Application Modernization for OpenVMS Customers

Brett Cameron
May 2018

Abstract

Many OpenVMS users run large, complex, business-critical custom written software applications that have served them exceptionally well for many years and continue to do so, but for whatever reason these applications now need to interoperate and exchange data with other external systems via new industry-standard mechanisms, or applications might require a new or alternative user interface to bring them into line with other systems. The options available are to replace the existing OpenVMS-based application environment or to modernize ("rejuvenate") it in some way so that it can operate in the new technology environment and continue to serve the business, potentially adding even greater business value. In this talk, the Brett will introduce the topic of application modernization and will discuss some of the different approaches and methods than can be taken to modernize custom OpenVMS application environments. Finally, Brett will introduce a set of services that can be provided by VMS Software Inc. to help OpenVMS users to modernize their application environment and retain their considerable and valuable investment in OpenVMS technology, so that it to take advantage of new software technologies and better interact with other systems.

AGENDA

- **Introduction (description of the problem)**
- Approaches to modernization
- Common situations
- Common issues and how to deal with them
- How can we assist?
- Summary

Description of the problem

As per the abstract…

The are many OpenVMS users running large, complex, business-critical bespoke applications that have served the business well for many years and continue to do so, but for whatever reason these applications now need to interoperate and exchange data with other external systems via new industry-standard mechanisms or applications might require a new or alternative user interface to bring them into line with other systems. The options available are to replace the existing OpenVMS-based application environment or to modernize ("rejuvenate") it in some way so that it can operate in the new technology environment and continue to serve the business, potentially adding even greater business value.

Modernization drivers

- Preservation of valued existing assets
- Reduce liability
  - Hardware availability and support issues
  - Software support issues
  - Availability of technical skills
- Increased revenue/profit
  - Reduce time-to-market for new products and services
  - Improve customer satisfaction
  - Enhance operational capability/efficiency
  - Create new revenue streams through new capabilities
- Reduce costs
  - Reduce spending in software licensing, maintenance, support, ...
  - Reduce labour costs
  - Reduce operating costs
    - Power consumption, rack space, ...

Simply upgrading to the latest hardware, operating system, and layered product versions does not fully address such matters (although it may help in certain areas, such as hardware maintenance and support costs, application performance and scalability, and so on).

Modernization drivers

- As applications age, they become more expensive to maintain and the risk associated with their operation increases
  - Entropy takes hold
  - If timely and appropriate action isn't taken, this can become a very big and expensive problem

CHAOS RULES THE UNIVERSE, SCIENTISTS CALL IT ENTROPY...
EVERYTHING IS BREAKING DOWN TENDING TOWARD GREATER AND GREATER DISORDER.
IT'S GREAT TO BE ON THE WINNING SIDE.

http://potatoci.blogspot.co.nz/2016/08/entropy-art-from-god.html

Any of these things sound familiar?

- Multiple deployment processes
- Weak deployment processes
- Weak testing processes
- Multiple build procedures
- Poor build procedures
  - When was the last time you did a full rebuild?
  - Can you do a full rebuild?
- Multiple code management systems
- No code management system
- Multiple code streams
- Multiple skill sets required to maintain code
- Ever increasing support costs
- Unsupported software
- Unsupported hardware
- No common data dictionary
- Hidden functionality
- Unknown (and therefore untested) functionality
- Inability to expose functionality as services
- Integration at any level is generally problematical
- Difficult to access data
- Inconsistent data
  - Poor referential integrity
- Old data
  - Takes up space
  - Causes confusion
- Outdated user interfaces
- New I/O and UI devices hard to support
- Integration issues

AGENDA

- Introduction (description of the problem)
- **Approaches to modernization**
- Common situations
- Common issues and how to deal with them
- How can we assist?
- Summary

Approaches to modernization

- Relearn
  - Capture details of intellectual property invested in applications
  - Enable investment to be preserved and carried forward through other modernization activities
- Re-factor
  - Code optimization (improve run-time efficiency)
  - Improve maintainability
  - Always a good idea
- Re-host
  - Migrate applications to lower-cost platforms without significantly changing business logic
- Re-interface
  - Create new interfaces to better leverage and extend application features and value
    - New user interfaces
    - Integration with other systems
    - Standards-based interfaces
- Re-architect
  - Re-engineer applications leveraging new technologies
- Replace
  - Replace existing applications with COTS packages
  - Redevelop applications from scratch
- Retire
  - Decommission applications that no longer provide any business value

An overall modernization strategy may encompass one or more of the above activities.

