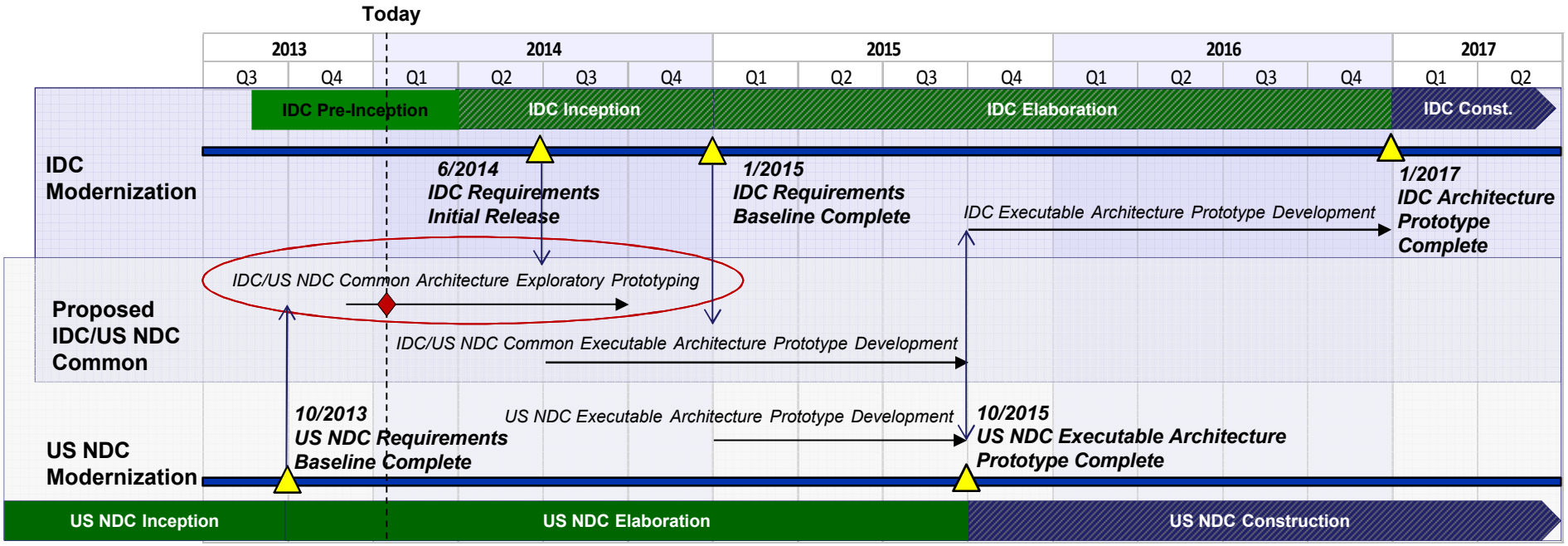
Sandia National Laboratories

- The existing US NDC modernization project scope includes definition of a baseline system architecture and development of an executable architecture prototype
- Under the proposed joint IDC/US NDC modernization project, a common architecture will be defined to support the core requirements shared by the two systems
  - The common architecture will be extended for each system to address the unique requirements of each
- Architecture deliverables include the following:

	Architecture Deliverable	Description
Elaboration Phase	System Architecture Document (draft)	Document significant decisions, structure and behavior of the common software & hardware architecture
	Use Case Realization reports (draft)	Baseline with draft content for all use cases, complete reports for all architecturally significant use cases
	Executable Architecture Prototype	Demonstration of the key features of the architecture

IDC / US NDC Common Architecture Definition- Timeline

- In accordance with the RUP process, baseline architecture definition and executable prototype development will be completed during the elaboration phase for each system
- In order to meet the US NDC modernization project schedule, architecture definition activities are planned to begin Q1 CY 2014
  - The US NDC modernization project elaboration phase is scheduled to complete in October 2015



