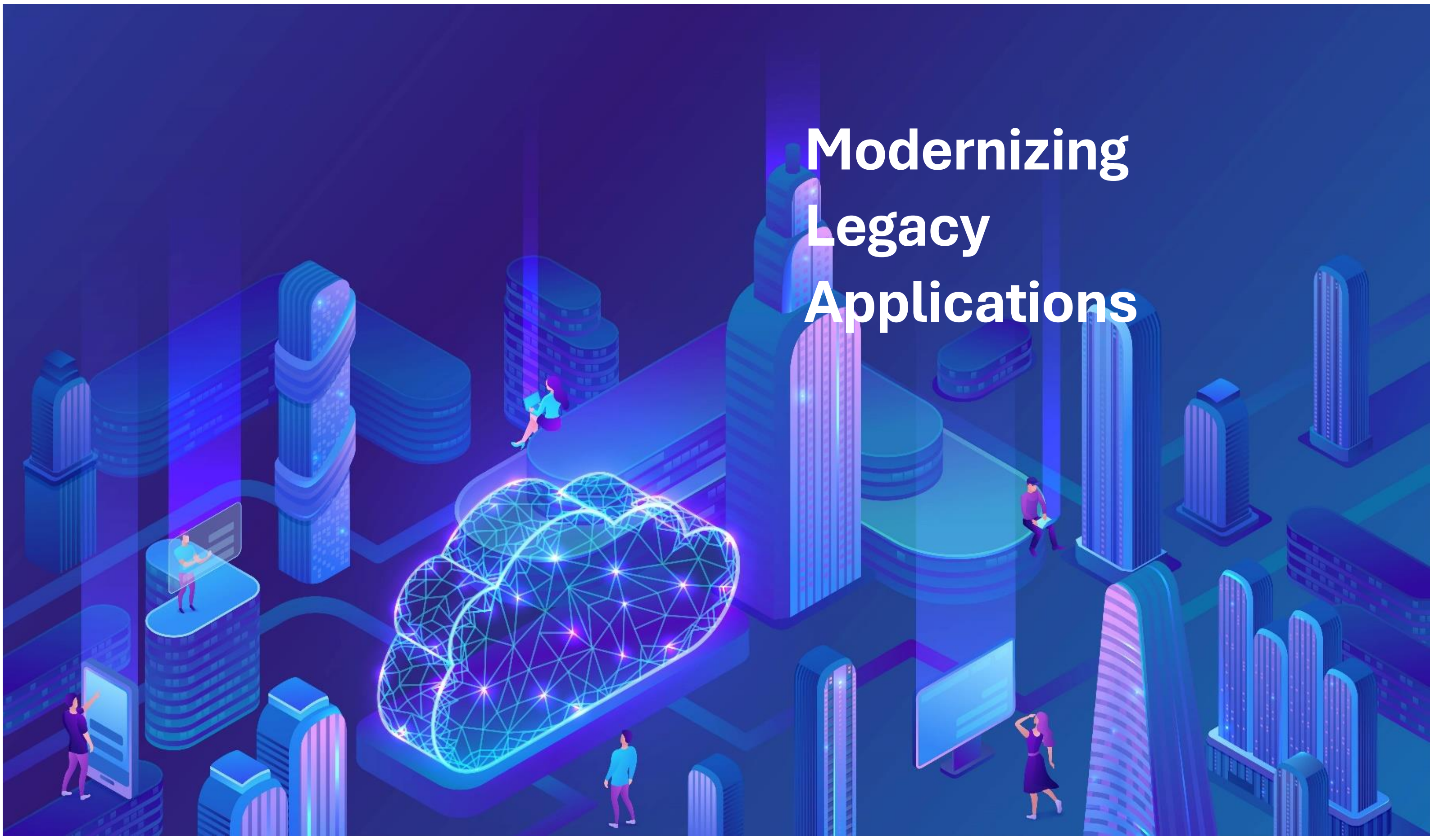
Modernization Strategy Paper



Introduction

This article is based on Gartner's articles on Application Modernization, including Choosing the Right Approach to Modernizing Legacy Systems and using Continuous Modernization to Optimize Legacy Applications. It also includes notable case studies and trading as a global consultancy.

This approach is designed to inform CIOs and executives about what to do with legacy applications, which are common and critical in both the public and private sectors.   We have often advised the government why we advocate for gradual modernization programs to remove pieces and encapsulate components, releasing them early and often over big-bang or rip-and-replace rewrites. These articles align with this strategy.

We also make a case for the government that “modernization” programs often be presented as CAPEX expenses, like any other modernization strategy is, e.g., big bang.

Both are methodologies to achieve a de-risk technology surface area and get better business value from the application portfolio managed by IT. If continuous modernization, with incremental change, driven by business pain points, delivers better outcomes, is more cost-effective, and is less risky than alternative strategies, Execs and Engineering Leaders should understand this; not focus on immediate ROI, and use effective methods to align business with IT, and employ continuous delivery, containerize where possible, and bring together platform and product teams.

70% of digital transformation programs fail to meet original expectations. How can we improve this success rate?

Choose the Right Approach to Modernize Your Legacy Systems

Each of the application modernization and cloud migration approaches varies substantially in terms of scope and effect. Software engineering leaders should use Gartner's evaluation framework to select an approach to produce the desired results with acceptable cost, risk, and impact.

Key Findings

- The approaches to application modernization and cloud migration vary significantly in scope, effect, risk, cost, and impact. Approaches have specific use cases, differing in which layers of an application can be improved.

- Software engineering leaders often pursue modernization or cloud migration efforts without determining why and how an application poses an obstacle to digital business - leading them to select an approach that does not resolve the underlying problem.

- Without a clear understanding of the root cause of the problem and the effect of different modernization approaches, software engineering leaders may select an ineffective approach that wastes time and effort and will negatively impact business outcomes.

Recommendations

Software engineering leaders should:

- Recognize the differences in scope and effect to understand each application modernization approach clearly.
- Determine what functional, architectural, and technological issues need to be resolved by performing a cause-effect analysis using the six drivers of modernization.
- Choose the right approach for each application by using Gartner's evaluation framework.

Introduction

The 2023 Gartner CIO Survey showed that organizations will 46% increase spending on application modernization (top 4 technology spend), and 50% will increase cloud platform spending (top 3 in technology spend). 47% will decrease investments in legacy infrastructure data center technologies, illustrating the transition to modern technology platforms.

Application portfolios deteriorate over time and gradually lose business fitness, innovation support, and agility while becoming increasingly expensive, complex, and risky. Software engineering leaders and enterprise application leaders can use various approaches to continuously improve the fitness and value of their applications so that they can better support changing business demands.

