SAND2014-20635R

US NDC Modernization

SAND-xxxx
Unclassified Unlimited Release
December 2014

US NDC Modernization Iteration E1 Prototyping Report: User Interface Framework

Version 1.1

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

SAND-xxxx
December 2014

US NDC Modernization Iteration E1 Prototyping Report: User Interface Framework

Randall R. Lober

Version 1.11
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185

ABSTRACT

During the first iteration of the US NDC Modernization Elaboration phase (E1), the SNL US NDC modernization project team completed an initial survey of applicable COTS solutions, and established exploratory prototyping related to the *User Interface Framework (UIF)* in support of system architecture definition. This report summarizes these activities and discusses planned follow-on work.

REVISIONS

Version	Date	Author/Team	Revision Description	Authorized by
1.0	3/18/2014	US NDC Modernization Team	Initial Release	M. Harris
1.1	12/19/2014	IDC Reengineering Team	IDC Release	M. Harris

TABLE OF CONTENTS

US NDC Modernization Iteration E1 Prototyping Report: User Interface Framework 3

Abstract 3

Revisions 4

Table of Contents 5

1. Overview 7

2. Schedule 7

3. Motivation 8

4. User Interface Framework 8

4.1. Definition 8

4.2. Design Goals 8

4.3. Constraints 9

4.4. Iteration E1 Prototyping Activities 9

4.4.1. Initial COTS Survey 9

4.4.1.1. NetBeans UIF 10

4.4.1.1.1. NetBeans UIF and Swing Widget Toolkit 11

4.4.1.1.2. NetBeans UIF and JavaFX Widget Toolkit 12

4.4.1.2. Eclipse UIF and JFace Widget Toolkit 12

4.4.1.3. Qt Creator UIF and Qt Widget Toolkit 13

4.4.1.4. wxWidgets Widget Toolkit 14

4.4.1.5. XUL Widget Toolkit 15

4.4.2. Exploratory Prototyping 15

4.4.2.1. Exploratory Prototype: NetBeans UIF using Swing Widgets 16

4.4.2.1.1. Configuring NetBeans for Development 16

4.4.2.1.2. Extending the Prototype for Seismic Waveform 2D Plotting 17

4.4.2.1.3. Evaluating Concurrency within Prototype Display Widgets 18

4.4.2.1.4. Enabling and Evaluating Netbean's RCP Plugin Usage 18

4.4.2.1.5. Assessing Usability, Performance, and Customization of the Prototype 19

4.4.2.1.6. Conclusions on the Prototype and NetBeans RCP 20

4.5. Follow-On Work 21

Appendix A. Supporting Tables 22

1. OVERVIEW

The US NDC Modernization project statement of work identifies the definition of a modernized system architecture as a central project deliverable. As part of the architecture definition activity, the Sandia National Laboratories (SNL) project team has established an ongoing, software prototyping effort to support architecture trades and analyses, as well as selection of core software technologies.

During the first iteration of the US NDC Modernization Elaboration phase (E1), spanning Q1 - Q2 FY2014, the prototyping effort included initial COTS surveys and exploratory prototyping addressing three core elements of the system architecture:

1. The *Common Object Interface (COI)* provides the system and research tools with access to persistent data via an abstraction of the underlying storage solutions.

2. The *processing control framework* provides for the definition, configuration, execution and control of processing components within the system, supporting both automated processing and interactive analysis.

3. The *User Interface Framework (UIF)* provides a flexible platform for the definition of extensible graphical user interface (GUI) components & composition of GUI displays supporting users of the system and research tools.

This report summarizes the iteration E1 prototyping activities of the SNL project team *specific to the User Interface Framework*. E1 prototyping activities for the processing control framework and COI are described in separate reports.

2. SCHEDULE

This report summarizes the UIF prototyping work completed during the three-month period from December 2013 to February 2014, based on the following schedule.

Period	Activity
December 2013	OSS/COTS survey
January – February 2014	Initial Exploratory Prototyping

3. MOTIVATION

Prototyping provides input critical in the definition of the system architecture, supporting selection of core software development languages and technologies, identification of architecture constraints & assumptions, and definition of high-level design patterns. In addition, the prototyping activity provides a foundation for development of the executable architecture deliverable.

