OpenLegacy

WHITE PAPER

How to Leverage Legacy Systems at the Speed of DevOps

openlegacy.com

How to Leverage Legacy Systems at the Speed of DevOps

Introduction

DevOps is Spreading Rapidly, but Selectively

The promise of DevOps is real. Needed functionality is coming online faster, teams are working together better, and issues are being resolved closer to the moment they are detected. But only where DevOps has been applied systematically. New applications and interfaces that rely upon legacy applications can struggle as the velocity of delivery is slowed by systems that were not designed for, and do not easily support, agile and DevOps methodologies. Organizations in every vertical need a way to access legacy data and business logic without creating more code and slowing down the DevOps process.

47% lower cost over 5 years

Amount organizations would save running workloads on connected mainframes by moving to a more distributed environment

IDC, "The Business Value of the Connected Mainframe for Digital Transformation", 2018

Current Environment – Backend

Large monolithic legacy systems tend to have many interconnected pieces and to be 'fragile' in an overall sense, with one application impacting several other systems, and changes needing to be validated across all of these impacted systems.

While the stability of legacy systems and data is renowned, the hurdles to getting changes implemented on those systems is equally infamous. As custodians of key business data; legacy systems developers, admins, and DBAs know well that they are not responding at the rate the business would like. The problem is that attempts to deliver changes more quickly are thwarted by the very complexity that makes these systems valuable. The architectures of ten or twenty years ago are not suited to the type of rapid response expected today.

Current Environment – Agile

Agile development and DevOps methodology both aim to deliver software to the business at a faster pace, one in tune with today's constant changes in business environment. By making shorter goals, agile allows for developers to focus on one problem at a time, and to see test results quickly after they check in their code. By increasing cross-functional communication and automating everything from build through deployment, DevOps aims to accelerate the rate at which those changes are presented to users.

In modern application development, agile development teamed with DevOps streamlines the development and delivery of applications. The communi-

> While the stability of legacy systems and data is renowned, the hurdles to getting changes implemented on those systems is equally infamous.

cations changes new development methodologies encourage keep business owners aware of the current state and any outstanding issues.

This all works well, if the application is completely wrapped in agile and DevOps, with rapid iterations offering many incremental changes to the application over time. The mantra of Silicon Valley is definitely an adjunct to agile and DevOps – "Fail fast" means implement it, find the problems, and get to resolving them.

But that is the opposite of the view that has guarded legacy systems forever—controlled change coupled with thorough test environments have kept legacy systems stable and reliable.

OpenLegacy bridges the gap between DevOps and legacy development practices. By offering a method to generate Java based microservices with REST accessible APIs, the system exposes legacy data to DevOps teams. DevOps can continue to "fail fast" and "fail small" since OpenLegacy's microservices architecture limits changes required on legacy systems. In this environment, changes are made at the microservice, or by combining microservices, just like any other API can be enhanced or combined. No changes need to occur on the legacy backend.

The Problem

Why Backend Work is Generally Slower

A problem occurs when a given application has dependencies on older architectures that are designed and maintained with a more robust approach. The systems that hold all of your customer and billing data have been built and modified for years with an eye to stability. And systems that have been rock-solid for years should not suddenly start having performance and quality issues.

The complexity of systems that have grown over the course of a decade or two can be stunning. They were designed for one initial purpose and, as that purpose has changed and grown, complexity has also grown. While the data housed in these systems is critical, and the business logic they exercise over that data is unique to the application, this complexity can make changes take much longer than a fresh new agile project. Even simple changes to legacy systems can require extensive testing as the impact of those changes to other parts of the overall software architecture are evaluated.

Organizations want new web and/or mobile interfaces to legacy systems. Creating such an interface seems relatively simple on the surface, but the relationship of UI to backend systems is symbiotic. The creation of a UI will require new functionality or changes to the underlying data store. This is a natural process that occurs as an understanding of how users interact with the new UI becomes clear.

These scenarios end up with a rapidly iterating front end that can turn out changes in days or weeks backed by a system that is designed with the idea that changes will take months. When user feedback results in change requests for the backend, the entire agile/DevOps process goes on hold.

What is needed is a platform that grants access to the wealth of data stored in legacy systems without requiring a massive amount of new development on those systems—a framework to give DevOps the business logic that applications need. The framework must also be flexible enough to serve that information where ever DevOps teams need it. For example, OpenLegacy's approach builds APIs on top of microservices that are

composed of standard Java stacks. That means Java developers can get to the data they need without having to wait for the legacy team to develop more code.

