Comparing COBOL Application Porting Approaches

By Al Gates and Anton Groom
With contributions from Frank Calvillo, Dan Kroeger, and Rich Trapp

Managed Business Solutions
(970) 224-1016
al.gates@thinkmbs.com
anton.groom@thinkmbs.com

Introduction

Your COBOL enterprise applications running on HP3000s that have been a key asset to your company now need to move to a new environment. Maintaining the asset value of these applications will require a real investment sometime in the next five years. The fundamental decision you now face is one of balancing the costs of migrating the applications against their value as company assets in a new environment.

Migrating your COBOL applications from an HP3000 architecture to a new platform requires far more than simply converting your hardware. This may be surprising; after all, you can take a C^{++} application running on an old HP Netserver and run it on the new HPQ Proliant server without the level of effort being talked about for HP3000 migrations. The reason is that it's not only the HP3000 hardware that is being retired, but also a number of core applications and tools that run in the HP3000 environment are being retired along with it.

Foremost amongst these is the MPE operating system, which is taking with it all of the system INTRINSICS called by COBOL and other programs. Also being retired is the HP Image database, which was a highly integrated part of most HP3000 COBOL applications, and losing this database system means losing all of the COBOL code that accessed it. The HP COBOL programming language and its compilers are also being retired with the HP 3000 and this requires that all your COBOL code be converted to a

version of COBOL that will compile on one of the new operating systems. Unfortunately these new versions of COBOL do not support all features supported by the HP e3000.

This paper first describes a process for developing a migration strategy for your HP e3000 enterprise applications. It then provides a review of potential new environments, a crucial choice in choosing an application porting approach. Assuming that you decide to port your applications to a new environment, the paper then discusses migration tools and services available to support this effort, resulting in equivalent, or improved, functionality. In its conclusion, the paper emphasizes the key decision points in a migration effort.

The Migration Process

Overview

The migration process for developing a migration strategy for your HP e3000 enterprise applications includes five major steps:

- Create a server inventory
- Create an application inventory
- Perform a migration assessment of each application
- Determine migration sequencing
- Create an executable migration plan

Each of these steps is described in the following sections.

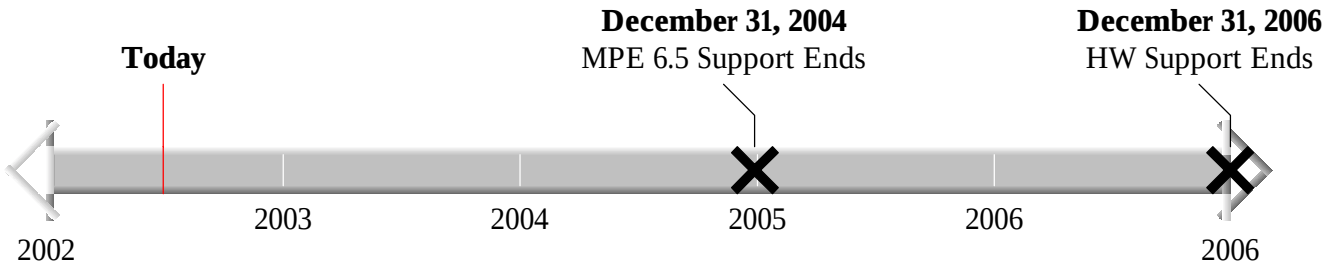
Create a Server Inventory

The most prudent first step is to begin by taking a thorough inventory of your existing COBOL enterprise applications and underlying systems. This is a practical action that can be started immediately by the technical staff, without having to wait for management to begin the strategic planning activities, and will provide essential data to be used in the planning stage.

The server inventory phase requires the compilation of a full inventory of the hardware and software that comprise the enterprise application environment, such as the hardware model, version of the operating system, and third party software acting as job schedulers, backups, or utilities. It should also provide a list of Independent Software Vendors (ISVs) associated with each environment application, and any support contracts for each application. This inventory can be compared against HP's e3000 Support Plan, which is shown in Figure 1 below (as of July 20, 2002). Based on your server inventory, and the figure below, you can identify when support ends for your configuration, and lay out a migration timetable based on the end of HP support.

As you can see from Figure 1, all HP3000 hardware will be supported until December 2006. This includes 968, 978, 988, 939, 959, 969, 991, 995, 996, A-Class, and N-Class servers. MPE 6.0 will no longer be supported after October 31, 2002; MPE 6.5 will no longer be supported after December 31, 2004. All later versions of MPE will be supported until December 2006.