4. USER INTERFACE FRAMEWORK

4.1. Definition

The User Interface Framework (UIF) is a software mechanism[1] providing a standard architecture, a set of application programming interfaces (APIs), and a runtime environment supporting development, integration and deployment of modular, extensible graphical user interface (GUI) software components. The UIF framework consists of the following separate components:

- UIF Framework development environment and standardized architecture
- A widget toolkit providing standard graphical elements for GUI display development

4.2. Design Goals

- Provide a single unified UIF to be used for all internal system displays.
- Support a consistent, modern user interface standard for all system graphical user interface display elements.
- Support a responsive and customizable user interface.
- Provide a modular, component-based architecture for development, integration, and deployment of extensible GUI components. Support the multiple application languages defined for the modernized system.

[1] J. Burns, et al, *System Architecture Document Version 1.0 DRAFT*, Sandia National Laboratories, Albuquerque, New Mexico 87185, March 2014, p. 14.

4.3. Constraints

- Performance: the UIF will support the user interface performance requirements
- Extensibility: the UIF will enable developers to build new GUI features and code plugins (via standard APIs). The UIF cannot preclude the ability to access underlying system functionality via a command-line-interface (CLI).
- Interfaces with other frameworks: the UIF will interoperate with the COI (retrieval and storage of configuration and processing results) and the processing control framework as appropriate.
- Reuse: the UIF will prefer Open Source Software (OSS) and other Commercial Off-The-Shelf (COTS) solutions to custom software development where available.
- Standardization: the UIF will prefer solutions based on open standards wherever possible.

4.4. Iteration E1 Prototyping Activities

The E1 prototype work focused on ensuring an array of UIF candidate solutions that meet at least a majority of the design goals and constraints outlined for the UIF were considered. Methods for identifying the survey set for UIF candidate solutions included capability, industry presence, and local development experience within the broader Sandia software community.

This approach involved identifying and surveying a moderately sized COTS set of UIF solution candidates, then applying the knowledge gained to execute a hands on evaluation with the selected UIF solution candidate from the survey evaluation stage.

4.4.1. Initial COTS Survey

For the UIF prototype E1 effort, two main considerations were key: 1) identifying an application framework (UIF) and 2) identifying an accompanying graphical widget toolkit to deploy. A given UIF may be able to accommodate more than one graphical widget toolkit type. This means that in the evaluation documented below there are two different evaluations of the NetBeans UIF: the first paired with the Swing widget toolkit and the second paired with the JavaFX widget toolkit.

UIF Solution	Widget toolkit	UIF language
NetBeans	Swing	Java (RCP)
NetBeans	JavaFX 2	Java (RCP)
Eclipse	JFace (SWT)	Java (RCP)
Qt Creator	Qt	C++
N/A	wxWidgets	C++
N/A	XUL	XML & Java

Additional details describing the UIFs and widget toolkit combinations that were considered is contained in **Table 1. UIF and Widget toolkit Summary** in ***Error! Reference source not found.***.

Multiple communities of developers exist with strongly preferred UIF application frameworks and widget toolkit solutions that can meet most of the design goals, constraints, and requirements for the modernized system. No single UIF candidate solution meets every design goal out of the box (specifically the CLI availability constraint), however it is estimated that the strongest candidates could be extended to meet all the design goals.

Some of the surveyed candidates are considered rich client platforms[2] (RCP), which provide additional development and deployment advantages such as the enhanced ability to integrate and deploy independent software components while the RCP developers manage the cross-platform support burden.

Of the most promising top three UIF solution candidates, two were of the RCP type (NetBeans and Eclipse).

4.4.1.1. NetBeans UIF

NetBeans is the strongest candidate within the Java oriented UIF solutions. NetBeans is under active development (Oracle) and has extensive support options and documentation.

The NetBeans UIF provides all the advantages of an RCP - specifically the ability to leverage the presence of NetBeans on most modern versions of major compute platforms such as Windows, Linux, Solaris, and Mac OS X. If the preferred compute platform of choice is not listed the source code to NetBeans is available for download and compilation.

[2] http://en.wikipedia.org/wiki/Rich_client_platform

As an RCP, NetBeans enables the efficient development, integration, and deployment of plugins and templates. This provides advantages to a development team that may be geographically dispersed as it enables the rapid sharing of code plugins and templates. The code plugins also allow the option of being deployed from a locally controlled server.

