

Curing the Legacy Migraine: Implementing a Successful Migration Plan

By Craig Abramson

A migration project involves a huge amount of planning in order to be successful. This article addresses the many considerations surrounding legacy migration, including deciding on the best method to convert legacy data, determining how to rebuild a component or entire application on the new platform, and other key decisions that need to be made before you get the job done.

Mainframe. UNIX. Linux. Windows. Most companies' IT environments have evolved into a heterogeneous mix of these primary operating platforms. The challenge that companies face in this environment is when to develop applications, where to run them, and how to migrate applications from one platform to another to achieve greater efficiency and a more effective balance of data processing resources.

WHAT IS LEGACY MIGRATION?

What was once a cutting-edge application has now become cumbersome. Perhaps performance has dramatically decreased over time, costs have escalated, and/or new tasks are unable to be completed. You are unable to add the features that you need to the application because of the platform it resides on, but it is still critical to the business processes of a company. Instead of eliminating and replacing it, another more

viable option could be to move, or migrate, the application to a different system and platform. The purpose of this migration would be to retain the application while gaining additional functionality and possibly reducing maintenance costs on the new platform. Additional reasons for choosing to migrate instead of rebuilding the application on a new system from the ground up include budgetary constraints and the overall time it would take to complete the project.

CONVERTING THE LEGACY DATA

One of the keys to successfully migrating an application to a new platform is the ability to convert the data. For example, if you're migrating an application from the mainframe to UNIX or Linux, you'll need to convert the data from EBCDIC to ASCII and possibly rewrite JCL into shell scripts. In order to make this a smooth transition, it's important to first examine the data on the legacy system. This involves

several steps, starting with identifying and setting up different categories for the data. Instead of trying to move all the data over to the new system, you're determining which data has potential value and needs to be moved. Although this can be a cumbersome task, it will eventually help you reduce the amount of data you'll need to migrate so that the project will take less time to complete.¹

The next step is to go through each data category that you've created and evaluate if it's critical to the objectives of the application. Once this is accomplished, you may be able to eliminate up to 80% of the legacy data. You'll then have to analyze the remaining data in order to determine if it is used in other applications, preventing you from migrating to a new platform. If this is not the case, you can determine the costs associated with converting the remaining data to the new system. This step includes analyzing the quality of data, how accessible it is, and how long it will take to convert. Now you can add up the costs for each category and determine if it's within the budget.²

On the surface, the conversion of mainframe data seems simple—merely a translation from EBCDIC to ASCII with FTP or DD or some other UNIX utility. Unfortunately, mainframe data seldom consists of character data alone. It frequently includes packed decimal (COMP-3) and other binary fields, mixed with character. Translating binary fields from EBCDIC to ASCII with utilities like FTP or DD is a sure way of destroying them. Of course, you could unpack the binary data on the mainframe and translate everything to ASCII. But that takes time and a lot of work and doesn't give you a usable layout as output. One of the quickest, easiest and most cost-effective ways to convert the data is to use third-party software.

SOFTWARE CONSIDERATIONS DURING THE MIGRATION PROJECT

There are several questions that need to be addressed before choosing the right tools for the legacy migration project. These will help you select the necessary software for converting the legacy data and to start thinking about how the application is going to run on the new platform. The questions include:

What is the most common programming language used in the applications you plan to migrate? If the language is COBOL, conversion is normally quite easy. Other languages will need further investigation. Once a language

compiler has been selected, it is important to confirm that the features available in the old version are also available in the new one. It is critical to verify that there aren't any missing features that are crucial to your processing. You should also check the performance of the new compilers that you're considering.

How much do your applications depend on the use of Job Control Languages? If you're migrating a batch application from a mainframe, you should determine the number and type of tasks that are performed through Job Control Language. The amount of Job Control Language that must be converted to shell script is important. Although equivalent commands are likely to be available in the UNIX shell, high-volume conversion can be time-consuming.

What will you use to store your data on UNIX? If your database is SQL-based, it's more likely that conversion will be relatively easy. If not, the conversion effort could be substantial. Capacity, performance and other critical areas such as data security should be checked.

What software products have you come to depend on in your application? Most likely, you've incorporated third-party software products that play a vital role in your application. You can make a list of the software used and contact the vendors to see if it is available on the new platform. If so, make sure that all of the key functions are included and ask questions about performance and vendor reliability. If the software isn't available for the new system, then consider competitive products. Keep in mind that conversion can be very time-consuming.

