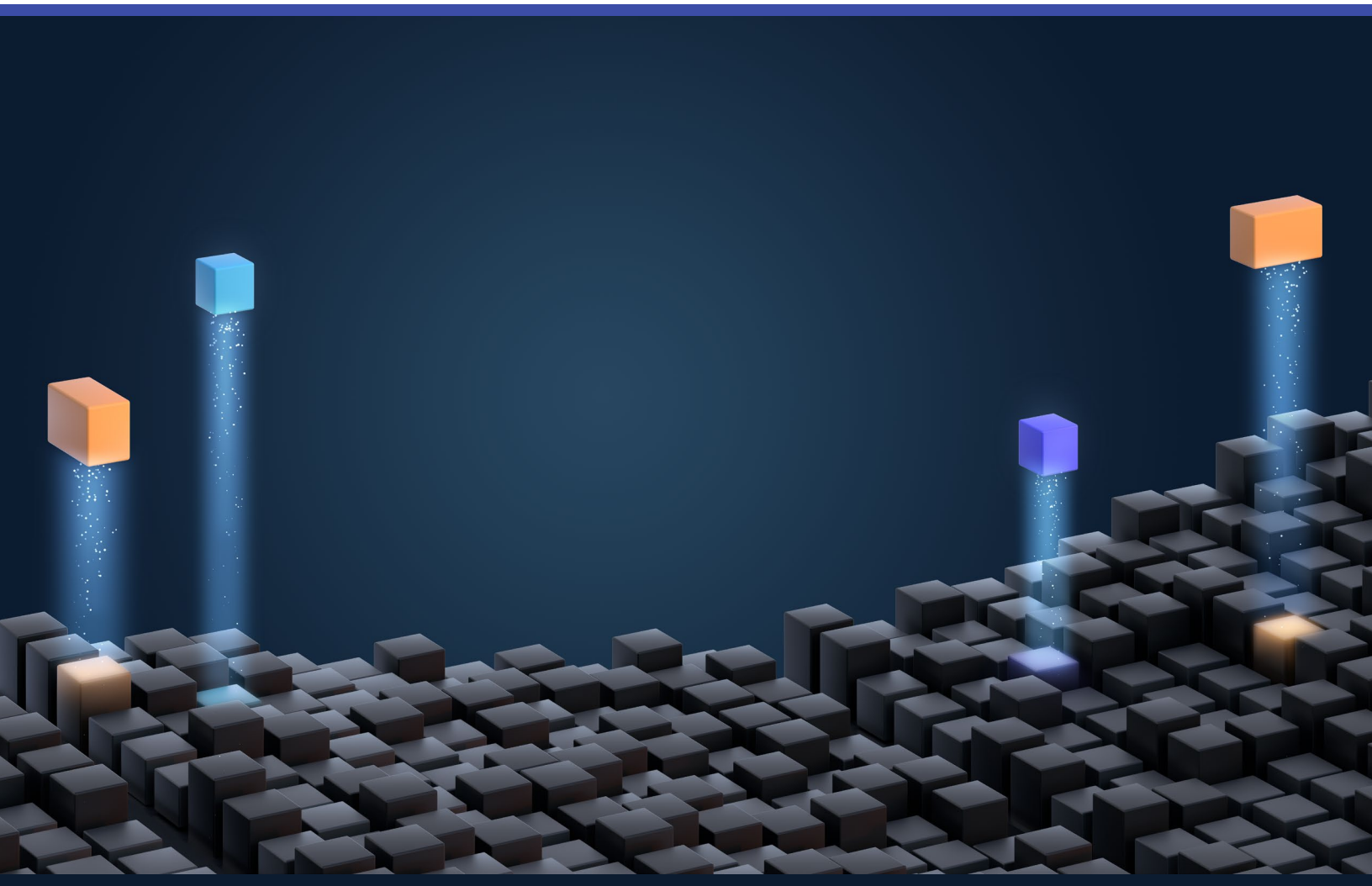
OpenLegacy

White Paper

How to Quickly Deliver Digital Services From Legacy Systems

Ensure Your Future Competitiveness



www.openlegacy.com

01

Introduction

"Legacy systems" like mainframes and midrange systems are common in well-established companies who have been building their markets and brands for decades. Emerging tech companies are forcing established enterprises to more closely monitor their competitiveness and find new ways to address long-standing technical challenges.

