Migrating Applications from Legacy Platforms to Windows

A White Paper

Summary of Experiences from 25 Years of Performing Platform Migration
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Emulation</td>
<td>3</td>
</tr>
<tr>
<td>Conversion</td>
<td>4</td>
</tr>
<tr>
<td>COBOL</td>
<td>4</td>
</tr>
<tr>
<td>Database</td>
<td>5</td>
</tr>
<tr>
<td>KSAM and Sequential Files</td>
<td>5</td>
</tr>
<tr>
<td>Screens</td>
<td>5</td>
</tr>
<tr>
<td>Job Streams</td>
<td>8</td>
</tr>
<tr>
<td>Summary</td>
<td>8</td>
</tr>
<tr>
<td>Contact</td>
<td>9</td>
</tr>
</tbody>
</table>
OVERVIEW

Unicon Conversion Technologies, now in its 25th year of performing platform migrations, has seen many changes in the onwards and upwards path of data processing technology. It is particularly interesting to note that prior to 1998 all our customers were asking us to migrate their mission critical applications to UNIX platforms. What a difference ten years have made. Today our customers’ new platform of choice is predominantly WINDOWS. This is true for migrations from the smaller DEC VAX systems up though the midrange HP3000 systems to the large IBM Mainframe systems. However, having said that, it is still true to say that over the years we have migrated more customers to a UNIX platform than a WINDOWS platform, but today that ratio is changing fast.

Our migration process was specifically designed to migrate to WINDOWS, UNIX or LINUX. There is no difference to Unicon as to which of these platforms the customer wants to migrate to; we can accommodate each. The decision as to which platform to migrate to is purely the customer’s choice. So why today do our customers primarily choose WINDOWS? Each customer is unique and each has their own reasons; from finding it easier to recruit WINDOWS knowledgeable staff than UNIX staff, to consolidating all systems onto one readily available platform throughout the organization, obtain improved interoperability, develop in a range of languages, move to a SQL Server database or wanting to establish a Visual Studio .net shop; but perhaps the general overall underlying reasons can be summed up in three words: Cost, Agility and Performance.

So what does a migrated application look like after it is migrated to WINDOWS?

To a large extent that question is answered by what method is chosen to perform the migration.

There are fundamentally two main migration options – Emulation and Conversion.

EMULATION

Emulation is where a software layer is added to a WINDOWS platform to make it emulate a legacy platform. This emulation software runs as a middleware layer on top of the Microsoft WINDOWS operating system sitting between the application code and the WINDOWS operating system. The emulation layer makes the WINDOWS platform look like the original legacy platform to the application code. In theory the application code can be lifted “AS IS” from the legacy platform and run on the emulation layer on the WINDOWS platform because the application code thinks it is running on the original legacy platform. Well it’s never quite that simple, but that’s the general idea.

Our migration offerings between 1985 an 1995 were entirely based on emulation solutions and consequently we have a lot of experience with emulators. But in 1996 we decided to stop offering an emulation solution and we replaced it with a conversion solution. We found that emulation was an appropriate solution for the short term but it did not provide a sound foundation for the long term future. Its main benefit was that it maintained the old legacy environment on the new WINDOWS platform, but staff retention and staff recruitment of qualified personnel with the old legacy skill sets became progressively hard to maintain. But perhaps most significantly the emulation layer became a major inhibitor for the customer to enhance the migrated application to take advantage of the many new feature functions of the WINDOWS environment and typically at some future time the emulation
solution had to be replaced with a long term solution and the whole migration effort had to be repeated.

There are certainly times when the emulation solution is appropriate, particularly if the customer intends to phase out the application within a couple of years.

But this paper is concerned with the options and processes of performing a long term migration solution and as such it will concentrate its discussions on the conversion solution option.

**CONVERSION**

Conversion is where the application components (COBOL, Screens, JCL, Data, etc) that were used on the legacy system are converted to their equivalent WINDOWS components. The converted applications code is no longer a legacy platform application it has become a true WINDOWS application. The components that will be used on the new WINDOWS platform are chosen by the customer; COBOL, Database, Screen deployment, JCL language, etc.

A converted system is built from the WINDOWS components that have been specifically chosen by the customer.

