Considerations for an OpenVMS Application Migration and Modernisation Strategy

Moving OpenVMS applications to Linux, Windows or Unix

OpenVMS: time to plan for the future

OpenVMS, originally called VMS (Virtual Memory System), was first released in support of the VAX-11/780 in 1977. During the 1980s and early 1990s a VAX/Alpha minicomputer running VMS was quite popular since the operating system was well suited to high speed, real-time applications. Even now, OpenVMS continues to drive numerous mission-critical business and operational systems. These reliable, high availability systems have proven their worth to numerous organisations, including nuclear power plants and financial services firms. Quite often, these organisations have invested money and decades of time to customise their OpenVMS applications to meet very specific needs.

In spite of OpenVMS's attributes, more and more IT executives are realising the limitations of their legacy applications in the context of their digital transformation strategies. Legacy languages and hardware are hard to support, maintaining them incurs ever-increasing costs, and the software often doesn't integrate well with modern IT systems. It's not surprising that today nearly half of IT executives view application modernisation as one of their top five priorities.

These issues serve as a prompt for IT leaders to closely examine the functionality of their OpenVMS systems and to seek the right modernisation strategy for their business. This paper looks at the available options for migration and the benefits that might be achieved through modernisation.

Factors driving decisions

There are five key factors to consider when assessing options for an OpenVMS application migration and modernisation strategy. How each application ranks against these factors will help determine the direction to be taken and the target end state.

Application Criticality

Perhaps the most important factor is how mission-critical the application is to the business. For example, does the application provide custom or unique business functionality that cannot easily be replaced? The answer to this question will help guide a decision on whether the application should be retained, replaced, rewritten or even retired. If the OpenVMS application has been customised to meet the needs of the business and is mission-critical, a migration and modernisation initiative may be the best strategy.

OpenVMS Application Migration & Modernisation

OpenVMS: time to plan for the future

OpenVMS, originally called VMS (Virtual Memory System), was first released in support of the VAX-11/780 in 1977. During the 1980s and early 1990s a VAX/Alpha minicomputer running VMS was quite popular since the operating system was well suited to high speed, real-time applications. Even now, OpenVMS continues to drive numerous mission-critical business and operational systems. These reliable, high availability systems have proven their worth to numerous organisations, including nuclear power plants and financial services firms. Quite often, these organisations have invested money and decades of time to customise their OpenVMS applications to meet very specific needs.

In spite of OpenVMS's attributes, more and more IT executives are realising the limitations of their legacy applications in the context of their digital transformation strategies. Legacy languages and hardware are hard to support, maintaining them incurs ever-increasing costs, and the software often doesn't integrate well with modern IT systems. It's not surprising that today nearly half of IT executives view application modernisation as one of their top five priorities.

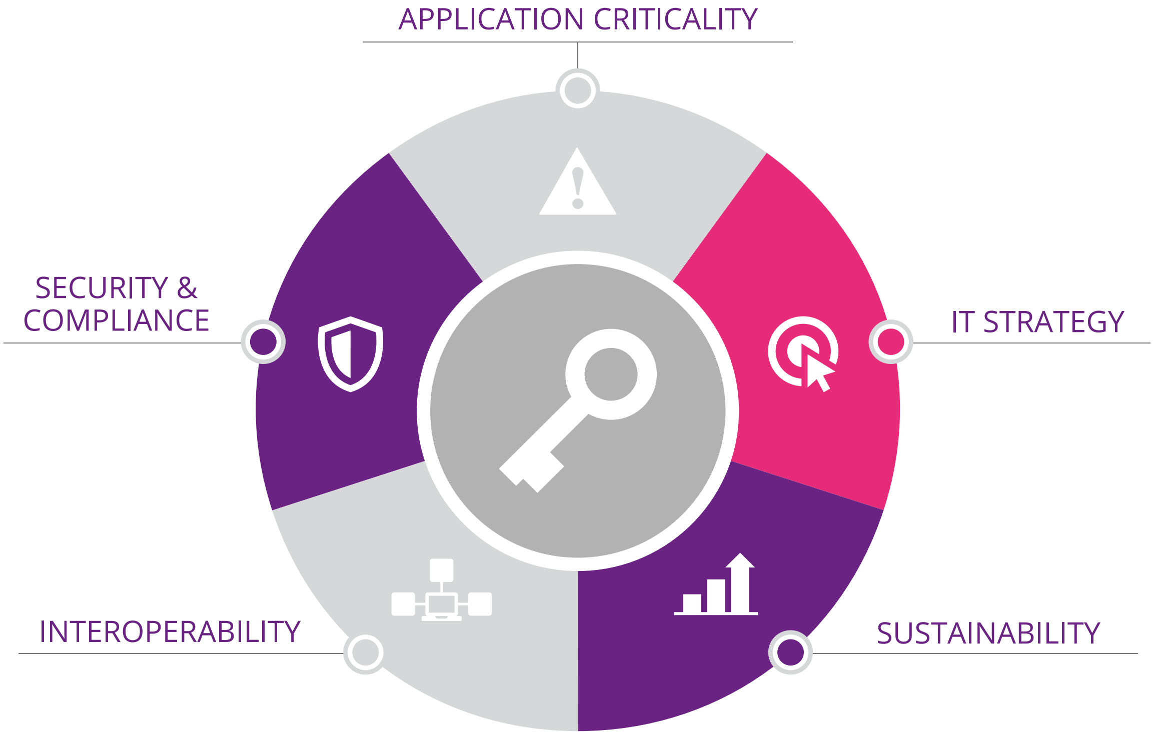
These issues serve as a prompt for IT leaders to closely examine the functionality of their OpenVMS systems and to seek the right modernisation strategy for their business. This paper looks at the available options for migration and the benefits that might be achieved through modernisation.

