A Business Guide to Legacy Application Modernization For The Cloud Era

EFFORT (TIME & COST)
BUSINESS IMPACT
RETAIN/ENCAPSULATE
REHOST
REPLATFORM
REFACTOR
REARCHITECT
REBUILD
REPLACE

Power Up Your Jakarta EE

User Guide

Contents

Guide Updated: **March 2025**

Digital transformation, largely enabled by the commoditization of massive computing resources through cloud computing, has changed the way applications are developed, deployed and maintained. Organizations can now handle faster application development iterations to meet ever-changing and increasingly complex customer expectations.

A number of key events over the past years created a paradigm shift to more digital economic activities across the globe. For example, the customized digital experiences pioneered by the likes of Amazon and Netflix have created a de facto standard of user experience that all organizations aspire to in order to acquire and maintain customers. However, not all enterprise applications are positioned to take advantage of the massive benefits offered by the modern digital transformation paradigm.

Some organizations are saddled with legacy applications that will need to be modernized before they can take advantage of the benefits of the digital transformation brought about by the cloud. This paper looks at key guidelines on modernizing such applications for the cloud native era.

We start by defining and looking at the characteristics of legacy applications as well as what modernization and its benefits are. We also explore the various options to realize legacy application modernization for the cloud native era. By the end of this paper, you will have a clear framework to assess the need for and undertake application modernization for your organization to fully leverage the latest technologies.

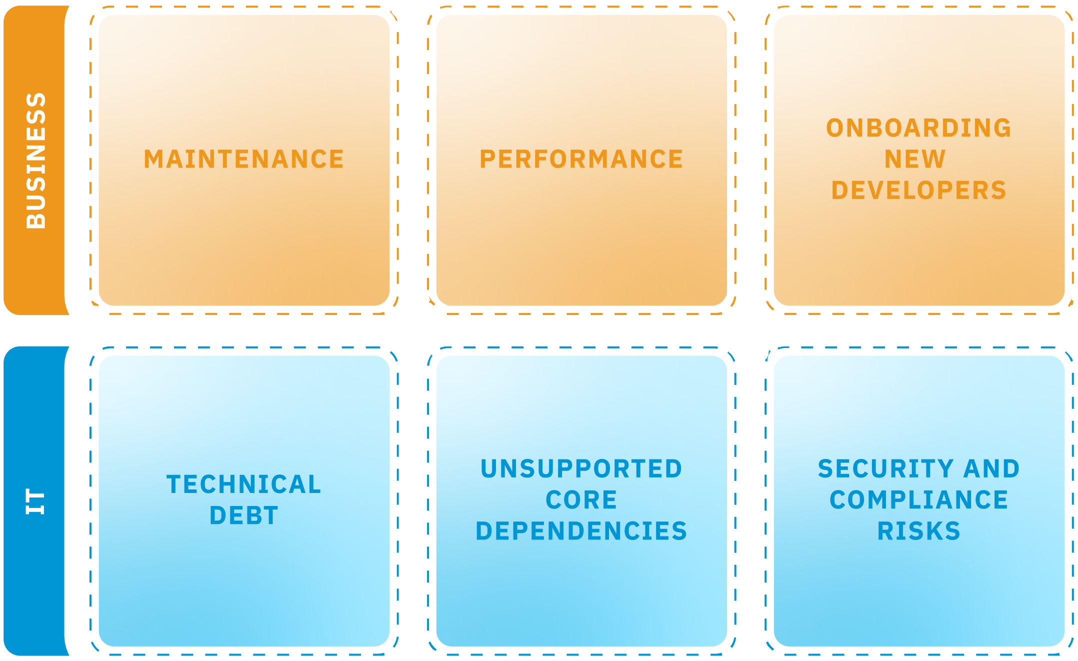
What Is A Legacy Application?

A legacy application can be defined as one that was developed with older technologies and/or using outdated development paradigms. For example, Gartner defines it as “an information system that may be based on outdated technologies, but is critical to day-to-day operations” of an organization. A legacy application then, is one that is currently being used by an organization but is in need of modernization within the context of contemporary application development and deployment paradigms.

A legacy application doesn’t have to necessarily be one written in an older programming language. An application written in a contemporary language like Java can be considered legacy if the underlying technology, like J2EE is no longer available and/or supported. In effect, an application can be classified as legacy when it is unable to take advantage of modern digital transformations enabled by the cloud to deliver exceptional customer experiences at reasonable costs to the organization.