What screen display interfaces are you currently using? If your current interface can't be ported, considerable redevelopment and staff retraining may be necessary. However, a new interface may be worth the time and effort if it allows your end users to work more efficiently.

Which security features and utilities are available on your system and how are they used? It's easy to forget about the tasks that security software and system utilities perform, but it's critical to check that the same facilities will be available in the new operating environment. If they are, determine how efficiently they will perform. If the security software or system utilities are provided by outside vendors, contact them about availability on the new platform.

The answers to these questions will give you a general idea of what software and functionality is needed on the new platform. Keep in mind

that determining all of the functions of the legacy application can be a difficult, if not impossible, task. After all, the application probably includes a combination of custom code and software that has been added over a long period of time with little documentation. By discussing the application with the end users to see exactly how they're using it, you can uncover key functions that aren't readily apparent as well as new features that should be added.

Performing an architectural assessment will also improve your understanding of how the system is constructed from a procedural flow, user interface and data structure perspective. The assessment will help you determine key issues that are relevant to data migration, end user perspective and related transformation issues. You'll gain a better understanding of how the legacy application has been built and the flow of data throughout the system, document interface points to related applications, and much more. The assessment process is divided into three stages: analyzing the flow and construction of the application, extracting and analyzing user views, and extracting and documenting data structure designs.³

The next step is to perform a functional assessment overview, which uncovers the functional capabilities and weaknesses across legacy applications and data structures. During this assessment, you will have to document the functionality of the legacy application, determine any functional similarities and differences between the legacy application and the architecture of the new platform, and decide if a component of the legacy application can be reused under the target architecture.⁴

CREATING A LEGACY MIGRATION PLAN

After analyzing your requirements for converting legacy data and determining what functionality the application on the new platform needs to provide, it's time to create a migration plan. In order to ensure the successful migration of the application, you can first break it up into standalone components that can be deployed on the new system. You are building a new collection of programs to handle a portion of the legacy application.⁵ If the key software programs that you use on the legacy application are also available on the new platform, it makes the migration much easier. You should be able to achieve the same processing requirements and similar performance results with the software running in the new application.

CITISTREET SAVES MILLIONS PER YEAR BY EFFICIENTLY MOVING ITS BENEFITS APPLICATION FROM MAINFRAME TO UNIX

The conversion process can also be broken up into two steps. First, you can translate the existing code, migrate the data, and perform the necessary testing. Then you can add the new functionality. This approach allows you to first verify that the new application provides the same functionality as the legacy application. Once this is confirmed, changes can be made to improve the program logic and structure.⁶

Another approach would be to migrate the whole application in a single step, but unless the application is very small, this process is very risky. After all, if you fail, the entire application fails. You'll also only have one system up and running using this approach. By building the application incrementally, you can reduce the risk of overall failure. Replacing one component that fails is much easier to do than reworking the entire application. You can also decide to only rebuild a key component of the legacy application on the new platform. Any write access to the old component would be redirected to the new system. This would be done if you can identify a component that needs additional functionality that is not available on the legacy system. By only replacing the one component, you're reducing the costs and risks of the migration project, but you're also not receiving any performance benefits because of all the disk writes that will need to be performed.⁷

Whichever method you select, it is important to test the new application or component and make sure that it meets your business requirements. This includes validating the data to ensure that it is successfully being converted to the new platform language. The application also depends on the availability of the data, so it's also critical to implement a backup and recovery strategy. When choosing a backup and recovery tool, there are several factors to consider:

- ▼ Determine what your backup window is and then test a product to see if it can perform in that timeframe. One way to increase the speed of the backup is to select a tool that performs differential or incremental backups.
- ▼ Determine how quickly a backup solution will be able to restore your files.
- ▼ Determine how easy it is to restore a file—does the tool use a graphical user interface (GUI) operable by almost anyone, or do you need complex commands?
- ▼ Consider platform compatibility—in a legacy migration project, it is essential that a high-performance backup is compatible with a variety of platforms such as NetWare, UNIX and Windows. This also allows for scalability.
- ▼ Determine security levels—a backup solution should have various levels of encryption and device distribution to protect the data as it travels across a network.

IMPLEMENTING THE PLAN

You've acquired the necessary tools to convert the data from the legacy system to the new platform. You've developed the plans needed to build the application, including figuring out the budget, adding software, writing custom code, and determining the approach to take. Management has approved the project, and now it's time to implement it. This can be a very time-consuming process fraught with trial and error. There may be a mistake in custom coding that needs to be corrected or the legacy data wasn't formatted properly after you converted it. One way to avoid common mistakes is to learn from the experiences of other companies that recently implemented a legacy migration plan.