Factors driving decisions

There are five key factors to consider when assessing options for an OpenVMS application migration and modernisation strategy. How each application ranks against these factors will help determine the direction to be taken and the target end state.

Application Criticality

Perhaps the most important factor is how mission-critical the application is to the business. For example, does the application provide custom or unique business functionality that cannot easily be replaced? The answer to this question will help guide a decision on whether the application should be retained, replaced, rewritten or even retired. If the OpenVMS application has been customised to meet the needs of the business and is mission-critical, a migration and modernisation initiative may be the best strategy.



OpenVMS Application Migration & Modernisation

Sustainability

There are fewer experienced OpenVMS resources available in the marketplace than ever before. This not only creates a potential risk, but impacts decisions about what end state should be targeted in any migration and modernisation initiative. It is important to analyse the existing OpenVMS skills that are available to support and operate not only the current environment, but also to what extent these resources will be accessible in the future. In addition to the human resources, the technical aspects of new releases must be considered.

Interoperability

The extent to which an OpenVMS application supports external interfaces and adheres to the standards within an organisation for interoperability architectures should be considered when looking at the desired end state for a migration or modernisation project.

In many cases, OpenVMS applications and data have become "islands" in an organisation's IT landscape. Certainly, integration of external applications with OpenVMS applications is feasible; however, these mechanisms are becoming exceptions in an enterprise integration strategy. Building integration points and maintaining them can become expensive and prevent organisations from achieving a standard architecture for application interoperability.

Security and Compliance

Security and compliance requirements represent another potential risk point in current OpenVMS applications. As organisations institute company-wide security standards and IT governance controls, OpenVMS applications often fall into a security and governance "silo." While usually secure within their own right, OpenVMS applications lack the ability to integrate into a broader security and control framework within a modern organisation.

IT Strategy

Lastly, OpenVMS can no longer be considered a strategic IT platform for an organisation. Companies are seeking to standardise their data centers around virtualised hardware and software stacks that can be deployed rapidly at very low cost ("computing pods"), that are capable of being scaled up or down as IT capacity requirements change. OpenVMS applications do not lend themselves to this strategy. As organisations begin to deploy private and/or public cloud architectures, the growing gulf between OpenVMS and this future IT direction becomes a liability.

Examining these five factors helps an IT organisation assess the relevance and reliance associated with OpenVMS applications. The good news is these factors can be addressed, and risks mitigated, with a well-designed migration and modernisation plan.

Re-host, re-architect or both?

There are many solutions which can be employed when moving OpenVMS applications and data to a new platform. These solutions range from "re-hosting" to a more invasive "re-architecting" of the applications. Between these two extremes it is often beneficial to consider a hybrid approach: re-hosting portions of the application that can be easily moved and re-architecting those components where a new end state technology or functionality is desired.

Re-hosting

Re-hosting (sometimes referred to as "lift and shift") describes a migration where minimal changes are made to the underlying technology of the application and database. The benefit of re-hosting is that it minimises changes to the application and ensures the preservation of functionality, business logic and business processes after the migration. Because of this, the testing required to verify the migration is minimised, and the cost/time for end-user re-training is eliminated.

An example of re-hosting would be recompiling an application written in Fortran (which might use DCL for the batch processes and a Record Management Services (RMS) file system) into a new Linux or Windows environment using an Open Systems DCL shell to support the DCL command files and RMS data within an Open Systems indexed file system.