So what are the WINDOWS choices for each component?

**COBOL**

By far the most prolific programming language in use today in the business community is COBOL; and of course COBOL is the dominant programming language on all the legacy platforms including WANG VS, DEC VMS, HP3000, IBM AS/400, IBM Mainframe and UNISYS Clearpath. COBOL accounts for some 80% of all computer code.

Since 1985 there were four main WINDOWS COBOLS – RM COBOL, ACUCOBOL, Micro Focus COBOL and FUJITSU COBOL.

But in May 2007 Micro Focus acquired ACUCOBOL when they bought ACUCORP. Shorty after they also acquired RM COBOL when they bought LIANT. ACUCOBOL is very similar to RM COBOL but has more features and functions. Micro Focus immediately incorporated the RM COBOL product into ACUCOBOL and stopped selling RM COBOL. Micro Focus is now incorporating ACUCOBOL into the Micro Focus COBOL product and although it still sells and supports ACUCOBOL to the old ACUCOBOL ISVs and existing customers it does not sell ACUCOBOL to new customers.

So today there are two main WINDOWS COBOLS --- Micro Focus COBOL and FUJITSU COBOL.

Both Micro Focus COBOL and FUJITSU COBOL are excellent WINDOWS COBOLS. They support both batch and online programming. For online work they both provide full support for COBOL Screen Section Screens, WINFORMS and WEBFORMS. (See SCREENS below) They are both highly integrated into the Microsoft Visual Studio IDE and the .net framework, and both support full object programming. They both have very efficient and effective debugging tools, both within an online Visual Studio Solution and within their own batch runtimes.
DATABASE

The two main databases our customers have chosen on WINDOWS are Microsoft SQL Server and ORACLE; with most choosing SQL Server. The following text discusses SQL Server as the WINDOWS database but the discussions would refer just as well to ORACLE.

The legacy databases (network, relational or hierarchical) converts directly to a SQL Server database. Each master table or segment becomes a SQL Server table. The Master / Detail relationships are maintained as before since an equivalent relationship is established in SQL Server between the Master tables and their corresponding Detail tables.

In SQL Server, if you have a table that is related to another table by a “foreign key” then the two tables are linked together so that a record in the “Master” table cannot be deleted if there are “detail” records for that master key in the “Detail” table and a record cannot be inserted into a “Detail” table if there is no “master” record in the “Master” table for the foreign key.

The records in the legacy tables become records (rows) in the corresponding SQL Server table and the fields in the records become corresponding fields (columns) in the SQL Server records (rows).

The migrated SQL Server tables are normal relational tables that can be accessed and processed as can any other SQL Server table.

The converted COBOL programs are modified to perform the exact same access functions as the programs performed on the legacy databases except that they are now using SQL statements to access the SQL Server tables.

Both Micro Focus COBOL and FUJITSU COBOL provide full support for both embedded SQL statements and stored procedures and both provided excellent performance.

KSAM AND SEQUENTIAL FILES

The Indexed files and sequential files are converted to standard COBOL ISAM and sequential files. The indexed files can be converted to SQL-Server tables and the program access logic is converted to SQL command to access the new SQL server tables. There is no logical difference in processing.

SCREENS

Screens are perhaps the most important aspect of a successful migration project because this is the primary interface to the user community. It is imperative to ensure that the migrated screens satisfy the need of the users. The screens can be converted to look and feel almost exactly as they did on the legacy platform or they can be enhanced with additional GUI functionality. Sometimes the screens are migrated AS IS for one department and with GUI functionality for another department. For example, users in a department who are used to speed typing data into the system may find it very detrimental to their job function to have to deal with the point and click of a mouse. On the other hand some users may find it very beneficial to be given selection list boxes and dropdown selection boxes to help them better manage their data input process.
On the legacy platform the online users typically connects using a directly connected proprietary terminal or by using a terminal emulator running on a PC connected over a network.

The user application screens are processed by the legacy system in one of two ways CONVERSATIONAL OR PSEUDO CONVERSATIONAL.

PSEUDO CONVERSATIONAL is generally the domain of the IBM and UNISYS Mainframes.

