Mainframe Migration Strategy

Table of Contents

About this Document ................................................................. 3
Migration Overview .................................................................... 4
   Infrastructure Migration ......................................................... 4
   Application Migration ............................................................ 5
Migration Strategies ................................................................. 6
   Refronting ............................................................................ 6
   Replacement .......................................................................... 6
   Rehosting ............................................................................ 7
   Rearchitecting ................................................................. 7
   Interoperation ....................................................................... 8
   Retirement ............................................................................ 8
Migration Methodology Overview ................................................. 9
   Assess & Design Phase ....................................................... 9
   Build & Implement Phase ................................................... 10
   Manage & Support Phase .................................................... 11
Migration Methodologies ............................................................. 13
   Assess & Design .............................................................. 13
   Build & Implement .......................................................... 14
   Manage & Support ........................................................... 20
Migration Project Planning .......................................................... 24
   Defining Scope .................................................................. 24
   Understanding Risks and Planning Contingencies .................... 24
   Organizing the Project Team .............................................. 26
   Developing a Project Plan ................................................... 26
   Closing the Project ............................................................ 29
Contact .................................................................................. 30
About this Document

This document provides a high-level summary of Transvive’s strategies and proven, best-practices methodologies for migrations.

While specifics may vary by the technology and business logic involved, migration projects all have many commonalities and can, therefore, be approached with certain common strategies and methodologies. Strategies and methodologies presented in this document are suitable for projects ranging from the smallest data conversion to the largest legacy migration project with a repeatable and systematic approach that ensures predictability and success.

Transvive’s strategies and methodologies cover the entire migration project lifecycle and include the following:

- Assessing the current environment to migrate
- Planning for a migration project
- Architecting a new target environment
- Implementing a migration by using available tools and processes
- Managing the newly migrated environment
Migration Overview

In general terms, the term “migration” refers to the movement of technology from older or proprietary systems to newer, more versatile, feature-rich and cost-effective applications and operating systems. In addition to cost savings, migration results in improved interoperability, reliability and manageability. Examples of migration include, but are not limited to, the following:

- The migration of common off-the-shelf (COTS) software from one platform to a larger or smaller similar platform
- The migration of data from one database to another, possibly similar data storage technology
- The migration of a custom-written application from one platform to a different platform and operating environment

For the purposes of this document, we will define two broad categories of migration:

- Infrastructure Migration
- Application Migration

Infrastructure Migration

The term “Infrastructure Migration” refers to the process of migrating all layers of the computing platform, not only the applications that support business functionality. This can be a complex exercise that will have a greater impact on the entire IT operations than other strategies would. For example, Infrastructure Migrations can include changes to the following:

- Applications that support business functionality
- Application infrastructure that supports the applications, such as web servers, application servers, middleware and database technology
- Third-party products provided by ISVs
- Application architecture, e.g. multithreading
- Computing and storage platforms, e.g. SAN or attached storage
- Network infrastructure
- Facilities infrastructure, such as power, ventilation and cooling
- Management policies
- System monitoring and management tools
- Locally written scripts to manage applications and data
- Administration and support staff training
Application Migration

The term “Application Migration” applies to applications rather than infrastructures. In particular, it usually applies to custom-written applications and refers to modifying or normalizing the code of an application so it can be recompiled and deployed on a new hardware platform that supports a different Operating System (OS). Application Migration is inherently associated with modifying the code base of an application so that the functionality provided by the Application Programming Interfaces (API) of the existing OS and supporting software products is replicated in the new target environment.

Application Migration typically requires minimal understanding of the logic or functionality of the application. It is a somewhat mechanical effort for making the application compatible with the new environment. An Application Migration strategy requires the integration of the application with a new development environment, as well as with a new operating system. While source code, scripts and data are moved, compilers, source code repositories and software tools are replaced by new versions that are compatible with the target platform.

When migrating an application, any supporting third-party software must also be migrated. If the software is not available on the new platform, similar software must be found and integrated into the application. Should the amount of integration become excessive, the migration might begin to look less like a rehosting and more like a rearchitecture effort (see Migration Strategies below).
Migration Strategies

In broad terms, there are various strategies available for a migration effort, including the following:

- Refronting
- Replacement
- Rehosting
- Rearchitecting
- Interoperation
- Retirement

The Transvive team has completed multiple projects involving all of these strategies.

Refronting

Rather than rewriting an entire application, a programmer might be able to change just the data entry portion of the application. Refronting or adding a more aesthetic interface to an existing application without changing the functionality, is an option. Users will have access to the same data but will be able to access it in a more efficient fashion without the use of expensive terminals, cabling or peripheral interconnects. When desired and appropriate, a browser-based solution can be developed. Web-enabling an application can provide significant cost reduction.