Using technological innovation, new players in almost every industry are finding new revenue sources in avenues that the established enterprises have less incentive to exploit. By streamlining and automating processes, these new companies can squeeze out savings and efficiency that the larger enterprises cannot match.

Some highly regulated industries, such as banking and insurance, are seeing emerging technology companies design compliance with the new data privacy regulations into their applications and interfaces, something that the incumbents are hard-pressed to do.

For example, "InsurTech startups are testing the waters on a host of potential game changers. These include using deep learning trained AIs to automate and handle the tasks of brokers and find the right mix of policies to complete an individual's coverage. There is also interest in the use of apps to pull disparate policies into a single platform for management and monitoring, creating on-demand insurance for micro-events like borrowing a friend's car, and the adoption of the peer-to-peer model to both create customized group coverage and incentivize positive choices through group rebates."

Ironically, some of the world's largest, oldest and most successful companies face the greatest technical challenges as they explore ways to bring their legacy, back-end systems up-to-speed with the current digital era. Connecting legacy systems to the digital world using APIs and microservices is proving effective.

While 82% of insurers have digital solutions, the bad news is only 32% think they are being effectively used.[1] Various legacy applications were developed in an age where no one anticipated the digital requirements of today. For example, most legacy IBM i (AS/400), midrange systems and applications depend on third-generation programming languages (3GL) such as COBOL or RPG, which are so out-of-date that the industry is running out of professionals who understand them.[2] Furthermore, technology investments, such as Service Oriented Architecture (SOA) and Enterprise Service Bus (ESB), were the best technology options at the time, but now fail to keep pace with the digital world.

55%
of all consumers prefer a digital experience[1]

32%
of insurers think their digital solutions are being effectively used.[1]

Insurance companies need to have the lowest rates, and provide them in milliseconds. They need to provide top-tier customer service—and provide insurance agents with instant customer history during every interaction. No matter the competitive strategy , you still have to serve your customers in best way possible—this requires innovation.

Several recent case studies show the specific ways in which these legacy applications are delaying innovation and improvement to the customer experience.

- AIG found their IBM i-powered application was unable to serve quotes to aggregation services. Even after six months of development, it still took up to three seconds to deliver quotes—long enough that most aggregator services still refused to accept their input. Using APIs to create web services solved this for AIG.

- Union Bank needed the ability to create and deploy APIs and microservices quickly and automatically for their long-term growth and differentiation. They needed to seamlessly integrate their legacy mainframe system into their new event-driven architecture, but due to layers of complex middleware, this was expected to take months. APIs and microservices were used to connect directly to CICS transactions.

- Virtually every department in Liberty Mutual needs to access data for compensation and claims management. Over the decades, hundreds of green screens were built to accommodate a multitude of users across divisions with thousands of different scenarios. The green screens were old and slowed down data access and entry, significantly affecting productivity and ultimately sales. Modern approaches separate the presentation layer from the business logic, allowing process modernizations and improvements.

Competing based on price isn't enough to win new customers and decrease churn anymore. In a worrisome survey for the insurance industry, 41% of insurance consumers indicated that they'd switched companies because their old carrier was lagging behind the times.[4] How can insurance companies keep up?

Insurance companies need to have the lowest rates—and provide them in milliseconds. They need to provide top-tier customer service—and provide insurance agents with instant customer history during every interaction. Insurance companies need a solution that can rapidly bootstrap their legacy applications to a state capable of this new demand for information and speed.

Digital services can help organizations quickly innovate in the near term and replace or modernize their legacy applications on a more relaxed timetable, without the difficulty, expense and risk of failure.

02

A brief introduction to Digital Services

For clarity, here are some definitions of the relevant terms and concepts.

APIs: APIs have been around for a long time, and in fact have been behind some of the biggest innovations in history, such as the Google API that enables third party organizations incorporate map data into their websites and application. In this regard, an "application programming interface" did just that—it enabled data from one location to be integrated into another.