The IBM & UNISYS operating systems are pure batch based systems, dating from the late 1960’s. They cannot directly support on-line terminals. So each company developed TP Monitors to provide on-line connectivity. The TP Monitors run in one of the batch regions and enable terminals to share the resources of that region. These TP Monitors enable the Mainframes to support terminal connections. IBM has CICS and IMS/DC and UNISYS has TIP and COMS. Although deferent in their individual implementations they all follow the same premise which in today’s new lingo is called STATELESS processing models. When the user logs onto the system a screen is displayed on the terminal but no program is running on the server. The user enters the data into the screen and pressing a function key invokes a program to service the transaction. The program loads, performs its work and presents the next screen to the terminal at which time the program terminates.

Sound familiar? Yep, it’s very like good old WEBFORMS running under a .net environment.

So the old CICS / IMS / TIP / COMS are stateless processes, just like WINDOW .net. These legacy screens are all maintained separately from the COBOL programs. They are all block mode maintained as BMS, FMS, DPS or SDF screens respectively. So by converting these screens to ASPX WEBFORMS we can implement a native .net environment without any requirements for any emulation or middleware layers.

CONVERSATIONAL is the generally domain of the mid range legacy systems; WANG VS, IBM AS/400, HP3000, DEC VMS, etc.

On these legacy platforms the user logs onto the system and then (probably through use of a menu system) calls the application program that contains the screen(s) that the users want to work with. The terminal maintains a permanent connected to the server throughout the user session. The application program loads into memory and starts processing from the first logic statement in the program. The program continues to process each statement along its logic path until it comes to a statement that presents a screen to the user terminal to display information or to allow entry of data; in either case the program presents the screen to the terminal (in the accept case the program waits for the user to enter the data and hit a function key or the enter key) and the program will continue on to the next instruction in the logic path.

Thus, the HP3000 online application code is procedural code. Throughout the process the program is fully resident in the memory (instate) and it is always in command of the processing, and the screens will be presented to the terminal by the program in the order determined in its logic path.

Although the screen processing can be performed by using COBOL DISPLAY and ACCEPT statements, most legacy programs use some form of block mode screen mechanism to perform their screen functions. For example the HP3000 uses VPLUS forms and the DEC VMS uses FMS or DECFORMS which are maintained separately from the COBOL programs.
There are three ways that CONVERSATIONAL screens can be converted to WINDOWS screens; they are COBOL SCREEN SECTION screens, WINFORMS and WEBFORMS.

COBOL SCREEN SECTION screens are the standard way of processing screens in COBOL. Although SCREEN SECTION screens have been around for decades they were officially incorporated into the ANSI COBOL Standards late in the 1990’s. The SCREEN SECTION comes after the LINKAGE SECTION in the DATA DIVISION of the COBOL program. The screen is defined in the SCREEN SECTION and presented to the terminal by DISPLAY screen-name and ACCEPT screen-name statements in the COBOL program. SCREEN SECTION screens are entirely contained in the COBOL program and maintained as part of the COBOL program. SCREEN SECTION Screens are character mode only; there are no GUI facilities. (ACUCOBOL SCREEN SECTION screens did have an extensive GUI facility but that product is no longer generally available.)

WINFORMS are from Microsoft. They are created and maintained in Visual Studio, using the standard Visual Studio IDE and of course they have full GUI facilities. So WINFORMS are maintained completely outside the COBOL program. When the program wants to use a form it INVOKES (calls) the form.

The calling of a SCREEN SECTION screen or a WEBFORM screen in a converted COBOL program is similar in concept to calling a screen on the original legacy platform. The program is in full control and the screens are presented to the user terminal as the program executes along its logic paths. This is a normal procedural code process just like the legacy COBOL. Just like on the legacy system the user logs onto the WINDOWS server from their PC which is connected over a network. The only difference is that instead of using terminal emulation software to connect to the server the user connects using a WINDOWS Terminal Services connection or a Citrix WinFrame connection.

WEBFORMS, like WINFORMS, are from Microsoft. They are created and maintained in Visual Studio; they have full GUI facilities and are maintained completely outside the COBOL program.

But that is where the similarity ends. WEBFORMS are full ASP .net and therefore employs a completely different method of processing.