Figure 1 – HP Support Time table

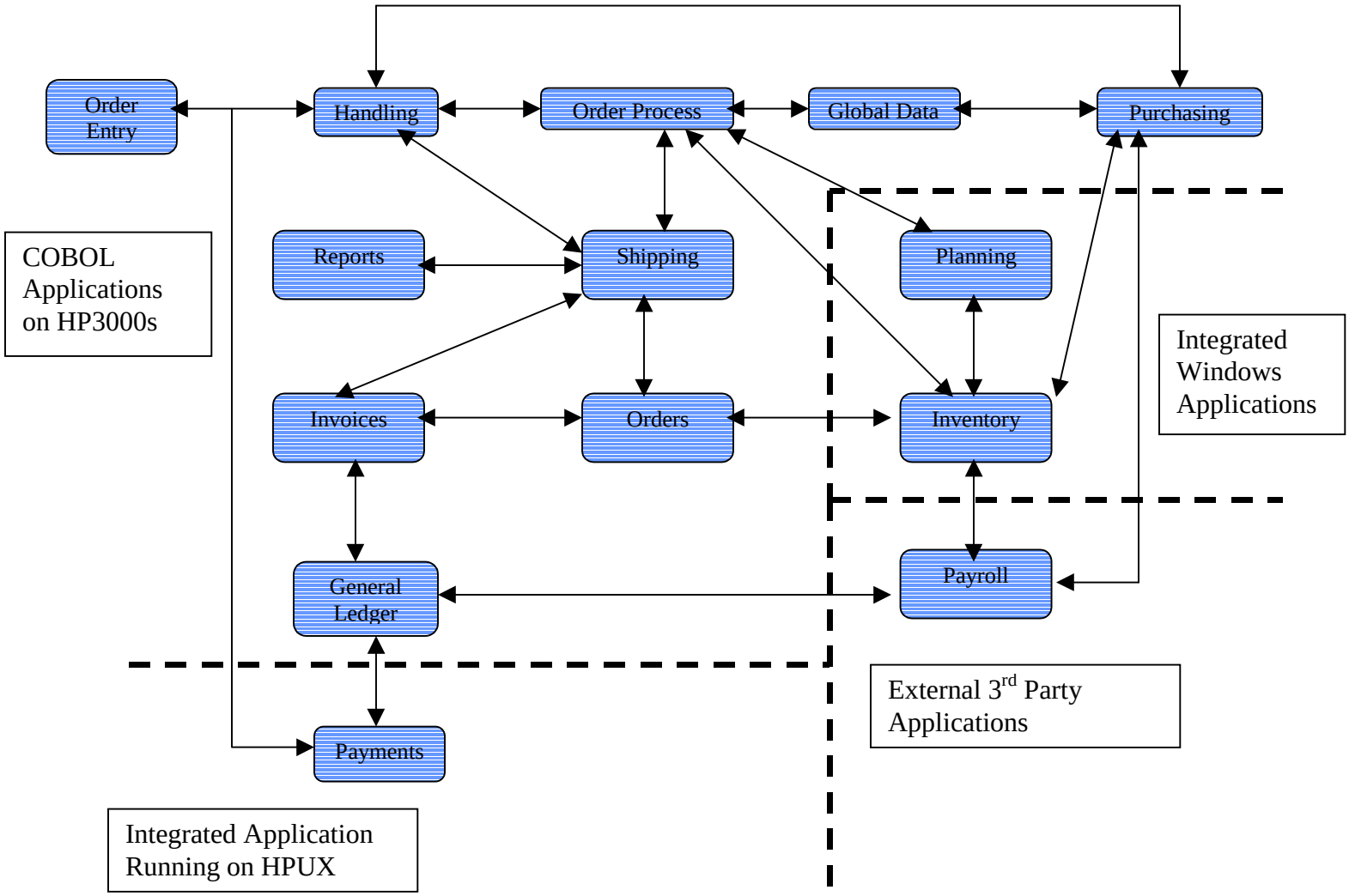


Create an Application Inventory

The applications inventory phase requires that you first identify all the applications that will need to be migrated, as well as all other applications that constitute the enterprise system. Details on organization, networking, third party applications, intrinsics, and external references must be documented. One deliverable of this stage is an application topology, such as the one illustrated in Figure 2. The topology shows the existing applications (boxes) and their data flows (arrows), which shows at a high level how data moves between components in your enterprise applications. This will be crucial information when building a step-by-step migration plan. As individual applications are migrated to the new environment, all of its interfaces will need to be maintained or addressed, temporarily or permanently, as part of the move.

The topology should include all components of the enterprise applications, whether they are running on an HP3000 or not. Supporting the topology is a list of application details, including counts of modules, lines of code, user interfaces (screens & reports), tables in the database schema, along with lists of cross-application interfaces, system dependencies, and associated ISVs.

