INFORMATION TECHNOLOGY ASSETS
MIGRATION GUIDE

Moving valuable applications and data to the cloud

Mobilize.NET
Your Code. New. Again.

WHY MIGRATE?

Entropy is the thermodynamic principal of the universe's unstoppable motion from order to disorder.

Information technology (IT) assets have their own entropy as they age; this entropic decay can be thought of as the legacy problem. Physical assets like laptops, monitors, and servers show their age visually—what looks more dated than a CRT monitor on a desk?

The legacy problem

Software and data, on the other hand, may show fewer visual clues of their age unless one looks under the hood at the source code of applications or scripts and schemas of a database or data warehouse. Legacy assets may look fine on the surface but be crumbling below.

Legacy assets are expensive, risky, and dangerous

As IT assets age into the legacy stage, they may continue to be perfectly functional yet create undesirable side effects, including:

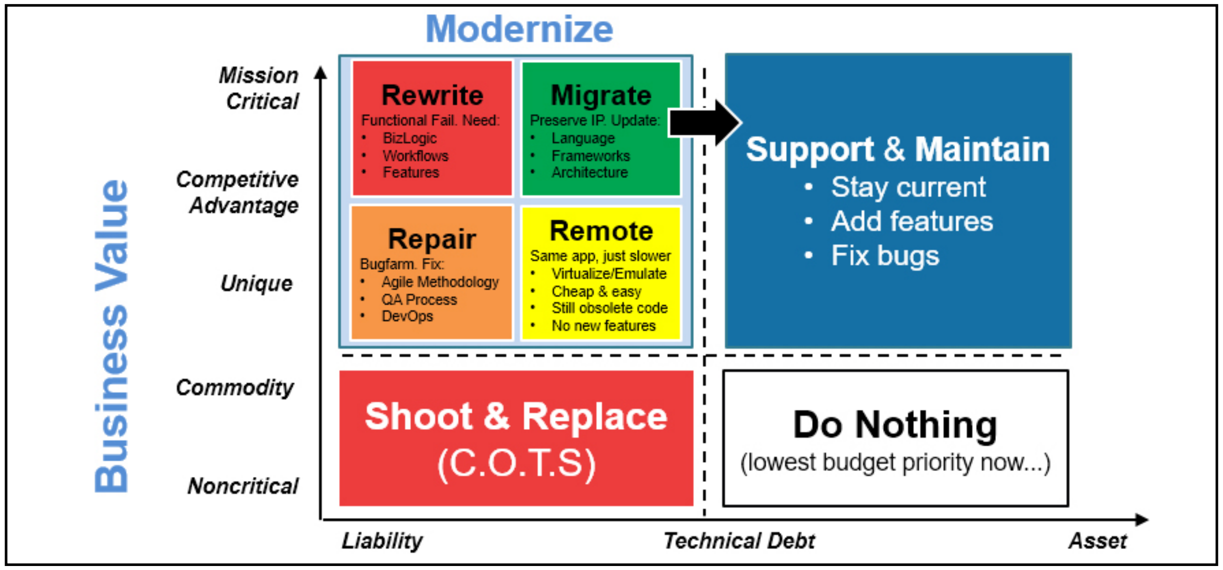
- Human resources cost more: as developers trained in old programming languages and tools age out of the workforce, replacement costs soar and ongoing maintenance of apps becomes expensive.
- Older applications and database systems frequently reflect security models from their origin date, providing inadequate protection against modern, more sophisticated attack vectors.
- Systems based on out-of-vendor-support platforms are at risk of sudden failure when OS updates or patches are deployed.
- Similarly, out-of-support components and dependencies can place the organization into non-compliance with IS/IT governance requirements.
- Finally, legacy systems can block the organization from implementing modern, efficient engineering practices.

Should you migrate your assets?

Deciding how to deal with legacy assets implies a methodology to evaluate each asset—both applications and data assets—in terms of both business value and technical quality. Gartner's TIME approach puts applications into four buckets: Tolerate, Invest, Eliminate, or Migrate. Similarly, the Mobilize.Net Portfolio Analysis process uses similar quadrants: Modernize, Support & Maintain, Shoot & Replace, or Do Nothing.

