LTI

Let's Solve

A Larsen & Toubro Group Company

Whitepaper

Legacy Modernization

Realign. Rebuild. Reengineer

Author: Ritu Raj Tripathi

Abstract

One old habit that large enterprises and traditional players cannot always continue with retaining legacy systems. In today's fast-paced business environment, with new-age technologies at play and the emergence of new and agile entrants in the market, some spring cleaning is a must.

Introduction

"If you want something new, you have to stop doing something old." – Peter F. Drucker

Technology is evolving at a never-before pace. Every day, tech giants are launching new digital products and services, which have accelerated business agility, innovation, and adaption to the dynamic business environment.

We are living in an era of ever-demanding consumers who want everything at the click of a button. That's why no company or business can take them for granted in this competitive business scenario. Instead, enterprises need to align their offerings, products, services, and experiences in line with the aspirations of the customers.

Successful companies like Uber and Amazon are already doing it by ingraining a culture of transformation in their DNA. A proof of it can be seen in the number of deployments and releases these organizations are churning out every week, day, a minute, or even a second. These kinds of rapid deployments and releases are possible because there is corresponding modern microservices-based agile software architecture to underpin these products and platforms.

Digital Economy: Transition to Transformation.

Digital technologies are disrupting the very core of business functions. It has brought traditional players and the new era companies, born in the digital-native world, on a level playing field. These new startups don't have to carry the heavy baggage of legacy core systems.

Apple did it with the iPod, Aggregators like Uber, Ola, Oyo transformed commuting and accommodation. Netflix changed the entire entertainment ballgame. Apps like Zoom and Microsoft Teams are making holes in the aviation business with remote working and meetings becoming the new default.

Is Legacy an Asset or a Liability?

To answer this, let's use some numbers to gain perspective.

92 of top 100 banks and 9 out of top 10 Life and Asset Insurance players rely on IBM mainframe for their core business operations.

The efficiency and reliability of these systems are time-tested. So, replacing them is a difficult decision for any CIO. Nonetheless, these systems have limitations. The major one being their inability to integrate with the latest digital technologies and ecosystem. Does this mean we have to live with these constraints and limitations? Fortunately, the answer is no.

Time to Build a New Legacy

It is only prudent that enterprises make the best use of their legacy core assets. They can select many options based on the context, application maturity, and aspiration of the organization. The organization should opt for an insightful SWOT analysis to select a modernization approach and strategy.

Traditional and established players are at a critical juncture in today's competitive environment. New-age digital disrupters are building a brand-new, fortified IT landscape, while the older players are playing catch- up with legacy systems. It's time they integrate innovative and digital technologies to leverage their existing IT investments. So, does that mean, the legacy IT needs to start from scratch? Fortunately, legacy is a viable and practical solution, but only if companies are looking to align their existing systems with the ever-growing demand for digital technology. For this, companies will have to chalk out an IT journey to digital from legacy. This comprises selecting an optimum mix of applications, digital touch-points, and an economic and logical action plan to accelerate this journey.

A Case for Modernizing Legacy

In this new era of digital transformation, you can't win with obsolete tools and outdated technology. Your business must find a niche and also leverage the strength of the partner ecosystem.

It is not the question of if but when to adapt to these new digital technologies like API, microservices, AI/ ML, IoT, blockchain, cloud, advanced analytics, RPA. We all know what happened to Kodak and Nokia when they refused to adapt to the changing times. Success today is hinged on the effectiveness of leveraging these digital technologies for your business.

Gartner predicts that every dollar invested in digital business innovation through to the end of 2020 will require enterprises to spend at least three times that to continuously modernize the legacy application portfolio.

Legacy modernization or app modernization is a strategy to transform IT assets and core legacy systems with digital enablement within the existing IT landscape or framework. There is no one-size-fits-all strategy but rather it depends on several factors, challenges and drivers.

When Does One Need Legacy Modernization?

1. When aligning current-day business strategy with IT systems

Today, companies need to align their IT products, process data infrastructures and drive business outcomes in the fast-paced business environment. Legacy applications can become a chokepoint with delayed batch cycles, slow processing and lackluster web integration with new-age platforms. This leads to slower rollouts and time to market.

2. If the TCO factor is giving you a tough time

The cost for mainframes and software license fees can go into millions. Plus, they have limited capabilities and a high cost of maintenance. If your enterprise is undergoing this situation, it's time to switch to legacy modernization.

3. When there are databases involved

There is a big mismatch between the capabilities of traditional legacy systems and the current business expectations and demands. They are complex and high maintenance, since a lot of systems use batch and online macros to facilitate access to these databases.

