NCCSD Systems/Data Workgroup Vendor Webinars – October 18, 25 and November 1

COVER PAGE for Answers by Vendors

Attached please find the answers provided by Conduent, who was invited by the Systems Workgroup of the National Council of Child Support Directors (NCCSD) to present educational webinars on two of the newest approaches to modernizing state child support systems.     These two topics are “replatforming/refactoring” and “low code/COTS”.   Note that some vendors are responding to both topics, and some are responding to only one.

IMPORTANT: Even though these are educational sessions, the vendors may be providing some proprietary information in their answers.   *Therefore by opening these documents you are agreeing to treat the information as confidential.*

Vendor Name: \_\_**Conduent \_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Please enter your responses into this document, but feel free to send any other attachments as well.

Questions:

1. **Please explain how your company defines both re-platforming and refactoring.**

Our definition of re-platforming is the act of moving an existing system from one technology platform to another without changing the business functionality of the system. The focus is to move to the new platform without significant changes to the underlying processes, unless needed to allow the migration to the new platform. For most states this will typically be moving from a mainframe platform to a .NET or Java environment supported by appropriate hardware and software configurations. Re-platforming keeps system functionality and user interactions the same in the legacy system minimizing disruption to end users.

Refactoring is changing a system, or components of a system, with the goal of retaining behavior of the system to achieve structural improvement and improved system performance. For example, code may be refactored to improve readability, make it easier to maintain, or make batch processes run more quickly. The focus of refactoring is usually on the programming code, but it can also affect the other structures of a system such as data storage, communications and third party software tools.

For some re-platforming projects there may be a need for refactoring to also be performed. For example, high volume system processing in batch programs may not achieve the required performance when converted and deployed on a new platform. These programs and processes may need to be changed or redesigned to obtain faster processing on the new platform.

Listed below are key characteristics for each of the modernization methods:

**Re-platforming**

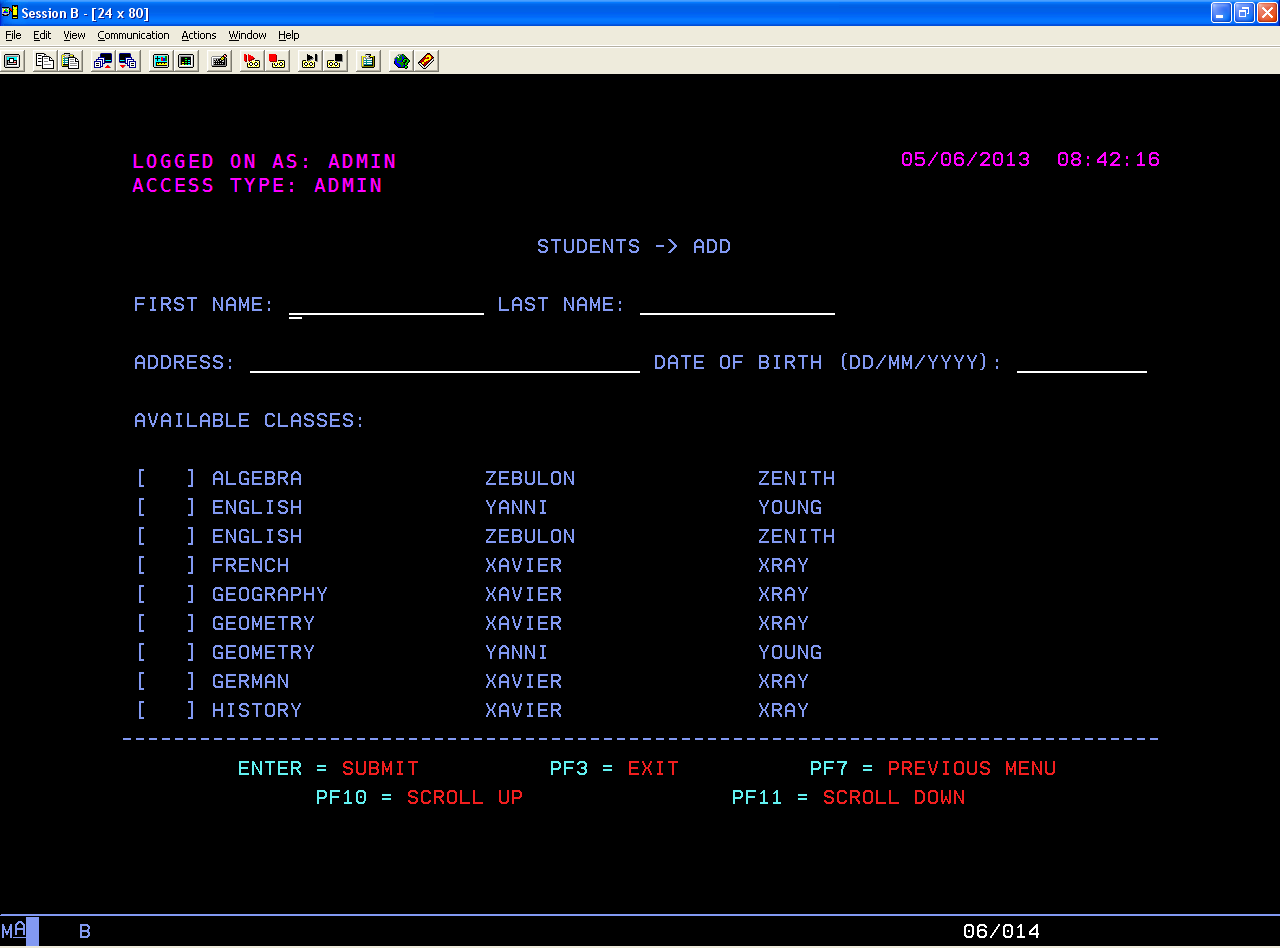
* Existing business processes are retained and do not require changes
* Replace hardware and software that is obsolete, hard to maintain, or expensive to maintain
* Frequently performed using automated conversion tools for data and code, but can be done manually
* Module or program structure is not usually changed