With WEBFORMS the user PC’s are not connected to the Server through the use of any terminal services software. The WINDOWS server must have a Web Server installed on it. The user PC’s use their Web Browser and the internet (or intranet) to connect to the Web Server. The user typically has an ICON on their screen that starts the app. When the user clicks on the ICON it fires up an instance of the Web Browser and gives it the URL of the Web Server and the name of the WEBFORM to be displayed. The form displays on the user PC. Now remember we fired up a WEBFORM not a COBOL program. The form is presented onto the terminal but there is no application program running on the server. The user enters the data into the form and presses a function key (or screen button, etc) and the WEBFORM creates an event request and sends it to the Web Server. The Web Server receives the request and the WEBFORM “code behind” processes the request by calling an application program to service that request. When the application program has performed its function, perhaps to obtain account information from the database using the account number entered by the user, it terminates and returns that information to the WEBFORM “code behind”. The WEBFORM “code behind” populates the form with the information and sends the form to the terminal.

The main thing to appreciate is that it is the WEBFORM that calls the application program to perform an event request. The application program performs the request and terminates. The program responded
to the form. The form is in control, not the program. When the form is on the screen there is no application program running on the server. This is stateless processing. An application program cannot “call” a form to display on the terminal, it can only respond to a call from a form and return to that form. The form however can INVOKE (call) another form to be displayed. This process does not fit the procedural code of the legacy CONVERSATIONAL online programs.

Basically there are two competing methodologies involved here, the stateless processing of the WEBFORM and the instate processing of the application procedural code. In the stateless process the forms needs to be in control of the programs and in the instate process the programs needs to be in control of the forms.

To make it work the legacy application programs have to be re-engineered into objects containing the logic required to service a form request. This is a massive undertaking.

To enable a WEBFORM front end to work with a converted procedural program back end, Unicon created an interface manager called PCIM. (Procedural Code Interface Manager). PCIM is a group of .net classes and WINDOWS API’s that very effectively enable a full ASP.net WEBFORM internet front end to work seamlessly with the converted procedural code back end. What is most important is that ongoing development can be performed either using the PCIM facility or directly using newly created .net server classes to service the WEBFORM front end.

**JOB STREAMS**

A successful migration project isn’t only concerned with COBOL application programs – it must also migrate the batch JCL and the utilities they use (Report Writer, Security, Job Scheduler, Communications, Sort, etc). The legacy job streams can be converted to whatever job command language (JCL) the customer chooses. Virtually all our customers choose to convert to .bat JCL files for WINDOWS, but any WINDOWS JCL language, such as Perl, can be chosen. The utilities can be converted to whatever WINDOWS replacements the customer may choose. Typically for WINDOWS security is handled by Microsoft Active Directory. With the database in SQL Server there are many reporting facilities available for use including Microsoft’s own SQL Server Reporting Facility and products like Crystal Reports. WINDOWS has a basic job scheduler and Unicon has a job manager utility which is often sufficient for most customers but if a more comprehensive product is required I would recommend Global ECS from Vinzant Software. WINDOWS of course is highly proficient at providing all forms of communications – TPC/IP, Serial, FTP, NDM, etc. All the communication API calls are very well explained at the Microsoft MSDN site. For an external WINDOWS SORT utility I recommend COSORT from the company of the same name. UNICON can also provide a very comprehensive print queue utility. There are so many options available for these utility functions in a WINDOWS environment that finding the perfect option is never a difficult task. The main thought to consider is that successful migration is the process of migrating the entire eco system that was on the legacy platform to the new WINDOWS platform and the converted .bat scripts can be designed to perform the exact same logical functions as the original legacy JCL.

**SUMMARY**

No one shrink wrapped prepackaged solution is right for everyone. One size does indeed not fit all. Our customers have been successful in their migrations because they chose their new components that were right for their site, for their management, operations, developers and users; and that fit both their
current and future needs. Component conversion allows the customer the flexibility to choose the right mix to create a sound investment and foundation for the long term future.

One final thought – it may be quite surprising to consider that at Unicon the conversion process is all done through using automated converters.

CONTACT

Mike Howard
Unicon Conversion Technologies, Inc.
www.uctnet.com
949-454-1554 office
714-883-3332 cell
mike@uctnet.com