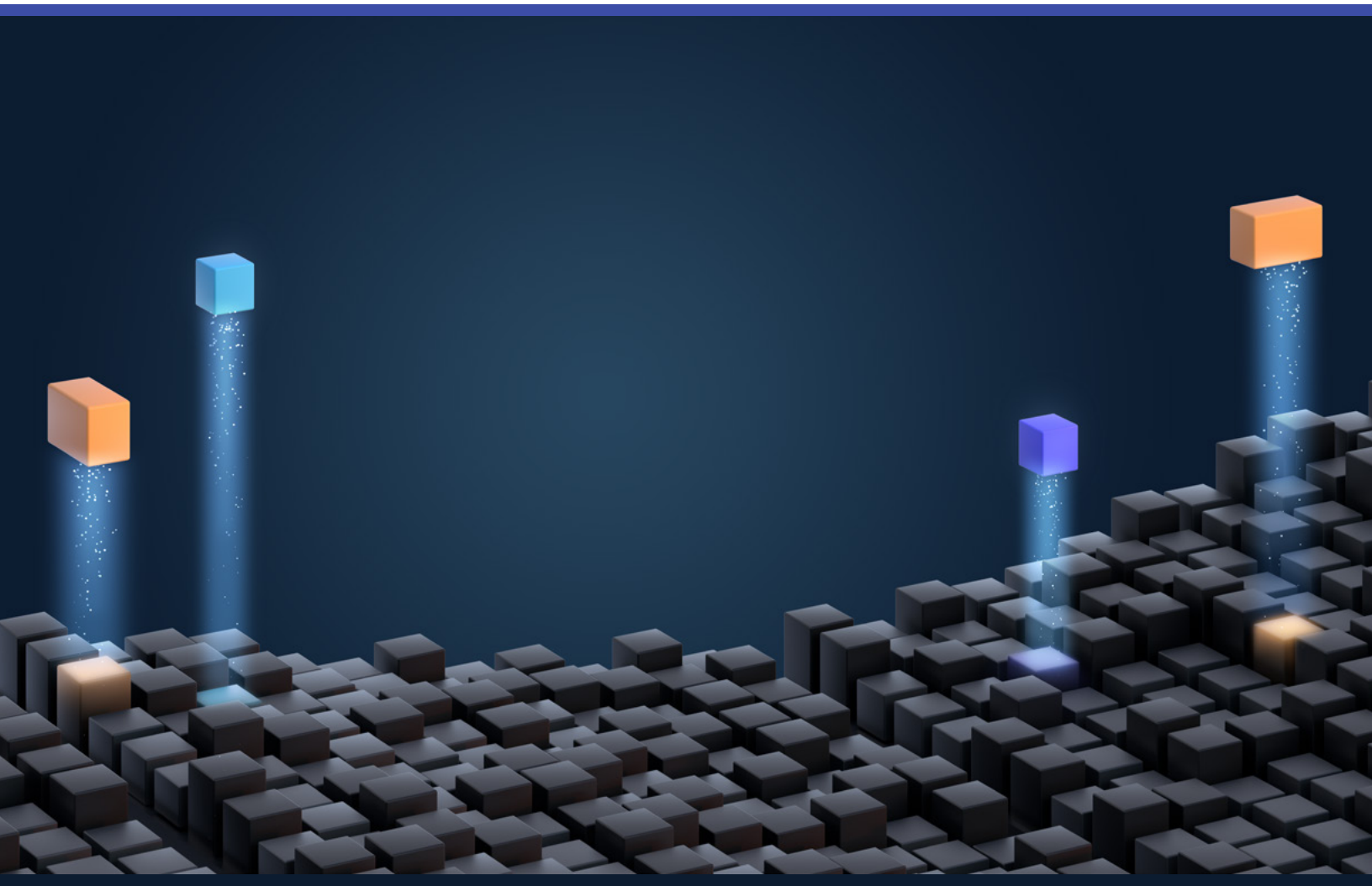
OpenLegacy
Digital to the Core

White Paper

How to quickly deliver digital insurance services from monolithic legacy systems

Ensure your future competitiveness in the age of ‘InsurTechs’



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. However, like the "FinTechs" (new, innovative finance technologies) who are starting to threaten the status quo in the banking industry, "InsurTechs" are forcing incumbent insurance companies to more closely monitor their competitiveness and find new ways to address long-standing technical challenges.