There is no one-size-fits-all approach to improving an application. Each approach - rehost, replatform, rearchitect, rebuild, and replace - serves a different purpose and varies substantially regarding effect, value, cost, risk, and impact.

When software engineering leaders choose the wrong approach, they will waste time and effort and fail to resolve the underlying problems in their application portfolio. The consequences of choosing the wrong approach include budget overruns, late delivery, low business benefits, business process disruptions, unmaintainable code, and early termination of the modernization initiative.

Which application modernization or cloud migration approach should you use? Software engineering leaders should evaluate all modernization and migration approaches and use Gartner's evaluation framework to choose the approach that solves the application's problem and meets the organization's acceptable risk, cost, and impact level

Application Modernization and Cloud Migration Approaches

The five main application modernization and cloud migration approaches vary substantially in scope, effect, risk, cost, and impact. Most modernization programs start with cloud migration, but there are exceptions.

Modernization approaches differ in their ability to change the technological, architectural or functional aspects of the application:

- Rehost and replatform allow changes to the technology platform on which the application is running. These approaches can solve problems caused by the technology.
- Rearchitect allows changes to the technology platform and the application's code structure. These approaches can solve problems in the technology and architecture domains.
- Rebuild and replace allow functions and features to be changed or added. These approaches can solve problems in the functional domain and provide the opportunity to change technology and architecture.

Besides the five main application modernization and cloud migration approaches, software engineering leaders may evaluate alternative options when modernization or migration is not worthwhile.

The choice to modernize/migrate or not is often made in a preceding application rationalization or application portfolio assessment initiative

Rehost

Lift and Shift Redeploy the application to other infrastructure (physical, virtual, or cloud) without recompiling, modifying the application code, or modifying its features and functions

Replatform

Lift and Reshape Migrate the application to a new runtime platform. It may include minimal changes to code for compatibility with the new platform but no changes to the code structure or the features and functions it provides.

Rearchitect

Reengineer, refactor, Restructure and optimize existing code without changing external behaviour to remove technical debt and improve nonfunctional attributes and structure. This can include a shift to a new application architecture or platform to exploit better capabilities, code transformation, or refactoring.

Rebuild

Rewrite, redesign Rebuild the application. This includes the possibility of creating a like-for-like replacement of the application and preserving its scope and specifications.

Repurchase

Drop and shop Replace the application with a COTS or SaaS solution, considering new requirements and needs.

Retire

Decommission the application, often as part of a replacement scenario.

Encapsulate

Capture the application’s data and functions and make them accessible as services via an API. This approach leverages and extends the application's features and value while hiding implementation specifics behind the interface.

Retain

Tolerate, and leave as is due to too much effort and risk, and application is nearing retirement

Determine What Functional, Architectural, and Technological Issues Need to Be Resolved

What are our main issues and pain points with the current application? What is the root cause of these problems?

Drivers for Modernization

Define what problems you are trying to solve. Refer to Gartner's 6 common drivers of modernization from business and IT. Use these drivers to evaluate the application and define the issues.

Business Drivers

From a business perspective, the three main drivers for application modernization are:

- Business fit: The application no longer meets current business requirements.
- Innovation: The application constrains the business from leveraging new business opportunities or addressing disruptions.
- Agility: The application and its supporting ecosystems cannot keep up with the pace of change, or those changes may come with an unacceptable level of cost/ risk.

IT Drivers

From an IT perspective, the three main drivers for application modernization are:

- Cost: The total cost of operating, maintaining, and changing the application is too high in relation to its business value.
- Complexity: The high complexity of the application creates various problems and is a major factor in maintainability as it impacts the time, cost, and risk of implementing changes.
- Risk: The application poses security, compliance, supportability and scalability risks. In older application platforms and languages, the risk of a skills shortage is often a major concern.

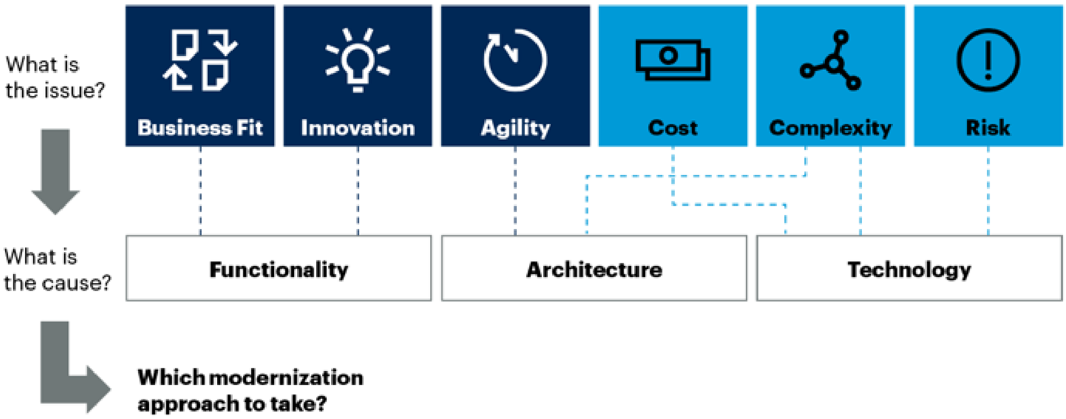
Combining business and IT drivers creates a turning point where application modernization is required. Define the main issues you are trying to solve in terms of these six drivers and add more detail to describe the issues or impediments clearly.

Identify the Cause of the Issues

The drivers can help to locate the probable cause of the issues. An application's issues can originate from three main layers: its technology, architecture, and functionality. In most cases, the main drivers for modernization point to the possible root cause:

- Cost, complexity, and risk issues are most likely caused by the technology used to build and support the application.
- The functionality most likely causes business fit and innovation support issues.
- The architecture and structure of the application most likely cause agility and complexity issues.

Identify Cause of the Issue as Basis for Selection of Approach



Evaluation Framework for Each Application Treatment

After identifying the likely cause of the application's issues, software engineering leaders can determine whether they need to fix its technology, architecture, and/or functionality aspects. This will be the basis for selecting the best approach.