Solving the stack problem.

Migration using automated tools addresses a specific problem set with both application and data assets: how to update the technology stack of assets that are providing real value but are based on outdated technology. As a general rule of thumb, applications or database systems that are working well and providing value but are based on out-of-support or obsolete components, languages, OS versions, or libraries benefit from automated migration to new technology. Assets that are also technologically lagging but are also not providing value are candidates for rewrite.



PLANNING THE MIGRATION

A successful migration project begins with a well-crafted plan. Thorough planning can help reduce or eliminate unforeseen problems and roadblocks down the road.

Recommended migration planning process

1. Inventory the application and data asset portfolio. This preliminary step can be time consuming but is extremely vital to the modernization effort. Capture metadata such as owner, technology stack, purpose, user list/description, business value rating and technical quality rating.[1]
2. Classify each asset according to your quadrant methodology. Perhaps the most important category is the "sunset" or "retire" quadrant; one large IT organization was able (following an exhaustive inventory process) to retire approximately 30% of their custom applications, replacing them with low-code or no-code alternatives. Carefully consider whether to migrate or rewrite assets which are still vital but suffer from technical debt or obsolete stack. Rewriting promises enhanced functionality and customer value but is fraught with risk for both budget and schedule.[2]
3. Prioritize projects based on a value to cost analysis. Value includes business function, size of user base, and potential risks of doing nothing. Costs analysis should include data- and experience-based resource and time estimates as well as opportunity costs (for example, if your best and brightest are rewriting legacy applications, they are unavailable to add customer value to more important apps).

1 While it's important to inventory and classify physical IT assets (desktops, servers, laptops, etc) that's beyond the scope of this paper.

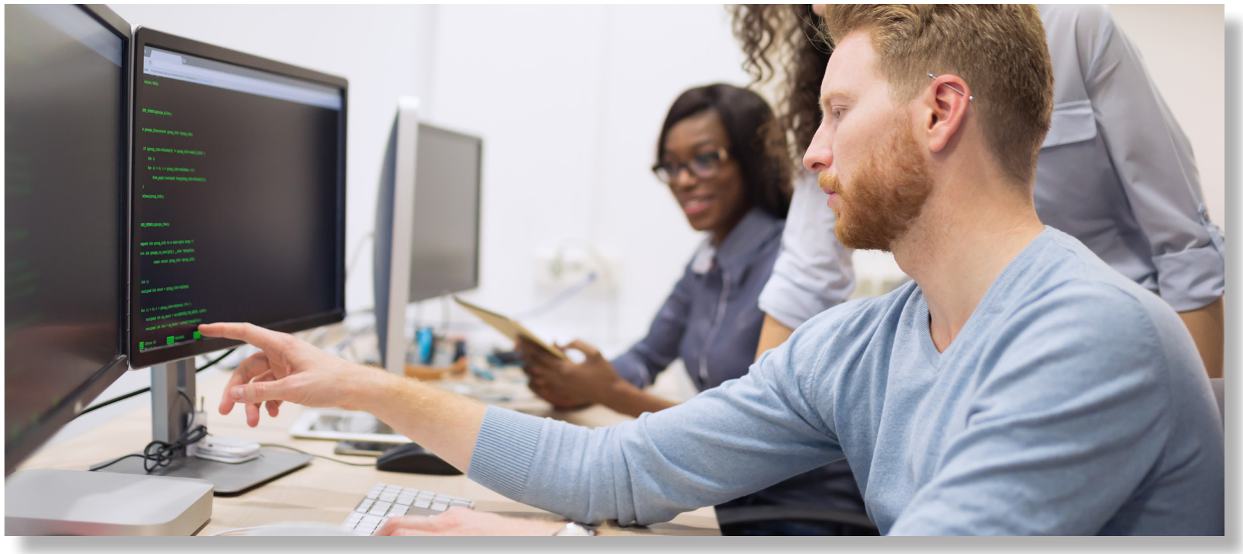
2 https://www.atspoke.com/blog/it/reasons-for-it-project-failure/

The Mobilize.Net Migration Blueprint

With over 20 years experience migrating application and database solutions, Mobilize.Net has developed a proven migration planning process: the Migration Blueprint. This one-week on-site engagement by senior engineering staff identifies critical issues with the migration project including:

- Current and planned system architecture
- Existing components, dependencies, and libraries
- Test plan including developing test cases for both existing and planned applications
- Project breakdown including milestones, resource requirements, manual effort required, and new engineering/development where needed.