According to Investopedia, "InsurTech" refers to the use of technology innovations designed to squeeze out savings and efficiency from the current insurance industry model. InsurTechs are exploring avenues that large insurance firms have less incentive to exploit, such as offering ultra-customized policies, social insurance, and using new streams of data from internet-enabled devices to dynamically price premiums according to observed behavior.

"In addition to better pricing models, InsurTech startups are testing the waters on a host of potential game changers. These include using deep learning trained AIs to 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 one 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 insurance 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.

Not surprisingly, almost 60% of insurance executives agree that they are falling behind the times technologically.[1] Various legacy applications were developed in an age where hardly anyone could have anticipated the digital requirements of today. For example, most legacy IBM i (commonly known as AS/400), midrange systems and applications depend on fourth-generation programming languages (4GL) 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, and add complexity and layers.

41%
of insurance consumers switched companies because their old carrier was lagging behind the times[6]

60%
of insurance executives agree they are falling behind technologically[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.

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

- A major publicly-traded insurance company developed an "agent portal" which exposed business processes for its workers. The organization wanted to add more services to the portal, but the portal was based on an IBM i application. This meant that adding more services took months at a time—and in the meantime, the company's competitors were gaining.[3]

- Insurance aggregators have become a critical way for companies to get their price quotes in front of consumers. Unfortunately, one company found that their AS/400-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.[4]

- Although an insurance company was able to modernize most of their offerings, their auto insurance offering was left as a legacy application. As a result, agents were forced to switch between a web browser and an antiquated "green screen," which impacted productivity and customer responsiveness.[5]

Having the lowest rates 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.[6] 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.

Application Programming Interfaces (APIs) and microservices are worth careful consideration for many organizations facing these concerns. "Microservices and microservices architectures will certainly have a place in the future architectures of insurers, insurance software vendors, and most likely any participant in the insurance industry," says analyst firm Celent.

APIs and microservices 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 APIs and microservices

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	Microservices 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
Source: Celent, "Honey I Shrunk the Services: Microservices and insurance," December 2015

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's API Integration and Management software, is designed for legacy systems and automates most of the hard coding labor.

Microservices: Essentially, microservice architecture is a method of developing software applications as a suite of independently deployable, small, modular services in which each service runs a unique process and communicates through a well-defined, lightweight mechanism to serve a business goal. 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 microservices 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.

The use of microservices is intended to result in a fully modular application. If a single microservice fails, the rest of the application will continue to function. If single microservice is updated, added,

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.

or deleted from the overall application, the rest of the application should need no further adjustment. Lastly, the use of microservices should result in a cloud-native application. The application should still function if its component microservices are located on different servers and in different clouds. Different application components should even be able to 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 insurance companies.

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.

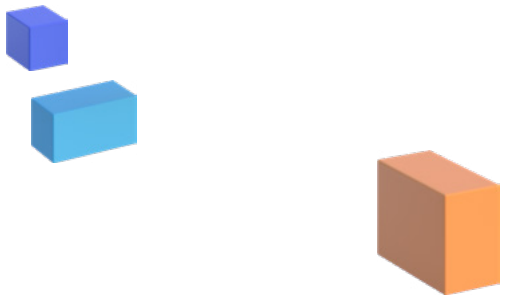
As far as microservices are concerned, the most important attribute for insurance organizations is not just the speed of development, but also the fact that different parts of the application can be developed at different speeds. One of the major drawbacks of developing for legacy architecture is the 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 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 independent changes to their applications, without slowing down to meet another team's pace. As a final result, customers will be able to enjoy new features and improved functionality at the rate they've come to demand.