4. When you need to accelerate documentation and business processes

Legacy systems were made for enterprises to function a decade ago. They are not made for the ever-demanding market scenarios and documentation requirements of today. This makes it difficult for an easy transition or migration of these systems.

How to Go About Legacy Modernization

There are multiple approaches to modernization – here are some prominent ones:

Rehosting

This is a lift-and-shift based approach where applications are redeployed to other physical, virtual or cloud infrastructure. It requires no recompiling, reconfiguring and readapting the application codes, features and functions. That's why this approach may not have any significant benefits apart from the cloud-like cost optimization.

Replatforming

This approach is based on migrating an application to a new runtime platform. While it may enjoy some of the cloud-native services, it will make minimal changes to the code or adapt to the new platform. Hence, it is just a cosmetic improvement but will not yield the core benefits of modern architecture.

Replacement

This approach is normally followed when the core systems become obsolete, complex and counter-productive for the enterprise, which leaves the enterprise with no choice but to replace them. It can be done with another packaged solution which stems from a modern IT architecture and can help business leverage a digital application ecosystem. This new system could also be a SaaS-based solution.

Re-engineering

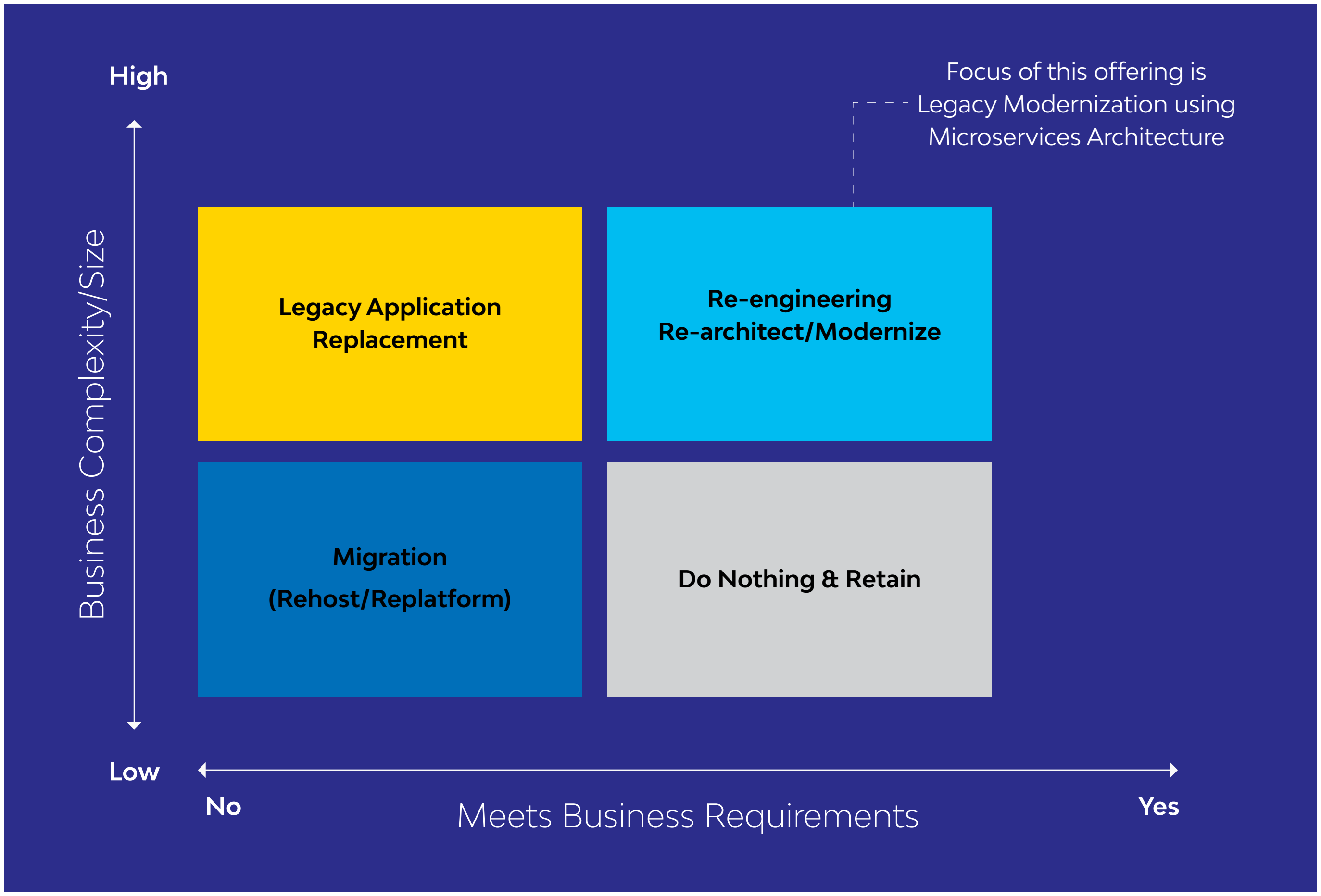
This is the core area of focus where an application is fundamentally transformed and modernized in true sense. It gets redesigned to take advantage of all modern architectural patterns, built on top of cloud-native technologies like containers, microservices, service mesh, API management, etc. Application functionality is architected to mimic the domain and capability that the organization offers as a product or service. The build and deployment pipeline is automated to take advantage of DevOps.