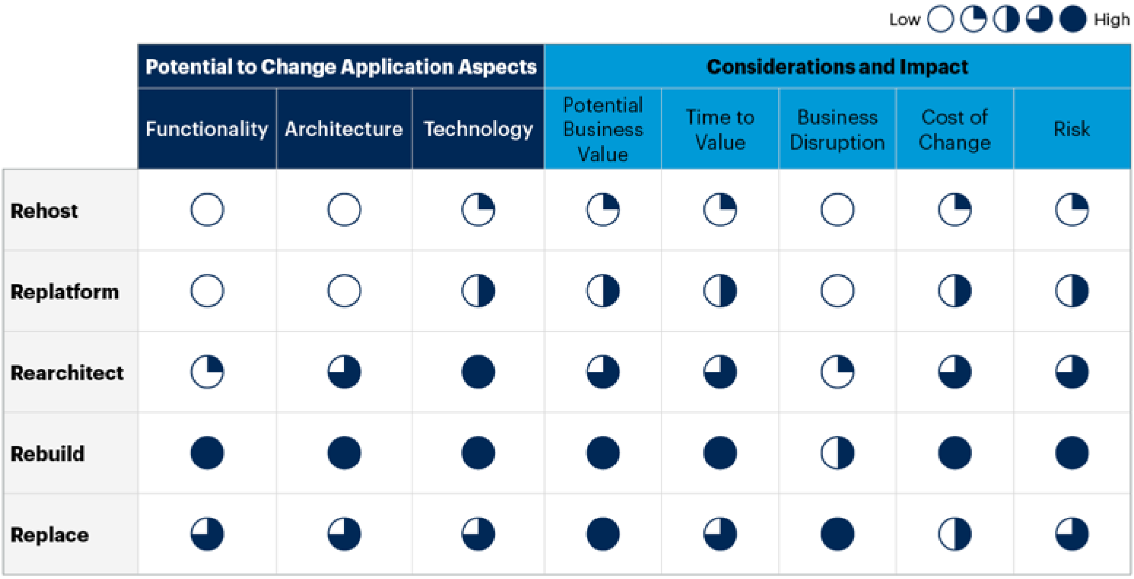
Figure 1 below outlines Gartner's framework for selecting the right application modernization and cloud migration approach. It lists each approach and identifies how much it can contribute to solving problems in technology, architecture, or functionality. For example, rearchitecting an application component can solve technology and architecture issues, but it has limited value in solving functional problems. Rebuilding an application gives you the greatest opportunity and flexibility to improve technology, architecture, and functionality.

Selecting the Right Approach

Modernization approaches also differ regarding risk, cost and impact on business and IT. The considerations and consequences for each approach include:

- Potential business value: Stakeholders often overestimate the potential value of a modernization or migration approach (for example, they expect that a replatform will increase business fit or agility). It is important to be clear about the potential business value of each approach.
- Time to value: Implementation time, and therefore time to value, differs for each modernization and migration approach.
- Business disruption: How does the modernization approach impact the business processes it supports? Is retraining required, or business process redesign? For example, a replacement of a COTS or SaaS solution will have a significant impact on business processes and user experience.
- Cost of change: This refers to the IT-related effort and cost of modernization and migration.
- Risk: This refers to the risk level of executing the modernization and migration.

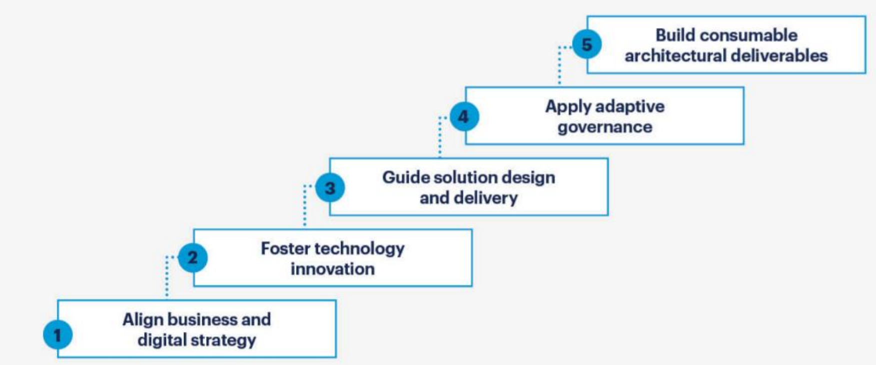
Software engineering leaders should use this framework to develop suitable modernization approaches. They should discuss the selected approach with all



Low ○ ◔ ◑ ◕ ● High

	Potential to Change Application Aspects			Considerations and Impact				
	Functionality	Architecture	Technology	Potential Business Value	Time to Value	Business Disruption	Cost of Change	Risk
Rehost	○	○	◔	◔	◔	○	◔	◔
Replatform	○	○	◑	◑	◑	○	◑	◑
Rearchitect	◔	◕	●	◕	◕	◔	◕	◕
Rebuild	●	●	●	●	●	◑	●	●
Replace	◕	◕	◕	●	◕	●	◑	◕

stakeholders and set expectations by making everyone aware of the modernization's scope, effect, outcomes, and consequences. The considerations and consequences must be evaluated and weighed to decide if modernization or migration is worthwhile and which candidate approaches provide the best balance between value and effect against cost, risk, and impact.


1 Align business and digital strategy
2 Foster technology innovation
3 Guide solution design and delivery
4 Apply adaptive governance
5 Build consumable architectural deliverables

The bottom line: Choose the application modernization and cloud migration approach that delivers improvements to the relevant application layers — technology, architecture, and functionality — and has an acceptable risk, cost, and impact profile.

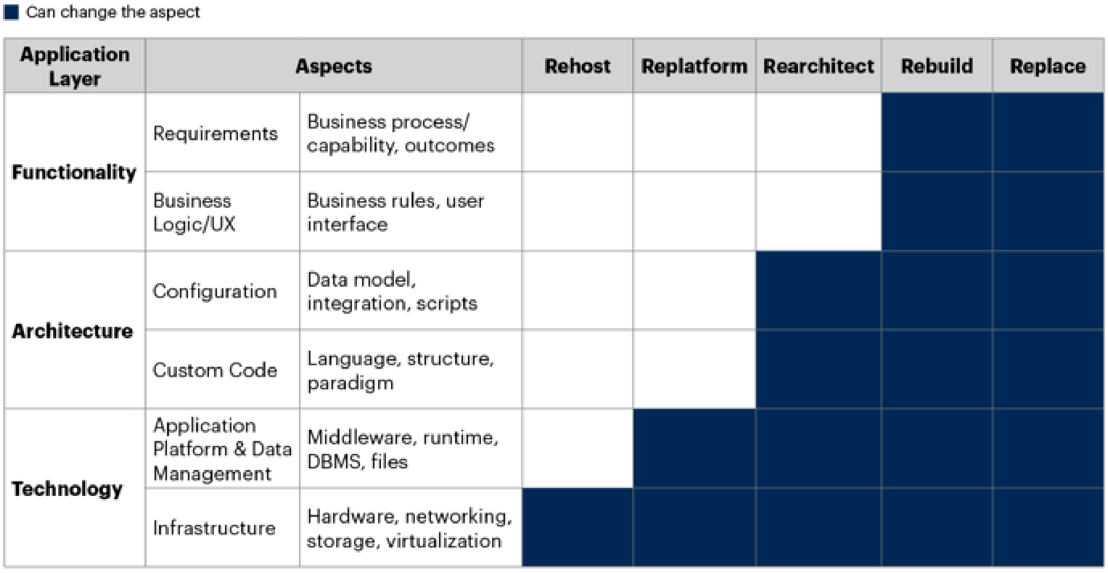
Evidence

2023 Gartner CIO and Technology Executive Survey: This survey was conducted to help CIOs and technology executives overcome digital execution gaps by empowering and enabling an ecosystem of internal and external digital technology producers.