03

Placing microservices and APIs in an insurance context

Looking back at the introduction, there are three circumstances where insurance companies were stymied by the task of adding modern functionality to a legacy infrastructure. In every instance, the company in question was able to solve their problem using microservice-based APIs:



Problem	Solution
Adding services to a portal based on an AS/400 backend	The insurance company was able to use an API connector which automatically scans and parses green screens in order to take the output from their IBM i application and place it into the context of their portal. No COBOL modification or re-write was required, and the first new service was in place within hours. As a result, their agents were able to rapidly understand the new interface and perform job responsibilities more productively.
Serving quotes to insurance aggregators from an AS/400-based application	The insurance company was able to use an API that defined web services on top of AS/400 transactions. This let them extend the reach of their legacy systems into the cloud. They were rapidly able to serve quotes to browser-based applications in just 300 ms and are now included in all major insurance aggregators.
Modernizing an antiquated auto insurance application based on COBOL	The insurance company was able to expose a service from their AS/400 claim management system that presented all the reports related to a specific claim within the main auto insurance web applications. The initial proof-of-concept was completed in just five days. In production mode, insurance agents were able to realize a time savings of up to 30%.

Table 2. Problems and API Solutions

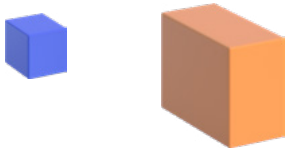
When insurance companies begin API and microservices adoption, some truly unique innovations are introduced. Another good real-world example is Liberty Mutual. The insurance giant has created an API portal that delivers car accident data to AI and machine learning developers. One of the first results 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 API integration, involving a complex chain of private, partner, and public APIs with added mobile integration, is just the tip of the iceberg when it comes to all that this technology has to offer.

In these examples, three different insurance companies were able to quickly leverage decades-old legacy systems for modern digital applications—without any changes to the systems themselves.

04

When and when not to use microservices

Microservices and APIs can strongly benefit insurance companies, but unlocking value from microservices means knowing when to apply them and how to create them. It also means learning where microservices and APIs can be 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.
Source: Celent, "Honey I Shrunk the Services: Microservices and insurance," December 2015

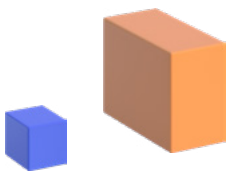
05

Legacy system: love it or leave it?

While insurance 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?" APIs and microservices help answer another big question: "How do we innovate faster, right now, today?"

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

APIs and microservices are a viable, reliable way to get business done to block InsurTechs and other competitors from going after your revenues and your customers.



[1] *Information Age, "Insurance is Falling Behind When it Comes to Technology Expectations," February 2017*

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

[3] *OpenLegacy, "Leading Insurance Firm Launches Innovative Agent Portal Connected to Core iSeries Application," accessed February 2018*

[4] *OpenLegacy, "Car Insurance Firm Reaches New Customers with New Online Quote Service," accessed February 2018*

[5] *OpenLegacy, "OpenLegacy Helps Improve Productivity Through an Integrated Applicated for Insurance Agents," accessed February 2018*

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

[7] *Techemergence, "How America's Top 4 Insurance Companies are Using Machine Learning," September 2017*

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

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

About OpenLegacy

OpenLegacy's Digital-Driven Integration enables organizations with legacy systems to release new digital services faster and more efficiently than ever before. It connects directly to even the most complex legacy systems, bypassing the need for extra layers of technology. It then automatically generates APIs in minutes, rapidly integrating those assets into exciting newinnovations. Finally, it deploys them as standard microservices or serverless functions, giving organizations speed and flexibility while drastically cutting costs and resources. With OpenLegacy, industry-leading companies release new apps, features, and updates in days instead of months, enabling them to truly become digital to the core.