Code is refactored to suit the requirement of a loosely coupled regime.

There should be a holistic assessment of the application landscape and portfolio to map the applications against these two key dimensions:

a) The ability of applications to meet business requirement
b) Its complexity and size.

Applications and systems will end up being plotted in one of the four quadrants and the strategy will depend on where particular application falls into. Modernization approach and strategy will also factor in many other aspects like the overall strategy of the enterprise, cost, ROI, TTM, partner ecosystem, spread across geography, verticalization, etc. The decision matrix is depicted below



High
Focus of this offering is Legacy Modernization using Microservices Architecture
Business Complexity/Size
Legacy Application Replacement
Re-engineering Re-architect/Modernize
Migration (Rehost/Replatform)
Do Nothing & Retain
Low
No
Meets Business Requirements
Yes

How to Go About Legacy Modernization

Re-engineering is the inside-out transformation approach, which means redesigning the IT architecture from the core components to building a new set of functionalities. This can be done by refactoring the code and leveraging new technologies, systems and platforms, which lead to better competencies, scalability, agility and adaptability to modern-day IT requirements and technologies.

There are two ways of achieving this transformation through re-engineering:

A Case for Modernizing Legacy

In this approach, all old applications are completed scrapped and rewritten using modern architecture and cloud-native technologies like clustering and microservices. Here are the pros and cons of this approach:

Pros	Cons
• No dependency on legacy code structure • Clean and futuristic architecture	• Huge investment required and difficult to justify ROI • Risk associated with new system and functionalities with no option of rollback • Prolonged wait time for business before the reengineering goes live • Big bang approach which requires dedicated bandwidth/ attention of stakeholders

Incremental upgrade and coexistence

In this approach application is analyzed and logically decomposed using one of the patterns, This can either be by decomposing by subdomain or by capability. Instead of a complete overhaul, certain functionalities and features are planned, picked and prioritized for upgrades in a phased manner. This prioritization should be based on business outcomes or the criticality of the feature. Here are the pros and cons of this approach:

Pros	Cons
• Less risky as compared to the big bang approach • Can fall back on the old implementation using the strangler approach • Incremental load increase to the new system and coexistence • Quick win and benefits realization by business • Faster time to market as the upgrade of individual features take less time compared to the whole rewrite	• Coexistence of old and new systems can lead to increased operation and maintenance cost during this period • Both skillsets (new and old) required during this period of coexistence

Reengineering – Step by Step. Stage by Stage.

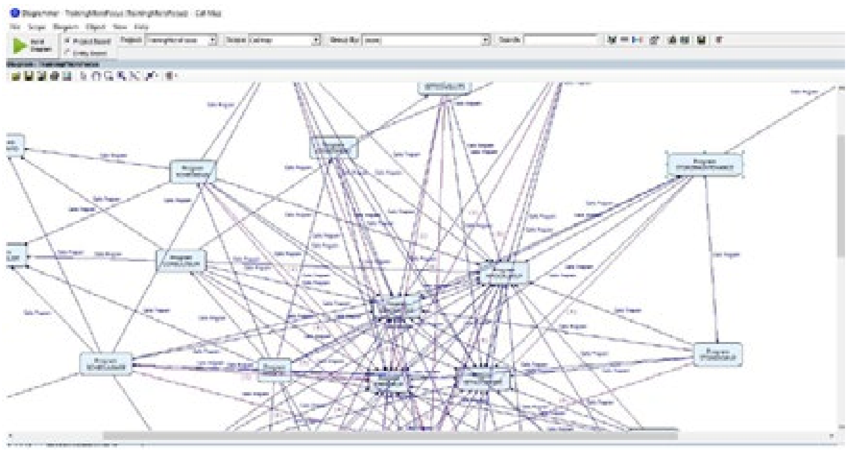
Irrespective of whether you opt for a complete rewrite or incremental approach, there is a set of best practices and sequential phases for modernizing any application. It is advisable to bank on proven tools, accelerators, and frameworks to fast track this engineering cycle. Here are the phases/ stages that best describe the approach recommended to achieve the benefit of app modernization:

Discovery

Organizations may or may not have the latest documentation to depict the as-is state of the system. Even if it is available, there is a good possibility that it is not up-to-date and does not cover all aspects of complex monolithic systems, which would have evolved over the years. There are many tools available (like micro focus Enterprise Analyzer, Eclipse, Altova Umodel, etc). These tools help in the following ways:

- They reverse engineer the complex monolithic application
- Generate reports depicting object hierarchy and structure
- UML diagrams showcase relationship and interactions
- Save effort and time by tool-driven code analysis
- Expedites modernization initiative by drawing the as-is state

Some of the typical output diagrams are depicted below as a sample.



Design

Modern application architecture is heavily underpinned on microservices and container-based architecture. These applications need to communicate effectively with each other and be a part of the whole ecosystem. This demands the design approach to be API-led and API-first. Such an approach leverages the following best practices and processes during design:

- Open API specification compliant tools to define and design APIs
- Clear separation of API specification and implementation
- API gateway pattern to ensure security and centralized control
- Decompose monolith by sub-domains or capability
- Use of a 12-factor architecture principle (Summary of them is described below)

Sr.	Principle	Description
1	Codebase	One codebase tracked in revision control, many deployments
2	Dependencies	Explicitly declare and isolate dependencies
3	Config	Store config in the environment
4	Backing services	Treat backing services as attached resources
5	Build, release, run	Strictly separate build and run stages
6	Processes	Execute the app as one or more stateless processes

7	Port binding	Export services via port binding
8	Concurrency	Scale-out via the process model
9	Disposability	Maximize robustness with fast startup and graceful shutdown
10	Dev/prod parity	Keep development, staging, and production as similar as possible
11	Logs	Treat logs as event streams
12	Admin processes	Run admin/management tasks as one-off processes

Implementation

Once specification and design are ready, you are all set to implement new features and functionality using cloud-native principles. The following best practices can be used during this phase:

- Leverage Java-based entity object generator for the legacy code
- SDK/ adopter-based approach to connect with the legacy system
- Tool to create project chassis for microservices (Example.: spring cloud framework)
- Lightweight business process flow framework for headless implementation
- Creation of container compatible docker image for deployment

Build or Deployment Automation

Automating the build and deployment process is an integral part of the modern microservices-based architecture. It is recommended to use the following automation and DevOps practices:

- Tool for setting up CI/ CD pipeline (Jenkins)
- Use of standard code repository and version control (Git)
- Use of standard build tool (Maven)
- Automation of testing (Junit)

- Static code analysis (SonarQube)
- Integration with defect and requirement management tool (Jira)
- Use of service registry (Eureka or env specific)
- Container for deployment (Kubernetes, EKS, GKE, AKS)

Coexistence and Scaling

Once new feature functionality is ready and deployed in a container-based environment, it will be expected to coexist with part of a large monolith application that is yet to be modernized. Strangler pattern helps in routing part of the requests to new microservices and slowly increases the load based on the confidence of successful processing. Contrary to the legacy environment, it is very easy to scale in a containerized environment via a horizontal scale-out approach. Policy-driven or autoscaling is possible to scale up or down based on user/transaction increase/decrease.

The Bottom Line

As part of this paper, I have tried to put together different drivers of legacy modernization and the role it plays in the digital transformation journey of an enterprise. There are many approaches to the legacy modernization process, which vary according to the context, maturity level, and the future roadmap of individual organizations.

Although there are many cloud platforms, which offer their variants of containerized services, it is advisable to opt for an approach that does not lead to a vendor lock-in agreement. Therefore, the enterprise strategy should be aligned to a multi-cloud approach and suited for hybrid development. Also, an incremental approach should be followed rather than a big-bang approach which will reduce the risk and also generate business benefits in a shorter duration. Technology is evolving rapidly and hence it is preferable to leverage the available tools and platforms to strive for short-term gains which is aligned to larger vision.

References

https://12factor.net/

https://martinfowler.com/

https://docs.aws.amazon.com

Building Microservices (O'Reilly book)

https://azure.microsoft.com/en-in/resources/designing-distributed-systems/

Ritu Raj Tripathi

Practice Head - API
Leading API & Microservices practice within digital integration, LTI.

- Two decades of experience in Information Technology services and products. His area of expertise includes digital transformation, integration, API, Microservices, cloud native technologies, App Modernization, and large & distributed program execution.

- BE from NIT Allahabad (MNNIT), EPLM from IIM Calcutta