Figure 2 – Sample Enterprise Application Topology



Perform a Migration Assessment of Each Application

The deliverables from this assessment should include a migration strategy and plans for each application in your enterprise suite, the scope of the migration, and any identified migration issues.

In this phase, it is important to thoroughly assess how well the application is serving the needs of the users and the enterprise as a whole. This should be done based on the following keys:

- Functional adequacy (is the application satisfactory or does it need enhancement)
- Maintainability (how difficult or costly is the application to maintain)
- Web-enabling (is there a need to implement web-clients)
- Integration requirements (is there a need to integrate the application with others)
- Incompatibility issues (list known issues and potential problems)

If the application is serving the needs of the enterprise well, then there is a stronger benefit to porting the application. If it is not serving the enterprise well, this may be a good opportunity to explore new options, such as purchasing a new application, or re-engineering the application.

Provided below is a basic decision process to help guide you through this migration assessment phase. A decision tree leads you to one of four migration paths for each application:

1. **Stay** – Keep the application in the current environment for as long as possible, or for a determined length of time
2. **Port** – Migrate the application to a new environment
3. **Build** – Re-engineer the application in a new environment
4. **Buy** – Purchase a new version of the application from an Independent Software Vendor (ISV), or roll the application into an already purchased application.

The basic decision tree is based on the following ten questions.

1. **How mission critical is the application?**
2. **Are there contractual, legal, or business reasons for migrating or for the status quo?**
3. **Is the expected life of the application greater than five years?**
4. **Does the application meet business process requirements?**
5. **Is the application custom built or purchased from an independent software vendor (ISV)?**
6. **Does the ISV have a migration path for the application?**
7. **Should this application's functionality be rolled into a different or larger pre-existing application?**
8. **What is the IT department's strategy regarding new platforms?**
9. **Does the IT department own the source code?**
10. **Are the application source code and application environment portable to the target platform?**

Figure 3 shows the paths of the basic decision tree, and helps to illustrate the migration decision process.

The first step is to determine the business criticality or level of importance of the application(s) to be migrated. All mission critical applications must be migrated, while non-mission critical applications must be assigned a level of importance and this will be used to balance against a cost estimate to migrate that application. For all but the most unimportant applications you will continue on with the decision process, for the unimportant applications the strategy is to "stay".

The next step is to determine whether there are any legal or business requirements that absolutely determine whether the application must be maintained "as is", or be migrated. For applications that must be maintained "as-is", this is the end of the process and the strategy is to "stay". Those applications with requirements to be migrated can skip the next step, whereas those applications with indeterminate status for this question must undergo the next step.

The next step is to determine the expected useful lifespan of the application. For any applications for which the answer to this question is "5 years or less", the strategy is to "stay". It may be possible with sufficient non-HP support and service to stretch this window period out to 8 to 10 years, but it is not recommended, and will not be discussed in this paper. All applications with longer expected lifespans move on to the next step.

This step requires that you consider the data collected in the application assessment phase and determine whether the application is meeting the needs of the enterprise. If the answer is no, then the next step is to do a thorough market search for a non-HP e3000

application that does meet the enterprise's needs. If one (or more) is found, then the strategy is to "buy". If none are available then the strategy is to "build" a custom application in a new target environment.

If the existing application meets all, or enough, of the enterprise's needs then the path followed depends on whether the application was supplied by an ISV or whether it was custom built. Assuming it was ISV supplied then the next step is to determine whether the ISV has a defined migration path. If so, the strategy is to "port" the application using the ISV's defined migration path. If the answer is no, then the next question is whether or not you own the source code for the application. If the answer is no, then the next step is to determine if there is an existing application that will duplicate (or improve) the suitability of the existing application. If the answer to this is yes, then the strategy is to "buy", if the answer is "no" then the strategy is to build.

If the answer to the source code ownership question is yes, then the decision path arrives at the same point as if the application was custom built and not from an ISV. At this point you must look to the company's IT strategy and see how well defined it is regarding target environment. For a strict IT strategy the next step is to determine whether the application can be ported, using one of the many available tools, to the new environment. If the answer is yes, the strategy is to "port", if the answer is no, then the strategy is to "buy" an application that will duplicate (or improve) the suitability of the exiting application in the IT defined target environment.

If there is no IT strategy, or it allows enough leeway in target environment, then the next step is to determine if the application is portable to any of the possible target environments. This determination will be a significant effort as many environment combinations and possible tool sets must be assessed. If the answer is yes, then the strategy is to "port", and if there is no way to port this application, the strategy is to "buy".