The viability of re-hosting depends not only upon having access to a native language compiler in the new environment, but also a framework that supports the proprietary aspects of the OpenVMS world. In the above example, support for DCL and RMS in the Windows or Linux target platform would be required.

Fortunately these Compatibility Frameworks are readily available in the market. They introduce a software license component in the new environment, but the reduction in overall migration cost more than pays for the licensing fees.

In addition to the Compatibility Framework accelerating the speed of a migration, most re-hosting solutions include tools to automate the adaptation of code and data often required for use with the target compiler or operating system. Continuing with the above example, in moving VMS Fortran to an Open Systems Fortran, there will likely be some differences in the syntax in use compared to the syntax supported by the new compiler. The proper re-hosting solution provides automated tools to adapt the code so that it will work with the new compiler. This speeds up the process and delivers predictable and consistent results compared to trying to perform this work manually. In addition to the speed of migration, a proven migration toolset decreases the cost and risk by reducing the number of human resources and amount of testing required.

Therefore re-hosting typically provides a faster, lower cost and less risky option for OpenVMS application and data migrations. Rehosting is not moving to an emulated hardware solution. This approach typically uses a software/hardware emulation of VAX or Alpha sitting on top of either bare metal or Windows or Linux. In all cases it still uses OpenVMS as the operating system and therefore does not address the inherent reasons to move away from OpenVMS.

Re-architecting

Re-hosting is not always the best solution. In some instances the OpenVMS technologies simply cannot be supported in the target environment. Or, it may be desirable to change an underlying technology to a more modern end state, or to add/change functionality.

There are varying levels of automation tools depending upon the re-architecting required. For example, when using DECforms, there is not a re-hosting, no-change option for moving the user interface to Windows or Linux – so the user interface will need to be re-architected. If the goal is to preserve the basic look and feel of DECforms, there are tools that will migrate the forms and associated escape routines in a relatively automated manner. However, if the goal is to convert DECforms to a browser or graphical client, this will require some amount of manual re-engineering.

A common re-architecting requirement is to transform the programming language in the OpenVMS application to a more modern code base. An example would be transforming Pascal to C, or Fortran to C#. There are commercial tools available to automate this process. However, it should be noted that when moving from a procedural language (e.g. Fortran or COBOL) there are varying degrees of how much the conversion is able to achieve a true Object Oriented (OO) end state. Typically, the more a “pure” OO code base is desired, the less automated the conversion will be – increasing testing time and project costs.

Hybrid solutions

Given the status of most OpenVMS applications, many organisations adopt a hybrid approach: re-hosting components that can be easily supported in the new environment, and re-architecting aspects for which a re-hosting solution is not readily available, or where functional/technology modernisation delivers added benefits to the business.

Breaking down the problem

OS layer

Linux or Windows?

Migrating OpenVMS applications can be seen as an opportunity to move to the standard infrastructure platform of the organisation. The options to move to Unix, Linux or Windows are all available and technically viable. The choice of which platform best suits the organisation may be driven by your available internal skill sets and software standards, such as your preferred development environment and target database, and, of course, cost.

If applications are being re-hosted using a Compatibility Framework, then re-training of OpenVMS programmers and operators can be minimised by choosing to preserve some of the technical interfaces used in the OpenVMS environment.

However, re-hosting should always provide a native implementation in the target environment in order to benefit from the economic and governance advantages of system standardisation.

Data layer

Migrating OpenVMS data provides an opportunity to move to the organisation’s standard database technology. The desire may be to move the OpenVMS proprietary RMS data to a Relational Database Management System (RDBMS). However, Compatibility Frameworks offer Open Systems native implementations of RMS, whether indexed or sequential files. Using a native indexed filing system to replace RMS might be an attractive alternative that reduces costs and minimises the need to add performance overhead when compared to moving the data to a RDBMS.

RMS

RMS is probably the most common file management system used on OpenVMS. It is an integral part of OpenVMS system software and its procedures run in executive mode. RMS supports the following four types of record level access:

- Sequential Access
- Relative Record Number Access
- Record File Address Access
- Indexed Access

And, the following record types:

- Fixed length
- Variable length
- Variable record length with fixed length control blocks

Stream files (records separated by termination characters)

- STREAM: Records terminated by CRLF
- STREAM_CR: Records terminated by CR
- STREAM_LF: Records terminated by LF

If the goal is to migrate RMS files along with the application programs that use them, there are really only two viable migration options:

- Move the RMS data to a Compatibility Framework that provides equivalent RMS file handler functionality in the target operating environment.
- Re-architect the data into a native relational database such as Oracle or Microsoft SQL Server.

The benefit of the first approach is that it maintains the RMS I/O statements in the migrated application, so there is no need to re-engineer the I/O statements and the program logic will remain unchanged.