Approaches to modernization

Cost, complexity, and risk…

Relearn:

- Simply updating documentation and test cases can yield considerable benefit for low cost and essentially no risk. This updated information can then be put to good use for subsequent modernization activities.

Refactor:

- Incremental refactoring of code can yield considerable benefits for reasonable (and incremental) cost at low risk. Refactoring can also be a preparatory step for other modernization activities.

Integration:

- Integrations with other systems can often be implemented cost effectively and can yield considerable business benefits (if done correctly).

Migration:

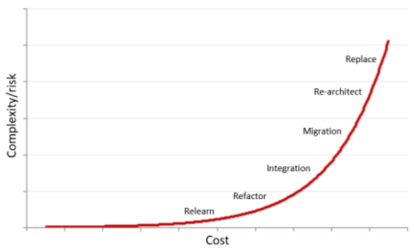
- Migration of applications to new platforms can range from medium-cost and low-risk through to high cost and high risk.

Re-architect:

- Re-engineering applications is typically costly and risky.

Replace:

- Replacing existing applications with COTS packages can be expensive and risky, but can also be very successful if planned and implemented properly.



Approaches to modernization

- No cookie-cutter solutions
  - Every situation is different
  - Can’t just go and buy a modernization solution off the shelf
  - Often need to get quite creative
  - It’s not a perfect world
  - Expect the unexpected

- But…
  - Your problem is probably not as unique as you think

Some key considerations

- Replacement or modernisation…
  - Does the application do the job?
    - Functionality
    - Performance
    - Maintainability
    - Capacity
    - Reliability
    - Ability to integrate with other systems
  - Cost of ownership
    - Do you know how the numbers stack up? You might be pleasantly surprised!
  - Return on investment
    - How soon do you want payback on your investment?
  - Availability of skills
  - End-of-life technologies

There are invariably many factors to be considered, and all of these factors are generally in one way or another inter-related.

So what should you be doing?

Does the application do what it needs to do and does it do it well? Is it maintainable? Is is cost-effective? Answers to these and related questions will help to drive your modernisation strategy.

I'VE NEVER SEEN YOU DO ANY REAL WORK AROUND HERE, IRV. HOW DO YOU GET AWAY WITH IT?
I WROTE THE CODE FOR OUR ACCOUNTING SYSTEM BACK IN THE MID-EIGHTIES. IT'S A MILLION LINES OF UNDOCUMENTED SPAGHETTI LOGIC.
IT'S THE HOLY GRAIL OF TECHNOLOGY!!
YOU BOYS MAY FIND A LITTLE EXTRA IN YOUR ENVELOPES THIS MONTH.
© 1994 United Feature Syndicate, Inc.

AGENDA

- Introduction (description of the problem)
- Approaches to modernization
- **Common situations**
- Common issues and how to deal with them
- How can we assist?
- Summary

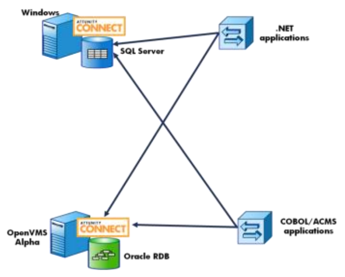
Some common modernization scenarios

- Move from VAX to Alpha or Integrity
  - Not so common, but we do get customers wanting to integrate VAX-based applications with other environments
- Move from Alpha to Integrity
- Replace green screens with web interface or GUI
- Integration
  - Web services
  - Message queuing and/or transactional middleware
  - Data access
  - User interface
- Use of Open Source technologies
- Single sign-on (external authentication using LDAP/ACME, …)
- Database migration
- Introduction of new technologies
- Programming language conversions
- …

Need to start thinking about x86!