Transvive employs an architectural model that allows new components to invoke old, migrated components with minimum change to the migrated components.

Replacement

With the replacement approach, the legacy application is decomposed into functional building blocks. Once an application is broken down in this manner, portions of a generic and often complex, custom-written legacy application can be replaced with a common off-the-shelf (COTS) application. Of course, the package must be able to run on the target OS.

When considering replacement as a strategy, the replacement of the application’s code with a new third-party software product is an option. A more powerful option might be to adopt a new application solution and change the organization’s business logic when it no longer yields competitive advantage to match the package’s optimum business process.

Replacement can also be applied to components within an alternative strategy. Interestingly, as a strategy, it potentially yields the highest benefits and involves
the highest degree of cost, yet when applied to components within an alternative strategy, it can be a quick and low-risk strategy.

**Rehosting**

Rehosting involves moving complete applications from a legacy environment with no change in functionality. Rehosting offers the advantage of low development risk and enables familiar legacy applications to be quickly transferred to a more cost-effective platform that exhibits lower total cost of ownership (TCO) and a faster return on investment (ROI). Extensive retraining of users is not needed because the architecture, interface and functionality do not change. Rehosting is an excellent approach for companies desiring to decrease their maintenance and support costs.

Rehosting does not change the application or the architecture. This means that new technology that is available in the target environment might not be properly utilized without some modification of the application. Rehosting is a preferred solution when the current business logic and business process remain competitive in the enterprise’s markets and are worth preserving. Rehosting offers the possibility of using cost savings accrued through switching development and runtime environments to fund full rearchitecture projects, when warranted.

**Rearchitecting**

Rearchitecting is a tailored approach that enables the entire application architecture to migrate to the new OS, possibly using new programming paradigms and languages. Applications poor in IT effectiveness and functionality are the best candidates for rearchitecting. This approach is best used when time is not a major factor in the decision.

Using this approach, applications are developed from scratch on a new platform, enabling organizations to significantly improve functionality and thereby take full advantage of the full potential of a target system. Rearchitecting opens the opportunity to improve the business logic and processes and to change the developer productivity model.

The downside to this approach is that it requires new or additional training for users, developers and technical staff. In addition, rearchitecting requires the most time and is the most error prone of all the possible solutions.

Despite these problems, rearchitecting is perceived to be the correct strategy and these problems become project risks. Transvive mitigates these risks by the application of appropriate business acceptance testing with internal and external users.
Interoperation

In certain cases, it might be advantageous to leave an application where it is and surround it with new technology when it is required by an enterprise. Interoperability is a strategy that should be considered if business requirements are being met, IT effectiveness is high and the environment is capable of interacting with current technology.

Unfortunately, business drivers – for example, the existence of a leasing or outsourcing contract – might dictate that an application should stay where it is for some period of time. This is one of the risks of abandoning the IT environment to a third party. Over time, outsourced applications become orphans within the IT infrastructure. They are not fully integrated into the IT environment, most likely do not have a development staff and run on outdated hardware that is no longer cost effective.

Many independent software vendors (ISV) provide technology that enables legacy applications and storage technology to interoperate with newer technology. Intelligent adapters exist that support interactions between the mainframe and modern computing alternatives.

Retirement

Changes in technology can obviate the need for specific functionality in an application or an overall solution. As middleware or third-party products mature, they might render the functionality implemented in the application obsolete. In this case, legacy utilities or legacy application functionalities can be retired because they are no longer required or are implemented elsewhere in the solution.
Migration Methodology Overview

Transvive has developed a repeatable and systematic methodology for migration that helps ensure predictability and success of migration projects. Transvive follows a structured architecture methodology built on best practices.

Following a structured architecture methodology is different from building a solution in an ad hoc fashion in that it requires a plan upfront for much of the work that needs to be done. In addition, the application of the methodology ensures collaboration with all of the stakeholders involved in a migration effort to develop an architecture that will address their needs as well as the business objectives.

This approach assumes the presence of a business case, a risk log and a project startup plan, which includes a project governance model. Fundamentally, this is a project methodology, which means that a project justification must have been successful. This methodology should be supplemented with formal project planning activities.

The first two phases are best undertaken in accordance with a gap analysis methodology as follows: document the current solutions, design a new solution and document the changes needed to achieve the required improvements. The third phase concerns the operation of processes within the production/post-production space. The processes and infrastructure to support manage and support phase will need to be architected, designed and implemented.

Although it may seem to be the case, there is no stately progression from one phase to the next. In practice, migration projects will iterate within phases and reenter preceding phases to improve the deliverables of those preceding phases.