Exploratory Prototyping – Overview

Sandia National Laboratories

- In order to support architecture definition for the US NDC modernization project, the project team has initiated an exploratory prototyping effort
  - This work will also support IDC and common architecture definition activities under the proposed joint IDC/US NDC modernization project
- The purpose of exploratory prototyping is to inform architecture trades and analysis – e.g.
  - Definition of high-level design patterns
  - Selection of target development languages
  - Selection of key software mechanisms (e.g. processing control, persistence, user interface frameworks) & tools
  - Adopt/Buy/Create trades

Exploratory Prototyping – Current Activities

Sandia National Laboratories

- Current activities (through March 2014) include the following:
  1. Survey of open source software (OSS) and COTS software resources for select areas of the architecture
  2. Assessment of candidate solutions based on survey findings
     - Includes basic prototyping
- Initial survey and assessment is focused on three architecture areas:
  - The Common Object Interface (COI)
  - Processing control framework
  - GUI framework

Timeframe	Activity	Description	Deliverable
Dec 2013	Initial OSS/COTS survey	Identify candidate open source and other COTS solutions supporting the selected architecture areas of focus: COI, automated processing control framework & GUI framework	Survey & Assessment Report (Mar 2014)
Jan – Mar 2014	Follow-on survey and assessment of Select Candidates	Evaluate select candidate solutions, developing basic prototypes and benchmarks as needed; additional survey work as needed	

Current Areas of Focus – Common Object Interface

Sandia National Laboratories

- Definition: A software mechanism providing access to persistent data within the common IDC/US NDC architecture
  - Provides an abstraction of the underlying data storage solutions, and includes the following major elements:
    - Application interfaces for Search/Create/Read/Update/Delete operations
    - An application data model that is independent from the physical representation of persistent data
- Design Goals:
  - Decouple the application data model from the physical data model
    - Minimize dependencies between the application and underlying data storage solutions
  - Support multiple storage solutions & deployment configurations as defined for the modernized IDC & US NDC systems*
  - Support a minimal set of application languages as defined for the transitional and end-state modernized systems*
  - Prefer COTS & OSS solutions where appropriate

* Development languages and storage solutions have not yet been fully defined for the common modernized architecture. One objective of exploratory prototyping is to inform these trades.

Current Areas of Focus – Automated Processing Control Framework

Sandia National Laboratories

- Definition: A software mechanism providing for the development, configuration, execution and control of automated processing components within the common IDC/US NDC architecture
  - Includes messaging between processing components (data and flow of control)
- Design Goals:
  - Provide a fault-tolerant, scalable, distributed processing model
  - Provide for definition, configuration & deployment of automated processing components
  - Provide capabilities to facilitate integration of new processing algorithm implementations
  - Support multiple deployment platforms and a minimal set of application languages as defined for the transitional and end-state modernized systems*
  - Prefer COTS & OSS solutions where appropriate

* Development languages have not yet been defined for the common modernized architecture. One objective of exploratory prototyping is to inform this trade.

Current Areas of Focus – GUI Framework

Sandia National Laboratories

- Definition: A software mechanism providing for the development, configuration, execution and control of Graphical User Interface (GUI) displays supporting users of the modernized IDC & US NDC systems
  - Includes e.g. GUI widget toolkits & GUI builders, UI application development frameworks (e.g. RCP plug-in architecture) and support for applicable design patterns (e.g. Model View Controller)
- Design Goals:
  - Support a consistent (common-look-and-feel) modern user interface standard for all display elements
  - Provide an extensible GUI architecture
  - Provide support for a minimal set of user interface languages as defined for the modernized US NDC system*
  - Prefer COTS & OSS solutions where appropriate

* Development languages have not yet been defined for the common modernized architecture. One objective of exploratory prototyping is to inform this trade.