In general, there are three kinds of API—private, partner, and public. Private APIs expose data from legacy systems to agents within the company, as in the example above. Partner APIs expose data to vendors, and public APIs expose data to customers. Organizations that are new to the technology typically start with private APIs.

	SOA	Digital Architecture
Primary Challenge	Reduce integration costs Breaking down walls between legacy, monolithic, vertical systems	Increase speed of delivery Increase the opportunity for concurrent development
Additional Benefit	True multichannel or omnichannel experiences Efficiency and reuse in application assets	Cloud-native scalability, reliability, and fault-tolerance Omnichannel experiences and augmented experiences Efficiency and reuse in application assets
Attitude to Code	Code is expensive and slow to build, test, and deploy Externalise workflow, rules, and orchestration to a middleware layer	Code is cheap and fast to build, test, and deploy Code is faster and more flexible than rules and workflow. Make code faster! Integration of RESTful services can be done efficiently at the client and in additional microservices

Table 1. Comparing SOA and Microservices Architecture[3]

APIs are essentially a software contract. When an API receives a request phrased in a certain way, it retrieves a certain kind of information. In an insurance context, imagine a small software program that retrieves a customer's name, address, and claims information when an agent inputs their ID number.

Some APIs are easier than others, depending largely on what data needs to be shared, and where the data is stored. Traditionally APIs from legacy systems required developers to navigate their way through complex layers of technology to access the legacy data.

Modern approaches, such as OpenLegacy, are designed for legacy systems and automate most of the hard coding labor.

Digital Services: Digital services are a way of delivering information through the cloud. A microservice architecture helps deliver digital services faster and is a method of developing software applications as a suite of independently deployable, small, modular units. Netflix, eBay, Amazon, the UK Government Digital Service, Forward, Twitter, PayPal, Gilt, Bluemix, Soundcloud, The Guardian, and many other large-scale websites and applications have all evolved from monolithic to a digital services architecture.

An API does not require a microservices architecture, but there are some benefits of doing so in many cases. "Architects who are proponents of a microservices architecture will often cite the availability and responsiveness of systems built in this style—perhaps citing Netflix and other massive scale, high availability systems. Chief Digital Officers and business leads, however, typically talk about the speed of change that can be brought through using processes, libraries, and tools associated with microservices architectures," says Celent.

One of the major drawbacks of developing for legacy architecture is the fact that any change in the underlying legacy application will be reflected in the more modern components—often in ways that break the entire application. Microservices remove this constraint.

The use of microservices is intended to result in a fully modular application. If a single digital service fails, the rest of the application will continue to function. If a single digital service is updated, added, or deleted from the overall application, the rest of the application should need no further adjustment. The application should still function if its component digital services are located on different servers and in different clouds. Different application components can scale independently.

Many developers, upon the first introduction to microservices, will immediately compare them to SOA. While SOA is another kind of service, and the end result of SOA often ends up looking a lot like microservices architecture, the process of developing SOA is quite different, and is often unsuitable for cloud-native development.

On the face of it, SOA provides a loosely-coupled architecture that lets developers build resilient applications which are independent of legacy infrastructure. In practice, however, the development precepts of SOA make it unwieldy for large enterprises. SOA is not independently-deployable like microservices. The application components in SOA can share services among themselves, while also remaining part of the same application.

In other words, SOA adds APIs to the enterprise by combining them into a monolith, and then places that monolith as an additional layer across pre-existing monoliths. As a result, coding, testing, and deploying SOA can be just as cumbersome as before, even though the result can appear similar to a microservices architecture.