Why Agile Work is Necessary

While the legacy systems and their development methodologies worked in an environment where the rate of change for all organizations was slower, and the demands of users were restricted to a small subset of the target market, the growth of the Internet and increasing competition make faster development of new systems and interfaces a business priority.

Agile and DevOps answer this need, allowing the business to respond to new competitive threats or to take advantage of a competitive opportunity in a timeframe suited to today's business environment. The catch is that the speed of delivery must be systemic. If a large portion of an organization's product portfolio is difficult to change and requires weeks or even months of testing after changes, the competitive business edge can be lost.

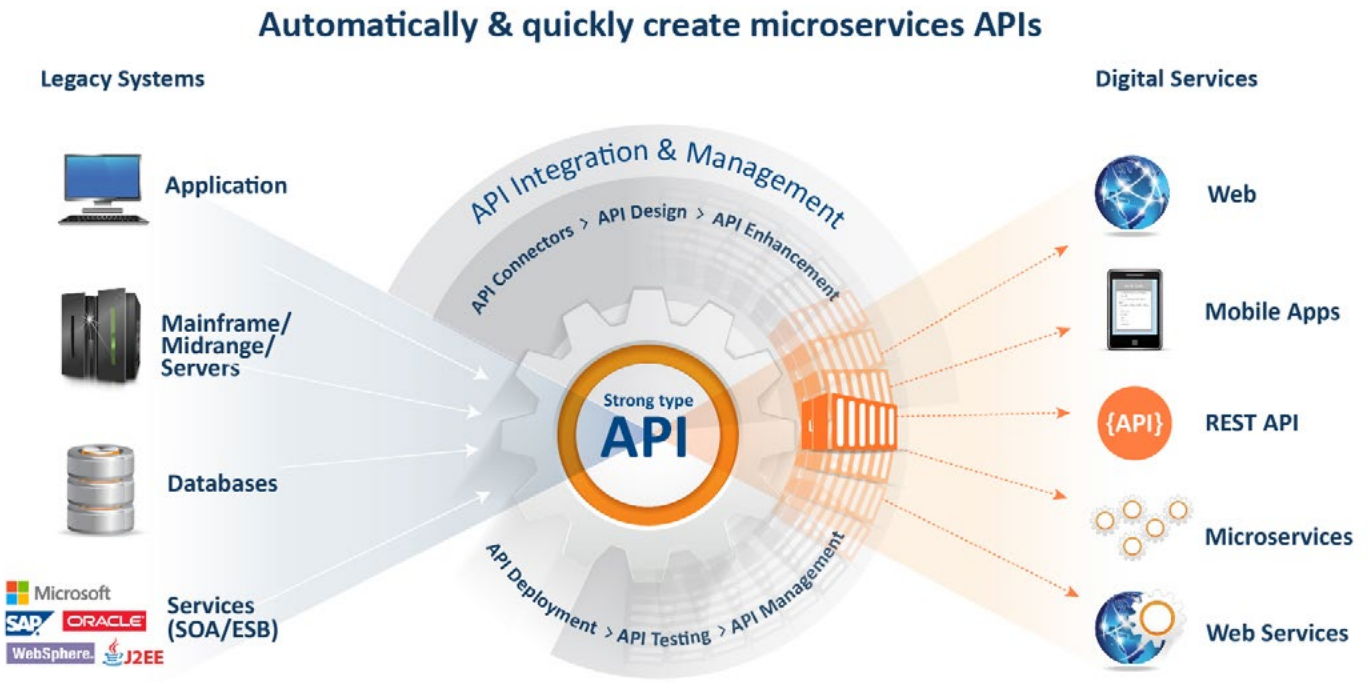
Utilizing agile development, DevOps methodologies, and OpenLegacy to bring legacy data into the agile and DevOps fold, an organization can start to deliver new functionality that utilizes legacy data and systems at the speed expected in a modern business environment.

Past Solutions and Their Legacy

In the not-too-distant past, Service Oriented Architectures (SOA) were used to address the slow rate of change in legacy systems. The underlying motivation for SOA was to provide access to data in legacy systems while speeding development by moving new development into the client/server tier. The problem is that once the data was manipulated in the client/server environment, that environment became part of the overall legacy infrastructure. Now some changes were being made in the legacy environment, while other changes were being made in the SOA environment, and those changes needed to be available at the same level of reliability.