Status - Common Object Interface (1 of 3)

Sandia National Laboratories

- The COI survey work to date has focused on open-source Object Relational Mapping (ORM) & similar data abstraction solutions
  - ORMs generally provide an object-oriented abstraction layer supporting application access to data stored in Relational Database Management Systems (RDBMS)
  - Relevance to the COI is based on the assumption that at least alphanumeric data will continue to be stored using relational DBMS solutions
- Note that ORM solutions do not themselves provide a complete COI solution
  - Waveform data may continue to be stored outside the RDBMS, necessitating additional access interface software
  - Data caching and other performance optimizations may be required that are not provided by the ORM solution
  - Follow-on exploratory prototyping will address these issues as needed

Status – Common Object Interface (2 of 3)

Sandia National Laboratories

- ORM and other solutions surveyed thus far include the following:

Solution	Summary	Advantages	Disadvantages
Hibernate	Java Object Relational Mapping (ORM) OSS	Prevalence, strong development community, standard ORM features, support for relevant RDBMSs, optimization support (e.g. lazy initialization, fetching strategies)	Supports Java only
OpenJPA	Java ORM OSS	Prevalence, strong developer community, standard ORM features, support for relevant RDBMSs	Supports Java only
Apache Cayenne	Java ORM OSS	Standard ORM features, caching support, schema mapping support (UI tool), support for relevant RDBMSs	Supports Java only, less prevalent than Hibernate & OpenJPA, development appears to be inactive
Apache Empire-DB	Java RDBMS Abstraction OSS	Support for relevant RDBMSs	Supports Java only, weaker abstraction of the physical schema (exposes SQL), less prevalent & mature (Apache incubator project)
Apache Torque	Java ORM OSS	Standard ORM features, Support for relevant RDBMSs	Supports Java only, less prevalent
ODB	C++ ORM OSS	standard ORM features, support for relevant RDBMSs, optimization support, STL & Boost integration	Supports C++ only
QxORM	C++ ORM OSS	standard ORM features, support for relevant RDBMSs	Supports C++ only, more complex interface than ODB (heavy use of macros and templates), less prevalent than ODB, sparse documentation

Status – Common Object Interface (3 of 3)

Sandia National Laboratories

- General findings include the following:
  - None of the solutions surveyed provides cross-language support
    - The solutions surveyed support only Java or C++ interfaces
  - Java solutions are more prevalent than C++ solutions
  - All surveyed solutions support multiple RDBMSs: Oracle, PostgresSQL, MySQL, SQL Server, SQLite
- Follow-On work:
  - Based on initial survey results and available resources, follow-on assessment & prototyping work through March 2014 will focus on Hibernate (Java) and ODB (C++)
  - Additional survey work will be conducted as needed based on assessment findings and additional candidates identified

Status – Automated Processing Control Framework (1 of 3)

Sandia National Laboratories

- The initial survey focused on two types of framework:
  1. Java Enterprise Application frameworks (Spring & WildFly)
     - Provide a robust set of enterprise application features for java applications
     - Implementation of the Java EE specification (WildFly) or similar alternative (Spring)
  2. Stream Processing Frameworks (Apache Storm, Samza, S4)
     - Provide distributed, fault-tolerant, scalable processing control for analysis of real-time stream data

Status – Automated Processing Control Framework (2 of 3)

- Processing control frameworks surveyed thus far include the following:

Solution	Summary	Advantages	Disadvantages
Spring	Java enterprise application framework OSS	Prevalence, strong developer community, robust enterprise application feature set	Supports Java only, greater complexity than stream processors
Java EE – Wildfly (JBoss)	Java EE application framework OSS	Prevalence, strong developer community, robust enterprise application feature set	Supports Java only, greater complexity than stream processors
Apache Storm	Stream processing OSS	Supports both Java and non-Java (C++, Python) processing components, strong feature set (fault tolerance, scalability, durability & state persistence), most popular and widely-used stream processor, relatively simple API	Relatively new (2011 Apache incubator), less prevalent than Enterprise Java solutions
Apache Samza	Stream processing OSS	Strong feature set (fault tolerance, scalability, durability & state persistence), relatively simple API	New (2013 Apache incubator), less prevalent than Enterprise Java solutions & Storm, currently supports only Java
Apache S4	Stream processing OSS	Strong feature set (fault tolerance, scalability, durability & state persistence)	Supports Java only, apparently few users and little development interest