The NetBeans platform offers reusable services common to desktop applications, allowing developers to focus on their application specific code. These reusable services include user interface management (full suite of menus and toolbars), user settings management (user customization), storage management (saving and loading any kind of data), window management, wizard framework (supporting step by step dialogs), progress reporting, NetBeans Visual Library (extensible Java widget library), and an integrated development tool suite (including grammar sensitive editors).

NetBeans has historically supported the Swing widget toolkit although the next generation widget toolkit JavaFX has been available for a few years (starting with NetBeans 6.1) and will become the default supported widget toolkit for NetBeans at some point in the future. Oracle is committing to supporting both Swing (as part of the Java SE specification) and JavaFX for the foreseeable future.

The NetBeans deployment support is predominately targeted to desktop applications. As an RCP, NetBeans is held in high regard by the Sandia development teams that were surveyed. NetBeans has earned the reputation of being a very efficient UIF to configure for new development efforts, having a reasonable learning curve for developers. As a development environment, NetBeans was seen as slightly less reliable or robust than Eclipse.

4.4.1.1.1. NetBeans UIF and Swing Widget Toolkit

The Swing widget toolkit has extensive market presence and active developer communities. The Swing widget toolkit is beginning to see some slowdown in developer discussion threads as the newer JavaFX 2.0 is gaining more attention. This would pose more of a concern if Swing did not easily integrate into code bases alongside JavaFX development.

Swing is currently the primary Java GUI widget toolkit providing a more sophisticated set of GUI components than earlier generation Java GUI widgets (Abstract Window Toolkit - AWT). In addition to expected GUI components such as buttons, check boxes, and labels, Swing provides more advanced components such as tabbed panels, scroll panes, trees, and lists. Swing components are platform independent.

Pairing the market share dominant Swing widget toolkit with the NetBeans UIF results in a very strong candidate.

4.4.1.1.2. NetBeans UIF and JavaFX Widget Toolkit

The JavaFX widget toolkit is gaining market presence and is the most modern of the Java widget toolkits. JavaFX does not currently support Solaris.

While an attractive toolkit, it is not surprising that it has its share of growing pains related to the new code base. A pertinent example is the new JavaFX 2D plotting capability - its feature set and appearance are top rate. The issue is the design of the new plotting capability made the assumption that every point on a curve is a separate object - this decision drastically impedes scalability for any curve plotting with more than 1000 points.

The JavaFX widget toolkit is a decent widget toolkit and may become the widget toolkit of choice in the near future. At this time it is still changing rapidly and growing, as such it has some challenges and is not as stable as Swing or JFace/SWT.

The combination of NetBeans UIF and JavaFX widget toolkit will eventually become the candidate of choice, but that time is not yet here.

4.4.1.2. Eclipse UIF and JFace Widget Toolkit

Eclipse/JFace is a strong runner up to NetBeans/Swing among the Java oriented UIF solutions; and the JFace/SWT[3] widget toolkit has extensive market presence and developer communities. Eclipse is under active development and has extensive support options and documentation. IBM has contributed significantly to the Eclipse open source effort and continues to help drive its development.

The Eclipse UIF also provides all the advantages of an RCP - specifically the ability to leverage the presence of Eclipse on most modern versions of major compute platforms such as Windows, Linux, and Mac OS X.. As an RCP, Eclipse also enables the efficient development, integration, and deployment of plugins and templates.

JFace/SWT is a very popular development widget toolkit with millions of downloads of their open source software base. General comments among the development community indicate stability is a higher priority among the committing and contributing community of Eclipse as opposed to rapid innovation and change. SWT will leverage native platform widgets when possible and this can help boost performance compared to Swing - however Swing performance has increased every year so this may no longer be a strong consideration. Due to a design decision by Eclipse, customization is subject to

[3] Standard Widget Toolkit (www.eclipse.org/swt/)

constraints imparted by that same use of native platform widgets. When compared to the most recent JavaFX 2.0 widget sets, the JFace/SWT widget toolkit feels slightly dated.

The Eclipse platform offers comparable reusable services common to desktop applications, allowing developers to focus on their application specific code.

The Eclipse deployment support is targeted to desktop applications and browsers. Eclipse is a solid RCP, although its reputed setup difficulty is hindering its broader adoption. As a development environment (once configured), Eclipse is viewed as a very strong candidate.