The result was a complex world where not only was there a large super-integrated legacy application sitting on the backend, the middle tier was now a large super-integrated legacy application also. Both became slow to change, and changes often had to span both tiers, increasing the amount of work and the number of teams involved.

Utilizing microservices and single data access APIs, OpenLegacy sidesteps these issues by presenting a given application with the data it needs, when it is needed. Developers writing in the language of their choice can say "I need the customer number so I can



> Utilizing microservices and single data access APIs, OpenLegacy sidesteps these issues by presenting a given application with the data it needs, when it is needed.

find their last order", and call the customer API to grab data. There is no mainframe coding, no middle layer of business logic that must be maintained and inter-relationships to manage, just a call to get the needed data, and move along.

OpenLegacy enables this ability to get the data and move along with a system of connectors that allow legacy data to be pulled into microservices easily. While many backend connectors come preconfigured (For example, CICS or DB2), others can be custom designed if there is a need. This offers ease of use for the common cases, and task-specific implementation for custom cases.

No matter which type of connector is needed, in the end, agile users are presented with an API implemented in Java and residing on the edge of a microservice. At that point, the code and resulting API fits into the existing agile and DevOps architecture.

Making the Two Work Together

Backend Strengths

Legacy systems grew to be legacy because they get the job done. The workhorse of most modern enterprises, core systems hold the data that represents customer base, sales, stock levels, and more. These systems have reliably serviced the organization for a long time because they're good – or at least good enough – at what they do.

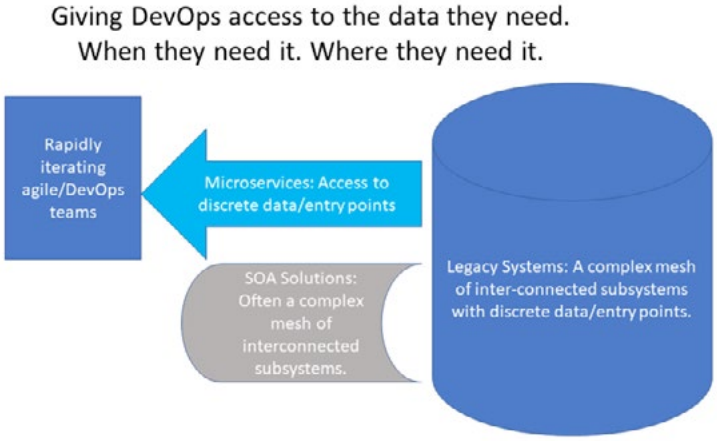
DevOps Strengths

Both the business and customers are demanding a faster rate of change though. The whole goal of digital transformation is to increase the rate of change to meet the demands of today's environment. The ability to rapidly improve the user experience is something that enterprises in nearly every vertical are striving for. Agile plus DevOps are the tools most organizations choose to approach that goal. By offering quick iterations with a highly automated build/test/deploy environment and repeatable/reusable processes and artifacts, DevOps increases the rate of change, and often increases stability by removing room for human error.

The Best of Both Worlds

Those monolithic legacy applications need to continue operating as they have in the past, but agile and DevOps teams need something that can move at the pace of modern business.

Since replacement of backend systems is not only expensive, but also prohibitively long, it makes the most sense to actually delivery on the promises SOA made. Enabling access to backend data, with small, manageable pieces that can be re-assembled as needed in a more agile environment is the key to improving productivity and delivery timelines without making things worse.



While the legacy systems continue to store and protect mission-critical data, OpenLegacy extends those core systems for access by agile teams. Using a microservices architecture with APIs that are both easily generated and customizable by Java developers, monolithic legacy applications can serve data where and when it is needed. OpenLegacy uses connectors to tie backend systems into the microservice, and API generation to generate the frontend API, relieving the need to develop access code in the most common access scenarios. Even in a highly flexible environment requiring support for platforms like cloud and mobile. OpenLegacy's integration with standard agile and DevOps tools such as Jenkins means that it is truly part of the DevOps solution architecture.

OpenLegacy helps teams adapt backend access with microservices and DevOps outside the monolith. Part of this solution is integration with modern development and DevOps tools. Based in Java, with Eclipse support, Jenkins integration, and targeting for Docker, developers will find existing skill-sets extend to legacy applications and data. This integration helps DevOps teams gain the benefits of stable, secure datasets and procedures while making application delivery faster.