Note 1: Application and Application Components

This research discusses an application as the target of modernization and cloud migration. The target can also be a component or module of an application. Gartner recommends an incremental approach to application modernization and cloud migration by focusing on the components of an application instead of a big-bang modernization of the whole application. This implies that components of an application should be modernized or migrated individually.

Modernization approaches can differ for components of the same application. For example, some components need a rebuild, while others only need a replatform to resolve the issues they are causing.



■ Can change the aspect

Application Layer	Aspects		Rehost	Replatform	Rearchitect	Rebuild	Replace
Functionality	Requirements	Business process/ capability, outcomes				■	■
	Business Logic/UX	Business rules, user interface				■	■
Architecture	Configuration	Data model, integration, scripts			■	■	■
	Custom Code	Language, structure, paradigm			■	■	■
Technology	Application Platform & Data Management	Middleware, runtime, DBMS, files		■	■	■	■
	Infrastructure	Hardware, networking, storage, virtualization	■	■	■	■	■

Determining Best Approach for Modernization

Digital transformation has made it imperative for application leaders to find effective ways to modernize legacy systems. The biggest challenge? Knowing the risk-to-reward ratio before acting.

The Gartner three-step evaluation process directs the approach to application modernization; the best approach depends on the problem you're trying to solve

1. Evaluate legacy systems using drivers

These are the issues, concerns, or impediments the legacy application creates due to its technology, architecture, or functionality.

From a business perspective, three drivers come into play: business fit, value, and agility. Suppose the legacy application does not meet the new requirements of a digital business. In that case, it must be modernized to fit properly and upgraded to provide greater business value. Applications that lack the agility to keep pace with the demands of digital business may be a cost or risk liability.

The three other drivers come from the IT perspective and involve cost, complexity, and risk. If the total cost of ownership (TCO) is too high, the technology too complex, or security, compliance, support, or scalability are being compromised, it's time to modernize. The best modernization opportunities are those with multiple drivers from both a business and an IT perspective.

2: Evaluate modernization

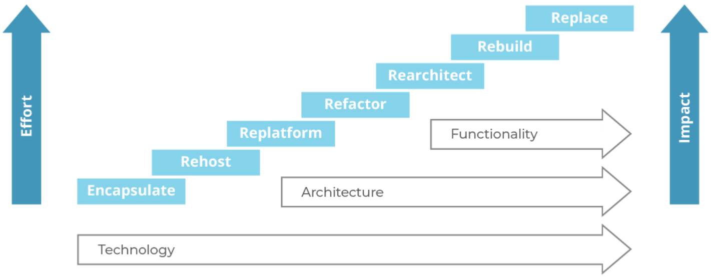
Consider modernization options once the opportunity is selected and the problem is identified. Gartner has ranked seven options by the ease of implementation (the easier the option, the less risk, and impact it will have on the system and the business processes; the harder, the more risk and impact).

1. **Encapsulate.** Leverage and extend the application features by encapsulating its data and functions, making them available as services via an API.
2. **Rehost.** Redeploy the application component to another infrastructure without modifying its code, features, or functions.
3. **Replatform.** Migrate to a new runtime platform, making minimal changes to the code but not the code structure, features, or functions.
4. **Refactor.** Restructure and optimize the existing code to remove technical debt and improve non-functional attributes.
5. **Rearchitect.** Materially alter the code to shift it to a new application architecture and exploit new and better capabilities.
6. **Rebuild.** Redesign or rewrite the application component from scratch while preserving its scope and specifications.
7. **Repurchase.** Replace the former application component, simultaneously considering new requirements and needs.

3. Choose the modernization approach with the highest effect and value

You can choose the modernization approach with the highest effect and value by mapping the seven options regarding their effect on technology, architecture, functionality, cost, and risk.

Modernizing legacy applications means choosing between rearchitecting, rebuilding, or replacing. Rearchitecting has medium costs and risks, whereas rebuilding or replacing provides the best results with higher costs and risks. The key is to weigh all options to help identify the extent to which each will have the desired effect—with the minimum effort and maximum positive impact.



Pace-Layered Application Strategy

Aligning Exec with IT

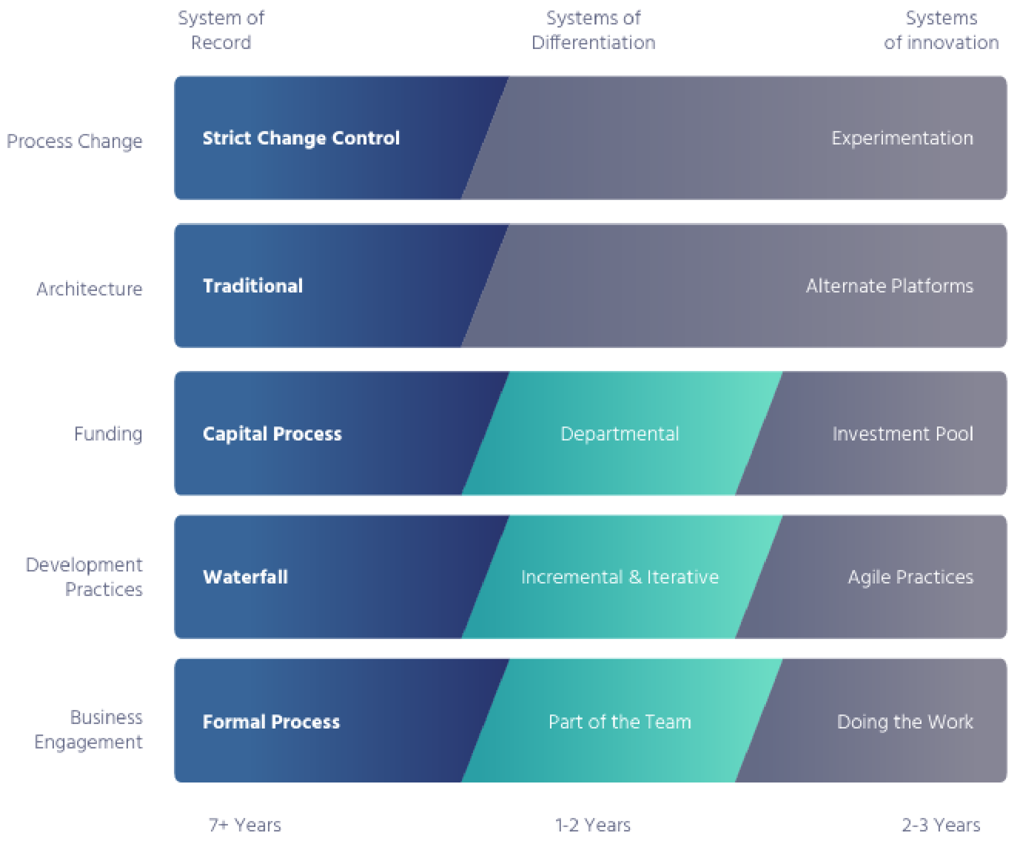
The Gartner Pace-Layed Application Strategy categorizes applications according to their organizational role to support business change, differentiation, and innovation.[1] This categorization aims to apply different IT governance procedures to different types of applications and accept that different expectations regarding the rate of change should be applied to different applications.