Eclipse has garnered the reputation of being a challenging UIF to configure for new development efforts, having a steep learning curve for developers. This assessment is shared among all the Sandia development teams we surveyed for background information. JFace/SWT is a strong widget toolkit candidate.

The combination of Eclipse and JFace/SWT makes a solid candidate that is still popular and very proven over time.

4.4.1.3. Qt Creator UIF and Qt Widget Toolkit

Qt Creator is the strongest candidate among the C++ oriented UIF solutions, and the Qt widget toolkit has strong market presence and engaged developer communities. Qt is under active development (Digia) and has extensive support options and documentation. With Qt 5, governance of the Qt framework development has moved to the open source community at qt-project.org.

While not a Java RCP, Qt Creator does have standard supported builds for most modern versions of all the main compute platforms and additionally the main mobile platforms such as Windows, Linux (32 and 64 bit precision), and Mac OS X, iOS, and Android.

Qt is a very popular development cross-platform widget toolkit with a large active development community. Of note is the fact that Qt is the UIF used to develop and deploy the KDE Linux windowing system (6 million lines of code not including Qt). The KDE development community is the second largest Free Software community behind the Linux kernel community. The Qt UIF has a reputation for a stable API. Qt also powered the Symbian mobile operating system previously used by Nokia (Nokia sold Qt rights and licensing to Digia in 2012). Qt widgets are native and hence quite high performance. From Sandia development team interviews Qt was ranked as having the strongest cross-platform consistent look-and-feel for GUI behavior (when compared to Eclipse).

The Qt Creator UIF offers comparable reusable services common to desktop applications, allowing developers to focus on their application specific code.

The Qt Creator deployment support is targeted to desktop applications and mobile operating systems. As a development environment, Qt Creator is viewed as a strong candidate with a reasonable developer learning curve. Qt is a strong widget toolkit candidate.

The combination of Qt Creator and Qt makes a strong solution candidate if your solution space leans toward C++.

4.4.1.4. wxWidgets Widget Toolkit

wxWidgets (formerly wxWindows) is a C++ widget toolkit and tools library for creating GUIs for cross-platform applications. It is not an RCP or a true UIF in the sense of Qt Creator. wxWidgets is under active development (open source community) and has open source and commercial support options and documentation.

A recently contributed package (NuGet) now supplies a template allowing developers to build wxWidgets applications within Microsoft Visual Studio. wxWidgets supports GUI builds on the main compute platforms such as Windows (all versions since Windows NT), Linux (all modern versions), other Unixes, and Mac OS X (all modern versions).

The wxWidgets widget toolkit offers comparable reusable services common to desktop applications, allowing developers to focus on their application specific code. wxWidgets has a reasonable community of developers, although not on a par with the Java RCP candidates or Qt.

wxWidgets is a native mode toolkit in that it provides a thin abstraction to a platform's native widgets, contrary to emulating the display of widgets using graphic primitives. Calling a wxWidgets native element on the target platform will tend to result in a "more" native looking interface than toolkits based on Java Swing toolkits. This is also reputed to offer performance benefits (when compared to Swing). wxWidgets is not limited solely to GUI development; it also includes an inter-process communication layer and socket networking functionality.

The wxWidgets widget toolkit has an active if smaller community although as a potential candidate it would be a second tier contender.

4.4.1.5. XUL Widget Toolkit

XUL (pronounced ZOOL) stands for XML User Interface Language. XUL is a user interface markup language developed by Mozilla. As XUL is implemented as an XML dialect, it allows for the construction of graphical user interfaces written in a similar manner as web pages. XUL can be used to construct cross-platform applications such as Mozilla Firefox. The layout engine Gecko interprets XUL and renders the user interface. XUL relies on multiple existing web standards including CSS, JavaScript, and DOM. Developers with backgrounds in web page design find XUL relatively easy to learn.

XUL has no formal specification and does not operate with non-Gecko implementations. It does use an open-source version of Gecko. While XUL is not an RCP or a true UIF, it provides an experimental XULRunner build to allow developers to build their applications on top of the Mozilla application framework.

XUL defined widgets are portable and can move easily between any platforms that support Mozilla applications.