**Refactoring**

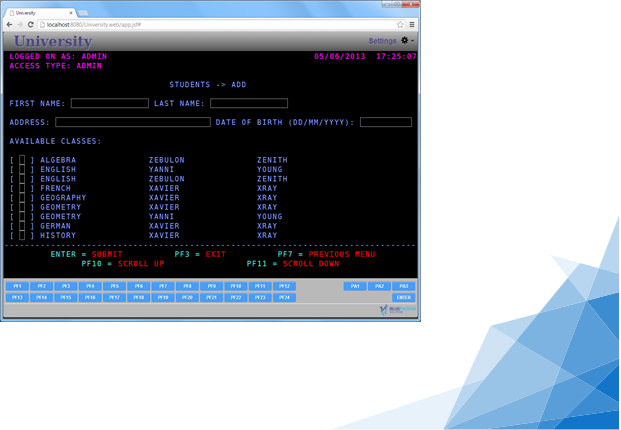
* Improve the structure of the source code and architecture with the result to make the code more efficient, scalable, maintainable or reusable, without actually changing any functions of the program itself.
* Improve code readability and make the debugging process efficient and easy to maintain
* Leverage the features and functions of the technology platform to improve the application business, performance and thereby reducing the overall cost of application maintenance
* Achieved through automated tools and/or manual programming

1. **Are you able to share any screen shots of a "before" and "after" implementation of this approach?**

The following are examples of a mainframe screen showing the before and after migration to a browser-based platform.



*BEFORE: Screen example of a mainframe screen before re-platforming*

*AFTER: Screen example after re-platforming to a browser-based platform*

Examples of before and after code conversion are provided in this section for a very basic batch COBOL program. You can see that some of the COBOL code structure in this example is brought forward into the new program structures. The transition of existing COBOL programmers to the new platform language is fairly straight forward once they know the new programming language because the programs will have some similar structure. An n-tiered architecture and some level of object-oriented code can be achieved through automated conversion tools, but some existing program structure will be carried forward into the new system.

