OpenVMS Migration
Design and experiences

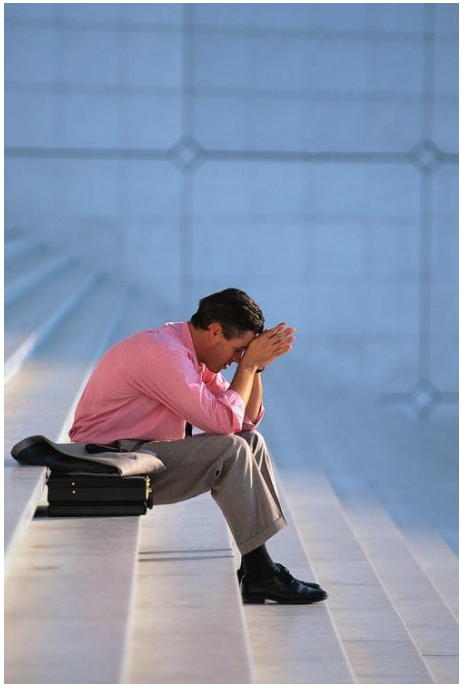
Colin Butcher
XDelta Limited
+44 (0) 117 904 8209

Objective

Explore key issues when migrating from OpenVMS

"If it was easy, most people would have migrated from OpenVMS already"

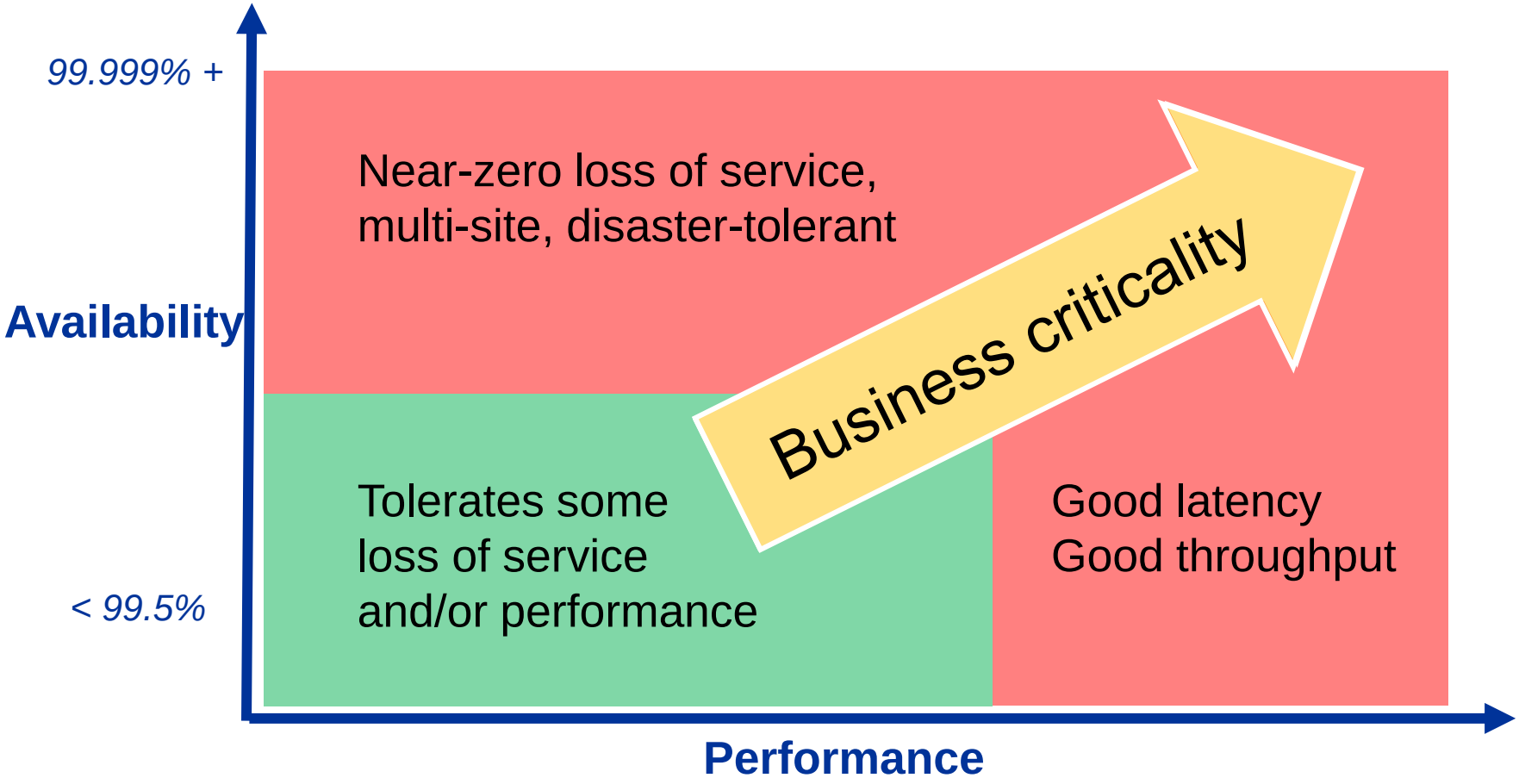
"OpenVMS is a very capable system which people have exploited to the full"