The XUL widget toolkit offers a common set of reusable services useful to desktop applications, although it is not clear of the depth of these services and features are really comparable with the Java RCP candidates, Qt, or even wxWidgets.

XUL deployment support is predominately targeted to browsers and some desktop applications.

The CEO of Mozilla has recently made public comments indicating interest in moving away from XUL and toward an Android based tool - this information alone would likely disqualify XUL as a viable future-proof candidate.

4.4.2. Exploratory Prototyping

In addition to online background research to help with initial candidate rankings, the UIF prototyping team conducted interviews of multiple Sandia software teams that use a variety of the UIF solution candidates across different technical discipline areas.

As an additional objective for the interviews with the various Sandia software teams, the UIF prototyping team searched for potential partner Sandia software teams who would be willing to provide consulting and startup support for the prototyping effort. This approach was viewed as the most efficient way to leverage the limited labor resources and generate the maximum level of insight in the limited time available. Out of the Sandia teams interviewed, three teams were developing (currently released to customers) Java Swing applications. One of the three teams employed NetBeans, one team used Qt Creator and Qt, and one team was using Eclipse and JFace/SWT.

Out of the COTS survey set of UIF candidate solutions a more extensive evaluation was made of the NetBeans UIF using the Java Swing widget toolkit candidate.

For this evaluation the UIF prototyping team partnered with a Sandia software team developinga Java Swing application using the NetBeans UIF.

The anticipated activities for the more in depth exploratory prototyping were to construct several typical seismic style GUI displays and in the process assess the effectiveness of the selected UIF for

- Usability (subjective but important insight capture)
- Performance
- Integration effort
- Plugin API standardization and support
- Inter process communication
- Configuration and setup effort

4.4.2.1. Exploratory Prototype: NetBeans UIF using Swing Widgets

A local UI development project at Sandia was interested in collaborating with the US NDC Modernization prototyping effort. This collaboration was advantageous to the US NDC effort as it would help us rapidly bootstrap into a functional framework that already had several interesting graphical widgets close to ones that we were wanting to evaluate - these widgets included tree and tabular data display widgets and a zooming World Wind mapping widget. The work would entail the integration of locally available software components (such as a 2D waveform trace plotter) from other environments and would exercise the integration and extensibility of both the project's software architecture and the NetBeans development environment.

4.4.2.1.1. Configuring NetBeans for Development

Our target prototyping environment was an Ubuntu Linux environment expected to be somewhat similar to the delivered system's eventual deployment environment. Since the development environment was a true RCP (NetBeans), we had the advantage of being able to develop on our choice of platforms. Current development platforms for the Sandia project include Windows and Mac OS X workstations with deliveries of the released application to a Windows platform and deliveries of a related code project to a Linux environment. Our plan was to develop on any of the Windows or Mac platforms and deliver to the Linux platform. We utilized the Sandia project's code repository to build a new prototype application (proto-usndc) based on the framework's existing feature set.

Since the Sandia project already had a complete development environment operational, the initial configuration management setup, code build system, and platform configuration was very smooth and straightforward. This capability would imply the introduction of new software developers would be an efficient and straightforward exercise using this system.

Rating: "Significantly above average" for the environment configuration and setup effort at this initial stage of the effort.

4.4.2.1.2. Extending the Prototype for Seismic Waveform 2D Plotting

The proto-usndc prototype application had many widget elements of interest and applicable to our evaluation of NetBeans and Java Swing for seismic analysis applications (several tabular, time based and tree structure data presentation displays, and a real time mapping display).

The proto-usndc prototype application did not have any 2D plotting capability sufficient to support the display of waveforms. Another Sandia UI software project did have a reasonable waveform plotting capability implemented in Swing, so the team decided to use that project's plotting code in the proto-usndc prototype. After consulting with the entire team on the code inheritance layout for the proto-usndc prototype, the initial proto-usndc application was configured to support five existing display widgets (tree viewer, timeline viewer, waveform viewer, globe viewer, table data viewer). This effort involved about three workdays.

The existing 2D waveform-plotting module was then modified to adapt to the Sandia project's generalized data object (gdo) structure that is used to manage all the displayed objects in the GUI. The gdo structure is a very capable design that enables concurrency throughout the GUI for different views and display widgets of the same data in different formats.