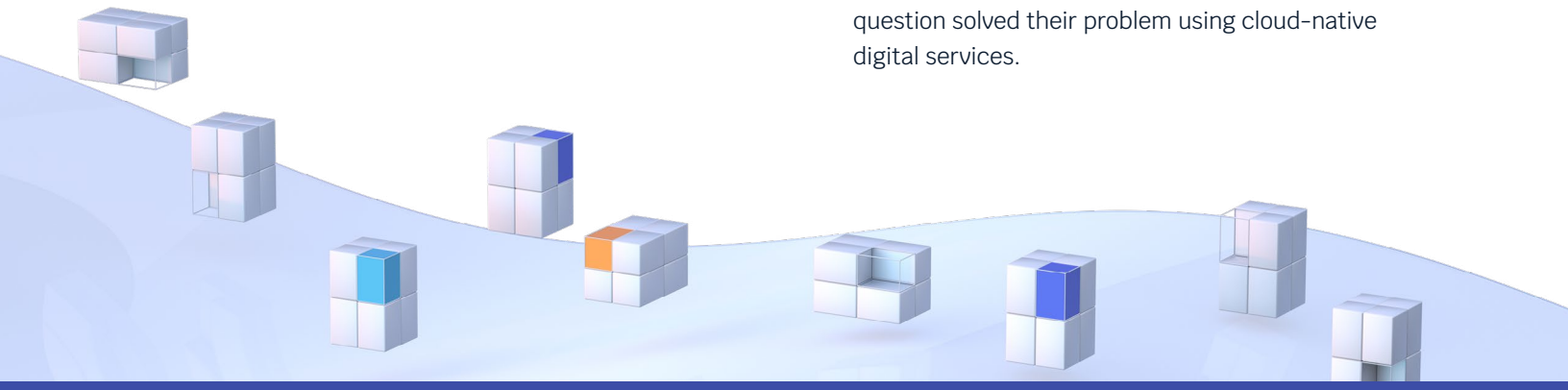
As far as digital services are concerned, the most important attribute for organizations is not just the speed of development, but also the fact that different parts of the application can be developed at different speeds. One of the major drawbacks of developing using legacy architecture is any change in the underlying legacy application will be reflected in the more modern components—often in ways they break the entire application.

Microservices remove the speed constraint. Microservices have only a loose dependency on legacy applications. If the legacy application is changed, the microservices components will keep running without errors, and vice versa. This lets independent development teams make changes to their applications, without slowing down to meet another team's pace. As a final result, customers enjoy new features and improved functionality at the rate they've come to demand.

03

Placing microservices and APIs in context

Looking back at the introduction, there are three circumstances where companies were stymied by the task of adding modern functionality to a legacy infrastructure. In every instance, the company in question solved their problem using cloud-native digital services.



Problem	Solution
Serving quotes to insurance aggregators from an IBM i-based application	AIG used an API that to define web services on IBM i transactions. This extended the reach of their legacy systems into the cloud. They served quotes to browser-based applications in just 300 ms and are now included in all major insurance aggregators.
Integrating a legacy mainframe system with a modern event-driven architecture	Union Bank decided to implement an event-driven architecture using Apache Kafka. They were able to connect directly to the CICS and create and deploy APIs and microservices, quickly and automatically, and bypass complex middleware. In addition, they were able to directly call an external API from a CICS mainframe application, allowing bi-directional operations between the mainframe and the digital architecture.
Modernizing an antiquated claims insurance application based on COBOL	Liberty Mutual exposed a service from their claim management system that presented all the reports related to a specific claim within the web applications. The initial proof-of-concept was completed in just five days. In production mode, insurance agents realized a time savings of up to 30%.

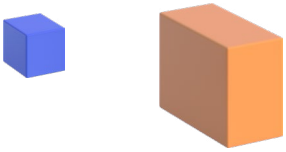
Table 2. Problems and API Solutions

When companies begin their digital services adoption, some truly unique innovations are introduced. A good real-world example is Liberty Mutual. The insurance giant created an API portal that delivers car accident data to AI and machine learning developers. One of the first results is expected to be an application which lets customers automatically appraise damage to their vehicles—simply by taking an image of a car accident with their smartphone camera. This kind of deeply-sophisticated integration, involving a complex chain of private, partner, and public digital services with added mobile integration, is just the tip of the iceberg when it comes to all that this technology has to offer.

04

When and when not to use digital services

Digital services can strongly benefit companies, but unlocking value from microservices means knowing when and where to apply them. It also means learning where microservices and APIs are most beneficial to the overall software architecture.