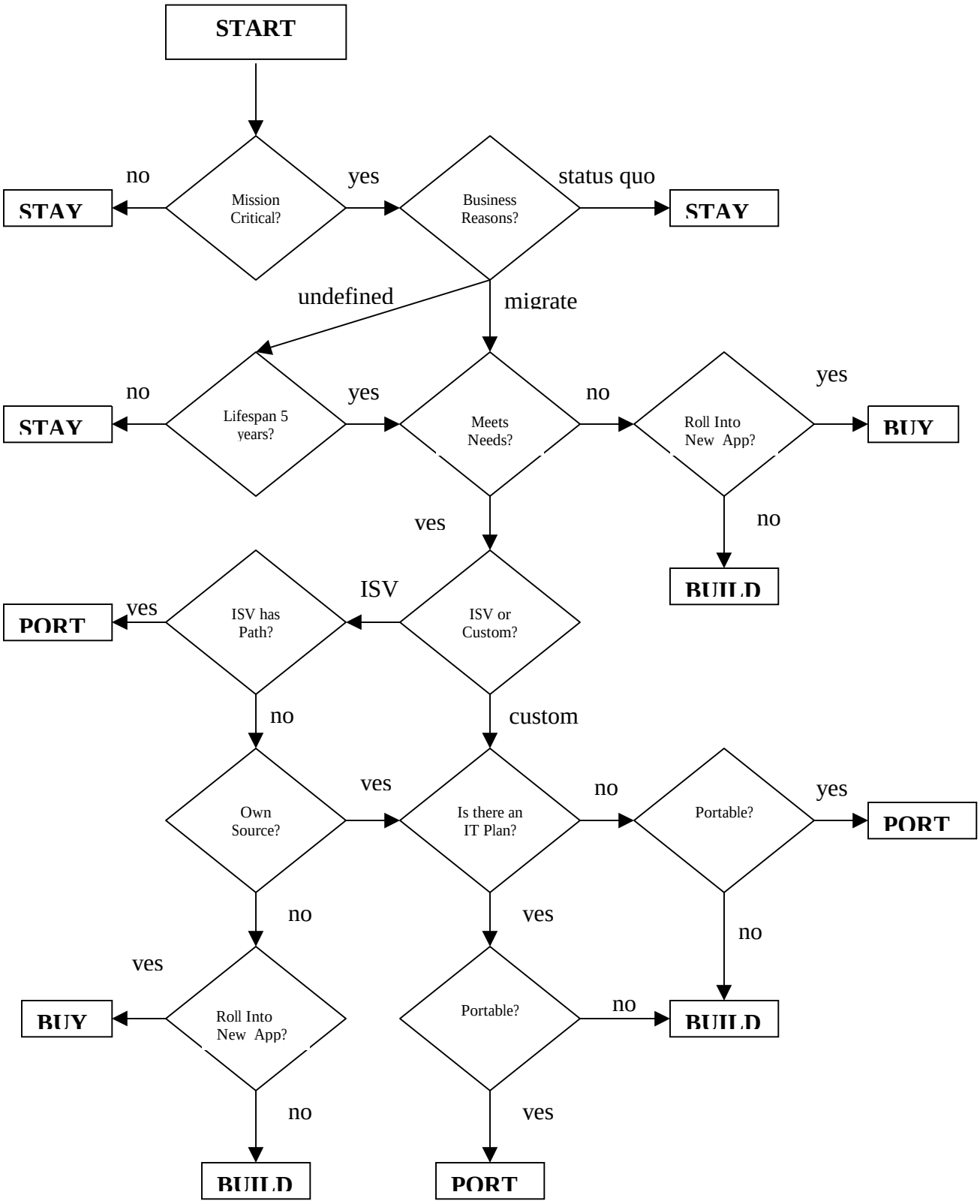
Determine Migration Sequencing

Once you have identified whether you choose "stay", "port", "build" or "buy" for each of your applications, you can then lay out a sequence for migrating each of the applications based on the topology you have drawn. The goal is to maintain the best possible integration of applications throughout the migration effort. Using the application topology, you can identify groups of applications that are relatively decoupled, and migrate the application in these groups. This will minimize the effect and effort to address temporary and permanent interface changes between applications.

Create an Executable Migration Plan

Once you have identified a sequence for the migration, and have set the timeline for your migration (as was done during the server inventory), you can then identify resource requirements over time for the migration effort. Now that you have identified the strategy for migrating each of the applications, the sequence and timing for the migration, and the resource requirements, you are ready to document this into an executable migration plan.

Figure 3 – Basic Decision Tree for Choosing a Migration Path for Applications



New Environment Analysis

Overview

One of the most important factors in your migration planning is choosing a new environment for your enterprise applications. This choice will have a profound effect on your migration strategy, and the viability and process for porting your COBOL application.