Example: complex data-level integration

- A Large European government department
- Data-level integration with new Microsoft-based application environment
- Heterogeneous environment
  - Microsoft SQL Server and .NET
  - RDB, ACMS, and COBOL on OpenVMS
- Attunity Connect used to facilitate all database operations
  - Query both SQL Server and RDB from OpenVMS application code via a common interface
  - Uses the Attunity ODBC API
  - Distributed transactions
  - Simultaneous updates to SQL Server and RDB databases



Example: complex Alpha to Integrity migration

- Large mobile telecommunications provider

- Large and complex code base (> 3M LOC)
  - COBOL with embedded SQL
    - Some Pascal and C code
  - ACMS
    - Large number of tasks
    - High transaction rates
  - Remote Sybase database (on UNIX)
  - Microsoft Visual C/C++ GUI
  - Bespoke Java integration layer

- Key solution aspects
  - Replacement of Sybase client with Open Source FreeTDS API
    - Sybase client not available for OpenVMS Integrity
    - Binary translation possible but not considered a viable longer-term solution
  - Embedded SQL pre-compiler developed as part of the solution
  - Replacement of Sybase OpenServer components with custom TCP/IP-based interface
  - Use of WSIT to facilitate integration with corporate ESB
  - Heavily customized WSIT Velocity templates used for code generation

Note that we had to get rather creative here, which is often the case on large-scale modernization projects. Tools and techniques developed for this project could be reusable on other projects. This project was far more complicated than most Alpha to Integrity migrations; however no such migration should be considered trivial, and proper processes and planning are just as important as the technical solution when it comes to achieving a successful result.

Example: incremental modernization

- New Zealand government department
- Application suite initially developed by DEC in 1988
  - Rally, COBOL, C, RDB, …
  - Additional applications components implemented mid to late 1990's
- Migration from VAX to Alpha 1997
- Migration from Alpha to Integrity 2011
- Original Rally screens replaced with Visual Basic GUI front end 1997
- Custom-written HTTP-based RPC-style client-server interface
  - "WebRPC"
  - Web services before web services ☺
- FreeTDS used to access SQL Server from OpenVMS environment
- Data-level interface to Oracle Financials (UNIX)
- Further introduction of web technologies 2012
  - Replacement of existing RPC middleware with gSOAP-based web services 2013
  - Replacement of Visual Basic screens with new .NET-based user interface components
  - Web service-based integration with other systems

Example: introduction of web services

There are many organizations using web services on OpenVMS in business-critical production environments…

- Spanning multiple sectors
  - Finance
  - Healthcare
  - Telecommunications
  - Tourism
  - Manufacturing
  - Entertainment
  - ...
- Involving integration with OpenVMS applications written in various languages and using various other technologies
  - C, COBOL, FORTRAN, BASIC, Pascal
  - ACMS, DEC/BEA/Oracle MessageQ
  - ...

Example: introduction of web services

Some examples…

- A large healthcare provider uses web services on OpenVMS to integrate a .NET application used by doctors across the US with their back-end systems
- A steel manufacturer uses web services to communicate between their OpenVMS based systems and factory cranes fitted with on-board computers running a .NET application
- A large European travel company uses web services with Apache to provide a web services interface to their ACMS application as well as calling web services from COBOL code within their applications
- Several other ACMS customers have implemented similar solutions
  - ACMS tasks can be viewed as services
  - Exposing ACMS tasks as web services can be largely automated
- An Australian customer is calling an internet-based web service from an existing COBOL application
- Another customer in Australia is invoking a web service from an existing application and is using SSL for additional security
- The previously-mentioned New Zealand government department
- ... and many more

AGENDA

- Introduction (description of the problem)
- Approaches to modernization
- Common situations
- **Common issues and how to deal with them**
- How can we assist?
- Summary

Solubility

- Issues can be technical and non-technical
  - Most technical issues are soluble
  - Some non-technical issues may be insoluble

More often than not, modernization projects fail or go badly wrong not for technical reasons but for reasons such as poor planning, poor estimation, wrong skill mix, weak project management, lack of governance and/or executive sponsorship, and so on. As with any project, good people and proper processes are invariably more important than technology!