The iteration of activities can occur within and between phases. The phases outlined here should not necessarily be confused with stages within the project plan. The project planning methodology will also mandate certain activities as projects transition between stages.

Assess & Design Phase

The initial assessment and design phase in a migration project addresses the following technical startup-planning tasks for the project:

- Assessing the environment to ensure that all of the assumptions made during project justification have been proved and that all of the requirements and dependencies for the architecture are documented. The following types of assessment tasks occur during the architect phase:
  - Assessing the technologies used
  - Assessing processes used
Assessing people skills needed

- Designing and architecting a migration solution, which includes the following types of tasks:
  - Identifying the degree of change required
  - Identifying service level goals
  - Documenting design goals
  - Creating a component and technique map
  - Refining high-level designs
  - Creating a transition plan
  - Developing a configuration management plan
  - Creating a system I/O map
  - Creating an acceptance test plan
  - Planning test strategies
  - Prototyping the process
  - Designing a training plan for the new environment

**Build & Implement Phase**

The build and implement phase of a migration project is the active development of a new environment during which the following types of tasks occur:

- Porting applications to the new operating system, which includes the following types of tasks:
  - Creating a target build environment
  - Building a new application for the target platform
  - Deciding whether to support backward compatibility

- Migrating data to the new environment by transferring or converting it. This step includes the following types of tasks:
  - Transferring data
  - Transforming data

- Creating the production environment, including the following types of tasks:
  - Building the production facilities environment
  - Building the production platform
  - Building the application infrastructure

- Testing the migrated environment, which includes the following types of tasks:
  - Building the test environment
  - Creating the test plan
  - Performing unit testing
  - Performing regression testing
Mainframe Migration Strategy

- Performing integration testing
- Testing performance

- Refining the migrated solution
- Training end users and staff

The key deliverables of this phase include working components and documentation. In addition, the implement phase might possess an iterative structure and might include prototyping. In particular, prototyping is useful for data migration and difficult or risky conversion techniques. The implement phase can be iterated to allow incremental development. This allows a number of drivers to influence staging. A migration solution may be developed incrementally because different management environments are being migrated in different stages, with training done before the quality assurance installation; or it could be that the solution’s architecture permits a component-based delivery or that the phasing might be based on project or business resource constraints.

Manage & Support Phase

The manage and support phase is about sustaining systems and applications at runtime and includes an amount of post go-live, onsite support and mentoring from key members of the Transvive project team to ensure a smooth hand off and transition and the continued success of the organization in maintaining systems moving forward.

Migration is fundamentally an implementation exercise, although the following management tasks are typically within the scope of a migration project:

- Assessing the current IT management infrastructure
- Addressing the critical gaps
- Extending the infrastructure to account for the migration

In some cases, this phase may focus on service delivery and support management to address other issues, including:

- Service level management
- Financial management
- Capacity management
- Availability management
- IT service continuity management
- Service desk
- Incident management
- Problem management
- Change management
- Release management
• Configuration management

Obviously, the manage and support stage starts when migrated code is placed into use. The management regime needs to be designed or the current management environment needs to be confirmed as being appropriate for use after the migration goes live.
Assess & Design

Assessing the Environment
Before undertaking any migration project, a clear understanding of the environment to be migrated is a necessity. This can be a problematic task, often difficult to understand the full scope of the application composition and the associated supporting infrastructure, as well as the various interconnects and dependencies that might exist.

The goal of assessment is to ensure that all assumptions made during the project justification are proved and that all of the requirements and dependencies for the architecture are documented. This documentation is the deliverable for the project assessment of a migration project.

Assessment tasks must be performed thoroughly. Failing to uncover significant dependencies that could jeopardize the project’s viability can easily lead to expensive architectural revisions. On the other hand, the assessment must not be too detailed. Wasting time on minutiae that are not relevant to the migration can reduce project momentum and negatively affect the return on investment.

To provide complete documentation of the current environment, the following major topics must be addressed:

Assessing Technologies Used
When assessing the technologies used in an existing environment, focus on existing applications, the application infrastructure and the platforms on which the applications run.

Assessing Processes Used
In addition to assessing the applications and platforms affected by a migration, existing and new processes must also be assessed in order to ensure that the migrated application will integrate into the procedures executed by the enterprise.