The target environment is comprised of the hardware, operating system (OS), programming language(s), database type, enterprise architecture, and client interface options. These all interact in different ways. Some components are incompatible while others are tightly integrated. Your choice of hardware may limit your OS options, and vice-versa. Your choice of OS may limit your database options and programming language options. The benefits of one database may be offset by the drawbacks of the best-suited companion operating system. All of these trade-offs and benefits must be considered as a whole and not in isolation. The process of planning a target environment is a complex one, and requires that boundaries are defined, priorities are set and all the myriad pros and cons be carefully analyzed.

The first step in deciding on a target environment is looking for guidance in the official IT strategy of the company. Companies with a well-defined strategy will expend very

little time and money on this step but their choices will be limited, and their company's pre-defined target environment options may not allow for the optimum combination of components for their enterprise applications. In contrast those companies with no defined strategy can spend a great deal of time and money getting through this step, but should set themselves the goal of ending up with the optimum environment for the application.

For companies with some leeway in defining their target environments, there are a number of key factors to consider in compiling the pieces of the whole. The most important amongst these factors are: price, ease of use, support (cost & availability) scalability, security, reliability and integration with the other components of the enterprise application. These factors should be considered for each of the components that make up the target environment, including operating systems, databases, programming languages, and development frameworks.

Operating Systems

This paper will focus only on the following market leader operating systems, and they are Windows, Linux and HPUX. Windows is a Microsoft product that first appeared as Windows 1.0 in 1985, and currently is spearheaded by the Win2000 version for servers and the XP version for desktops. Windows systems popularity is built upon the strength of their graphical user interface, and the associated ease of use. Windows dominates the desktop market, with approximately 90% market share. Linux is a free open source implementation of Unix that runs on a number of hardware platforms. It was primarily

developed by Linus Torvalds, but has seen numerous evolutions and upgrades. Linux has become increasingly popular over the last couple years. HPUX is a proprietary HP version of Unix, developed and performance-tuned for use on HP servers. With the latest version, 11i, HPUX delivers a highly available, secure and manageable operating system, and a choice of four different HP-UX 11i operating environments, each tailored for a specific implementation: internet, enterprise, mission-critical deployments and technical computing environments.

The operating system you choose to target should be based on the company IT strategy and the available skill sets. But for those without a strictly defined strategy the selection of an OS can be based on a comparative analysis. The most popular choices for enterprise applications today are HPUX, Windows 2000 and Linux, but Unix is dominant and currently holds a market share of approximately 60% for application servers. However it is expected to lose ground to around a 50% share by 2006. (IDC 2002) Windows 2000 is a well-supported and popular operating system, which is integrated with the new MS.net architecture. Windows currently holds approximately a 25% share of the application server market and is expected to gain a few points over the next five years. Linux is gaining in popularity but currently holds only a 4% market share of application servers, which is expected to grow to a 10% share by 2006.

Table 1 illustrates our opinion of how the operating systems compare on a range of features.

Table 1 – Operating System Comparison

	Linux	HPUX	Windows
Price	5	2	2
Ease of Use	2	1	4
Support Cost	3	2	4
Scalability	2	4	3
Security	3	4	2
Reliability	4	5	3
Support Features	2	3	3

Please note: Tables presented in this paper use a relative scoring system (5 is best and 1 is worst). This means that we are not promoting one product over the other, but rather that we are attempting to present a comparative rating for each of the listed features. These ratings are subjective and based on our internal research.

Databases

Next, you need to consider which database platform is the best target. The three most popular databases implemented by companies migrating from the HP3000 environment are SQL Server, Oracle and Eloquence.

SQL Server is a Microsoft product, highly integrated with the Windows operating system, but entirely unsupported on Linux or HPUX. Some benefits of Microsoft SQL Server are:

- Ease of Use: Ease-of-use encourages fast user-adoption, improving interactive understanding of business-critical information across the entire company.
- Business performance focused: Provides an interactive, dynamic intelligence solution for different classes of users, from simple to complex, including content viewers, analysts and developers.

- Enterprise-class scalability: Integrates with and leverages almost any existing information source to establish a scalable, standards-based environment for developing business intelligence solutions.
- Secure: Provides safe exchange of business information through the ability to secure content to different users across the enterprise
- Rapid deployment: Administers easily for fast deployment and lower cost of support.

Oracle 9i is the latest (non-ERP) version of Oracle's database product line. Oracle runs on the following platforms: HPUX, Linux, Windows, Sun Solaris, and IBM. Some benefits of the Oracle 9i database are:

- Increased transaction processing (Scalability)
- XML and Java Support
- Advanced security features
- Backup and Recovery (Reliability)

These features allow Oracle 9i to be a competitive solution for any enterprise database application.