From an end user perspective most clients have duplicated the legacy system screens on the new technology platform. It costs less and minimizes the impact for users on the new system. For example, a mainframe screen accessed through a terminal emulator looks the same as the new technology screen accessed through a browser. The addition of navigation components, graphics (agency logo), and other standard screen components may be added as part of a re-platform project, but the layout of the data fields is usually similar to the legacy system.

The example that follows shows how a simple COBOL application with simple file processing is translated to Java, using our partner’s automated tool (CTU–Cobol To Universal). CTU converts the source language, COBOL in this example, to a proprietary “universal language” which is then used to generate the new platform code. This design makes the tool more flexible and able to handle a variety of technology platform conversions.

|  |  |
| --- | --- |
| COBOL | Java |
| IDENTIFICATION DIVISION.  PROGRAM-ID. FILES.  ENVIRONMENT DIVISION.  INPUT-OUTPUT SECTION.  FILE-CONTROL.  SELECT PROFILES ASSIGN TO PROFILEFILE.  SELECT WEDDINGS ASSIGN TO WEDDINGFILE  ORGANIZATION IS SEQUENTIAL.  DATA DIVISION.  FILE SECTION.  FD PROFILES.  01 PROFILES-RECORD.  05 NAME PIC X(20).  05 SURNAME PIC X(20).  05 MARITAL-STATUS PIC X(01) VALUE ‘M’.  88 MARRIED VALUE ‘M’.  88 SINGLE VALUE ‘S’.  FD WEDDINGS.  01 WEDDINGS-RECORD.  05 NAME PIC X(20).  05 SURNAME PIC X(20).  WORKING-STORAGE SECTION.  01 END-OF-FILE PIC X(01) VALUE ‘N’.  88 EOF-Y VALUE ‘Y’.  88 EOF-N VALUE ‘N’.  01 MARRIED-PEOPLE PIC 9(5) COMP VALUE ‘0’.  PROCEDURE DIVISION.  0100-OPEN-FILES.  \* OPEN PROFILES FILE  OPEN INPUT PROFILES.  \* OPEN WEDDINGS FILE  OPEN OUTPUT WEDDINGS.  0200-PROCESS-FILES.  \* READ RECORDS FROM PROFILES FILE AND  \* WRITE THEM INTO THE WEDDING FILE IF MARRIED  PERFORM UNTIL EOF-Y  READ PROFILES  NEXT RECORD  AT END SET EOF-Y TO TRUE  NOT AT END  IF MARRIED  PERFORM 0800-WRITE-WEDDING-RECORD  ADD 1 TO MARRIED-PEOPLE  END-IF  END-PERFORM. | package com.bphx.ctu.samples.files;  import com.bphx.ctu.af.core.\*;  import com.bphx.ctu.af.io.\*;  import com.bphx.ctu.af.util.\*;  public class Files {  private static final String PROGRAM\_ID = “FILES”;  public ProfilesTO profilesTO = new ProfilesTO();  public ProfilesDAO profilesDAO = new ProfilesDAO();  public WeddingsTO weddingsTO = new WeddingsTO();  public WeddingsDAO weddingsDAO = new WeddingsDAO();  protected boolean endOfFile = false;  protected int marriedPeople = 0;  public void run() {  try {  mainSubroutine();  } catch (ReturnException re) {  // normal termination of the program  }  }  private void mainSubroutine() {  openFiles();  processFiles();  closeFiles();  }  private void openFiles() {  // OPEN PROFILES FILE  profilesDAO.open(OpenMode.READ, “Files”);  // OPEN WEDDINGS FILE  weddingsDAO.open(OpenMode.WRITE, “Files”);  }  private void processFiles() {  // READ RECORDS FROM PROFILES FILE AND  // WRITE THEM INTO THE WEDDING FILE IF MARRIED  while (!endOfFile) {  profilesTO = profilesDAO.read(profilesTO);  if (profilesDAO.getFileStatus().isEOF()) {  endOfFile = true;  } else {  if (profilesTO.maritalStatus.isMarried()) {  writeWeddingRecord();  marriedPeople = marriedPeople + 1;  }  }  } |