OpenVMSmigration.com

- XDelta in collaboration with specialist organisations
- Complementary skills to solve difficult problems
- Strategic planning, technical leadership and project direction with clarity of vision and an eye for detail
- Ensure long term success through skills transfer
- Gartner (2009):
  - Identified XDelta as one of few companies world-wide capable of OpenVMS migration related projects

Business criticality



Current situation

Most OpenVMS systems purpose written:

- Tight integration with operating system and infrastructure
- Multi-site mission-critical capabilities

Well engineered operating system:

- Well structured and documented
- Scales very well from small to large implementations

Evaluate your options:

- Changing platform is something you do very infrequently
- New platform has to be stable for many years
- You will probably change platform again in 10+ years
- What level of support do you need for h/w and s/w ?

Where to start ?

Don’t start in the wrong place:

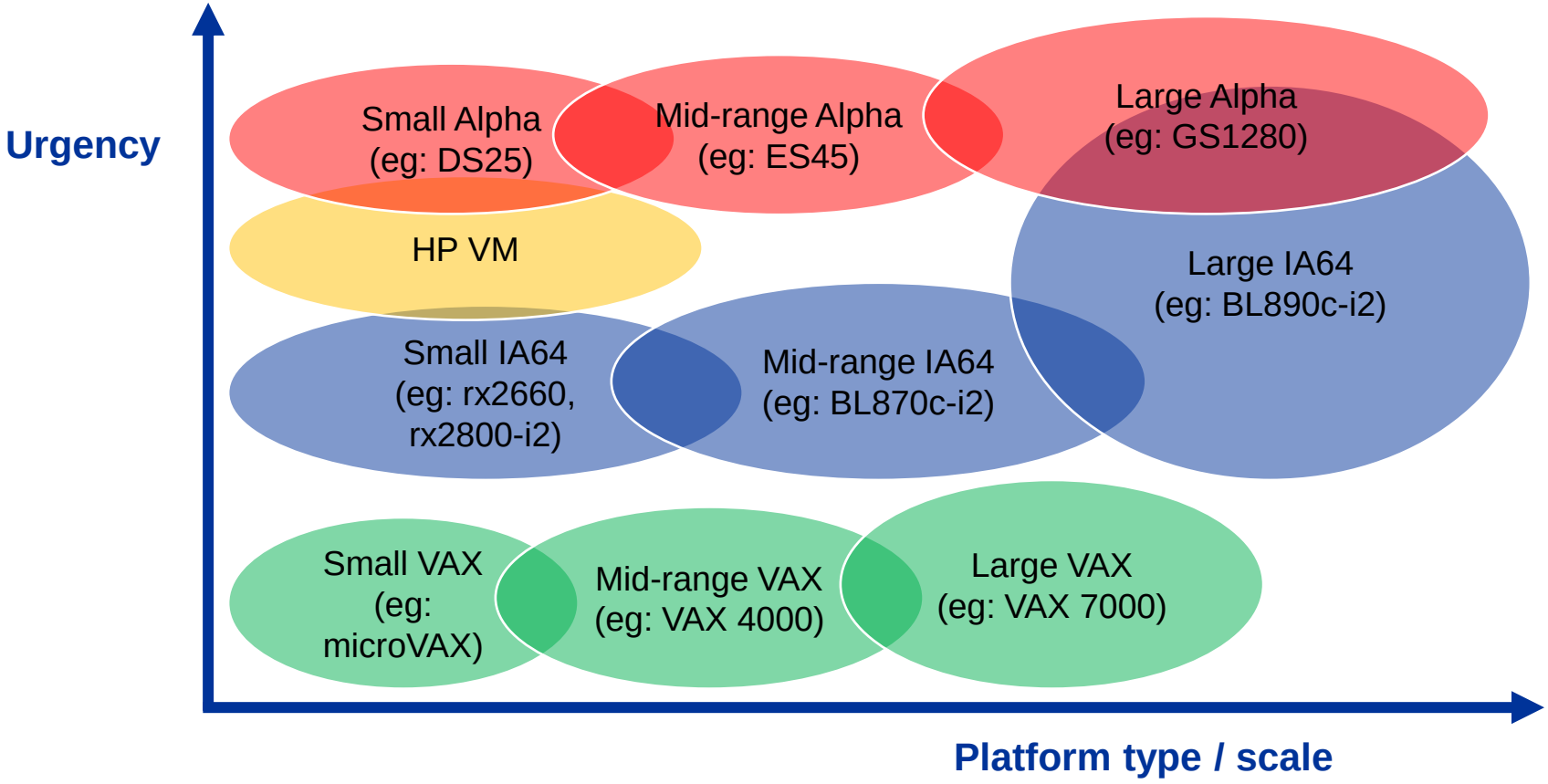
- Understand systems so they can be migrated without disruption
- It is a far bigger problem than just porting the code
- Use this as an opportunity to “do the right thing”
- Don’t rush - you’ll have to live with the decisions you make

There is a range of solutions - what works best for you ?

Migration

- Define criteria for success and how to meet them
- Design, plan and implement
- Test and demonstrate
- Transition and data migration with minimal disruption

Urgency



Complexity

Most systems have evolved over many years:

- PDP11 to VAX to Alpha to Integrity is not unusual

A range of languages:

- Basic, Fortran, Cobol, Macro32, C, C++, Pascal, Ada, Java, DCL, etc.

A variety of workloads:

- Interactive, Batch, Real-time, Backups, etc.