Common challenges

- Some common issues faced by modernization projects...
  - Incomplete or non-existent build procedures
  - Ridiculously convoluted build processes
  - Loss of business knowledge
  - Poor documentation
  - Poor executive sponsorship
  - Poor test coverage/procedures
  - Inexperienced modernization team
  - Business wanting functional change in parallel with modernization
  - Poor planning/estimating
- Some more specific technical challenges...
  - Missing code
  - Unsupported software (software not available for OpenVMS on Integrity)
  - Old code not compliant with newer compiler versions
  - Specialist drivers/hardware interfaces
  - Subtle current and target platform differences
  - Performance considerations
  - ...

Missing code

- Generally applicable to VAX to Alpha and Alpha to Integrity migrations
- Situations need to be assessed on a case-by-case basis
- Missing code may be from a long-gone vendor
  - Complete black box
  - Potentially a show-stopper, depending on what the code does
- Re-write or replace
  - Need requirements to work from
  - Cost, time, effort, ...
  - Ultimately the only solution
- Binary translation (AEST, TIE, ...)
  - Generally works
  - Often works very well
  - Only applicable to executables and shareable images
    - No good if your missing code is in object files or libraries
    - No good for privileged images
  - Can be performance problems
    - Alignment faults
    - I/O performance
  - Really a stop-gap measure/interim solution
  - May not be supported
    - No guarantees are provided

The bottom line is that missing code can be a big problem! If binary translation is an option, give it a try and you might be pleasantly surprised, but translation generally should not be viewed as a long-term solution.

Software not available on Integrity

- Software not available or perhaps not supported
- Binary translation
  - See comments on previous slide
- Replace
  - Try to find a viable off-the-shelf or Open Source alternative
  - Develop your own replacement

As with Alpha to Integrity migrations, software availability will be an issue for migration of some application environments to x86. Start planning now; we can help!

AGENDA

- Introduction (description of the problem)
- Approaches to modernization
- Common situations
- Common issues and how to deal with them
- **How can we assist?**
- Summary

How can we assist?

- Porting lab
- Architectural workshop
- Assessment of current environment
- Introduction of new technologies into an existing OpenVMS environment
- Application migration
- Application maintenance and support services

Porting lab

- Well-equipped secure laboratory environment
- Build and test applications
- Evaluate new software technologies
- Supported by skilled OpenVMS engineers

Architectual workshop

- A fixed-price on-site consulting service
- The main intent of the Architectural Workshop service is to:
  - Conduct in-depth discussions on the current OpenVMS environment and assist in the definition of future goals and how to achieve them
  - Provide information on current and future technology provided by VMS Software Inc. and its partners
  - Identify areas of further investigation that can be tested in proof of concept projects
- The workshop is driven by a jointly agreed agenda and the outcomes of the workshop are documented

This service should be performed as the first stage of a larger modernization initiative, as it encompasses all facets of the IT environment under investigation. It is conducted at a relatively high level and gathers information that may be put to good use in subsequent phases of a modernization program.

Assessment of current environment

- This service includes an examination of the current OpenVMS environment and provides recommendations regarding matters such as:
  - Operating system, layered product, and application tuning
  - Hardware configuration
  - Growth/usage (capacity planning)
  - Backup strategies
  - Availability
  - Security
  - DR strategy
  - Service level agreements
  - Modernisation roadmap/strategy
  - Risks
  - ...