A potential shortcoming is that the data may be difficult to access from other applications and industry standard query tools. However, there are now ODBC and JDBC gateway products available for migrated RMS data that provide relational database type querying and integration solutions.

The benefit of the second approach can be realised when integrating the migrated application and data with other applications and moving all computing to a common database standard. Re-architecting RMS data to a RDBMS is entirely possible. The key issue is minimising the amount of application code re-engineering required to access the relational database (as opposed to the indexed, record level RMS files). A comprehensive migration vendor toolset should provide mechanisms for migrating RMS to a RDBMS and managing the impact this creates on the application code.

Depending on the application's programming language, vendor solutions may generate intermediate I/O libraries that provide static SQL calls and map data I/O back to a call level interface, replacing the RMS I/O statements in the programs. Other options are more like "black box" solutions, providing optimised dynamic SQL I/O and again mapping to calls replacing the RMS statements in the program. Lastly, programs can be re-architected to use embedded SQL. This can be a large undertaking, but will provide a "native" implementation which may be more sustainable in the future. The embedded SQL option will also depend on the availability of SQL pre-compiler support for the application's programming language.

If the desire is to migrate RMS data to another application's database for archive purposes only, and the impact on the associated application's code is not a concern, there are data movement/ modernisation tools available to extract and load the RMS data into another database structure. These tools may need access to application code or copy books in order to help determine the RMS data definitions for the extract.

Oracle Rdb

Oracle Rdb is a RDBMS specifically for OpenVMS. Rdb was originally created by Digital Equipment

Company (DEC) in 1984 and was intended to be used for data storage and retrieval by high-level languages. In 1994, DEC sold the Rdb division to Oracle Corporation where it was rebranded Oracle Rdb. It currently runs on OpenVMS for VAX, Alpha and HP Integrity Servers.

While there is no corresponding version of Oracle Rdb on Unix, Linux or Windows, the obvious target end state for an Rdb migration is another relational database (e.g. Oracle or Microsoft SQL Server). While there are some changes needed to schema definitions, the conversion and migration are relatively straightforward given the right vendor tools.

The majority of the Rdb SQL syntax is fully supported by other RDBMS without change. However, there are a small number of changes to the SQL, including stored procedures, that will be required in order to get it to execute on the target RDBMS. Automated tools are valuable when making these changes, in order to save time/ effort and ensure accuracy.

There are two ways to combine SQL with OpenVMS host language programs. You can create a separate module for SQL language statements (SQLMODs) or you can enter them directly in the host language program and use a SQL precompiler. Changes will be required to these interface methods in the associated program code to use APIs available on the target platform. Rdb access commands can be different from standard ANSI-embedded SQL statements and the existing code structures will require adaptation in order to compile and function with the new RDBMS. Vendor migration tools should help with this process.

Other databases and data sources

In addition to RMS and Rdb, there are other databases in use on the OpenVMS operating system (e.g. Oracle and Ingres). There are also proprietary legacy database systems (e.g. CODASYL DBMS and Adabas). The migration options for these less common databases depend very much on whether they are supported in the target operating system environment. The process for migrating any proprietary, hierarchical, or network structured databases to a RDBMS is similar – but usually more complex – than converting RMS to a relational structure.

Obviously, if Oracle on OpenVMS is being used, then migrating to Oracle on a new target platform will probably be the easiest migration option. This should not limit thinking, however; there is no reason why a migration to a different target relational database that is currently being employed in an organisation cannot be accomplished.

Automated tools for such database conversions may be limited, but an experienced vendor should be able to understand the required conversion and may be able to develop automated tools.

Application program layer

The overall OpenVMS modernisation/migration project – whether it involves re-hosting, re-architecting, or a hybrid combination – is primarily driven by the requirements of the application program layer. In particular, the driver is what programming language the applications are currently written in and what programming language is desired as the end state for the future. There is a wide variety of mostly 3GL languages supported on OpenVMS, the most common being:

HP COBOL

HP COBOL is a common programming language found on OpenVMS for business applications and is one of the easiest to re-host to Unix, Linux or Windows. There is a range of Open Systems COBOL compilers that provide a “landing site” for applications written in HP COBOL. However, moving HP COBOL to an Open Systems COBOL involves more than simple recompiling, as there will be differences in supported syntax and less tolerance for non-compliant syntax.

In addition, OpenVMS COBOL code will be dependent on facilities of the OpenVMS environment. Aspects such as file and screen I/O will be alien to a new compiler and any OpenVMS system service calls will not be natively supported. These issues can be resolved with a Compatibility Framework that supports these functions in the new operating system and avoids changes to the application program code. Alternatively, these intrinsic functions can be re-engineered to functions supported by the target COBOL compiler. However, this re-engineering will take longer, adding to the cost, and be more risky.