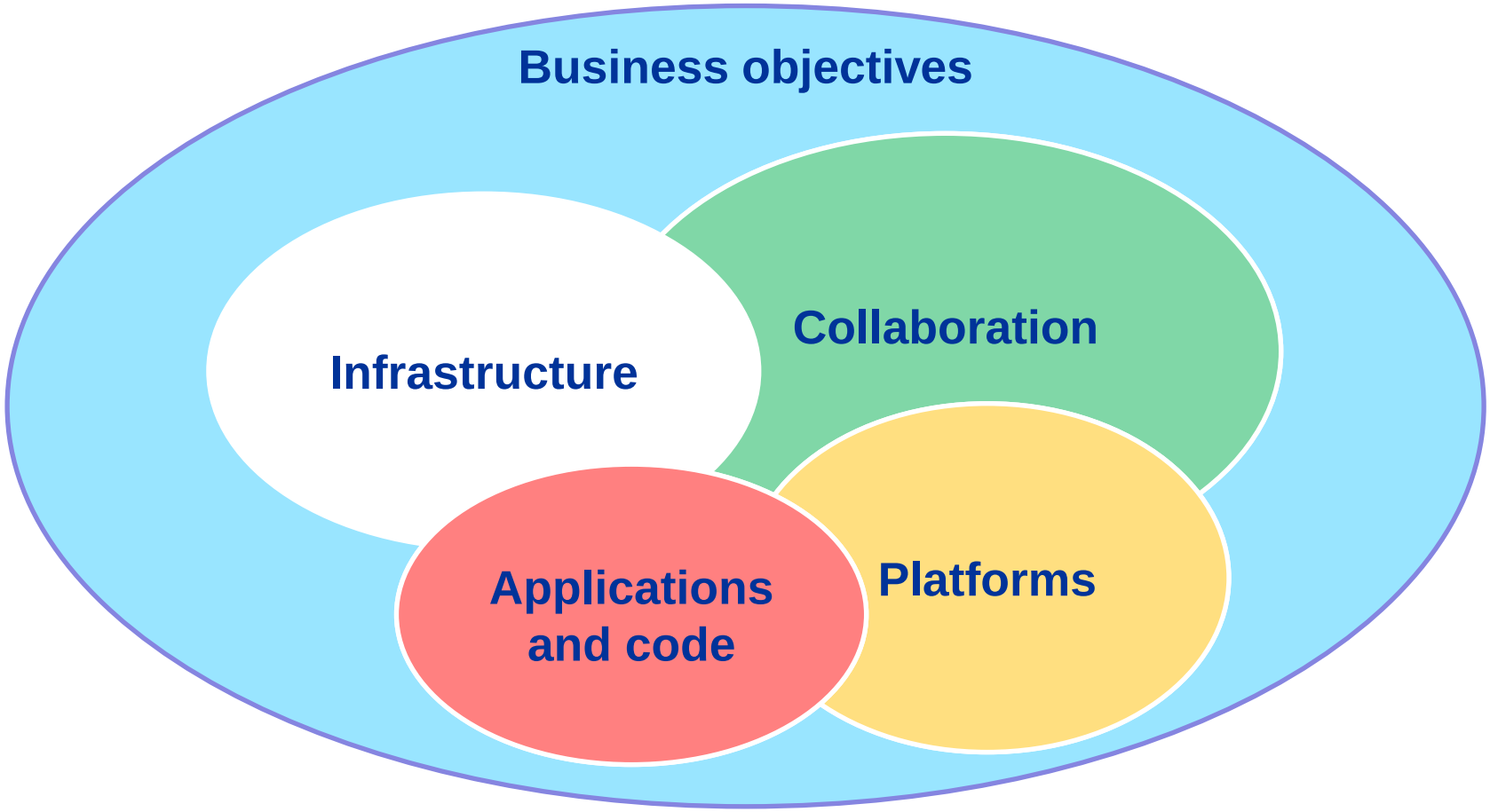
Set up in different ways:

- Naming conventions, Programming styles, DCL usage, etc.

People have moved on:

- Knowledge, Documentation, Design techniques, etc.

Holistic approach



System architecture

Platform irrelevant

- Business processes and logic flow
- Business requirements for security, availability, performance, expansion, connectivity, etc.

Platform independent:

- To what extent can we isolate implementation from platform ?
- Abstraction layers to hide platform specific details
- Choice of languages, etc.

Platform specific:

- Mechanisms that implement functions (eg: volume shadowing)
- System service calls (eg: file system, synchronisation, etc.)
- Requires detailed understanding of behaviour

Code conversion

Code analysis and conversion:

- Don't worry about how much code there is
- Most code uses a subset of available features
- Largely a problem with "old", less portable languages
- Language conversion may be less difficult than you think
- Maintain the target code, not the original source
- Be able to move it again in the future