Once the waveform-plotting module was adapted to the gdo structure, the new proto-usndc prototype application was able to display multiple waveform channels stacked vertically (one waveform channel for each imported waveform data object). The amount of effort it took an experienced developer to refactor the waveform-plotting module to accommodate gdo objects was approximately two workdays.

Rating: "Significantly above average" for integration ease and rapid prototyping capability, code reuse efficiency, and effectiveness of the gdo data structure.

4.4.2.1.3. Evaluating Concurrency within Prototype Display Widgets

Once the waveform-plotting capability was added into the new proto-usndc prototype application, the existing gdo architecture handled the management of data concurrency among the display widgets being used.

Every defined gdo object was visible within each display widget. When the user selected a particular gdo object in one display widget, the object would immediately highlight in the other display widgets resulting in an effective visual feedback pattern to the user.

The concurrency of the data objects among the different graphical perspectives reinforces the different attributes of the objects and more effectively informs the user of the nature and available parameters associated with the data objects being evaluated or analyzed. The effort to complete this concurrency capability was captured in the effort to enable the gdo object data structures in the previous task.

The proto-usndc application was a single-threaded NetBeans application and due to this and the available time an assessment of inter-process communication was not performed for this prototype.

Rating: N/A inter-process communications.

Rating: "Significantly above average" data concurrency and effective display widget usage reinforcing data relationships with alternative data viewpoints.

4.4.2.1.4. Enabling and Evaluating Netbean's RCP Plugin Usage

NetBeans includes modules to enable user controlled options and external code integration through plugins and templates. Specific exercises were completed to test the plugin capabilities of NetBeans and the proto-usndc design.

A new signal filter was defined in the vehicle of a code plugin. Once the user imported a series of seismic waveforms, the default proto-usndc filter is employed (this filter can be raw or "no filter").

Once the new plugin was constructed and checked into the proto-usndc version control system, it became available as a NetBeans module and was visible through the Plugins menu item. The user selects the Plugins menu item and is asked to download, install, activate, deactivate, uninstall, or delete one or more plugins. Proto-usndc remembers which plugins have been activated, downloaded, installed, or uninstalled.

When the user elects to download a new plugin, available plugins are updated automatically and are visible via the NetBeans Update Center (which can be configured to be simply a local server or file system). Once the desired plugin is downloaded, the user selects to install (the user is also prompted for a license agreement affirmation). Then the user can activate the new plugin and it becomes live, creating new GUI elements per its definition live while the current application is running. The application can immediately access the new features. No restarting is required. In our case the new plugin introduced a user-controlled Butterworth filter capability to enhance the existing waveform data objects.

Rating: “Significantly above average” for plugin API standardization and support, live code updating, and central plugin deployment and tracking.

4.4.2.1.5. Assessing Usability, Performance, and Customization of the Prototype

Overall the usability of proto-usndc exceeded the team’s expectations. The modern GUI elements were effective, usable, and attractive. Many features were available simply due to the NetBeans RCP that was supporting the application. These features included the ability to customize window size, window position, and whether a window was docked or floating disconnected from the rest of the proto-usndc GUI.

The user could modify the relative position of windows by dragging them to different snap locations on the floating set of connected display widgets or the user could simply “tear off” any given display widget and move it to a location of their choice anywhere on the screen. The application includes a command to reset the graphical display to its default. Additional customization capabilities included the ability to set font sizes and background panel colors and control proxy settings for network access to the NetBeans Update Center server.

The GUI performance is a subjective ranking. However, in the team’s view the performance was very solid, though the amount of test data was minimal at this time. The graphics performance was considered in the resizing and relocating of GUI display panes, the plotting and updating of 2D waveform trace plots, and the updating, zooming, tiling, and responsiveness of the globe 3D mapping display widget. The GUI responded equally well whether it was running on a Windows 7 native workstation, and Windows 7 VM, or on a 2010 MacBook Pro running OS X 10.8. We were unable to complete the performance testing within the Ubuntu

Linux environment due to Java graphics library complications. On the platforms where the Java graphics libraries were operational, the performance was good. The challenge of the Linux graphics libraries is addressed in the next section.

Rating: “Above average” for usability, customization, observed performance.