OpenVMS Application Migration & Modernisation

Whichever approach is decided upon, vendor tools can provide a high level of automation for the code changes required for the new COBOL compiler.

HP Fortran

HP Fortran is used in a significant number of manufacturing, process control and banking applications running on OpenVMS. There are Fortran compilers available on Unix, Linux and Windows, so re-hosting to a Fortran end state is feasible. The migration process and requirements are similar to migrating COBOL.

As with COBOL, adaptation of the code may be required to comply with the new compiler. A Compatibility Framework will avoid the need to re-engineer programs for file and screen I/O, as well as other OpenVMS intrinsics.

One option for migrating a Fortran code base is to convert it to a programming language that offers similar capabilities. Most Fortran conversions target C or C++ as the end state. This provides the benefit of re-architecting the application into a form that can be supported by more commonly available resources. These projects typically require extensive regression testing when compared to a re-hosting to the same programming language using an Open Systems compiler.

Whether re-hosting to a Fortran end state or migrating to a new language, vendor tools can provide a high level of automation for any code changes required.

HP Pascal

Unlike COBOL or Fortran, there is not the same support for Pascal in modern operating systems, so applications written in Pascal almost always require transformation to another programming language. Conversions to C, C++, Microsoft C#, and Java are all feasible. However, depending on the nature of your code base and the application functionality required, certain of these end state options make more technical sense than others.

Transformation Tools

3rd Party: Scheduler, Utilities	REVIEW / REPLACE	Re-host or Re-architect
Rdb, Oracle, Codasyl	DATA MIGRATION	Relational Database Mangement System
RMS, Indexed Files		Relational Database Management System or Indexed Files
COBOL, Basic, Fortran, Pascal, C	APPLICATION LOGIC	Re-host or Re-architect
DCL/System Calls	OPERATION ENVIRONMENT	Rewrite or DCL Shell/ System Call API
SMG, FMS, DECforms, Oracle Forms, Equipment	INTERFACES	Re-host or Re-architect
OpenVMS	PLATFORM	Linux/Unix/Windows

Advanced Compatibility Framework

Pascal provides better program structure controls than many procedural coding environments, but Pascal applications are not truly Object Oriented (OO). Converting procedural code to OO languages usually requires a trade-off between how pure an OO end state is desired versus the speed and cost of conversion. In some cases, it is just not realistic to use automated conversion and code generation tools to take a procedural 3GL code base to a "true" OO end state. However, if there is a willingness to compromise, automated conversion of Pascal to a modern programming language, particularly to C or C++, is quite feasible.

HP Basic (Basic Plus, Basic Plus 2)

HP Basic is a reasonably common language in the OpenVMS community. It is used in commercial, manufacturing and banking applications. Although there are Open Systems BASIC compilers and interpreters, they do not offer a high degree of syntax compatibility with HP Basic. HP Basic applications are therefore candidates for transformation to a more modern programming language. Specific tools are available to convert Basic to C++, Microsoft C# and Java. There are also general tools which can be tailored to convert HP Basic to other target programming languages.

Other OpenVMS programming languages

In additional to the languages listed above, OpenVMS supports other programming languages, such as Ada. In general, the same key factors discussed above will apply when assessing the migration options for such languages.

The first consideration is whether there are suitable compilers for the current language that, with the help of a Compatibility Framework, create a pathway to migrate with minimal changes to the code. The second consideration is how much ongoing maintenance and development work will be required for the application in the future. If a considerable amount of development is anticipated to keep the application current with business requirements, then transforming the application code to a modern development environment is probably the best approach.

The more the application code is changed during the migration process, the more regression testing will be required before "going live" and increased risk will be introduced into the migration project. This applies even if automation tools are being utilised during the re-hosting or re-architecting process.

Operation environment

DCL

Digital Command Language (DCL) is used to control processes in the OpenVMS environment. The migration of DCL command files is similar to migrating application program code. There are two choices available for the migration of DCL:

- Keep the DCL virtually as is and run it in an Open Systems DCL "shell" provided by a Compatibility Framework.
- Transform the DCL to a native Linux or Windows scripting language.

In practice, most users choose to keep DCL and run it in a vendor-provided shell on the target system. This has the merits of speed, low cost and low risk – avoiding the need for significant regression testing and re-training associated with transforming the code to native target script. Good DCL shells are capable of switching to native scripting languages as existing DCL modules are enhanced or modified. This allows the user to move to a native environment over time, rather than in one major transformation. DCL shells are typically more concise than native scripting solutions, while transforming DCL to native script will invariably result in generating multiple lines of script code for each line of DCL code.