Eloquence is an HP proprietary database product designed by Marxmeier Software AG. Eloquence runs on the following platforms: HPUX, Linux and Windows. Some benefits of Eloquence are:

- Close emulation of the HP3000 Image database. (Turbo Image Interface)
- Java Support
- Low Cost

Eloquence is primarily touted as an interim solution for migrated applications, but depending on the specific needs of an enterprise, it may be a suitable longer-term solution.

Table 2 – Database System Comparison

Table 2	SQL	Oracle	Eloquence
Price	3	2	4
Ease of Use	3	2	3
Support Cost	4	1	4
Scalability	3	4	2
Security	3	4	2
High Volume Data	3	5	2
Skill Set Availability	1	1	4
Warehousing	3	4	2

Please note: Tables presented in this paper use a relative scoring system (5 is best and 1 is worst). This means that we are not promoting one product over the other, but rather that we are attempting to present a comparative rating for each of the listed features. These ratings are subjective and based on our internal research.

Programming Languages

This section will focus on four programming languages, two of which are effectively the same language. The languages are Visual Basic, C#, C++ and Java. Visual Basic and C# are both designed for use with the .Net framework, and are effectively the same language with little more than syntax rules to tell them apart.

Visual Basic (VB) is a Microsoft product based on the BASIC language. VB was the first language to provide a visual IDE for developers, which facilitated rapid application development (RAD) approaches. In its earlier forms VB was an event-driven language, but in the latest release, VB.Net, it is fully compliant with object-oriented standards. VB started the current trend toward the visual IDEs that are now available for most programming languages. C# is Microsoft's newest language, fully integrated for the .Net framework and built to compete with Sun's Java language. C# is an object oriented language and provides type-safety, garbage collection, versioning support and scalability. C# and VB applications are only supported on the Windows operating system, and are therefore not easily portable across systems.

C++ is a high-level language built on the foundation of C, and was developed by Bjarne Stroustrup at Bell Labs. C was originally developed as a systems programming language, but became extremely popular for applications programming. C++ adds object oriented features to the original C foundation and is currently one of the most popular programming languages. Although it is a high level language, C++ is much closer to assembly language than its competitors, allowing for the development of very efficient code. C++ programs can be written for almost any operating system, but must be compiled specifically for each platform. Its source code is portable, but its executable files are platform specific. There can also be OS specific C++ calls that need to be addressed when moving to new platforms.

Java is an object-oriented language similar to C++ but simplified to eliminate features that have caused common programming problems, such as memory allocation, which is remedied by automatic garbage collection. Java source files are compiled into bytecode format files, which are then executed by a java interpreter and on a runtime environment called the Java Virtual Machine (JVM). The JVM is available for almost all operating systems, making Java a truly platform-independent language. This makes Java applications portable across systems with no need to modify the source code at all.

Application Frameworks

Another consideration is the programming architecture of the enterprise application. Application frameworks streamline and improve your enterprise application development environment. In this category, we will consider the J2EE architecture and the .Net architecture. These two frameworks dominate the market for applications that utilize the internet and have web client interfaces as part of the client-server design. The .Net architecture is integrated with the windows operating system and the IIS web server, while the J2EE architecture is integrated with the Java Virtual Machine (JVM) and runs on any operating system, and is used most commonly with the Apache Web Server, or in conjunction with applications like JRun and Jakarta's Tomcat. The fundamental difference between the two is that J2EE is a standard to which products are written, while .Net is a product strategy.

There is little difference between these two frameworks, however the J2EE platform has been around longer in its current form, while .Net is a newer platform built upon the

successful Windows DNA foundation. Both provide developers with a large number of code libraries and defined functionality, both support web services, and both have great numbers of skilled developers available to the market. Table 3 outlines the comparison between the two technologies.

Table 3 – Application Framework Comparison

Feature	J2EE	.Net
Framework Type	Development Standard	Product
Middleware Vendors	30+	Microsoft
Language Interpreter	Java Runtime Environment (JRE)	Common Language Runtime (CLR)
Dynamic web pages	Java Server Pages (JSP)	Active Server Pages (ASP.NET)
Middle-tier components	Enterprise Java Beans (EJB)	.Net Managed Components
Database access	Java Database Connectivity (JDBC) & SQLJ	ActiveX Data Objects (ADO.NET)
Communication Protocols:		
SOAP, WSDL, UDDI	Yes	Yes
Implicit Middleware	Yes	Yes

Integration

As discussed at the top of this section, the target environment is not simply a sum of its parts, but a complex matrix of interactions, dependencies and compatibilities that must be optimized to deliver the desired performance. Table 4 illustrates the compatibility between the operating systems and the databases (using “yes” or “no”) and highlights databases that are tightly integrated with an operating system (using “integrated”). Table 5 illustrates the compatibility between the operating systems and the programming languages (using “yes” or “no”) and highlights languages that are tightly integrated with an operating system (using “integrated”). These (and other) simple two-dimensional matrices can be used as guides in compiling the complex multi-dimensional target environment matrix. It is important to factor the skill sets of your developer and support

staff into this matrix, and include hardware and other resource limitations or dependencies in the final matrix.