The output of the Migration Blueprint is a detailed plan highlighting outstanding issues and recommendations for remediation. The level of detail is sufficient for an internal team to fully plan resource and schedule requirements, or to get a quote from an external vendor for a turn-key project delivery. Following the Blueprint project, many of Mobilize.Net's customers engage us to deliver a completely migrated application, including full visual and functional equivalence, as evidenced by passing a rigorous set of functional tests.



MIGRATING CODE AND DATA

Migrating code and data to the cloud opens new value possibilities and can reduce costs. Automated tools make it both efficient and low-risk.

Migrating custom applications

As discussed above, the first step in any migration project is a well-formed plan. Following the planning phase, the organization can focus on the resource or engagement model that best suits their specific situation.

Based on organization-wide IT priorities and available resources, management can determine what level of engagement makes sense for each migration project, from full ownership to complete out-sourcing.

Mobilize.Net has developed several customer-facing engagement models for application migration:

License only

Mobilize.Net provides licensed tools and technical support for the customer organization to perform all the migration work themselves. Best for organizations with experience in both the original and destination technology platforms and languages.

Compilation

Mobilize.Net migrates the customer code to the target platform and gets all the code to the compilation state. Some code gets stubbed out; this engagement model is best for organizations that have limited technical resources and want to reserve their staff for the last mile work.

Visual Equivalence

Mobilize.Net migrates the code, eliminates compiler errors, and updates the target code until all visual UI elements will render. Best for organizations who want to reserve their in-house resources for the final test and bug-fixing stage.

Functional Equivalence

Mobilize.Net migrates the code, performs all last mile work, and debugs the code until it passes the full suite of customer-provided test cases. Normally these tests are developed using the source application for the reference specification; functional equivalence is designed to be faithful to the UI and functionality of the original application.

For more information on the migration process, see "Choosing a migration partner" below.

The cloud is the goal

Following the advent of the personal computer in the 1980s, organizations large and small found these simple and affordable devices an ideal replacement for more expensive mainframe and mini-computer workstations, and the era of client/server applications development was born. In the 1990s, the availability of simple programming tools like Visual Basic and PowerBuilder made building front-end applications for database servers cheap and easy. Later, the release of Microsoft .NET further simplified creation of rich desktop applications.

The appeal of hosting applications in public cloud services like AWS or Azure was given a boost during the pandemic in early 2020 when workers suddenly were forced to do their jobs from home offices; many of these mission critical desktop applications were captive to a now vacant office. Organizations reacted by adopting a “lift and shift” approach to move those workloads into the cloud so workers could continue to access them. However, this lift and shift approach is only a partial solution, and doesn’t address the real issue of platform obsolescence at all. It simply moves the problem from on-premise to public cloud.

The real advantages of cloud deployment—DevOps, CI/CD, elastic load balancing, real time monitoring, containerization, and more—requires cloud native applications, not just desktop apps hosted in a virtual environment. Migration projects that do not re-platform desktop apps to be cloud native are wasting an enormous opportunity.

Migrating databases and data warehouses

Just as cloud deployment offers many benefits to applications, it offers equal benefits to data assets as well. Moving the data in any data warehouse (Teradata, Oracle, SQL Server and so forth) to the cloud is trivial with often the biggest challenge being the bandwidth necessary to push terabytes of data up to the provider—alternatively, data can be copied to external hard disks which can be shipped directly to Amazon or Microsoft.

A harder problem is the migration of on-prem data warehouses with complex scripts and stored procedures to a cloud environment. Snowflake® is making rapid gains moving customers from more traditional systems like Teradata and Oracle to their Data Cloud.