User Interface (UI) layer

How the migration and modernisation of an OpenVMS application's UI is handled depends on the mechanism currently in use for developing and maintaining the screens and the tolerance for cost/risk involved in a migration. Generally, the options are:

- Maintain the existing UI through the use of a Compatibility Framework.
- Transform the UI to an alternative technology using automated tools.
- Re-architect the UI into a modern look and feel using modern technologies.

FMS

HP FMS (Forms Management System) is a common UI development and management environment found on OpenVMS. Usually running on "green screen" VT100 terminals or emulators, FMS provides an asynchronous character-based interface for online OpenVMS applications.

OpenVMS Application Migration & Modernisation

As well as providing an API to develop forms for a UI, FMS allows programmers to embed subroutines into forms for field validation and processing purposes. Compatibility Frameworks can provide a means of preserving the FMS API in a Unix, Linux or Windows environment. This avoids re-engineering the UI components of an application, reducing the time and cost of a migration and the need to retrain users after the application is migrated.

SMG

SMG is an OpenVMS screen management library. Like FMS it provides support for a character based VT user interface. SMG is similar to the ncurses (new curses) library supported under flavors of UNIX. SMG provides the means to manage screen I/O independent of terminal hardware. However, unlike FMS, SMG doesn't use a Form definition for the screen I/O and doesn't provide a means of embedding logic into the screen layout.

Most OpenVMS migration frameworks on Unix offer an SMG emulation. Providing the same green screen character-based interface. Migrating SMG screens to Windows will require the use of a VT100 terminal emulator and terminal server capability to reproduce the green screen look and feel.

Because SMG uses inline commands in the application programs and doesn't offer a layer of abstraction from the application programs, it is much harder to modernise SMG screens to a native graphical user interface (GUI). Any attempt to do this will inevitably require significant restructuring and change to the application code. However, some degree of a GUI can be provided for SMG screens using screen scraping solutions.

DECforms

HP DECforms is a software product for the development and deployment of a forms-based UI for interactive applications running on OpenVMS. Not only does DECforms give the look and feel of a forms interface, it also supplies a robust set of dialog management and validation functions to control the UI at application runtime.

IFDL (Independent Form Description Language) is used to define the DECforms used by an application. Most migrations transform IFDL code into an alternative language with its own UI capabilities. For example, one common approach is to transform the IFDL into COBOL routines that provide a similar UI. This is a sensible approach if the core application code is written in COBOL. Alternatives exist to convert IFDL into other 3GL languages such as Fortran.

Alternatively, DECforms can be transformed into a modern UI such as ASP, JSP or HTML. However, the existing API may not always be preserved and changes to the application code may be required. This type of transformation will almost always require some retraining of users.

ACMS

ACMS is a transaction processing (TP) monitor that runs on OpenVMS. It is intended for businesses that require high performance, security, data integrity and both centralised and distributed processing. Providing similar TP capabilities for migrated applications can be an important issue for organisations using this product. While ACMS provides several mechanisms for external Windows and Unix applications to communicate back to it , there is no ACMS product that runs on Open Systems. Migrating an ACMS application requires replacement of its TP functionality with another engine. Oracle's Tuxedo has been commonly used to provide a TP environment for migrated ACMS applications.

Another route is to re-architect the DECforms interface and re-engineer the TP to a client-server or web architecture. This effectively removes the need for a pure TP function and replaces it with the combination of application and database functionality, or an application server to ensure transaction integrity.

Oracle Forms

Oracle Forms represents the other significant UI technology found in OpenVMS. Because it is also found in other legacy application environments, there are several vendors who specialise in providing migration solutions for Oracle Forms applications. These solutions, in conjunction with specific OpenVMS application migration offerings from specialised vendors, can provide a complete migration solution for OpenVMS Oracle Forms applications.

Third party products

When considering a migration, it is important to consider any third party products used in conjunction with applications on OpenVMS. Third party products can be divided into two groups: 1) those that interact with applications at run-time (e.g. Attunity data access solutions) and 2) those that provide operator and programmer utilities (e.g. HP's Datatrieve). The reason for making this distinction is that it is possible to employ different strategies based upon which group each product falls into.

As a starting point, an inventory of all third party products should be created – categorising them according to how they are used, and then mapping them to target solutions that meet the organisation's functional needs in the new environment.

If a third party product is providing run-time functionality to an application (e.g. a sort utility), then the impact of changing the API when moving to an Open Systems replacement product should be considered. The simplest solution is to try to map to an Open Systems version of the same product whenever this is an option.