Summary

Gartner suggests a fundamental disconnect exists in the strategies pursued by leadership, management, and IT. Management seeks applications that are easy to use and reliable. Leadership wants to mitigate risk and take advantage of new opportunities. IT aims to reduce cost, minimize complexity, and improve security. This leads to conflicting desires for continuous change and tight control of any change. The Pace-Layed Application Strategy aims to help this issue.

> "A methodology for categorizing applications and developing a differentiated management and governance process that reflects how they are used and their rate of change"

The idea of pace layering is based on the concept of shearing layers in building architecture, whereby different layers within the building change at different rates. One example of this is that while the structure of the building may last for thirty to one hundred years, the skin of the building may change to accommodate new styles or to incorporate more energy-efficient designs. With this in mind, the fundamental concept of the Gartner pace layer is that different types of systems should develop more quickly than others, and as such, they should be treated differently.



	System of Record	Systems of Differentiation	Systems of innovation
Process Change	Strict Change Control		Experimentation
Architecture	Traditional		Alternate Platforms
Funding	Capital Process	Departmental	Investment Pool
Development Practices	Waterfall	Incremental & Iterative	Agile Practices
Business Engagement	Formal Process	Part of the Team	Doing the Work
	7+ Years	1-2 Years	2-3 Years

Continuous Modernization to Optimize Legacy Applications

Legacy applications hinder continuous value delivery, but big-bang or rip-and-replace efforts are costly, risky, and time-consuming. Instead, application and software engineering leaders should use continuous modernization to minimize legacy application optimization's cost, risk, and impact.

Key Findings

- A rip-and-replace approaches to modernization are disruptive in terms of cost, risk, time and impact, factors which often outweigh potential benefits
- Legacy applications aren't designed for continuous delivery, with certain components creating obstacles that effect business operations
- Poor coordination between product and modernization teams and a lack of business leader engagement undermine change programs

Recommendations

To optimize legacy applications effectively, applications and software engineering leaders should:

- Use continuous modernization instead of rip-and-replace programs, addressing specific business obstacles caused by legacy applications.
- Identify, prioritize, and remove the most severe friction points—business capabilities with poor application support.
- Create a continuous culture by aligning with product and platform teams for "just-in-time" iterative improvements

Introduction

Digital transformation is ongoing. According to Gartner CIO surveys, 40% of enterprise processes were optimized digitally by 2023, with 25% of total revenue from digital products and services. 47% of the respondents selected "integrate, innovate and modernize enterprise applications" as a focus for the next 12 months, making it the third highest priority overall.[1] Modernizing legacy applications with a big-bang approach is often impractical due to cost, risk, and impact. Continuous modernization offers a better solution by managing legacy applications as assets and addressing obstacles that impede value delivery.

Legacy applications remain central to many organizations, containing valuable data and functions. However, they impose obstacles that affect business agility and velocity:

1. They were not designed to match the pace of digital business demand.
2. Modern platforms may better deliver some functions.
3. They are closed systems, making functions and data hard to access.
4. They have aged poorly, increasing cost, complexity, and technical risk while decreasing business fit, value, and agility.

Rather than discarding legacy applications, focus on addressing the 20% of functions causing 80% of the pain. Apply the 80/20 rule and think of legacy applications like an old car: replace or repair defective components to extend its life and value instead of scrapping it.

Use Continuous Modernization as an Alternative to Rip-and-Replace Programs

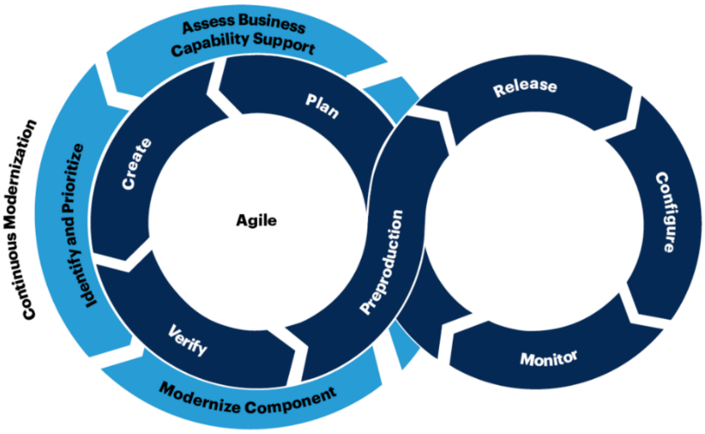
Applications and software engineering leaders should view legacy applications as assets to manage, not problems to eliminate A gradual, continuous approach is superior due to:

- Lower cost, risk, and impact than rip-and-replace efforts.
- Urgency, as digital business needs support quickly.

- Uncertainty, as it's hard to predict future digital business requirements.

Technology ages quickly, and requirements change constantly, creating new legacy applications.

Continuous Modernization Minimizes the Cost, Risk and Impact of Optimizing Legacy Applications



Implement Continuous Modernization

Continuous modernization is an iterative process to identify, prioritize, and remove obstacles from legacy applications. This process involves three key stages:

Assess and Prioritize

- Identify business capabilities supported by legacy applications and their contribution to business value.
- Map components of legacy applications to each business capability.
- Assess support for each business capability and identify modernization drivers, such as poor business fit or high complexity.
- Prioritize pain points using methods like TIME analysis, and create a backlog of modernization activities.

Transform

- Analyze and address the cause behind each friction point, selecting appropriate modernization approaches like encapsulation, re-platforming, refactoring, or replacing components.
- Execute the modernization activities to improve business support.

Refine and Repeat

- Reevaluate the backlog to include new or changed priorities.
- Select the next activity from the backlog and repeat the process.