Make Use of the Most Prevalent Skills

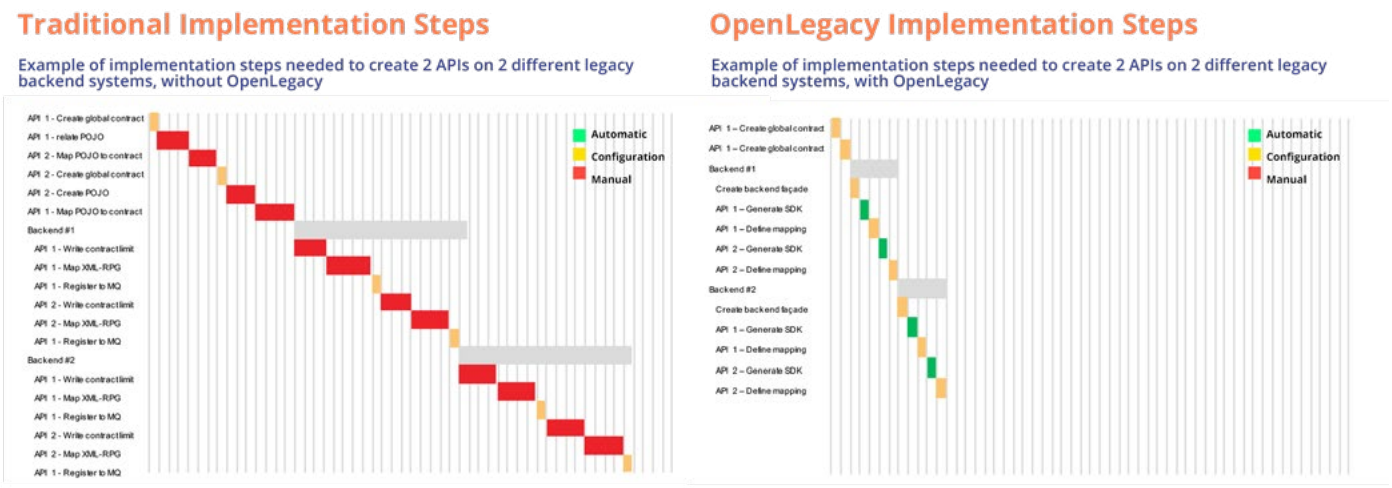
Most new developers entering the workforce are trained in modern languages like Java, Node.js and Python. They are not equipped to handle the environments in which legacy applications were developed and are maintained. OpenLegacy generates APIs that can be called from any language that supports SOAP or REST API calls.

That means developers and teams can use the toolset that best fits the goals of agile and DevOps, taking advantage of the languages they are already familiar with, but gaining access to legacy systems and data that power the enterprise.

Importantly, development on the legacy systems is not required. Since the pool of available developers for many legacy platforms – most notably the mainframe – is on a downward trend, not creating new source to be maintained for integration is important, as available legacy system staff will have enough to do.

With OpenLegacy, agile developers can generate the APIs, make changes required, and keep their project moving forward. With the development shifted to agile teams and platforms integrated into the DevOps lifecycle, the need for legacy changes is reduced or eliminated. This allows projects to use the data and subsystems that have worked for years, while getting the benefits of agile and DevOps development speeds.

OpenLegacy Reduces Implementation Steps to Accelerate Your Time-to-Market



OpenLegacy accelerates development utilizing legacy systems by automating otherwise manual steps, and reducing the number of steps.

Get More Benefits Than Time To Market

Security with DevOps

One of the issues with exposing the mainframe more directly to corporate systems is implementing sufficient data security. Most companies would be hesitant to provide an API that can modify customer data. A strong security model is needed that will allow access while restricting it to those users and apps that should be touching the data.

OpenLegacy meets these needs with OAuth2 support and data masking to offer only data an application needs, and not the entire table/file structure to be picked through.

Since OpenLegacy is integrated with standard DevOps toolsets, existing security infrastructure applies to generated microservices. Policies and standards that are enforced through the test environment can be applied directly to generated source, resolving the need for another set of security tools to be deployed. What works for the legacy system will still be there, and what works for agile teams will work for the microservices.

Repeatability/Portability for DevOps

DevOps requires the ability to repeat processes easily with predictable and consistent results. This is because automation depends upon the ability to know the final state, and the speed of iteration does not leave room for massive customization for each individual release. Increasingly, portability of systems is an imperative also. Choosing where to deploy an application based upon today's needs, with the flexibility to move the application at a later date, is becoming the standard.