If the intention is to have VMS Software Inc. help devise a modernization strategy and roadmap roadmap, this would typically be done as a follow-up activity to this assessment.

Introduction of new technologies

- This service covers such items as:
  – Investigation of the existing OpenVMS environment and establishment of criteria for the introduction of new technologies

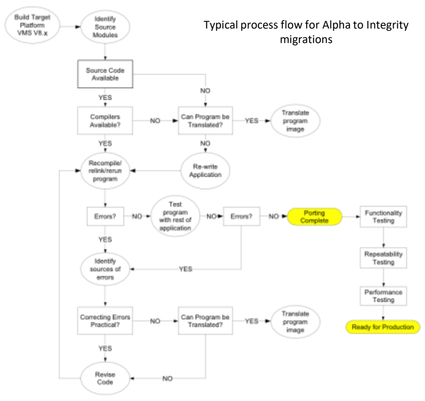
- And includes:
  – Selection criteria for a particular technology
  – Network infrastructure and ability to participate in Internet-based interactions
  – Existing platform presence and future platform plans
  – Identifying core business services and mapping these to existing or modified software applications
  – Examination of existing software applications to identify components that may be exposed as services to facilitate integration with other systems

Application migration

- Comprehensive service to help move OpenVMS-based applications from VAX to Alpha (maybe), VAX to Itanium (maybe), or Alpha to Integrity
  - Also need to start thinking about x86

- Potentially modify applications to participate in a modern heterogeneous computing environment
  - Although modifying applications while migrating them is generally not a good idea

The migration process advocates a phased approach to minimize risk and deliver the best possible solution. The migration lifecycle can be tailored to address the specific needs of each project, and is predicated on helping you to preserving and enhancing the OpenVMS-based environment.


Typical process flow for Alpha to Integrity migrations

Application maintenance and support

Many organizations with large custom-written OpenVMS applications no longer have sufficient software development expertise in-house to maintain and support these applications or find it difficult to find a suitably skilled service provider that is able to provide these services. VMS Software Inc. and our partners are able to provide cost-effective specialist application maintenance and support services for custom-written OpenVMS-based software applications and can tailor such services to meet specific requirements.

- Much like any other software support arrangement, except it's your code
- Bug fixes
- Enhancements
- Requirements definition
- Modernization
- Documentation
- Testing
- Source code control
- ...

AGENDA

- Introduction (description of the problem)
- Approaches to modernization
- Common situations
- Common issues and how to deal with them
- How can we assist?
- **Summary**

Summary

From a technology perspective there is absolutely no impediment to modernizing (rejuvenating) bespoke OpenVMS-based application environments.

- Exchanging information via web services or other integration mechanisms
- Leveraging Open Source software
- Participation in heterogeneous service-oriented environments
- Introduction of GUI and web-based interfaces
- Interaction with cloud services
- ...

- If you're on Alpha and intend to move to x86, start planning now!

We can help, be it with workshop sessions, short-term consulting assignments, or with project-based delivery of larger pieces of work.

Summary

- Just about every modernization project is different
- Imagination and creativity are often required (no off-the-shelf solutions)
- Consider your options very carefully
- Investigate what others have done
- Validate your ideas (proof-of-concept)
- Understand the numbers (ROI/TCO)
  - Does a particular option really make commercial sense?
- Good people and proper processes are at least as important as technology

Creativity and imagination (sometimes called hacking) is often an important aspect of modernization projects. There are no off-the-shelf solutions, and sometimes it is necessary to develop your own tools or integration components in order to achieve a successful result.

- Modernization projects often require a diverse set of skills
  - Technical
  - Project management
  - Testing
  - ...
- Ensure senior team members are suitably experienced
- Understand the consequences of "big-bang" versus incremental
- Ensure that you have strong executive sponsorship and business by-in
- Do not underestimate testing effort
- Developing your own tools may not be as difficult as you might think
- Open Source is your friend
- We can help!

Questions