Composability Focused

Composability allows systems to adapt quickly while staying resilient. Continuous modernization makes legacy systems more composable by replacing components with appropriate technology or encapsulating, reusing, and extending current functions.

For example, an inflexible pricing function can be replaced with a flexible rule engine, or an order-tracking function can be supported by implementing an API and building a web GUI on a suitable technology platform. Encapsulation, while using existing data and functions, enables us to eventually replace that part of the legacy system with minimal impact of extended function.

Continuous Culture

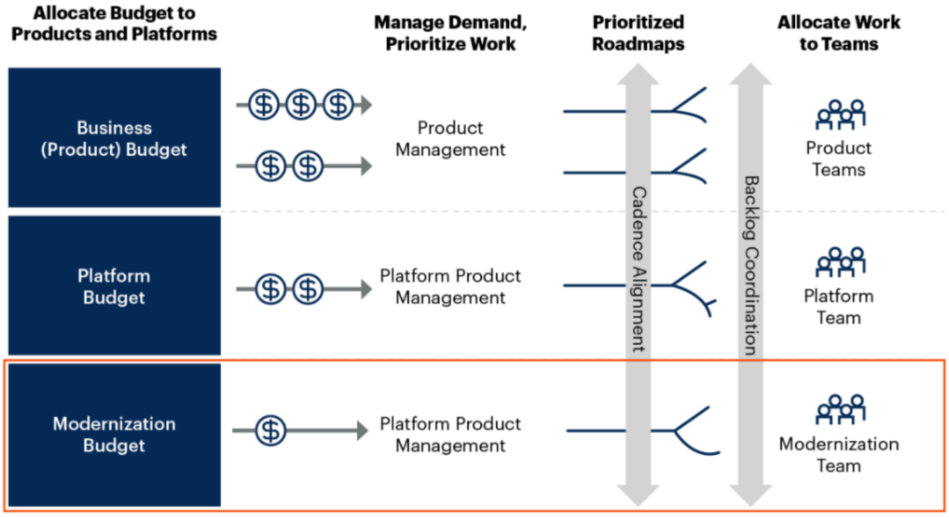
Coordinate efforts between product and platform teams to respond quickly to changes in business demand and new obstacles. Improve IT and business, ensure every team manages technical debt, and improve product and platform in each cycle.

Applications and software engineering leaders must work with business leaders to manage applications as assets. They must allocate resources for ongoing modernization, coordinate backlogs, and align delivery cadences to ensure timely support for product teams.

Fostering collaboration and creating a culture of continuous modernization can help organizations effectively support digital business and continuously improve their legacy applications.

Collaboration Between Product and Platform Teams

Continuous modernization recognizes that modernization is never complete. New legacy applications emerge as old applications are modernized, requiring ongoing updates. For example, old mainframe applications from the 1970s and 1980s are still in place in banks; during replacement, new technology emerges, such as client/server applications from the 1990s and Java applications from the 2000s.



Application Portfolio Management – Gartner TIME Assessment

Application Portfolio Management (APM) is crucial in the digital transformation era, ensuring IT investments align with business goals. As businesses strive with an ever-growing array of applications, this approach ensures IT investments align with the business, making APM a driver of digital transformation success. The TIME assessment is a strategic tool guiding businesses through their digital journey.

What is Gartner's TIME Assessment?

Gartner's TIME Assessment (Tolerate, Invest, Migrate, Eliminate) helps organizations categorize and evaluate their application portfolios to align IT strategies with business objectives:

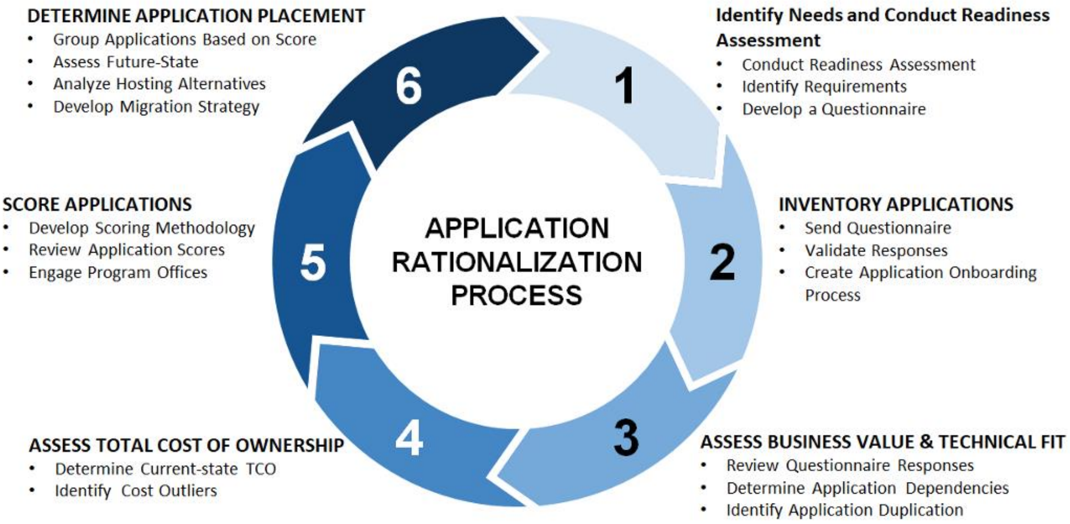
- **Tolerate**: Maintain functional but imperfect applications until a change is necessary.
- **Invest**: Continuously enhance and scale valuable, competitive applications.
- **Migrate**: Update or replace outdated applications to meet new needs.
- **Eliminate**: Remove redundant or low-value applications to free resources and reduce complexity.

Importance of the TIME Assessment

- **Cost Efficiency**: Optimizes IT spending by identifying where to allocate resources.
- **Reduced Complexity**: Simplifies IT landscape by eliminating redundancies.
- **Future-proofing**: Keeps businesses agile and ready to adopt new technologies.
- **Business Alignment**: Ensures IT decisions support business strategies.

Implementing Gartner's TIME Model

1. Inventory and Assessment: Document all applications and their metrics.
2. Business Input: Engage stakeholders for requirements and future plans.
3. Categorize: Classify applications into TIME segments through discussions and reviews
4. Plan and Act: Create action plans for each category.
5. Review Periodically: Regularly update the TIME assessment to stay aligned with business needs.