Assessing Skills Requirements
A key input into the selection of any migration strategy will be the skills of the existing IT staff. The existing skill level of the staff must be assessed and their abilities should be an input to the migration strategy decision. The assessment should focus on the skills at all levels in the organization: operators, administrators and developers.
Designing and Architecting a Migration Solution

When designing the target technology platform, the degree of change required by the enterprise must be understood. Changes might be in business functionality or more commonly, they might be changes in the service level. Within the context of this methodology, the design stage addresses issues of platform technology, process and people. The following tasks are involved in the design and architecture of a migration solution:

- Identifying the degree of change required
- Identifying service level goals
- Documenting design goals
- Creating a component and technique map
- Refining high-level designs
- Creating a transition plan
- Developing a configuration management plan
- Creating a system I/O map
- Creating an acceptance test plan
- Planning test strategies
- Prototyping the process
- Designing a training plan for the new environment

Build & Implement

With a successful prototype of the target environment in place, it is time to implement the migration. The steps involved in migrating the current environment to the target environment can include the following:

Porting an Application to a New Operating System

Porting is an activity of moving a custom application, written in one or more programming languages, to a new operating system (OS). This activity relates primarily to creating a new executable that is compatible with the new OS. This is accomplished by modifying the application programming interfaces (APIs) used by the software so that they conform to or match the APIs offered by the new OS while maintaining similar functionality.

The first step in porting an application is to identify which APIs are incompatible with the new OS and need to be rebuilt. Once an API is flagged as being incompatible, a solution for a compatibility library must be developed. This solution must be one that can be used to replicate the functionality that existed on the old platform. The source code of the application can then be modified to use the new API, as defined in the compatibility library, wherever the older incompatible version occurs.

Creating a Target Build Environment

When migrating custom-written applications, the development environment must also be migrated to the new target environment. The development environment is composed of the development tools, the source code for the application and
any third-party products that are required to create the application. It also includes the hardware to support the developers who will use the tools to create the application executable and the supporting application infrastructure. To migrate the build environment, perform the following tasks:

- Prepare the hardware environments, including backup and restore facilities
- Identify the software to use in the new environment
- Acquire a recent reference build log
- Plan to acquire and install tools and utilities

**Building a New Application for the Target Platform**

Once it is verified that the development environment is functioning correctly, the application build can commence. Much of the information about how an application should be built can be obtained from the make files and the reference build log that were created when the application was built on the old system. Building the application results in the creation of an application executable. The functionality provided by the application can depend on a number of different elements of the build process. The output of the build process usually depends on one or more of the following items:

- Header files
- Compiler options
- Symbol resolution
- Variable initialization
- Conditional compilation
- Precompilation

Key elements of the new application build include the following:

- Creating a Compatibility Library
- Modifying the Make Environment
- Understanding the Application Configuration
- Deciding Whether to Support Backward Compatibility

**Migrating Data**

With the applications taken care of, it is now time to handle the most delicate part of the migration implementation: data transfer or conversion. While most organizations don’t realize it, data migration is one of the most important parts of the migration exercise. Depending on the application being used and outage window, this activity can range in difficulty and effort from trivial to extremely laborious.

**Transferring Data**

The goal here is to get data from the legacy server onto the new target environment. While the physical format of the data might change, the logical
Mainframe Migration Strategy

(application) format of the data will stay the same. The difficulty of transferring data is determined by three factors: size, transfer window and rate of change. Looking at all these factors, a strategy for getting data to the desired location should be developed.

The size of data is the most obvious factor in data transfers. Size, in itself, is not a problem, but size and a small transfer window can create one. Problems caused by size are not limited to bandwidth. Staging and backing up data in a timely fashion can also be problematic.

The other variable involved in determining the difficulty of a transfer involves the data’s rate of change. Static data and read-only application data can make transfer much easier by allowing segmentation of activities. However, data that constantly changes, such as transactional databases or real-time data acquisitions, might be very difficult to transfer.

Regardless of the level of difficulty involved in the data transfer process, there are a number of decisions to be made. The most important of these is the general strategy for transferring the data. There are two common methods for transferring data:

- Network data transfers
- Media data transfers.

Regardless of the method chosen for transferring data, there is a basic methodology to be followed. In general, this process includes the following tasks:

- Plan the transfer process
- Perform functional testing
- Conduct performance testing
- Implement the transfer process

Following this process should ensure a successful data transfer.

Transforming Data

Sometimes there is more to contend with than just the physical transportation of the data from one platform to another. Data may need to be converted or transformed in some way to work with the new platform, application or both.

Data transformations will follow much the same process as the data transfers above: however, extra steps will be needed to process the data. These extra steps, usually called staging, can range from simple mapping to complete rekeying.

Data transformation tend to fall into three categories:

- Encoded data transformations
- Application transformations
Mainframe Migration Strategy

- Database transformations

Creating the Production Environment

The term “migration” implies the movement from one environment to a different environment. The new environment is referred to as the target environment. The application is only one component of the target environment. The facilities to support the new compute, network and storage platforms must be prepared. The platforms must be installed and configured and the application infrastructure that will support the application must be created. These activities should proceed in a parallel fashion with the migration of the application, to reduce the amount of time required for the migration effort.

