**NCCSD Systems Workgroup Vendor Forums – Q&A related to Refactoring/Replatforming**

Vendor Name: Accenture

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 replatforming and refactoring.

**Replatforming**: To change the hardware and operating system that a software system runs on without changing the structure or functionality. In terms of child support modernization efforts, this typically can be translated to ‘code conversion’. In addition, this can be further clarified as ‘line-by-line code conversion’. ‘Line-by-line’ code conversion means converting legacy code exactly to a new language. No re-engineering or re-structuring of the code occurs. The typical goal of this activity is to achieve a lower total cost of ownership while not investing time, money and energy in redesigning a system and business processes that have already been proven to meet the system users’ needs. It is widely understood, that this technique would not require an OCSE approved Feasibility Study or automated system re-certification.

**Refactoring**: Changing the structure of a program or system. This is considered because solely replatforming a language, line-by-line to a new language can be sub-optimal. For example, converting a largely procedural language, like COBOL, to an object-oriented language, like Java does not re-engineer the code to be optimized in this new paradigm of coding principles. It is only through converting, but then re-designing, or re-factoring that the language can we optimized for the more modern, object-oriented language. One deterrent to this approach is that it will likely require a re-examination and potential re-certification of the core application – as the underlying business logic has been re-engineered through the re-factoring process.

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

We consider code modernization to be a combination of replatforming and refactoring. In Michigan we modernized the Child Support court guideline calculation sub system by moving the code base to from Oracle Forms to Java. The project was focused on overhauling the end user UI. This could not be accomplished in Oracle Forms. New functionality that the business had needed for years could be implemented once Java was available. And a technical interface was created to allow the new guideline application to co-exist seamlessly with the legacy Oracle Forms child support system. A few before and after screen shots are included below.

***In the legacy application users had limited space to enter income details and were forced to have Monthly Income Amount***



**The modernized design includes details of each income record and actual income calculation based on how data is presented to user**

***Nearly every part of the old calculator saw a UI refresh, even if the underlying data was similar***



1. Under what circumstances does it make the best sense for a state child support agency to consider refactoring/replatforming 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?

State characteristics which lend themselves to a code conversion/refactoring approach to system modernization include:

* + Legacy system is stable.
	+ Legacy system functionality addresses State’s business needs and users are generally content with how the system operates.
	+ Business processes are stable and effective.
	+ Support exists within the State’s IT department for modern software and operating system platforms.
	+ Statewide adoption of modern software platforms is already underway, licensing has been procured and can be extended to the modernized application.
	+ There is a business need to move off the existing legacy platform. Examples include escalating licensing costs, software tools have reached end of life and are no longer supported by the vendor, finding developers for the legacy platform is prohibitively expensive, etc.
	+ Valuable staff have retired and taken with them deep knowledge of the legacy system which is not thoroughly documented.
	+ Legacy system contains business logic that has been enhanced and revised over the years and would be difficult to adequately capture in system requirements, but which need to be preserved in the new system.
1. Generally speaking, what should a state expect on the following: project timeframe, project cost, time to rollout statewide?

Statewide modernization projects could take anywhere from 12-24 months and cost from $5 million up to $10 million depending on size and complexity.

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

**Pros:**

* Effectively harvests legacy business logic. This is especially advantageous when current staff is less familiar with the system logic and would struggle articulating the details in business requirements.
* Replicates legacy system on new platform in short timeframe (~ 2years), helping to reduce IT costs associated with legacy system and infrastructure.
* Enables State to incrementally enhance functionality post code conversion, prioritizing areas with most significant business impacts.

**Cons:**

* ‘Line-by-line’ conversion replicates the legacy procedural code using syntax of a modern language, but does not take advantage of object-oriented principles (e.g. inheritance, encapsulation), does not structure the output as well-defined objects and services representing the business entities and functions, and the resulting code is not optimized for the new platform.
* Converted code, given it’s a hybrid of legacy and modern language concepts, is more difficult for a new workforce to enhance and maintain. In our experience, enhancing converted code requires significant rework of impacted and surrounding code.
* No functional system enhancements/ improvements are realized. Code conversion provides very little user/customer benefit or business value. The business pain points you experience today remain post conversion.
1. What are the most important things that a state should do to prepare for this approach?

In preparation for this system modernization effort, States should:

* Identify business experts across all CSE core capabilities who can be business analysts during the system modernization effort.
* Clean up any known data issues in the legacy system in preparation for data conversion to the new system.
* Correct, clarify, or create legacy system documentation to help support the system modernization effort.
* Halt ongoing updates to the legacy system.
* Define test cases that, when run successfully on the converted system, will build confidence across the user community.
* Set expectations with users that the initial effort is to replatform the technology and will not result in any enhancements to business functionality.
* Define key stakeholders and organize a governing body who can efficiently review project issues and make decisions to help sustain forward progress during the modernization effort.
* Decide between pursuing a ‘line-by-line’ Code Conversion approach or a Code Refactoring approach.
1. How does this type of child support system modernization effort fit with states who need to have an enterprise approach?

Code conversion simply moves the legacy CSE system to a new technical platform, it does not add any new functionality to address the need for an enterprise approach to Human Services. However, moving to a modern platform does better position the State to enhance or otherwise change the system. These changes can be targeted towards realizing synergies across other Human Services systems in the State (e.g. common case file, single process to apply for services, etc.).

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

Depending on the state of your legacy system and the objectives you are seeking from your modernization there are multiple approaches we could recommend.  These span from addressing pain points in the existing system (**Extend life**), **Re-hosting** the application on a lower cost platform, performing a **Language Migration** (as discussed earlier in this response), and **Re-architecting / rewriting** the application.



Related to the earlier modernization options (Extend life or Re-host), Accenture often leverages a technique called Digital Decoupling with our clients to enable faster delivery of new capabilities with minimal impact to the existing legacy system.  Through using Change Data Capture (CDC) tools we can enable an Event-Driven Architecture to create loosely coupled Microservices that can provide new capabilities by acting on data that has been ‘unlocked’ from the legacy systems.  These Microservices are independent and can be delivered and changed by small teams following agile methods.  What this means for your business is the ability to modernize select aspects of the application and provide new capabilities without having to replace the core of your existing system.

We recommend building Microservices in a cloud-native way, leveraging technologies like Kubernetes to build resilient services that scale automatically based on level of utilization.  In this way when utilization is low you pay less as less compute resources are used and when there are volume spikes the capability quickly scales to support the increased use.