Alternatively, the following example shows how the same simple COBOL program is translated to C#, using CTU.

|  |  |
| --- | --- |
| COBOL | C# |
| IDENTIFICATION DIVISION.  PROGRAM-ID. FILES.  ENVIRONMENT DIVISION.  INPUT-OUTPUT SECTION.  FILE-CONTROL.  SELECT PROFILES ASSIGN TO PROFILEFILE.  SELECT WEDDINGS ASSIGN TO WEDDINGFILE  ORGANIZATION IS SEQUENTIAL.  DATA DIVISION.  FILE SECTION.  FD PROFILES.  01 PROFILES-RECORD.  05 NAME PIC X(20).  05 SURNAME PIC X(20).  05 MARITAL-STATUS PIC X(01) VALUE ‘M’.  88 MARRIED VALUE ‘M’.  88 SINGLE VALUE ‘S’.  FD WEDDINGS.  01 WEDDINGS-RECORD.  05 NAME PIC X(20).  05 SURNAME PIC X(20).  WORKING-STORAGE SECTION.  01 END-OF-FILE PIC X(01) VALUE ‘N’.  88 EOF-Y VALUE ‘Y’.  88 EOF-N VALUE ‘N’.  01 MARRIED-PEOPLE PIC 9(5) COMP VALUE ‘0’.  PROCEDURE DIVISION.  0100-OPEN-FILES.  \* OPEN PROFILES FILE  OPEN INPUT PROFILES.  \* OPEN WEDDINGS FILE  OPEN OUTPUT WEDDINGS.  0200-PROCESS-FILES.  \* READ RECORDS FROM PROFILES FILE AND  \* WRITE THEM INTO THE WEDDING FILE IF MARRIED  PERFORM UNTIL EOF-Y  READ PROFILES  NEXT RECORD  AT END SET EOF-Y TO TRUE  NOT AT END  IF MARRIED  PERFORM 0800-WRITE-WEDDING-RECORD  ADD 1 TO MARRIED-PEOPLE  END-IF  END-PERFORM. | namespace com.bphx.ctu.samples.files  {  public class Files : AbstractProgramTemplate<FilesTO>  {  private const string PROGRAM\_ID = “FILES”;  public ProfilesTO profilesTO = new ProfilesTO();  public ProfilesDAO profilesDAO = new ProfilesDAO(new  FileAccessStatus());  public WeddingsTO weddingsTO = new WeddingsTO();  public WeddingsDAO weddingsDAO = new  WeddingsDAO(new FileAccessStatus());  protected bool endOfFile = false;  protected int marriedPeople = 0;  protected override void EntryPoint()  {  mainSubroutine();  }  private void mainSubroutine()  {  openFiles();  processFiles();  closeFiles();  }  private void openFiles()  {  // OPEN PROFILES FILE  profilesDAO.Open(OpenMode.READ, “Files”);  // OPEN WEDDINGS FILE  weddingsDAO.Open(OpenMode.WRITE, “Files”);  }  private void processFiles()  {  // READ RECORDS FROM PROFILES FILE AND  // WRITE THEM INTO THE WEDDING FILE IF MARRIED  while (!getEndOfFile())  {  profilesTO = profilesDAO.Read(profilesTO);  if (profilesDAO.GetFileStatus().IsEOF())  {  setEndOfFile(true);  }  else  {  if (profilesTO.maritalStatus.isMarried())  {  writeWeddingRecord();  setMarriedPeople(System.Math.Abs(1 +  getMarriedPeople()) % 100000);  }  }  } |

The converted code is readable and maintainable, following Java and C# standards and object-oriented concepts. COBOL business logic and comments are preserved so employees familiar with the original legacy application can easily understand the converted one. In this example, there is a separation between business logic (the program) and data layer (ProfilesDAO, ProfilesTO, WeddingsDAO, WeddingsTO).