Building the Production Facilities Environment

Modifying a building or the data center is a significant task. Depending on the changes required, this can take a great deal of time. Care must be taken to minimize the disruption to the existing IT operation when bringing in new power, knocking down a wall or increasing air conditioning efficiencies. Inspections and building permits might have to be arranged for and acquired in order to make structural modifications to real estate. In addition, new facilities might have to be created or leased to support the new environment. Heating and ventilation additions might not be feasible in some older environments and building design might limit the amount of heat that can be dissipated. Any limitations on the generating capacity of the local power utility should also be considered.

Building the Production Platform

Once the facilities are in place, the compute platform can be deployed. This platform might be integrated with existing storage and networking components or it might require its own storage and networking subsystems.

If existing storage is to be used, ensure that it integrates with the new environment at both the hardware level (fiber channel, SCSI and the like) and at the software level (file system type, volume management and so on). When using existing storage, be extremely careful to ensure that no data on the existing storage is lost or corrupted. If the platform is to use a new storage platform and networking gear, the impact on the existing environment can be minimized.

Once the storage platform has been configured, extensive testing must be conducted to verify that the system is performing correctly. These tests should be conducted with utilities supplied by the compute platform or storage vendor, not the application. It is critical that the newly installed storage be validated before data is installed.

Network infrastructure is usually easier to test and configure than are storage platforms. All switches, routers, hubs and load balancers must be correctly installed and configured. Sufficient loads for these components must be generated to verify that they are performing as intended.
If a larger, enterprise-class machine is used as the compute platform, the compute platform itself might require extensive configuration. Frequently, large enterprise class machines can be chosen for the new production platform because of their availability, scalability, dynamic reconfiguration and hot swap capabilities. In certain cases, TCO can be reduced and ROI improved by partitioning a large symmetric multiprocessor machine into smaller virtual machines.

These activities can and should take place while the application is being migrated. The production platform build is independent of the migration activity.

**Building the Application Infrastructure**

Once the platform is installed and the OS is up and running, configuration of the infrastructure for the application can commence. All the software used to support the enterprise and the application should be available in the new environment, including the following items:

- Directory name services
- Identity services
- Accounting packages
- Management infrastructure
- Backup facilities
- Third-party runtime products required by the application
- Databases

These products should be tested as standalone utilities whenever possible before the installation of the migrated application.

**Testing the Migrated Environment**

The value of testing or quality assurance (QA) cannot be overstated. Typically, QA is the last part of a migration activity or product release cycle, to test and verify that the system does what it is supposed to do. Various types of testing can be performed and the testing methodology can be extended so that it is not an activity that is taken on at the end of the project, but one that is integrated into the project from the onset.

**Building the Test Environment**

In addition to creating development and production environments, a testing environment is also required to support the activity of verifying that the migrated application functions as planned. Testing is rarely permitted in a production environment and the development environment is usually reserved for the software developers who are producing the new application and may not have all the tools or the capacity to support the tests required before an application is put into production.
As with product development, migration activities must conclude with testing to ensure that the application meets its requirements. It is important to carefully plan for the acquisition of additional hardware and software to test the application once the new solution is put into production.

The test environment requires all the same supporting software that is required in the production environment. One approach used to increase availability calls for the purchase of an identical system that might or might not be in a clustered environment. However, it might be possible to execute regression, unit or correctness testing on a smaller machine with limited capacity.

**Creating the Test Plan**

The types of testing that should be performed vary depending on the environment that is being migrated. When developing a test plan, Transvive uses the following resources:

- Old test cases
- Requirements documents
- Functional specifications
- Business process flows or definitions
- Change documents (customizations/enhancement/patch levels)
- Training documents
- Consultations with users of the application

**Performing Unit Testing**

Unit tests are typically small tests that verify the functionality of a class or function used during the creation of a larger application. These tests should be written before the implementation starts to be coded. Unit tests can be run in a batch mode and frequently have automated results analysis, meaning that the tester is informed of exactly what case failed, thereby enabling rapid verification of changes.

**Performing Regression Testing**

Regression testing ensures that an application’s functionality has not changed (other than as intended) after modifications or updates have been made to the environment. This is usually thought of as end-to-end testing. Typically, when software is tested, regression tests are optimized to examine only those components that have been changed. In the case of a migration, there can be no such optimization because significant changes will have been introduced to the entire environment. All aspects of the business process should be tested in a regression test. A regression test must not only ensure that the correct results are achieved but also that the operational documentation (run books) is still accurate.