Everybody has retirement dreams, and the key to making them come true is to have a sound retirement plan. CitiStreet, one of the largest global benefits delivery firms in the United States, helps people develop those plans and is an excellent example of how one company migrated a critical legacy application. The company serves more than 7.5 million participants, administers approximately \$180 billion in assets in the United States, has about 800,000 participants, and administers about \$2.5 billion in assets outside the United States.

One of the options that CitiStreet offers is a defined contribution plan in which each participant contributes a specific amount. The benefit payment in this plan is not defined, but depends on the amounts contributed and the investment performance of the account. With the number of corporate, government, health care, Taft-Hartley and not-for-profit organizations participating in this plan at CitiStreet, the company has to handle a large amount of data. In order to accomplish this processing, CitiStreet performs up to 5,000 batch jobs a day. Each batch job can contain in excess of 50 GB of data and consists of all of the financial and demographic information involved in record keeping a defined contribution retirement plan. The information is then used to create reports that provide detailed participant information.


Although the company was satisfied with the performance of their application, they needed to reduce their ongoing cost structures while still providing excellent service to their clients. "Part of this strategy was to move from a mainframe to a UNIX environment for the SunGard OmniPlus® defined contribution recordkeeping system," explained CIO Barry Strasnick. "Because CitiStreet supports some very large defined contribution plans, along with thousands of smaller ones, efficiency is critical to our operation." He was concerned that the performance might be an issue while making the transition from mainframe to an HP-Superdome running HP-UNIX. His goal was to achieve the same processing time results that he was getting on the mainframe.

A critical step in their legacy migration was to be able to convert the data from EBCDIC to ASCII. The company selected a product from Syncsort Inc. CitiStreet then had to find a tool to process and order the data once it was converted. Since Strasnick had previously worked with SyncSort on various platforms, he decided to test the utility. "SyncSort allowed our COBOL batch processing to be accomplished on the HP Superdome with the same or quicker time periods than on the IBM mainframe," he stated.

The new application included ordering the data within 15 to 20 minutes and converting large amounts of data from EBCDIC to ASCII over limited processing windows. CitiStreet's legacy migration project was successful because the company developed an effective plan by first analyzing their project requirements and then determining what tools were needed to achieve the desired results. Instead of trying to write a custom program, third-party software was used to handle the data conversion and the record keeping system was rebuilt on UNIX. The company was able to meet their performance goals as they migrated to UNIX, and in the process, saved millions of dollars.

SUMMARY

A migration project involves a tremendous amount of planning to ensure its success. From figuring out the best method to convert legacy

data to determining how to rebuild a component or entire application on the new platform, key decisions need to be made that will affect the costs and time involved. If the legacy application uses third-party tools that are available on multiple platforms, the transition to the new platform will be much quicker and easier. Performing architectural and functional assessments will also help improve your understanding of how the system is constructed and determine what functionality needs to be included. And after you've developed and implemented the migration plan, the application will once again be considered cutting-edge, offering the end users everything they need to get their job done. 

NaSPA member Craig Abramson is a technical analyst at Syncsort Incorporated, focusing on the latest data warehouse performance solutions. He has over seven years experience in the field working on projects dealing with data warehousing, database management and Web log processing.

¹ Girard, John. "Migrating Legacy Content" <http://www.cmswatch.com>, (April, 2003)

² Girard, John. "Migrating Legacy Content" <http://www.cmswatch.com>, (April, 2003)

³ Ulrich, William M. "Legacy Systems: Transformation Strategies", Prentice Hall PTR 2002, pp. 241

⁴ Ulrich, William M. "Legacy Systems: Transformation Strategies", Prentice Hall PTR 2002, pp. 242

⁵ Zoufaly, Federico. "Issues and Challenges Facing Legacy Systems"


<http://www.developer.com/java/article.php/1492531>, (2003)

⁶ Zoufaly, Federico. "Issues and Challenges Facing Legacy Systems"

<http://www.developer.com/java/article.php/1492531>, (2003)

⁷ Keller, Wolfgang. "The Bridge to the New Town: A Legacy System Migration Pattern"

<http://www.objectarchitects.de/ObjectArchitects/>, (2000)

Technical 

Supporting Enterprise Networks and Operating Environments

SUPPORT