A legacy application will generally be working and delivering business value as it was intended, but most likely it will be at a much more diminished rate relative to the rate of change in customer expectations. For instance, a forty-year-old insurance policy and claim management system written in COBOL can still be used to manage insurance products but it will be almost impossible or extremely expensive to use to deliver other, more modern customer experiences, like self-service, mobile and SMS integration among others.

Identifying an application as legacy and, as such, in need of modernization isn't always so straight-forward for organizations. The following points will help you assess the need for modernization of an application. The more an application ticks these checkmarks, the greater the likelihood of it being considered legacy.


BUSINESS
MAINTENANCE
PERFORMANCE
ONBOARDING NEW DEVELOPERS
IT
TECHNICAL DEBT
UNSUPPORTED CORE DEPENDENCIES
SECURITY AND COMPLIANCE RISKS

Legacy Application Evaluation

An application can be classified as legacy by a critical analysis of the overall value it brings to the business. This value analysis should consider two thematic areas: business and IT.

Business

From the business perspective, is the application a good fit, delivering business value and agile enough in its ability to meet ever changing customer expectations brought about by digital transformation? The following points should be analyzed from the business perspective.

Maintenance

How hard is it to maintain the application? How fast can bugs be fixed? What is the turnaround time for adding new features? Do the costs of doing these grow exponentially or linearly? How satisfied are the staff maintaining this application? Is the application living up to expectation in delivering its primary function?

Performance

How well does the system perform? Are end users of the system satisfied with its performance? Are the clerks using the system happy with how fast it allows them to get their jobs done? How well does the application scale to meet a surge in demand?

Onboarding New Developers

How fast can we onboard new developers? Are there enough developers with the skills to maintain the application? How fast can we replace existing developers if and when they leave?

IT

From the IT perspective, what are the costs associated with running and maintaining the application? How complex is every development task that needs to be carried out on it? What are the security risks associated with the current state of the application?

Technical debt

How much technical debt does the application have? How much is piled on with each feature or bug fix? How much does this debt impact the performance of the application? How much will it impact future development?

Unsupported Core Dependencies

Are there external core application dependencies that are no longer supported? Are there any integrations that are no longer available to the application? Can the application keep delivering its core value in the absence of these integrations? Can these integrations and external libraries be replaced? How much will it cost to replace them?

Security And Compliance Risks

How secure is the application? How much of the underlying infrastructure is no longer developed or maintained? How up to date are the core components of the application? How compliant is the application with regulatory requirements?

As you can see, some factors can intersect the business and IT perspectives. For instance, application maintenance transcends both business and IT perspectives. The more an application ticks these checkmarks or scores poorly on them, the greater the likelihood of it being classified as legacy and thus in need of modernization. Once the state of an application is clarified, the next logical question is what is legacy application modernization? And what benefits can it bring to an organization that undertakes it?

Legacy Application Modernization

After identifying an application as legacy, the next step is to modernize it. Legacy application modernization is a catch-all term that encompasses the gamut of processes involved in identifying an application as legacy to reworking it to take advantage of digital transformations and deliver superior customer experiences.

Technically, legacy application modernization, according to VMWare is "the practice of updating older software for newer computing approaches, including newer languages, frameworks and infrastructure platforms." Legacy application modernization then, is a way of renewing an old system with cloud native capabilities to deliver exceptional customer experiences.

Advantages Of Legacy Application Modernization

Identifying a system as legacy and accepting that it needs to be modernized may not necessarily be enough to get buy-in from all decision makers. So let us explore some advantages that can be attributed to legacy application modernization.

Costs Reduction

There are a lot of expenses associated with running a legacy system. These costs can include:

Specific Hardware

Certain old systems were designed to run on specific hardware. Such hardware might not be readily available on the market and thus very expensive to acquire and maintain. Modernizing the application could eliminate this cost driver because the modernized application can run on readily available commodity hardware.

Slower Turnarounds

The entire development cycle for a legacy application might be very slow due to the arcane processes involved in each step. Adding a new feature or fixing a bug might take a significant amount of time. Modernizing the application can help it take advantage of current DevOps paradigms for a faster turnaround.

Dearth Of Developers

Legacy systems might be built with very outdated frameworks or programming languages that have a dearth of competent developers to carry on maintaining the system. Modernizing such a system will help reduce such costs by taking advantage of readily available developers once the application is updated to a much current infrastructure.