**Performing Integration Testing**

Integration testing must be performed on hardware and software. When the hardware has been installed, optimal functioning must be ensured. We recommend that tests be conducted on the individual components (compute
platform, storage platform and network facilities) to verify their operation and limitations. For software testing, the integration of different software components can produce erroneous results. Wherever possible, interaction with third-party products and packages should be verified during the migration effort.

**Testing Performance**

Once it is verified that the migrated system is producing the correct results, test to verify that it is performing optimally in the new environment and verify that the new environment can support the SLAs required by the enterprise. This will involve stress-testing the application by applying differing workloads and measuring how the application responds in terms of throughput, latency, memory utilization and processor loads.

When possible, the migrated environment should be tested in parallel with the old implementation to ensure the veracity of the new implementation as the application might function correctly, but not efficiently. Whenever possible, record and compare measurements of performance on the old system in terms of system metrics, as well as application metrics, with similar metrics in the new system.

**Training End Users and Staff**

Although we have listed training as the last task of the implement phase, it should be addressed throughout the entire implementation, if not earlier. Training, whether in the form of informal on-the-job training and knowledge transfer or formal classroom lectures, is a vital ingredient for the successful completion of a migration project and the future service levels of the environment. It should be treated with the same level of assessment and planning as the code porting or data transformation.

Make sure to address the following strategic areas of learning to ensure a successful migration project:

- Awareness
- End-user training
- Platform training
- Process training

**Manage & Support**

The following management tasks are within the scope of this document:

- Assessing the current IT management infrastructure
- Addressing the critical gaps
- Extending the infrastructure to account for the migration
The new IT environment components must be integrated into the existing IT management infrastructure. The approach taken and the degree of effort involved with it will depend on the current degree of operational capability the organization possesses. A generalized strategy for addressing operational readiness should include these steps, which should be focused on the requirements and benefits case defined during the justification activities and the architect phase.

**Assessing the Current IT Management Infrastructure**

As a first step to ensuring that they can support the migrated environment, organizations undertaking a migration must develop a realistic understanding of their operational capability. Review the available standards and frameworks, looking for opportunities to leverage them as much as possible. A management framework and a maturity model should be adopted as the basis for both the assessment effort and subsequent improvement activities. A management framework describes what needs to be in place. A maturity model defines the evolutionary path for realizing the framework.

**Assess People Requirements**

When assessing the human aspects of a management solution, we focus on the skills that will need to be developed and exploited to manage and control the new environment. Obviously, new skills will need to be adopted and absorbed, which puts a focus on training and development processes that will enable the development of the appropriate competences to successfully control the new environment.

When significant changes are introduced, it is essential to emphasize communication and coordination between workgroups to limit the resistance to change and to increase the chances for success.

**Assess Processes**

To be proactive, evaluate the following processes and bring them to an appropriate level of maturity:

- Change management
- Problem management
- Implementation management
- Execution management

Obviously, the ability to introduce changes into the environment with minimal risk and at an acceptable cost is key to the successful migration of any technology. If the change management process is implemented successfully, this can be achieved.

For execution management (often referred to as operations management) to become more proactive in accepting and introducing changes, the following key areas should start to be addressed:
- Production control
- Resource administration
- Resource planning
- Service recovery

The ability to manage problems and incidents quickly and consistently is essential to avoid solving the same issue more than once. The focus of problem management, when shifting from reactive to proactive, must move to root cause analysis to enable the most effective solutions for existing issues.

**Assess Tools**
In moving to a more mature level of operational capability, an organization shifts from technology that is focused on monitoring the lower portions of the E-stack to technology that extends the monitoring coverage and facilitates proactive management of the IT environment. In applying the tools solutions model, Transvive would focus on the following items at each layer:

- Element and resource management
- Event and information management
- Service level managers
- Process workflow managers

**Perform Audits**
To determine the current state of IT operational capability, some type of audit should be conducted. Audits generally fall into one of two categories:

- Compliance audits
- Effectiveness audits

**Addressing Critical Gaps**
Improving operational capability is an organization-wide effort. Moving up the maturity scale requires the application of resources (time, skills and money) and the cooperation of the entire organization. Efforts to improve operational capability require senior management commitment.

Improving operational capability is an evolutionary, not revolutionary, activity. Various maturity models for IT operations can communicate the goals of capability improvement activities and define an incremental approach to realizing those goals. Organizations should focus on incremental activities with a quick return that are conducted within the context of a well-defined strategy. The “big bang” approach to building operational capability is strongly discouraged.