The table below illustrates the basic mapping from the procedural COBOL programming language to its object-oriented counterparts:

| Procedural COBOL | Java | C# |
| --- | --- | --- |
| Legacy program | Java class | C# class |
| Paragraph within program | Java method | C# method |
| File | Data Access Object (DAO) | Data Access Object (DAO) |
| File record definition | Data Transfer Object (DTO) | Data Transfer Object (DTO) |
| Table | Data Access Object (DAO)  Data Transfer Object (DTO) | Data Access Object (DAO)  Data Transfer Object (DTO) |
| Copybook | Data classes | Data classes |
| Working Storage | Data fields: Class properties  Group items: Class methods  Redefines: Java classes | Data fields: Class properties  Group items: Class methods  Redefines: C# classes |
| Map | Web client (JSF, beans) | Web client (ASP .NET MVC) |

1. **Under what circumstances does it make the best sense for a state child support agency to consider refactoring/re-platforming versus other possible means of modernizing its child support system? Are there any characteristics of either a state’s IT system or its business processes that lend themselves more to this approach?**

Child support agencies should consider refactoring or re-platforming under any of the following circumstances:

* High system operating costs (i.e. operate on mainframe or client/server platforms)
* Funding challenges to perform other modernization options
* Applications that were built using software products or tools that are no longer supported by the manufacturer
* Agencies with limited staff and or skills that are hard to replace due to dependency on mainframe or other aging platforms / products
* Unmaintainable or difficult to maintain systems
* Agencies that want to reduce modernization risk through shorter projects and automation of the conversion
* Agencies that want to leverage current software products and technologies for more secure and maintainable systems

The stakeholders should assess their needs with a analysis of the child support program and systems. The following list of action items could help arrive at a decision:

* Existing child support functional business processes and functions meet the needs of the program and can continue to provide the services
* State staff have the skills to maintain the application after implementation
* Budget limitations that prohibit a full scale system rebuild or procurement of COTS products
* Cost of upkeep of current system continues to climb and support for existing hardware and software products continues to be a challenge
* Compare cost of operations and maintenance – legacy system vs modernized system

1. **Generally speaking, what should a state expect on the following: project timeframe, project cost, time to rollout statewide?**

Our analysis and estimation model is based on certain key attributes that are relevant measures for re-platforming which include factors such as system requirements, volume of data, programs to be converted, type of statewide implementation, and staff required to support the initiative after implementation.

* 18 to 24 months for planning, assessment, hardware/software procurement, application code transfer (automated), testing and implementation
* Anywhere between $5M to $15M depending on the volume of source code, applications and data
* Data Conversion – quality of data and data integrity determines the duration for data migration and testing
* Implementation Strategy
  + Duration of pilot phase and validation of system functions prior to statewide rollout. An alternate implementation strategy would be to consider a big bang rollout after rigorous testing and validation with the benefit of reduced implementation costs and no data synchronization requirements.
  + Regional Rollout – Limit the rollout phases to short cycles and avoid cost overruns as two systems will required maintenance
* Size of code to be converted and data volume
* Number of commercial system components that don’t have a new technology platform equivalent (i.e. the COTS document generation system isn’t supported anymore and a completely different product will need to be used)

1. **Please list and explain the pros and cons, and any common pitfalls the states should know, for refactoring/re-platforming. What surprised you in your implementations?**

**Pros**

* Relatively low cost of the project compared to system transfer or commercial platform implementation
* Low risk if automation of over 70% can be achieved
* Very low impact on system users because the system still functions the same as on the legacy platform
* Migrating to a new technology and platform provides the opportunity to build add on services at a later stage
* Cost savings when aging mainframes are replaced
* Staff and required skillsets readily available in the market
* Existing legacy programmers that can code in the new technology will have an easier transition because some of the program structure is usually brought forward from the legacy to the new system.