Performance

A legacy application might have major performance bottlenecks due to a combination of factors, such as technical debt, outdated hardware, bugs and outdated runtimes. Modernizing legacy applications with cloud native capabilities can allow developers to take advantage of the commoditization of cloud infrastructure. As a result, it is easier to scale applications to meet demand, with the entire system becoming more resilient while increasing its uptime significantly.

Security

Modernizing legacy applications can eliminate whole surfaces of security risks, such as outdated libraries, runtimes and language. It can also support real-time application monitoring to enable faster reactions to security threats. For instance, a cloud native modernized legacy application can use modern continuous security scan services that will keep watching over all layers of the application, such as application code, deployment containers and platform. Modernizing to current versions of all application components means the application will have timely security patches and bug fixes for all such components.

Better Employee Experience

Oftentimes employees that work on legacy systems may not be very satisfied with their jobs because of how tedious dealing with such outdated software can be. Modernizing legacy applications with cloud-native capabilities can increase employee satisfaction. This, in turn, can help reduce employee churn.

Deliver Superior Customer Experience

Modernizing legacy applications for the cloud can enable an organization to deliver superior customer experience. A cloud-enabled application can take advantage of the massive array of cloud services to help predict, meet and exceed customer expectations, thereby increasing customer satisfaction, reducing the customer maintenance costs and increasing overall profitability.

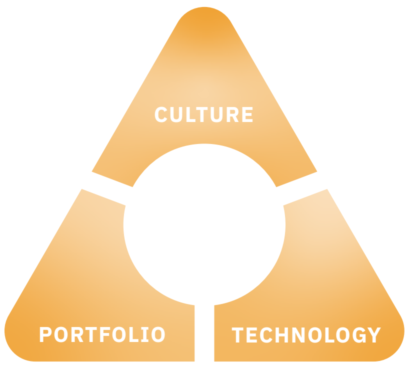
Reduced Compliance Risks

Every organization operates within some regulatory framework. And there may be key requirements that the organization must meet as part of delivering its core mandate. Legacy applications can be hard to amend to keep up with changing regulations. Modernizing them can significantly reduce the cost of regulatory compliance, as they can become much more amenable to new regulations. This, in turn, reduces the risk of incurring fines for regulatory breaches.

A study by NTT Data Consulting found that 63% of surveyed banks struggled integrating with external systems, and 78% of spendings on core deposit systems is on legacy application maintenance and compliance. Such major challenges leave little room for innovation, which can eventually lead to stagnation in quality service delivery.

This is by no means an exhaustive list of advantages that can be derived from modernizing legacy applications. Every organization can realize a lot more benefits that will be specific to their own context. Once the gains of legacy modernization have been considered, the next step is to explore a broad framework for planning and carrying out the modernization efforts.

Modernization Framework



Legacy application modernization can be a very complicated and expensive exercise that must be carried out with as much planning as possible. This section offers a framework that can help you plan a path to modernizing your legacy application. This consists of three broad thematic areas: culture, portfolio and technology.

Culture

Depending on the size of the specific application, modernizing a legacy system can be an exercise that impacts everyone in the organization. As such, it is essential to start the process by aligning the company culture to the process. Getting everyone in the company to be onboard, from bottom up, is critical to ensuring the success of the process. It is also important to set clear expectations before, during and after the process. This is because the modernization efforts, depending on the actual process chosen, could result in significantly different software. This means current users need to have their expectations calibrated to avoid major cognitive dissonance as a fallout of the modernization effort.

Internal teams may also need restructuring as part of the modernization process. This is important to communicate upfront and get buy-in from all concerned parties before the actual process begins. Having everyone aligned on how to proceed and what to expect is the single most crucial part of the process and should be handled with utmost care and tact. As every organization is the sum of its people, getting their buy-in before undertaking a significant exercise such as a modernization of a legacy application is paramount, as the process can have far reaching effects for everyone.