IT management is a process-driven activity. Often organizations react to issues of IT management by acquiring technology to manage the environment without considering the processes needed to operate the environment. The initial focus of efforts associated with improving operational capability should be on the definition and implementation of the processes to be used in managing the
environment. The process architecture should drive the tools and skills architectures.

To be successful, organizations must measure progress. Improving operational capability requires organization-wide commitment. As part of meeting that commitment, meaningful metrics to measure baseline capability must be used before starting improvement efforts. Then, periodically evaluate the effect of the improvement efforts. Investments in operational capability should be justified by corresponding improvements in key performance metrics. Established continuous improvement methodologies are great tools for enabling these projects of change.

**Selecting Tools for Managing the Migrated Environment**

The tools solutions model described above implies the application of sound systems design principles to include modular design, separation of function and well defined interfaces. As a result, the management tools architecture should be loosely coupled with the corresponding managed environment. The degree of dependency between the two architectures decreases the farther up the management framework.
Migration Project Planning

With a successful project justification and approvals in place, building a vital infrastructure that will make the migration successful is the next step. This section details the tasks involved in planning a migration project. From a high level, building the infrastructure involves the following tasks:

- Defining scope
- Understanding risks and planning contingencies
- Organizing the project team
- Closing the Project

Defining Scope

As with any complex project, scope must be tightly defined and zealously controlled throughout the project. Scope creep is the number one threat to any migration project. Migrations are complex projects when they have a proper scope and additional challenges can mean the difference between success and failure.

In addition, however, the scope should not be too tightly defined; shortcuts and omissions that occur during the architect phase will be uncovered during the implement phase, often resulting in a need to adjust the scope of a project. This flexibility to adjust scope, but only when absolutely required, can be the difference between the overall success and failure of a migration effort. Deviations from a scope should be fully documented with appropriate risks, mitigation procedures and contingency plans as outlined below.

Understanding Risks and Planning Contingencies

Risks are inherent in any IT project. However, migrations tend to be considered the most risky because of the number of interdependencies they involve and the amount of change they produce. To minimize risk in migrations, a risk management technique should be followed. While many different techniques exist, most include the following components:

- Risk identification
- Risk estimation
- Risk evaluation
- Mitigation and contingency plan creation

By choosing a risk management technique that has each of these components, risk can be actively monitored and controlled.
Because of the risky nature of migration projects, it is especially important to plan for contingencies in case the project runs into trouble.

Risk is decreased with knowledge gained through an assessment of the migration opportunity. The amount of effort expended during an assessment should be directly related to the level of risk the organization is willing to accept and the associated costs it is willing to pay, both in terms of labour and organizational disruption caused by the assessment activity itself.

This section explains how to identify and estimate risk in a project and how to evaluate that risk. It then presents a common risk management technique and its application to migration projects.

**Identify Risk**

All risks need to be stated so that they can be evaluated. There are generally two types of risk: business risk and project risk. Business risks prevent the project’s ability to deliver the desired business benefits. Project risks jeopardize the project from delivering its objectives.

**Estimate Risk**

Each identified risk must be assessed. This assessment measures the possible repercussions to the business or project from the identified risk. Each identified risk must also be categorized by the severity of its impediment to the success of the project. This too will be a numerical index representing a possible problem.

**Evaluate Risk**

After identifying risks and estimating their impact and severity, it is important to evaluate and prioritize the risks to the project. This task allows concentration on risks that pose the greatest threat to the project. Most methodologies prioritize risks according to the product of their numerical impact and severity ratings.

**Create Mitigation and Contingency Plans**

Once all the risks have been identified and prioritized, mitigation and contingency plans must be created. Mitigation plans reduce the risk of the problem occurring. Contingency plans outline the project’s recourse if the problem occurs. Some of these plans should be built into the best practices of the project itself. Contingency plans are formed at different levels, depending on the magnitude of the risk. The risk and contingency methodology should be refined and updated during each successive phase of the project.
Organizing the Project Team

The final piece of the migration project plan is the organization of people, their relationships and reporting structures. A typical project organization is composed of the following elements:

- Project sponsors and Project board
- Project office
- Quality office and Project assurance
- Testing team
- Development and Porting team
- Data Conversion team
- Infrastructure team
- Implementation team
- Operations team

Identify Resources

After detailing tasks and durations in a project plan, the resource profiles required to complete the tasks should be identified. Resource profiles are sets of skills that are required to accomplish groups of tasks within a project plan. Once all of the resource profiles for a particular project are identified, people with these profiles can be assigned to individual tasks and begin creating a project plan.

Developing a Project Plan

While every migration project will be different in objectives, scope, difficulty and level of effort, some standard steps and rules of thumb can be applied to any project plan.