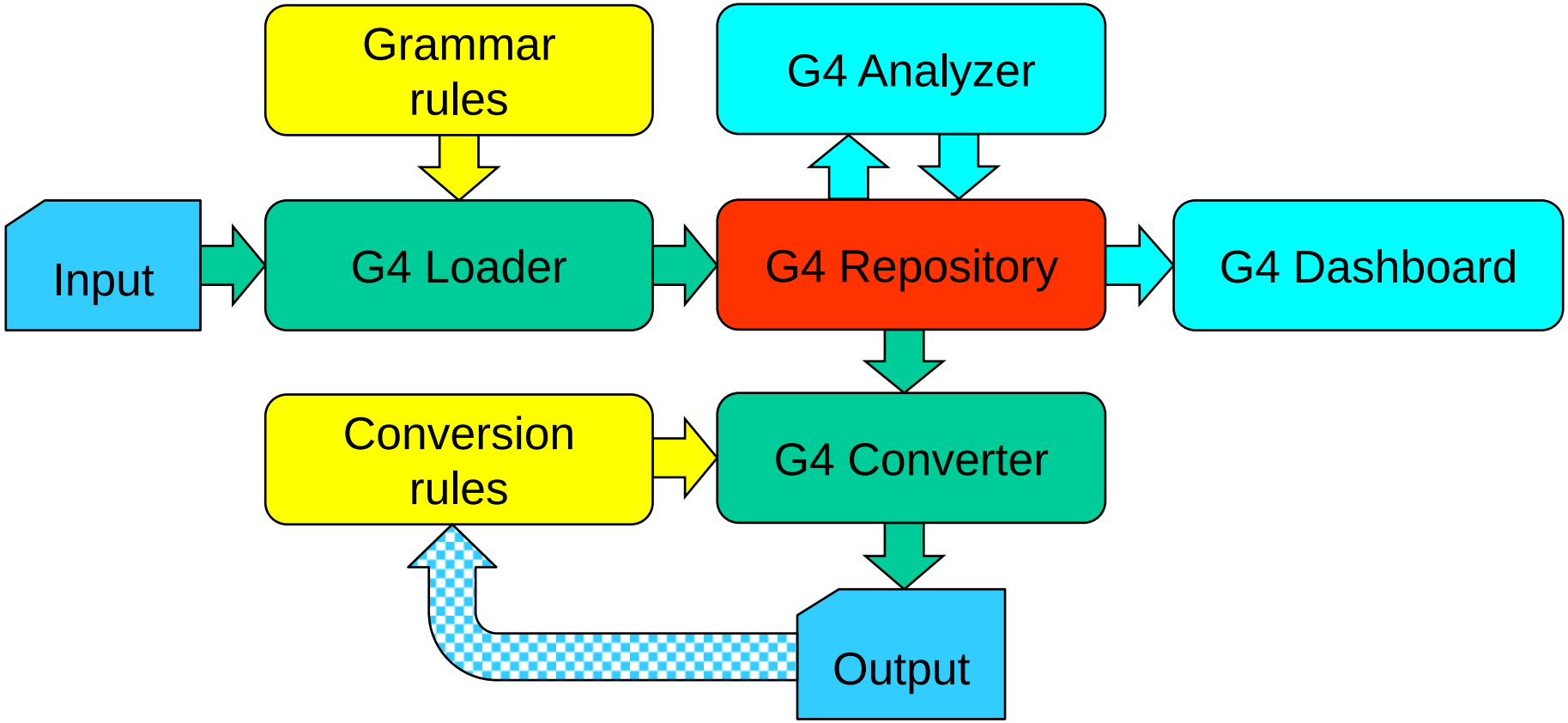
Abstraction layers:

- Hides platform specific functions

Code re-factoring:

- Removes "dead" code and cleans up flow of control
- Improve performance and testing

Cornerstone G4 Platform



Code migration

Code migration – language specific:

- Run-time libraries with standard interfaces

Code migration – platform specific:

- System services (eg: locking, process creation, logical names)
- IO operations (eg: ASTs, SMG routines)

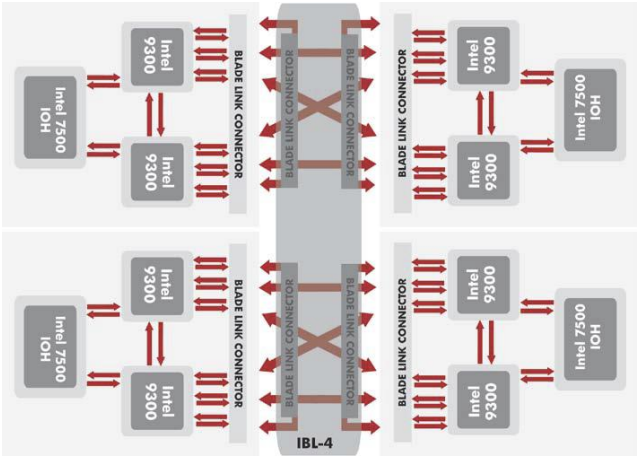
Code migration – file system and file naming:

- Mapping of file naming conventions
- RMS files (eg: indexed files)
- Data migration and data types

High performance

High performance systems:

- Native hardware for best performance
- Virtualisation - no
- Emulation - no



Performance constraints:

- Bandwidth and Latency
- Achievable IO operations rate
- Parallelism within the applications
- Memory usage and NUMA effects

Limiting factors are often outside the system itself:

- Network infrastructure
- Storage arrays and SAN fabric infrastructure
- Inter-operability with other systems

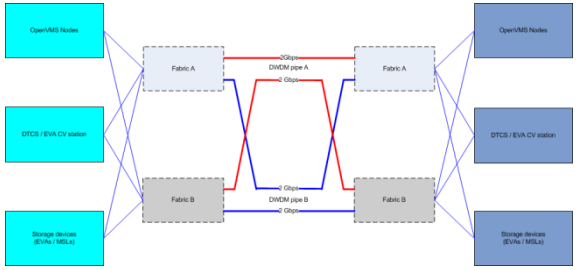
High availability

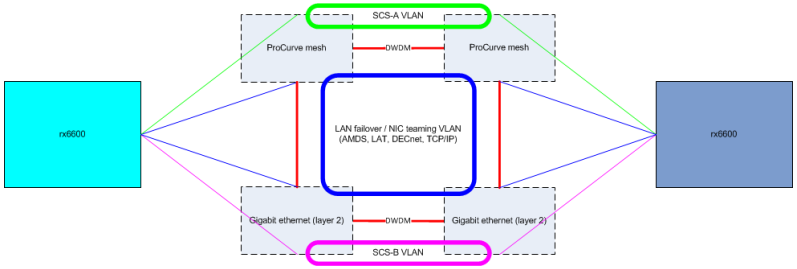
High availability systems:

- Carefully integrated with a consistent overall architecture
- No single points of failure
- Make best use of the capabilities of the individual components
- Understand the whole system and write as little code as needed

How to migrate with minimal disruption ?

The closer you get to zero downtime, the harder it is!





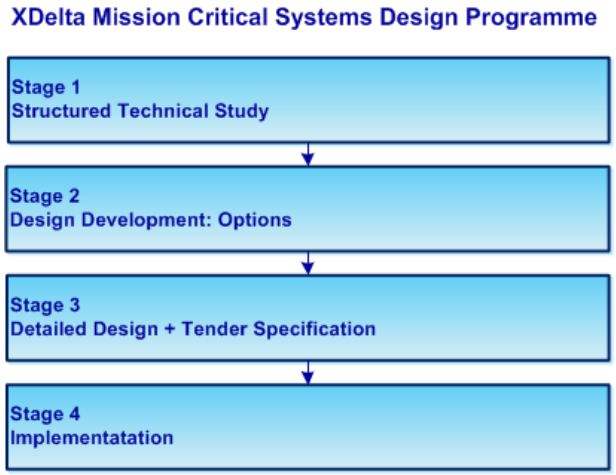
Project delivery

The holistic approach is essential:

- Business impact and risk
- Complexity and inter-relationships
- Systems engineering:
  - Structure; Design; Planning; Testing
  - Data migration; Transition
  - Support; Knowledge transfer
- Collaboration with your suppliers

Extra workload and different skills:

- Migration will consume resources
  - You may not have all the skills you need
  - You may not have the time you need
- Start early with a careful assessment
- Make assumptions if needed
- Take the time to do it well



OpenVMS migration - contact us

Thank you for your participation.

If you have questions, or need help, please ask!

Colin Butcher, Derek Webb or John Foster
XDelta Limited


+44 (0) 117 904 8209


migration@xdelta.co.uk