Table 4 – Operating System / Database Compatibility

	MS SQL	Oracle	Eloquence
Linux	No	Yes	Yes
HPUX	No	Yes	Yes
Windows	Integrated	Yes	Yes

Table 5 – Operating System / Programming Language Compatibility

	vb/c#.net	Java	C++
Linux	No	Yes	Integrated
HPUX	No	Yes	Integrated
Windows	Integrated	Yes	Yes

Migration Tool Analysis

Overview

Assuming that you have decided to port your COBOL applications and that you have chosen a new environment for your applications, the next step is to determine the best set of migration tools to use to efficiently migrate your application.

Migration Tools

There are four major migration tool and service vendors identified in this paper, who provide a total migration solution to address database, user interface, JCL, and source code migration. They are Neartek, Ordina Denkart, Sunguard Bi-Tech, and Transoft. The matrices below (Table 6) show which new environments each of the migration vendor's support.

TABLE 6 – Migration Tool Environments

Operating Systems

	linux	hpux	win
Neartek	Yes	Yes	Yes
Ordina Denkart	Yes	Yes	Yes
Sunguard Bi-Tech	Yes	Yes	Yes
Transoft	Yes	Yes	Yes

Databases

Table 2

	SQL	Oracle	Eloquence
Neartek	Yes	Yes	Yes
Ordina Denkart	Yes	Yes	Yes
Sunguard Bi-Tech	Yes	Yes	Yes
Transoft	Yes	Yes	Yes

Programming Languages

	vb	Java	C++	COBOL
Neartek	No	User Interface Only	No	Yes
Ordina Denkart	No	User Interface Only	No – See Note 1	Yes
Sunguard Bi-Tech	No	No	No	Yes
Transoft	User Interface Only	User Interface Only	No	Yes

Note 1: Ordina Denkart does offer Pascal and SPL translations to C++

Development Frameworks

	.net	J2EE	Open COBOL
Neartek	No	Yes	Yes
Ordina Denkart	No	Yes	Yes
Sunguard Bi-Tech	No	No	Yes
Transoft	Yes	Yes	Yes

Migration Techniques

Each of the migration vendors follows similar migration techniques for migrating COBOL code. These techniques are described below:

Programming Language

- The source code is moved to a new environment using an open system COBOL compiler.
- Alternatively, the source code is moved to a new environment in a new language. The new source code greatly resembles the original COBOL source code, even if the new language is fundamentally different, such as an object-oriented (OO) language.

MPE Intrinsics

- A library is used in the new environment to intercept COBOL MPE Intrinsic calls. These libraries use the operating system or a low level programming language (such as C) to handle the functionality of the MPE Intrinsic call. This requires the migration vendor's run-time library to reside on your system as long as the MPE Intrinsic calls remain in your source code.
- Alternatively, the MPE Intrinsic is translated into new environment code directly in the source code itself, therefore not needing a resident run-time library.

Database

- The tools create database tables in the new environment based on the schema of the Image database. They create indexes in the new tables based on Master and Automatic-Master tables in the Image database.
- Array fields are flattened into the tables as individual field names. For example, an array element containing up to five values is set up as five fields in the new table (field_1, field_2, field_3, field_4, and field_5).
- The tools then move the data from the Image database to the new database given the very similar structure.
- Other database migration tools allow you to build data maps between the Image database and the new environment database. The data maps then serves to move data from one database to another, either real-time or one time. Bridgeware, by Taurus Software, is an example of a tool that does this with Image and new environment databases.
- COBOL database access statements are handled with run-time library calls, much like the MPE Intrinsics above.
- Alternatively, COBOL database access statements are replaced directly in the source code with SQL statements.
- KSAM files are moved to equivalent files provided by the COBOL compiler on the new platform. One example of these is C-Isam files, provided by Microfocus COBOL.
- Flat files are moved to flat files in the new platform.
- Some tools will migrate KSAM and flat files into database tables.

User Interfaces

- Vplus calls are handled with run-time library calls which will allow the new screens to function in many different environments, such as character-based, graphics user interface (GUI) or web client.
- Many companies are choosing to re-engineer their user interfaces, while migrating the rest of the code. This allows them to move toward an open client-server environment, without re-writing the whole application, yet significantly upgrading the end user's view of the system.
- Direct I/O calls in COBOL programs port as is to open COBOL.