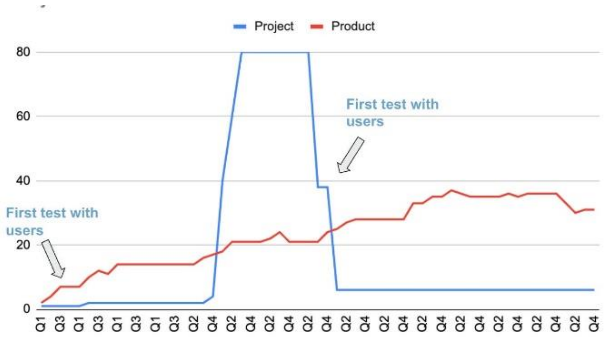

DETERMINE APPLICATION PLACEMENT
• Group Applications Based on Score
• Assess Future-State
• Analyze Hosting Alternatives
• Develop Migration Strategy
Identify Needs and Conduct Readiness Assessment
• Conduct Readiness Assessment
• Identify Requirements
• Develop a Questionnaire
6
1
SCORE APPLICATIONS
• Develop Scoring Methodology
• Review Application Scores
• Engage Program Offices
APPLICATION RATIONALIZATION PROCESS
5
2
INVENTORY APPLICATIONS
• Send Questionnaire
• Validate Responses
• Create Application Onboarding Process
4
3
ASSESS TOTAL COST OF OWNERSHIP
• Determine Current-state TCO
• Identify Cost Outliers
ASSESS BUSINESS VALUE & TECHNICAL FIT
• Review Questionnaire Responses
• Determine Application Dependencies
• Identify Application Duplication

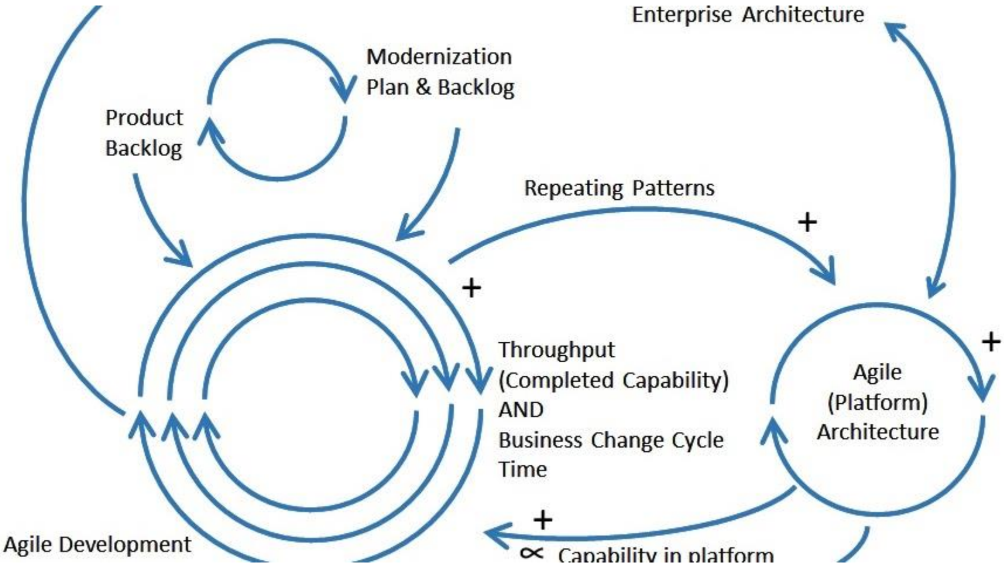
Business Case and Rationale

The business case includes the financial rationale behind a transformation, including cost, savings, and returns. It provides a forecast for return on investment and connects IT changes to business KPIs that ensure ongoing project funding. To build the business case, organizations must first. understand the focus and motivations around cost savings. They then need to develop a current state cost model per app to get to an average monthly cost per virtual machine and per database that can be compared with cloud hosting and drive decisions around IT spend alignment to business criticality.

Funding Model

Project vs product funding model, where the project relates to CAPEX, having a consistent team over time may look like an unwanted ongoing expense if we assume that development work is at some point "done," but that is not the case, then moving to a "product model" has several advantages and aligns better with continuous modernization[2]




Enterprise Architecture
Modernization Plan & Backlog
Product Backlog
Repeating Patterns
+
+
+
Throughput (Completed Capability) AND Business Change Cycle Time
Agile (Platform) Architecture
+
∝ Capability in platform
Agile Development

MicroServices

The number of mentions of “microservices architecture” plunged 42% between January 2019 and September 2020, according to Gartner social media analytics study.

This trend points to growing disillusionment with microservices architecture — a design paradigm that aims to increase agility, deployment flexibility, and precise scalability by applying service-oriented architecture and domain-driven design principles to distributed applications (i.e., microservices).

Gartner defines a microservice as an application component that is tightly scoped, strongly encapsulated, loosely coupled, independently deployable, and independently scalable. Microservices architecture can deliver great benefits, but success is often elusive due to misconceptions about why, when, and how to use it.

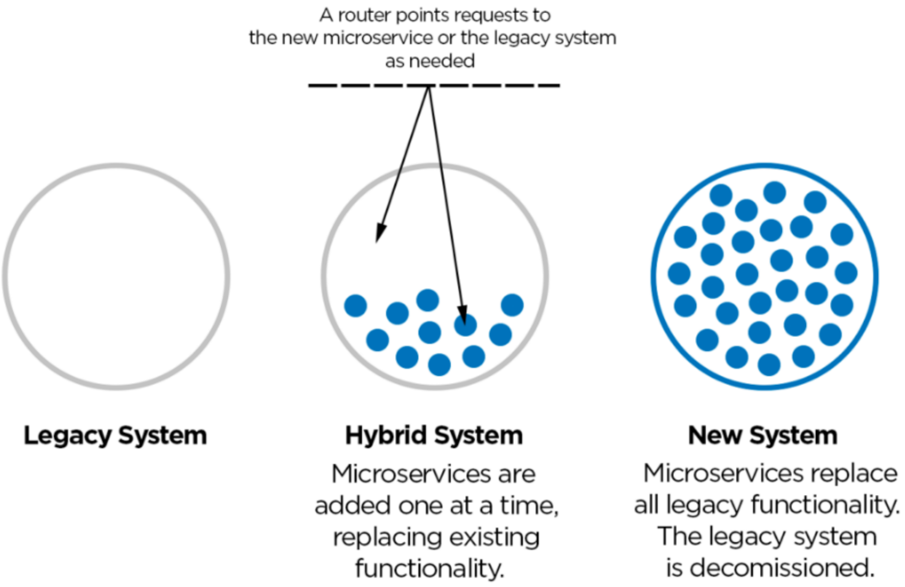
Microservices architecture can improve agility and increase scalability, but it isn’t right in all circumstances. In fact, it could be absolutely wrong. To improve the likelihood of success, software engineering leaders should answer three essential questions about microservices architecture before setting up a microservices platform: Why, when and how?

The ideal answer to “why” will result in a business case with a clear return-on-investment benefit. Software engineers most frequently adopt microservices architecture to enable continuous delivery of new application features. In fact, if you aren’t trying to implement a continuous delivery practice, you are better off using a more coarse-grained architectural model — what Gartner calls “Mesh App and Service Architecture” and “mini services.”’