Status – Automated Processing Control Framework (3 of 3)

Sandia National Laboratories

- General findings include the following:
  - Only one of the solutions surveyed (Apache Storm) provides explicit cross-language support for processing components
  - Enterprise Java solutions provide a more extensive feature set (e.g. full implementation of the Java EE standards) at the expense of greater complexity than stream processors, which are more narrowly focused on distributed processing control
- Follow-On work:
  - Based on initial survey results and available resources, follow-on assessment & prototyping work through March 2014 will focus on Wildfly (Java EE) and Apache Storm (stream processor) frameworks
  - Additional survey work will be conducted as needed based on assessment findings and additional candidates identified

Status – GUI Framework (1 of 3)

Sandia National Laboratories

- The initial survey focused on two types of solution:
  1. GUI Application Frameworks (JavaFX, Qt, WxWidgets)
     - Provide widget toolkits, GUI builders & IDE integrations supporting graphical display development as well as components supporting application development (e.g. data access, communication facilities & design pattern abstractions)
  2. GUI Rich Client Platforms (Eclipse/SWT/JFace & Netbeans/Swing)
     - Provide an extensible application architecture supporting integration of independent GUI components (e.g. OSGi plugin architecture) in addition to standard GUI application elements (widget toolkit, GUI builders, IDE integrations, etc.)

Status – GUI Framework (2 of 3)

Sandia National Laboratories

- GUI frameworks surveyed thus far include the following:

Solution	Summary	Advantages	Disadvantages
JavaFX 2	Java GUI application framework	Growing prevalence as the replacement for Swing, desktop and browser deployment options	Relatively new, less mature than SWT/JFace & Qt, does not include an extensible component/plugin-based architecture (other than future Netbeans integration)
SWT/JFace/Eclipse RCP	Java Rich Client Platform (RCP)	Prevalence, developer community, maturity, extensible plugin architecture	complexity & learning curve
Swing/Netbeans RCP	Java Rich Client Platform (RCP)	Prevalence, developer community, maturity, extensible plugin architecture	Swing will be replaced with JavaFX in the future, mature JavaFX integrations for Netbeans are not yet available, complexity & learning curve
Qt	C++ GUI application framework	Prevalence, developer community, maturity, possibly some performance advantages	Does not include an extensible component/plugin-based architecture (limited & less prevalent Qt-based RCP options exist – e.g. GCF)
WxWidgets	C++ GUI application framework	Maturity	Does not include an extensible component/plugin-based architecture, less prevalent than Qt and Java Frameworks

Status – GUI Framework (3 of 3)

Sandia National Laboratories

- General findings include the following:
  - The two prominent rich client platforms (Eclipse and Netbeans) provide an extensible OSGi-based plug-in architecture
    - The other frameworks surveyed do not provide an equivalent capability
  - All frameworks provide cross-platform support for Linux, Unix, Windows & Mac
    - Likely not a significant driver for the US NDC / IDC architecture
  - Java GUI solutions are more prevalent than C++ solutions
- Follow-On work:
  - Based on initial survey results and available resources, follow-on assessment & prototyping work through March 2014 will focus the surveyed RCP solutions (Eclipse and Netbeans)
  - Additional survey work will be conducted as needed based on assessment findings and additional candidates identified

Summary

- An exploratory prototyping effort is underway to support US NDC/Common architecture definition
- Initial survey and assessment results for the COI, automated processing control framework and GUI framework will be completed March 2014
- Under the proposed joint IDC/US NDC modernization project, continued exploratory prototyping will provide ongoing support for definition of the common IDC/US NDC architecture