Reevaluating Migration and Modernization Strategies

Choosing the Right Strategy

HP 3000 application users have been forced to evaluate the value of their existing so-called Legacy Applications. This is both bad news and good news. The bad news of course is that the discontinuance of the platform means that if the application is deemed valuable enough to retain and continue to use long term, one has to look at the prospects for getting continuing acceptable operational reliability from the system. It should be expected that over time staying on the HP 3000 platform will become increasingly more unreliable. For many companies, this means that they should consider migrating to another platform. The good news is the opportunity that those organizations have for extending the life cycle of the applications that will be retained.

Those organizations faced with migration decisions seem to be faced with two key decisions that appear to be in conflict. On the one hand, the safest, quickest, least risky and least expensive alternative for application retention and migration is to do a like-for-like migration. This means that the application is moved to a new operating system with minimal changes to code and functionality. On the surface, this idea appears to be in conflict with the desire to modernize the application and make more like applications that were designed more recently. Examples of changes that might be desirable are applying one or more of the standard modernization techniques:

These techniques include:

- Hardware Modernization (actual or virtual operating systems)
- Application/software modernization
    - User interface modernization
    - Connectivity and integration
    - Programming Language modernization
    - Database modernization
    - Application modernization
    - Application re-engineering

Transformix can show how its Open-Ended Transformation Approach facilitates the possibility of both seemingly conflicting goals to be met at an initial cost, risk level and time frame that is comparable with simply doing a like-for-like migration.

Choices made in the initial migration determine the organizations ability to have the organization evolve to meet future corporation needs. For example, Transformix can show that it is no more difficult to migrate to an RDBMS such as Oracle, IBM DB2, Microsoft SQL Server or Postgresql than it is to migrate

to Eloquence. However, the benefits of using any of these RDBMS's versus continuing to use only IMAGE –like functionality are considerable. That is, simply having an RDBMS available opens the door to a large number of modernization possibilities.

Transfo0rmix believes that migrations should result in Open-Ended extensible solutions that retain the core functionally of the existing application but at the same time that extend the life cycle of the migrated application. We believe that re-implementing applications on new platforms in this way can reduce implementation costs and difficulty, and the additional capabilities of new technologies can provide access to functions such as web services and integrated development environments. Once transformation is complete and functional equivalence has been reached the applications can be aligned more closely to current and future business needs through the addition of new functionality to the transformed application.

Transformix uses many technologies such as tools based migration, a group of tools that migration and provide an MPE-like runtime environment on modern platforms, and program transformation by software which results in re-implementation of older languages such as BRW or Suprtool into JAVA. Products such as the BRW translation and SuprtoolSQL have made the transformation process a cost-effective and accurate way to preserve legacy investments and thereby avoid the costs and business impact of migration to entirely new software.

With an intelligent and informed choice of migration tools and strategy, the life of an existing application can be extended to meet the continuing needs of an organization. There are several ways to make a wrong choice. One can try to do too much and fail or greatly exceed cost and time estimates. One can be "successful" while only achieving limited goals. While a non-informed choice of tools and approach might seem successful on the surface, it can lead to situation where in order to get a more modern application to users, the application will need to be migrated more than one time.

Legacy Transformation or Modernization

Legacy Transformation, or legacy modernization, refers to the rewriting or porting of a legacy system to a modern computer programming language, software libraries, protocols, or hardware platform. Legacy transformation aims to retain and extend the value of the legacy investment through migration to new platforms.

A legacy application is any application based on older technologies and hardware, such as mainframes, that continues to provide core services to an organization. Legacy applications are frequently large and difficult to modify, and scrapping or replacing them often means re-engineering an organization's business processes as well. However, more and more applications that were written in so called modern languages like java are becoming legacy. Whereas 'legacy' languages such as Cobol are top on the list for what would be considered legacy, newer languages can be just as monolithic, hard to modify, and thus, be candidates of modernization projects.

The goal of legacy transformation is to retain the value of the legacy asset on the new platform. In practice this transformation can take several forms. For example, it might involve translation of the

source code, or some level of re-use of existing code plus a Web-to-host capability to provide the customer access required by the business. If a rewrite is necessary, then the existing business rules can be extracted to form part of the statement of requirements for a rewrite.

How Business Software Has Changed

Independent of the operational reliability issues are considerations related to how the world views information systems today versus how they did when most HP 3000 applications were written. That is, an application designed today and how it is hosted would likely have a different structure than one originally written for the HP 3000. Some key areas of difference are shown in the chart in Figure 1. Therefore, even though it might make excellent business sense to retain the application in many cases it is desirable for the application to share the characteristics of more modern applications.

	HP 3000 MPE/iX	Modern Systems Environment
Hardware and Systems Software Modernization (actual or virtual operating systems)		
•		
•		
•		
Application/software modernization		
• User interface modernization	Terminal based	Web, Mobile phone, GUI
• Connectivity and integration		
a) OpenSSH and Security		
b)		
c) Outside connectivity – Social networking		
•		
•		
•		
• Programming Language modernization		
• Database modernization		
• Printing modernization		

Table 1 - Feature Comparison - MPE Application versus Modern Application

Open-Ended Transformation

It would be nice if software applications could be made to be fully adjustable. Making the core application that the user is responsible for more easily adapted is outside of the scope of this paper.

However, as Figure 1 indicates, there are many areas outside of the core application that can be accommodated external to the application code. Here are some examples in Table 2:

	Existing Application	Transformix Solution
Hardware and Systems Software Modernization (actual or virtual operating systems)		Nothing to do
• UNIX, Linux and Windows		
• Virtualization		
• Storage		
Application/software modernization		
• User interface modernization	Terminal based	VPLUS to XML allows VPLUS applications to have GUI, Web, Mobile phone, Interactive voice response front ends with no change to source code.
• Connectivity and integration		
a) OpenSSH and Security		FTP in job streams changed underneath to SSH. JCL does not change.
b) Network of application servers and shared database servers		
c) Interoperability between operating systems and multiple systems		
d) File and directory sharing - CIFS, NFS		
e)		
f) Outside connectivity – Social networking		
• Programming Language modernization		
a) COBOL		
b) Suprtool		
c) Speedware		
d) SPL		
• Database modernization		
e) IMAGE		RDBMS makes …
a) KSAM		Stored in RDBMS
b) MPE Message Files		Implemented with message queuing
• Printing modernization		Jasper reports so

		output can be PDF, EXCEL, CSV, Word, etc.

Table 2 - Transformix Modernization Approach

It is important to note that some of the modifications are byproducts of the Transformix approach. That is, the users simply let us know that they want to have TurboIMAGE and KSAM migrated to an RDBMS of their choice and that's it. That's what we do doing the migration. On the other hand, some of the changes are possible after the initial migration is performed because the required hooks are already in the software and were placed there during the migration. An example of the later is the replacement of VPLUS screens with either GUI or Web interfaces. When we migrate VPLUS forms we store them as XML objects. These same objects can be used to implement the users choice of user interface with no change to the core application.

Conclusion

Modernizing existing MPE applications can on the one hand be a byproduct of migration and on the other hand, migration can make future modernizations easier. This approach gives IT managers a way to accomplish more results for users with less investment of both time and money. As these orgainizations work to reduce risk and extend the life and versatility of their applications, Transformix is uniquely positioned help companies achieve their business and technical goals with a growing number of tools and skills

Contact Transformix Computer Corporation for more information at www.xformix.com.