Legacy, Hybrid and New Systems

This gives us a few clear options regarding how we approach containerization, encapsulating APIs, and focusing on interrelated components.

Following these general purpose strategic paths forward.


A router points requests to the new microservice or the legacy system as needed
Legacy System
Hybrid System
Microservices are added one at a time, replacing existing functionality.
New System
Microservices replace all legacy functionality. The legacy system is decomissioned.

Reference Shopify's Implementation of the Modular Monolith: Componentization

Once it was clear that Shopify had outgrown the monolithic structure and that it was affecting developer productivity and happiness, a survey was sent to all the developers working in our core system to identify the main pain points.[3] They knew they had a problem, but they wanted to be data-informed when devising a solution to ensure it was designed to actually solve the problem, not just the anecdotally reported one.

The results of that survey informed the decision to split up their codebase. In early 2017, a small but mighty team was put together to tackle this. The project was initially named "Break-Core-Up-Into-Multiple-Pieces," and eventually evolved into "Componentization."

Modular Components

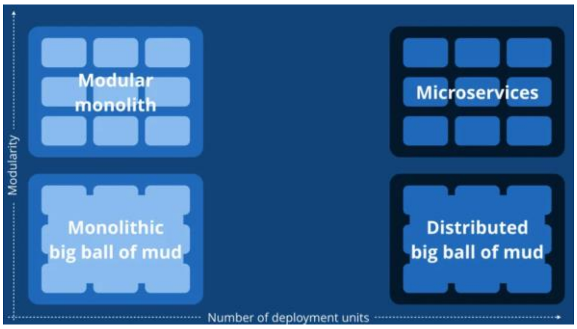
We wanted a solution that increased modularity without increasing the number of deployment units, allowing us to get the advantages of both monoliths and microservices without so many of the downsides.

A modular monolith is a system where all of the code powers a single application and there are strictly enforced boundaries between different domains

- Re-organize code by real-world concepts (orders, shipping, inventory, billing), rather than software concepts (MVC)
- Make code easier to locate and enable understanding of the individual pieces independently.
- Each component is structured as a "mini app," with the goal of eventually namespacing them as modules.

Decoupling

This new organization would highlight areas that were unnecessarily coupled, enabling goals of separation of concerns to be delivered by a series of discrete changes that separate APIs, based on business relationships, in a mono-repo, enabling teams to manage individual pieces, hence giving the benefits of microservices without the downside.


Modular monolith
Microservices
Modularity
Monolithic big ball of mud
Distributed big ball of mud
Number of deployment units

Steps to Componentization

Reorganizing the codebase by business domains: This restructuring facilitated better ownership and understanding of different areas within the code.

Isolating dependencies: By isolating dependencies, Shopify aimed to reduce coupling and enhance modularity, which made the codebase easier to manage and extend.

Enforcing boundaries: Clear boundaries between modules were established to prevent the spread of changes and maintain the integrity of each component. This approach helped make the development process more predictable and manageable.

Develop tools that automate the process, alongside high automated test coverage.

Koan Case Study: MetService

After a standard lift and shift, after 20 years of accumulating applications providing key weather data to the National Airline Carrier (Air New Zealand), the Road Safety Authority (New Zealand Transport Authority) and Transpower.

Modernization Portfolio Assessment
Review using custom tools, technical reviews, and interviews with product owners / technical people. Assess the entire application portfolio of over 350+ applications. Tools now include an open-sourced method of scanning running applications.

Roadmap and Working Model
Develop modernization strategy, business cases, ROI, and first 18 months of strategy; operating environment has massive technical debt, take out dependencies, iteratively, release often. The working model has our team working in sprints with MetService Teams

Modernization Delivery
Used API Gateway to replace technical debt and "out of support" from Broadcom gateway. Strategies include encapsulate, refactor, and rebuild as "cloud-native" services API Gateway (V2), Lambda, with high test coverage. Subsequently, retired Broadcom gateway and RedShield. Route 53, DNS records, and API Keys were migrated into AWS.

MuleSoft Extraction

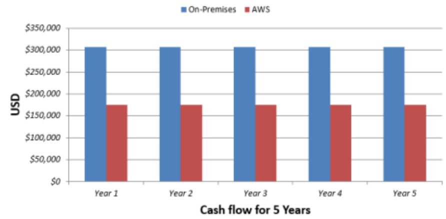
Remove MuleSoft (retire) from the operating environment, remove technical debt, and use containers. 'Modernize' by replacing custom mule code with typescript / nest.js, refactor code by porting Java into containers, and generating typescript code using data weave. Refactored / rebuild uses Nest.js in containers, replacing some functions with typescript.

Looking at applications holistically helps us to reduce complexity and technical debt in migration and modernization programs.

Migration Assess, Mobilize

MetService required a partner to scope and deliver 150 virtual machine migrations from the Auckland Data Centre into their current AWS tenancy. Koan used MAP funding to deliver an assessment phase and a mobilization phase. During the assess migrate phase, a hundred VMs were decommissioned, reducing costs, and operationalization efforts

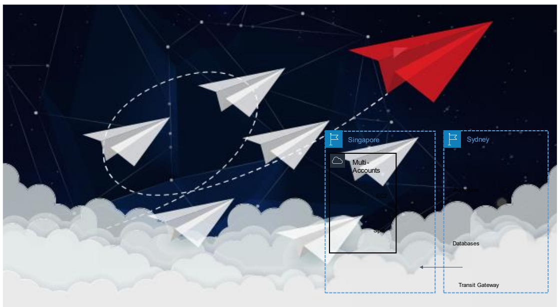
Total Cost of Ownership TCO
On-prem versus cloud computing costs were calculated and estimated using both 3- and 5-year TCO estimates. Modernization patterns applied to the core rehost strategies brought costs down.



Platform / Product Team Collaboration
By working with multiple teams, including BAU, Platform, and Product, we identified that the majority of on-prem services could be decommissioned. To de-risk the changes, various strategies were used to test virtual machines/services before turning them off. After starting with 150 VMs / services, the final tally of services, applications, and infrastructure services was reduced to 50.

Containerization

POCs were completed for complicated applications to containerize them with docker images and deploy with ECS to test and enable. This includes strategies for handling a two-way data exchange with Airways.



Koan is a global cloud and software development consultancy.

Koan has been building and deploying secure, compliant, enterprise applications for Global customers for 20 years.

Practices DecSecOps, Cloud, Modernize, Develop and Data-ML-AI.

[1] Layered Application Strategy and Why Should You Use It

[2] Congressional Testimony

[3] Deconstructing monolith designing software maximizes developer productivity