	Prerequisites
The Obvious or Hard Prerequisites	• Use of cloud or cloud-style infrastructure that enables swift, automated deployment • Adoption of DevOps processes and tools • Tools, frameworks, and/or libraries that allow for quick delivery of small applications
The Less Obvious or Soft Prerequisites	• Agreed expectations around speed of delivery of services to production environments • Understanding about the implications of eventual consistency and the accuracy of data returned in different environments • Hiring, acquiring, or partnering for multiskilled developers able to deliver functionality in this way • A capacity to deliver changes at different speeds in different environments

Table 3. Prerequisites to Adopting a Microservices Architecture.[3]

05

Legacy system: Love it or leave it?

While companies grapple with the really big questions, such as “Should I modernize or rip and replace my legacy systems?” and “What is the risk of failure?” Digital services help answer another big question: “How do we innovate faster, today?”

For some, this lightweight, open standards approach is a long term solution. For others, digital services facilitate future modernization or rip and replace options. Either way, APIs are a viable, reliable way to get business done collaborate with Insurtechs and block competitors from going after your customers.

Digital services are a viable, reliable way to get business done to block Insurtechs and other competitors from going after your revenues and your customers.

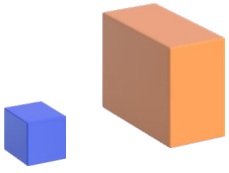
06

The Cloud: Fluff or substance?

Actually, the answer is both, but already leaning toward substance. As with all technologies that force a paradigm shift, the initial adoption phase is often characterized by hype and buzz.

Cloud technology has matured sufficiently that we can already gain from its advantages, yet it cannot wholly replace your legacy and core systems, which will remain important for the foreseeable future.

By automatically generating cloud -native microservice-based APIs and serverless functions with direct connections to these backend systems, OpenLegacy Hub allows you to benefit from the advantages of both your legacy sources and cloud-ready functions.



OpenLegacy Hub provides an automated and standard way of functionally-decoupling a monolithic core and legacy applications by creating a microservices architecture without the need to change the legacy application. With decoupled microservices, you can launch the digital services you need, while making it easy to modernize and migrate off the legacy system in the future because the microservice is a defined interface that is easy to leverage.

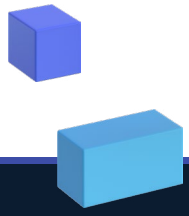
07

OpenLegacy on AWS

OpenLegacy Hub offers an innovative way of creating hybrid integrations between AWS and the most complex core and legacy systems by automatically generating microservice-based APIs and serverless functions with direct connections to these backend systems.

OpenLegacy unleashes the data and functionality locked in core and legacy systems by rapidly creating digital services decoupling monolithic applications into lightweight microservices and other end points. This enables customers to maximize the benefits of a composable enterprise and extend AWS's performance, scalability, reliability, and availability to all core and legacy systems.

OpenLegacy Hub supports serverless deployment with quick generation of NodeJS as Lambda functions, as well as container-based deployment through the automatic generation of cloud-native microservices-based APIs.



[1] *https://worldinsurancereport.com/wp-content/uploads/sites/6/2021/07/World-Insurance-Report-2021.pdf*

[2] *Reuters, "Banks Scramble to Fix Old Systems as IT 'Cowboys' Ride into Sunset," April 2017*

[3] *Celent, "Honey I Shrunk the Services: Microservices and Insurance," December 2015*

[4] *IBM, "Capturing Hearts, Minds, and Marketshare: How Connected Insurers are Improving Customer Retention," May 2015*

[5] *https://www.investopedia.com/terms/i/insurtech*

About OpenLegacy

OpenLegacy offers a cloud-first legacy modernization platform. OpenLegacy Hub delivers high ROI with a simple, disruption-free, method to generate, extend and manage digital services from legacy systems to the cloud. Jumpstart and optimize your modernization journey and follow it through, no matter the chosen strategy: modernizing in place (hybrid), rehosting/replatforming or even replacing and rewriting the entire application. Each can be simplified and automated to perfect the process drastically eliminating complexity, time, cost, and risk. OpenLegacy is used by many of the world's leading enterprises, including Citi, Scotiabank, Liberty Mutual, DBS, and Standard Chartered, to name a few.