However, this is not how IT has traditionally operated; taking the same steps is not guaranteed to get the same results. And moving an application has been a choice of last resort. Traditional application development targets an architecture, and releases become customized to take into account an array of variables (like installed packages) present on the target platform.

With OpenLegacy's approach, microservices address these needs. The ability to create a brand new copy of the microservice to replace the one being upgraded makes deployments predictable and results readily testable. The resulting microservices can be targeted where they are needed. Today's environment might have them deployed to an internal server, while tomorrow's might need the microservices in a cloud-hosted container. The ability to place microservices where they are needed offers both deployment flexibility and the largest pool of re-use options.

Performance

Traditional SOA based approaches suffered from performance lag due to the layers of processing occurring at different steps in a request. The legacy system would perform all of its business processing, return results, and the SOA would perform all of its processing. Solutions that used queues such as MQ would end up with additional lag as several legacy systems were invoked and processed then more processing was done on the "client" side (actually the server in the client-server world).

> By offering a method to generate Java based microservices with REST accessible APIs, the system exposes legacy data to DevOps teams.

By making use of microservices and allowing fine-grained access, OpenLegacy greatly reduces this performance hit. Data and smaller subsystems on the legacy system can be accessed directly, leaving the bulk of the processing to the application being developed. Think in the case of a database – ACID processes would occur, but the business validations required in most mainframe applications would only be necessary on an Insert or Update statement. For inserts and updates, validation is required that might need those extensive backend subsystems. For access of data, simple retrieval is normally sufficient.

These smaller, easier to utilize pieces of data are similar to other data used in the client application, making processing uniform and fast. Uniform processing is useful in agile environments, and fast application processing is desired in every environment.

Conclusion

Take Your DevOps to the Next Level

When implementing DevOps, whether across an application portfolio or across the entire organization, it's not an option to leave legacy systems and data behind. Likewise, putting legacy systems into the process flow without improving traditional development/deployment methods will slow agile and DevOps. Since speed and adaptability are the primary benefits of DevOps and agile, this slow process reduces their effectiveness.

When implementing DevOps, whether across an application portfolio or across the entire organization, it's not an option to leave legacy systems and data behind.

Different methods have been used in the past to make legacy systems more accessible to new applications, but all have suffered from weaknesses, some even fit the age-old phrase "The cure is worse than the disease".

Using microservices to access legacy data from modern development environments is an important part of modernizing an application infrastructure. Microservices make the applications more portable, while modern environments bring applications into the agile and DevOps worlds. Incorporating modern security mechanisms into the new environment extends legacy data protection to this new platform. The net result is new applications coming online that do not require legacy development efforts. Legacy systems become "just another data source", just like any other the agile system accesses.

The ability to generate APIs that reside in microservices and serve up data from legacy systems offers the best of both worlds. Agile development with modern DevOps tools, and access to core corporate data from legacy systems will bring legacy systems into the rapid development fold. Organizations like OpenLegacy have developed a software solution to help you achieve agility in the last bastion of long release cycles.

About OpenLegacy

OpenLegacy accelerates delivery of innovative digital services from legacy systems in days or weeks versus months. Our microservices-based API integration and management software reduces manual effort by automating API creation, simplifies the process by avoiding layers of complexity, and improves staff efficiency and API performance. Our software directly accesses and extends business logic to web, mobile our cloud innovations in the form of Java objects, REST APIs or SOAP. Most importantly, this process is not only fast, easy and secure, but also does not require special staff skills or changes to existing systems or architecture. Together, business and IT teams can quickly, easily and securely meet consumer, partner or employee demands for digital services without modernizing or replacing core systems. Learn why leading companies choose OpenLegacy at www.openlegacy.com.

OpenLegacy
OPEN INNOVATION

www.openlegacy.com
sales@openlegacy.com

Headquarters
11921 Freedom Drive,
Suite 550
Reston, Virginia, 20190

Dallas
Lewisville Vista Point North
405 State Hwy 121, Suite A250
Lewisville, TX 75067

Israel
3 Mota Gur,
Olympia Park,
Petah Tikva, Israel

Chicago
541 N. Fairbanks Ct.
Suite 2200
Chicago, IL 60611

Mexico
Av. Insurgentes Sur #730,
Col. Del Valle,
Delegación Benito Juárez,
Piso 2 México, DF. CP 03104

Switzerland
Rue Etienne
Dumont 1
Geneva, 1204
Switzerland