Mobilize.Net is a key migration partner for Snowflake, having built tools to unblock Teradata and Oracle customers from adopting the Snowflake data cloud. Our technology—similar to how our application migration tools work—parses SQL (DDL, DML, and everything else), PL/SQL (sprocs, macros, functions, etc.), and proprietary scripts (such as BTEQ and all its extensions in Teradata)—and generates Snowflake SQL, Python, and JavaScript fully commented code.

Automated code conversion using Mobilize.Net SnowConvert dramatically reduces the time, cost, and risk of moving from traditional data warehouse systems to Snowflake, making the eventual savings even more attractive.

CHOOSING A MIGRATION PARTNER

Migration projects fall into the category of large initiatives that are both *infrequent* and *consequential*.

Infrequent because most organizations have a sufficiency of experience both creating new software and data assets and maintaining those over time. Migrating to a new platform, however—except in some rare situations—is not something done often enough for the organization to build experience and expertise.

Consequential because typically only critical, large, and complex applications and data systems are candidates for migration or re-platforming. As described earlier, these systems continue to offer real business value to the organization, but are held back by old or obsolete languages, platforms, and runtimes. Failing to move the system forward to a more modern technology stack could have serious negative consequences for the organization.

Thus the choice of a partner for a migration partner is a critical decision in achieving a positive outcome.

DIY Migration

The "do it yourself" approach is popular with organizations that have deep experience in developing software applications or database/data warehouse systems. Independent software vendors (ISVs) are prone to making this their default choice, as software development is their metier. The key to a successful outcome with the DIY approach is to bring highly automated tooling into the fold, allowing the efficiencies of computer aided software engineering (CASE) to be a vital part of the project methodology.

The organization pursuing a DIY approach to legacy migration should look at all the available tooling and preferably test different tool sets on their own code base in order to make accurate appraisals of the efficacy of each and estimates of the effort that will be required for the total project. The vendor's level of support and experience should also be a major factor in any decision.

The use of internal resources for a migration project should be considered carefully. While seemingly simple (re-create an existing application or data system using a newer platform, language, or runtime), in reality these projects can turn out to be highly complex. Many organizations are reluctant to deploy their "best and brightest" on legacy projects instead assigning "less than the best" engineering talent, further adding to the risk of the project missing key requirements, schedule, or budget targets.

Offshore Migration

More and more technology creation and maintenance is being driven to offshore providers in countries like India and Ukraine. Systems Integrators (SIs) are eager to take on these kinds of large data and application migration projects, sometimes using their own tools to speed up the migration process.

Organizations choosing to follow this path should carefully review the selected vendor's experience and track record in projects directly similar to the legacy system to be migrated. The offshore promise may be to replace experience and/or advanced tooling with low-cost bodies thrown at the project. Poor decisions by the offshore team due to a lack of expertise and relevant experience may lead to long-term issues and costs for the organization that owns, operates, and will maintain the migrated code.

Vendor-assisted migration

The final option is to draw on the service component of a vendor who specializes—ideally exclusively—in legacy migration tool development.

For over 20 years Mobilize.Net has focused solely on building highly automated, AI-assisted tools to move legacy source code to modern platforms, languages, and runtimes. Beginning with the Visual Basic Upgrade Wizard, commissioned by Microsoft to be a key part of Visual Studio in order to ease the transition for VB6 developers to the .NET Framework, Mobilize.Net has subsequently built highly automated tools to move a variety of desktop programming languages to cloud native, as well as more recently building tools for migration to Snowflake Data Cloud.

But in addition to tools, Mobilize.Net has worked with hundreds of organizations—encompassing F500, state and local government, ISVs, and global SIs—to deliver turnkey migration projects of systems frequently in the millions of lines of code (LOC). Organizations that recognize either they lack or decline to utilize internal resources for these highly-specialized projects are able to get assistance from Mobilize.Net for partial to complete system migration, including unique customizations to further automate conversion of workload-specific code patterns.

Additional information and resources are available in the Mobilize.Net Digital Transformation Starter Kit here: https://www.mobilize.net/resources/guides/digital-transformation-starter-kit