Using automated tools to adapt the code components to work with a new third party product will reduce risk and cost compared to making these changes manually.

The need for third party utilities that have been frequently used by programmers or operators may disappear due to capabilities found in the new development or operating system environment.

Experienced OpenVMS migration consultants have already built most of the "from-to" mapping for the more commonly found third party products. Partnering with such a vendor at the planning stages in a project can save time and effort.

OpenVMS clustering

OpenVMS was one of the first operating systems to support clustered environments. Originally called VAXcluster, and now known as VMScluster, the capability was introduced in 1984. Clustering was initially used to expand the scalability and load balancing of an application. As hardware performance improved, clustering today is used more to provide high availability and rollover capabilities.

Many organisations still use VMScluster today. This does not prevent migration to another platform and operating system. There are clustering solutions for Linux, Unix and Windows environments (e.g. Red Hat Cluster Suite, VMWare and Veritas Cluster Server). With improvements in computing power and virtualisation of servers, clustering for performance and load balancing purposes is less of a requirement, and clustering in modern systems is mostly designed to improve application availability. Products such as HP Serviceguard, Sun Cluster and Microsoft Cluster Server are good examples of high availability clustering solutions.

Recognising why clustering is used with OpenVMS applications helps to determine if it will be required in the migration target end state. For example, if clustering is being done for performance purposes, it may make sense to design a non-clustered, virtualised server environment for the target end state. However, if high availability is a critical requirement, then an add-on cluster support product may be ideal for use with the target operating system.

Other interfaces

Another important consideration is the external interfaces supported in the OpenVMS environment. There may be TCP/IP connections into the system, or external file transfer tasks providing external data that are processed by the OpenVMS application. Similar interfaces to the application must be planned for in the target environment. Where possible, the new interfaces should avoid the need to modify the external applications in order to work with the migrated application.

Significant overhead can be added if customers or business partners also need to revise their applications due to changing interfaces after a migration.

Mitigating risk

So far, this paper has focused on the specifics of migrating each component making up an OpenVMS application set: the core of the migration project. However, there are other major factors in a migration – discovery, planning and testing – which are critical to success.

Discovery

Discovery is a critical step in any successful migration in that it determines the composition, size and complexity of the current OpenVMS environment. Making an inventory of application components creates a template upon which migration solutions can be applied to each piece. Knowing the size of each component provides metrics that are vital to accurately estimating the scope and cost of the migration project. In addition to knowing the quantities, understanding the interactions and complexities also provides valuable scoping information.

Application Understanding (AU) tools are available to automate a large portion of this phase of the project. The best migration solution vendors offer these AU tools as part of their solution set.

Once the application component inventory is in place, work can begin on analysing the source code to discover any potential migration issues. AU discovery tools frequently automate this process as well.

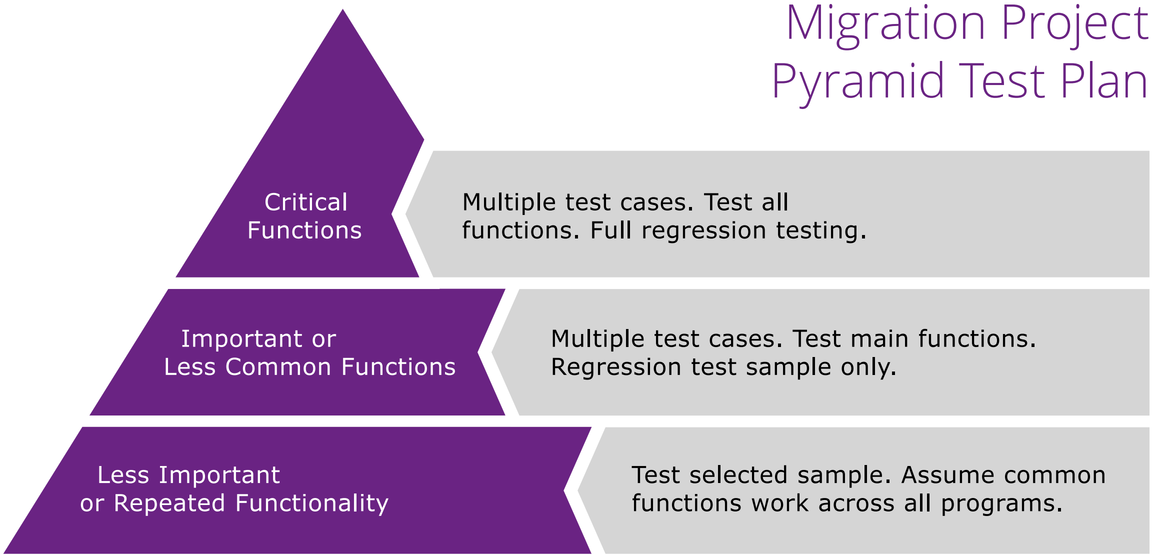
Planning

With a complete picture of the application set that will be migrated, the next step is to plan the migration process. The key in this phase is to ensure that the migration addresses every aspect of the application. Planning the migration completely is the single most important aspect for risk mitigation.