JCL

- The tools or service provider translates the JCL into the new environment's scripting language.
- Alternatively, the migration vendor will provide a JCL emulator on the new platform.

Migration Issues

As you port your COBOL application, there are risk areas to watch and mitigate against in the areas of MPE Intrinsics, Programming Languages, Performance, and GUI's:

MPE Intrinsics

Not all MPE Intrinsics used in your application may be covered by the tools used. The migration vendor will either take the opportunity to add the new Intrinsic to their toolset, or migrate these calls manually.

Programming Languages

In source code that we have reviewed, we have found SPL, Assembler, C, and 4gls mixed in with the source code. Some of these languages do not migrate automatically to new platforms and require manual re-writes. Many 4gls have migration paths to new platforms, such as Cognos and Speedware, but some do not.

Performance

Some applications pull data from several tables all at once as part of the component's functionality. When using Image, the application tends to retrieve data from tables one at a time, then put them together in the source code. When these database statements are migrated, they tend to want to continue to retrieve data from the new database one table at a time. If the new database is a relational database management system (RDBMS), some data retrieval performance can be lost. In native RDBMS applications, the code usually allows the RDBMS (its optimizer) to choose the order for accessing tables, because the optimizer knows how it can join the tables the fastest. Migrated applications, without some additional work, may not allow the RDBMS to make this decision. This can affect the performance of multi-table joins. This may be important if the datasets are large, such as 100,000s or millions of records.

GUIs

When character based screens are translated into GUI screens, real estate is often lost from the screen. If the original character based screen was packed with data, the screen will have to be re-organized in the new environment. Also, user impact may be greater

converting from character-based to GUI or web client, since keyboard entry may be replaced by mouse-driven functions.

COBOL Development Frameworks

There are three major COBOL compilers available in the new environments. They are Microfocus COBOL, ACUCOBOL, and Fujitsu's NetCOBOL.

Microfocus COBOL

Microfocus COBOL is the standard bundled COBOL compiler on the HP9000. It offers the ability to integrate COBOL into open architectures, by providing tools for integrating COBOL applications with the web, and new technology application components. They also provide access to SQL, and C-ISAM files, their equivalent to MPE KSAM files. In addition, it provides portability to a wide range of operating environments and hardware platforms, high performance native code that can be optimized for peak performance, advanced debugging and analysis tools, access to relational database management systems, and it enables communication between client server applications running on different machines.

ACUCOBOL

Like Microfocus, ACUCOBOL offers the ability to integrate COBOL into open architectures, by providing tools for integrating COBOL applications with the web, and new technology application components. They also provide access to SQL, and VISION Isam files, their equivalent to MPE KSAM files.

Other advantages of ACUCOBOL are their internal GUI development tools that use COBOL for screen development. Their tools provide a graphical GUI development environment on Windows that generates code that can be directly moved to Unix or other platforms. ACUCOBOL has built the handling of MPE Intrinsics into their compiler and has partnered with a Vplus migration vendor, Screenjet, to provide a means for migrating Vplus forms to the ACUCOBOL GUI environment. ACUCOBOL has also partnered with Transoft for HP e3000 COBOL migration services.

Fujitsu Sweet3000

Sweet3000 provides a migration path for companies that wish to migrate COBOL applications to either the Windows or Unix operating systems, and provides extensive integration for the .net framework, with Fujitsu's NetCOBOL for .net. This is a highly recommended tool for companies that have selected the .net framework as the foundation of their target environment.

Sweet3000 is designed to migrate HP3000 COBOL applications, including HP-VPLUS screen IO, HP IMAGE databases, COBOL data files, the most common HP intrinsics and HP3000 batch jobs. This process occurs with minimal changes to the application source code. The migrated application will be on the NetCOBOL platform. Very little of the conversion work takes place on the HP3000, with most of the work being done in the target environment.

Once you have identified your new environment, you can use the migration techniques and compatibility matrices above to work with migration vendors to lay out the migration approach that best meets your goals.

Conclusion

Your enterprise application suite is a valuable investment for your company. You have invested years, perhaps decades, of development time into them. Your users are familiar with them and consider the functionality an integral part of their job. The applications reflect your business rules and policies, and specifically function to make your business more efficient.

Now is the time to consider their future carefully. Using the migration process provided in this paper, you can choose whether it is best to 'Stay, Port, Build, or Buy'. If you decide to port the application, the next critical decision is to choose your new IT environment.

Once you know that you are moving the application to a specified environment, you can evaluate the migration vendors to help you get there and choose the best process to retain the value in your investment. As you undertake the migration, you can monitor and mitigate the technical risks that could occur during your migration effort.

At the conclusion of your migration effort, you will have set your migration goal with careful consideration, and followed the most efficient path to reaching that goal.