**Cons**

* Some of the legacy program coding structure is carried forward. For example, COBOL is a procedural language. Re-platforming will bring the structure forward into some of the new modules even though the new programming language supports object-oriented development
* Commercial components integrated into the legacy system may not have a new technology equivalent which will require a new product to be acquired or developed to replace the component.
* Some modules may not be able to be automatically converted which will require the functionality to be migrated by a programmer which is expensive and raises the level of risk incrementally.
* To more quickly and cost effectively re-platform a system some vendors will provide replacement or run-time modules that will be needed for the system to operate on the new platform. States should be fully aware of these modules, licensing costs, as well as how will the vendor keep them updated over the life of the system.

**Common pitfalls**

* Application code that is missing, doesn’t compile, or that produces different results as the compiled modules in the legacy platform
* Business processes that are inefficient in the current system are carried over and continue to produce the same results
* Data anomalies when uncovered can lead to additional data cleanup actions
* Ineffective and poor code structure could degrade system performance. Batch processing may be slower without some refactoring
* Re-platformed systems aren’t necessarily easier to modify than the legacy systems because they bring some of the legacy structures forward at the program level.
* Because some legacy code structure is brought forward in most re-platformed systems new staff will require assistance to understand the system and how it was converted.
* The lines of code will usually increase due to an automated re-platforming process.

**What surprised us**

* Application code that is missing, doesn’t compile, or that produces different results from the compiled modules in the legacy platform
* Replacement products for commercial system components don’t provide exactly the same functionality as those used on the legacy system
* Finding new code during the Assessment Phase that we didn’t know about when we bid the work
* Discovered that code in the legacy system wasn’t working correctly during system/user acceptance testing of the new technology system.

1. **What are the most important things that a state should do to prepare for this approach?**

States could begin an internal assessment initiated by the program management team supported by the IT department to determine the preferred approach. A few guidelines are as follows:

* Preliminary assessment and evaluation of current system in terms of operating costs, staff skillset and clear definition of objectives and expected results
* Assess and confirm existing business processes meet the program needs and do not require changes. In cases where the state determines business process changes are needed, then the overall implementation approach would have to be further analyzed to determine the cost of implementing the changes as part of the re-platform project.
* Work with the State IT department to identify the preferred technology stack, procurement guidelines, and timelines
* Review strategy to evaluate alignment with the long term plan for the agency and or state
* Define an approach and plan to handle system enhancements after modernization is completed
* Delete programs and utilities that aren’t used anymore to reduce cost, complexity, and testing effort
* Create a data cleansing plan and perform data purification to reduce cost and conversion issues that may arise during the data conversion phase
* Define criteria to measure code conversion for quality and maintainability. Establishing thresholds will help define the acceptance criteria during system implementation.
* Identify mainframe components that will need a corresponding product that runs on the new environment. For example, report management, data warehouse and retrieval, job scheduler, and name/address validation.

1. **How does this type of child support system modernization effort fit with states who need to have an enterprise approach?**

Re-platforming can help the agency move toward the goal of being on an enterprise platform in several key ways:

* If the agency is using a standard technology platform for the enterprise the migration of the child support system to the same technology can make data sharing easier, help the agency develop programming resources that understand the child support program and the enterprise technology, put data into structures that are easier to access, and eliminate unique system technologies that require specialized resources to support
* Re-platforming will force the agency to go through the process of identifying unused programs/utilities, develop a more updated set of test scenarios, update system documentation, and clean up system data
* New technology products may be available to exchange data in real-time with the enterprise system

Conduent has always viewed re-platforming/refactoring as a step in the modernization process. It should not be avoided even if the agency uses a commercial or custom developed enterprise platform.

1. **What haven’t we asked that we should have?**

We believe the most important and relevant questions have been documented. A few additional topics and questions to be considered would be:

* Will the state perform the testing which is a major issue as it pertains to staff availability and associated cost? Few states have well documented systems, so asking a vendor to come in and perform a functional test of the system will require additional cost.
* How many State resources will be needed for the project? Many clients see this as an IT project, but adequate conversion and system testing requires business knowledge.
* What percentage of automated conversion should I expect?

NOTE: Conduent has included an attachment as a PDF from our partner Modern Systems. This is a whitepaper named Refactoring with Automated Conversion: COBOL to Java or C#.