Testing

Testing is also critical. Most migrations require some modification to code and data structures. In re-architecting projects, entire code bases can be transformed into new languages. Therefore, every migration should include the classic testing

Migration Project Pyramid Test Plan



Critical Functions

Multiple test cases. Test all functions. Full regression testing.

Important or Less Common Functions

Multiple test cases. Test main functions. Regression test sample only.

Less Important or Repeated Functionality

Test selected sample. Assume common functions work across all programs.

OpenVMS Application Migration & Modernisation

methods: regression testing, systems testing, user acceptance testing and, in some cases, parallel testing.

The application migration testing pyramid approach

A testing plan should be created based on the strategy being adopted. If using a Compatibility Framework and automated migration tools, which preserve much of the logic flow of the application, it should be possible to create a "pyramid testing" plan. This recognises that the migration is not the same as a pure development project.

Thanks to the predictable outcome of automated tools, pyramid testing validates the migration process rather than testing each program that is migrated (i.e. if the migration works for a representative subset of programs, it can be assumed that it works for all programs). Pyramid testing still requires a reasonably comprehensive approach, but prioritises groups into different levels based on program criticality.

Delivering benefits

OpenVMS migration and modernisation projects have great potential to deliver benefits to a business while future-proofing applications – avoiding the need to undertake migrations again at a later date. In general, the benefits can be seen as:

Lowering the Total Cost of Ownership (TCO)

While it may not be obvious on the surface, the reality is the TCO of OpenVMS applications is 50-70% higher than modern application platforms. Labor costs are typically higher as skilled resources become increasingly scarce. The cost of application maintenance and development is higher when the costs of building interoperability with other systems and the lack of economies of scale compared to modern platforms are factored in. With higher costs and higher risks, migrating and modernising OpenVMS applications should dramatically lower the TCO associated with their functionality.

Avoiding catastrophes

Most, if not all, IT organisations have special disaster recovery provisions for their OpenVMS environment. As hardware platforms cease to be supported, surviving catastrophic hardware failures will become increasingly more difficult and expensive to address. After the stockpile of spare parts for Alpha and Integrity platforms has dwindled, the next step may be shopping for parts on eBay or similar sources. Clearly, this is an unacceptable level of operational risk for most organisations.

Sustainability for the future

The outcome of a well-planned and executed migration and modernisation initiative is to move OpenVMS applications to a maintainable, low risk, and low cost end state that is viable for the future. A well executed plan may have multiple steps that meet short-term migration needs but also deliver long-term benefits.

OpenVMS Application Migration & Modernisation

Application modernisation is a "team sport"

Most IT organisations have limited experience with complex legacy application migration and modernisations. Few organisations have spare resources with specialised skills sitting around who can dedicate themselves to such a project and still keep up with day-to-day responsibilities. Rarely are the specialised tools and methodologies to deliver a highly automated migration available in-house. For these reasons, most organisations seek specialist vendors to assist – and sometimes to take full responsibility – for delivering a successful project.

It would be simplistic to think that the vendor is only providing software tools and technology. Migration and modernisation projects are complex. A successful vendor must demonstrate that is has a proven and repeatable methodology – in addition to the right skills, sufficient resources and specialised technologies to ensure success.

In Summary

The goal of this paper was to look at the available options for migrating and modernising applications currently running in the OpenVMS environment – and to highlight the benefits that can be achieved through modernisation. Factors in deciding upon the right course of action were discussed, and certain details regarding the migration of specific components were highlighted.

Just as no two companies are identical, the options, challenges, and requirements of undertaking a migration and modernisation project demand a customised solution that leverages an organisation's existing resources while optimising both technical and business results.

If there is only one key takeaway from this discussion, it should be that migrations are complex. They are most successful in their deployment when a methodical, fact-based approach is utilised to assess the functional value and technical complexity of the current OpenVMS environment to arrive at the right strategic direction for the organisation. Specialised tools are available to automate large portions of this assessment and experienced migration vendors exist to assist in planning and executing the project. The result will be a project that avoids major operational risks and delivers business benefits with a considerable return on investment.

More information

w oneadvanced.com/us
t 770 933 1965
e hi@oneadvanced.com

3200 Windy Hill Road, Suite 230 West, Atlanta, GA 30339