Architect a Migration Solution

Architecture is the first stage of the project that we need to tackle. Transvive’s architecture methodology is divided into three major areas: assessment, design and prototyping. Each of these areas is aimed at advancing the solution toward a new design.

Assessment Activities

Assessment activities in this stage are straightforward. We need to investigate each layer of the E-stack within the solution and assess what likely changes are necessary to meet our goals of rehosting on a new OS. The output of each of these tasks will be a summary of the current situation, detailed information about key configuration items and an estimation of the level of effort required to move to a new environment.
Design Activities
Once the assessment has been completed, it is time for a review of the data gathered during the assessment and the development of an initial design. The review is a critical part of the design, whereby the raw data is turned into usable information by identification of key requirements within the gathered data.

The initial design takes into account the requirements gleaned in the previous task and matches them with technologies in the key systemic qualities. This combination of technologies becomes the initial design of the migrated solution.

While each company will have different processes, methodologies and standards for architectural design, the deliverable should be the same: an architectural report detailing the requirements and all components of the solution.

Prototype Activities
The next activities in the architecture provide a proof of concept for the initial architecture. This is an optional step, depending on the level of risk and the project’s mitigation plans. Many migration projects will include this stage and will result in the construction of a scaled-down version of the initial solution that can provide the functional validity of the solution.

Implement the Migration Solution
With the architecture stage completed, it is time for implementation. In the Transvive’s Architecture Methodology, implementation is divided into many areas:

- Application redevelopment and porting
- Data conversion and movement
- Building
- Testing
- Training

Each of these areas is crucial to successfully completing a migration.

Application Redevelopment and Porting Activities
In migration projects that rely on the porting or redevelopment of applications, this stage receives the most attention. Other migration projects, such as the migration from custom applications to COTS solutions, might not address it at all. These tasks include the process of developing, customizing or porting applications so that they execute in the new environment. The key activities here include the following:

- Creating the new build environment
- Modifying the application for the new build environment
- Compiling and building the application for the new environment
- Unit testing the new application
These activities vary greatly depending on complexity. However, the key criterion in this level of effort is the application or porting complexity.

**Data Conversion and Movement Activities**
Similar to the developing and porting activities, this set of tasks readies the data for the next environment. However, in addition to the possible conversion of data from one format to another, there may be substantial challenges in actually moving the data from one platform and application to another. This is especially true of databases or any other application that handles data outside the normal UNIX file constructs.

**Build Activities**
The build activities of the implementation stage are similar to those of any other project. Infrastructure must be constructed so that the applications can be hosted. Some of the environments that will need to be built include the following:

- Test environment
- Target environment
- Target platform
- Application environment
- Regression environment

**Test Activities**
One of the more important parts of a migration project is buried here in the middle of the project plan. Testing will make or break a migration project. Several types of tests need to be planned and executed:

- Unit testing
- Integration testing
- Validity testing
- Performance testing
- Transfer testing

**Training Activities**
The final activities in the implementation stage address the overlooked human aspect of migration projects: training. Several levels of training will need to be conducted to make sure that the newly migrated environment supports the business properly.

Although this activity set is discussed last, it should actually be started before the build tasks within this stage so that administrators have good knowledge of the new environment.
Manage the Migrated Environment

The final stage of migration is making sure that the migrated solution will meet the previous service level agreements (SLA). We do this by addressing the operation and maintenance procedures with the same approach as for the human and technology issues in the project. These are the key tasks in this phase:

- Evaluating operational procedures and tools
- Redeveloping subpar procedures and tools to meet expected performance
- Implementing new supporting tools

Between each stage of the project, these key project evaluation documents need to be updated and reassessed:

- Risks, mitigation and contingency plans
- Project actuals compared against the plan
- Review of key benefits and objectives validity

Closing the Project

After all implementation and management activities have been finished, it is time to disband the team and close the project. However, before this is done, a few key activities must be completed:

- Obtaining the customer’s formal acceptance of all deliverables from the project
- Evaluating the delivery of the output and solution against the measurable objectives outlined at the beginning of the project
- Capturing important lessons learned for the next migration project
- Filing all deliverables and documentation so that other parts of the organization can review the project

These activities effectively return control of the environment to the normal operations team, dissolve the project structure and feed the results of the migration to the project sponsors.
Contact

For more information about Transvive’s migration and modernization strategies and methodologies, contact us.

Transvive Inc
26 Soho Street, Suite 201
Toronto, ON M5T 1Z7
416-599-2900
info@transvive.com

www.transvive.com

About Transvive

For more than 10 years, Transvive’s professional services team has been providing mainframe solutions that significantly reduce operating costs and risks and increase efficiency and speed to market. Transvive has successfully led projects from simple GUI modernizations to large-scale system overhauls.