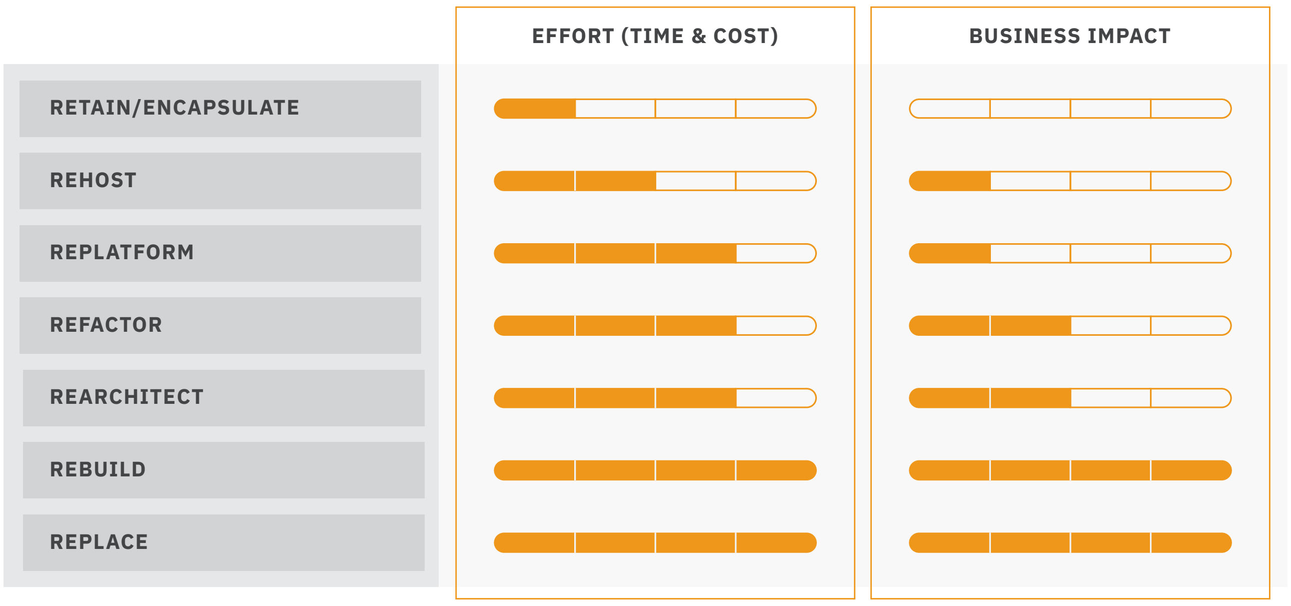
Portfolio

Legacy application modernization is a straightforward process if your organisation has a single application under consideration. However, when there are more than one, it is important to prioritize the resources available for the process in a way that maximizes the potential outcome. Legacy applications in the organization's portfolio should be prioritized in order of greatest returns and lowest cost. Such a matrix will help determine which application should be modernized first.

As almost every organization has limited resources, allocating some to a complex process as application modernization should be carried out after extensive considerations of all legacy applications in the portfolio. This will help allocate resources to the project with the best chance of succeeding.

Technology

Technology is the third element in the framework trio for effective legacy modernizations. It involves considerations on the various options available for the actual process. As every organization (and application) is different, there are multiple ways to carry out a modernization. Depending on the ultimate goal, a given technology choice or combination of choices can be made. There are seven broad options that can be used to modernize a legacy application. These are not mutually exclusive and can be combined. Let us explore them, from the least complicated and most economical to the most complex and expensive.



Retain / Encapsulate

This option entails retaining the current legacy system and exposing its core functions for consumption by greenfield microservices (or even monoliths) with built-in cloud-native features. As a result, the core features of the legacy application are abstracted behind microservices, which can offer all the benefits of cloud-native applications for users. Retaining/encapsulating is generally the easiest and most economical option. The legacy system can still be maintained/hosted as is, while applications consuming the exposed core functionality can be deployed to the cloud for better scalability, performance and resilience.

Rehost

As one of the cost drivers of running a legacy system is related to hardware and infrastructure, modernizing a legacy application could entail rehosting it on a different hardware infrastructure. The new infrastructure could be a new physical, virtual or public cloud platform. The goal is to optimize the legacy application to take advantage of hardware for better performance and security.

Replatform

Replatforming means migrating the legacy application to a more recent version of its runtime without making any extensive changes to the code and its structure. As such, this strategy involves updating the underlying application's framework, library or platform to take advantage of new features and improvements. For example, a legacy application written in Java 5 on the J2EE platform can be migrated to Java 11 and Jakarta EE 8 with minimal changes to the code structure. Such a seemingly minor update means lots of improvements for the application with all the optimizations that have gone into the JVM from Java 5 - 11. Upgrading to Jakarta EE 8 also means escaping all the security flaws present in the long unsupported J2EE runtime.

Refactor

Refactoring entails optimizing and restructuring the application internally, without impacting its external functionality. The goal of this type of modernization is to improve the internal performance and maintenance of the application without causing significant downtime. Refactoring and replatforming can be phased in a staggered modernization approach, where replatforming comes first, followed by refactoring after a period of time.