4.4.2.1.6. Conclusions on the Prototype and NetBeans RCP

While the predominate impression of proto-usndc was positive, as the prototype exercise began converging to its end some reappearing issues began to become too frequent to be dismissed as anomalies. The overall performance and efficiency of NetBeans RCP application development is evident. The environment has delivered and would make a solid solution for a permanent investment for the US NDC modernization effort. The overall code organization that was inherited from the existing Sandia project is also very strong and would make a good baseline for future development processes.

The impressions of Java Swing and its implementation within NetBeans are all very positive, and are examples of modern GUI design and effectiveness even allowing for the fact that JavaFX is slightly newer.

One apparent weakness that was observed in the proto-usndc prototype is the complexity that was introduced to support the 3D mapping widget using NASA’s World Wind Java module. When this module is accurately configured it is an effective and welcome addition to the seismic analyst’ display set. The key phrase is “when it is accurately configured”. In hindsight the solution should be a dedicated distribution manifest with packaged libraries installed in a dedicated project directory that includes the appropriate OpenGL jogl and gluegen-rt libraries for every targeted platform proto-usndc is to be deployed to. This issue can be rectified, although at this time this issue remains open.

In fairness, the genesis of the proto-usndc prototype came from a strong project whose support of their targeted platforms is very solid. The challenges we saw in attempting to deploy the proto-usndc prototype to additional targeted platforms should not be construed as a significant risk factor for this particular candidate in the search for the optimum development tool and end user RCP and user interface. If the US NDC Modernization effort elects to invest in NetBeans and Swing, the relevant deployment platforms will be defined up front and as new dependencies are created, their deployment requirements will be captured and resolved in a formal ongoing development process.

Rating: “Average” for deployment and high performance graphics OpenGL configuration and setup. Main difficulty is the ability to define and deploy the specific set of OpenGL graphics required jar files.

Final conclusion and rating: “Above average” for overall NetBeans RCP effectiveness and the proto-usndc gdo data structure design and GUI impact.

4.5. Follow-On Work

To build context around the results from the initial E1 prototype evaluation, additional UIF prototypes should be pursued.

Anticipated follow on efforts will pursue a comparable evaluation of the Java RCP Eclipse JFace/SWT candidate and potentially the C++ Qt Creator Qt candidate if additional Sandia UI software development teams are available for consulting and collaborative efforts.

Focus would be on the same design goals and constraints, although additional evaluation of multi-display performance and plugin management capability would provide more insight.

Additional E2 prototyping UIF investigations should address the question of evaluating the feasibility of a full web-based. Some promising solutions are available (Liferay) that need to be assessed for viability of a browser based UI for the modernized system.

A final area of interest is the investigation of viable methods to support the CLI capability that will accompany the GUI interface and how the two systems will interact and pose requirements on each other.

The development of this prototyping and evaluation data will assist in the eventual down selecting of a single candidate solution anticipated for the E3/E4 executable architecture phase.

APPENDIX A. SUPPORTING TABLES

Table 1. UIF and Widget toolkit Summary

Candidate Solution UIF/widget toolkit	URL	Summary Assessment
NetBeans/Swing	https://netbeans.org/ http://docs.oracle.com/javase/tutorial/uiswing/index.html	Advantages: NetBeans is the leading Java UIF candidate. Swing widgets integrate alongside JavaFX code. Large community. Disadvantages: Oracle dependence.
Eclipse/JFace (SWT)	http://wiki.eclipse.org/Main_Page http://wiki.eclipse.org/JFace http://www.eclipse.org/swt/	Advantages: Eclipse is the 2nd place Java UIF candidate. IBM supported. Very stable. Large community. Disadvantages: Eclipse learning curve is the most difficult. JFace/SWT is slightly dated compared to Swing and JavaFX2.
Qt Creator / Qt	http://qt-project.org/	Advantages: Qt is the leading C++ UIF candidate. GUI widgets are fast and native: strongest cross platform GUI behavior. Disadvantages: Not an RCP solution. Smaller community than Java.
NetBeans/JavaFX 2	https://netbeans.org/ www.javafx.com	Advantages: NetBeans is the leading Java UIF candidate. JavaFX2 has most modern Java GUI elements. Large community. Disadvantages: JavaFX2 2D plotting package is beautiful but has serious scaling issues. Oracle dependence.
NA/wxWidgets	http://wxwidgets.org/	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	https://developer.mozilla.org/en-US/docs/XUL	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.

This is the last page of the document.