Rearchitect

Rearchitecting means fundamentally altering the code of the application to adopt a new architecture, enabling it to fully leverage the benefits of the new design. For example, the J2EE application mentioned above can be rearchitected into a set of cloud-native microservices that, together, perform the same function as the current legacy application. As microservices however, the rearchitected application can take full advantage of the benefits offered by the cloud to enhance the customer and user experience of the application.

Rebuild

Rebuild entails redesigning or completely rewriting an application from scratch while preserving its core functions. Rebuilding differs from rearchitecting in that it changes how the application's building blocks are arranged while rearchitecting involves a sort of one-to-one rewrite. Rebuilding can be challenging because it should maintain the architecture of the application but rewrite it to take advantage of new the capabilities offered by the underlying language, framework or runtime.

Replace

Replacing is the most expensive and hardest of all the modernization options. It consists of eliminating the current legacy application and building a new one that considers current requirements. Fully replacing a legacy application is a very complicated and expensive exercise that can take a significant amount of time to complete. However, it is also the most effective of all the modernization options.

Deciding which modernization effort to pick is a domain-dependent decision. However, care must be taken to choose the most effective for a given context. For example, companies may be tempted to adopt an option without careful evaluation that might prove much more costly along the way than the cost of running the legacy application.

A Note On Microservices and Monoliths

Undoubtedly, almost all legacy applications are large, complex monoliths that were written years ago. Monolithic architecture was generally the de facto architecture of the time. You might be tempted to blindly replace your legacy application with microservices, as this is the current and most popular architectural option. However, it is much more valuable to pick an architecture based on the business value it brings in within your organizational context.

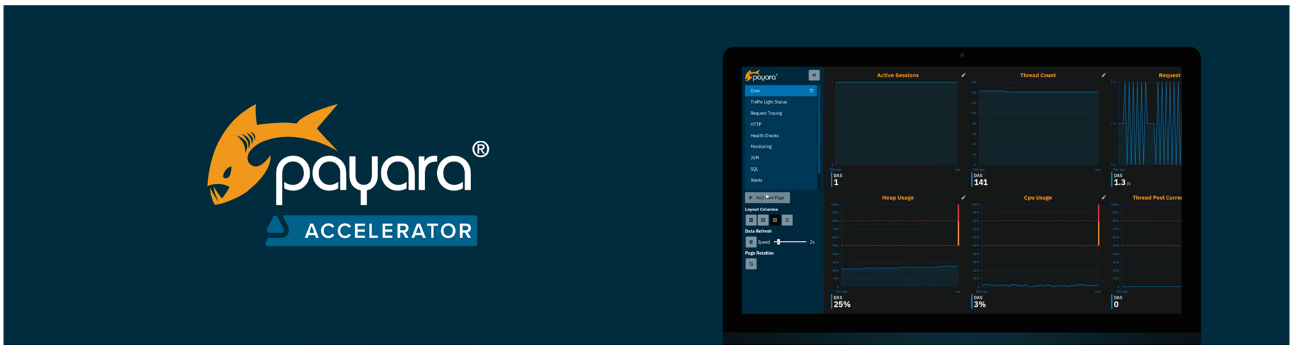
A legacy monolith application can be effectively modernized without needing to rewrite everything as microservices. Platforms like Jakarta EE allow all types of applications to take advantage of cloud-native features and benefits without tying you to a given architecture. In the end, to decide which modernization effort to choose is important to assess the legacy application within the context of organizational goals. Doing so will clarify what modernization costs the company can and is willing to incur.

Conclusions

Legacy application modernization is something every organization will eventually have to contend with as systems age. In fact, legacy applications will start costing more than the revenue they bring in the older they get.

The decision to breathe new life into existing software at the lowest cost possible is one that requires a lot of contextual analysis. The decision to modernize an application is a far-reaching one that should be taken within a framework that takes into account culture, portfolio and technology. By aligning company culture, setting out priorities on what to update first and identifying the technologies that are best suited to the specific applications, organizations can significantly increase the chances of a successful modernization exercise.

If you are currently considering legacy application modernization and need technical expertise and support to kickstart or complete the process, [Payara Accelerator]() can help you optimize and accelerate the project for better profitability. Reach out to us and we will be delighted to help you optimize your application stack for the cloud-native era to deliver superior applications and customer experience.



sales@payara.fish

UK: +44 800 538 5490
Intl: +1 888 239 8941

www.payara.fish