Raiders of the Lost Apps: An Adventurer's Guidebook for Uncovering Technical Debt through Modernization

An Intellyx Analyst Guide, for vFunction

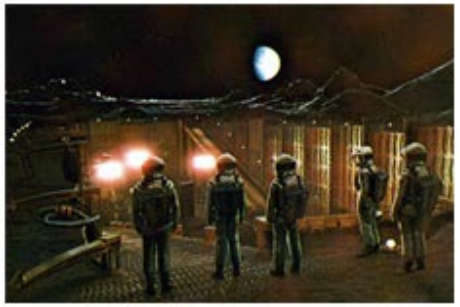
By Jason Bloomberg and Jason English

©2023 Intellyx LLC

Table of Contents

©2023 Intellyx LLC



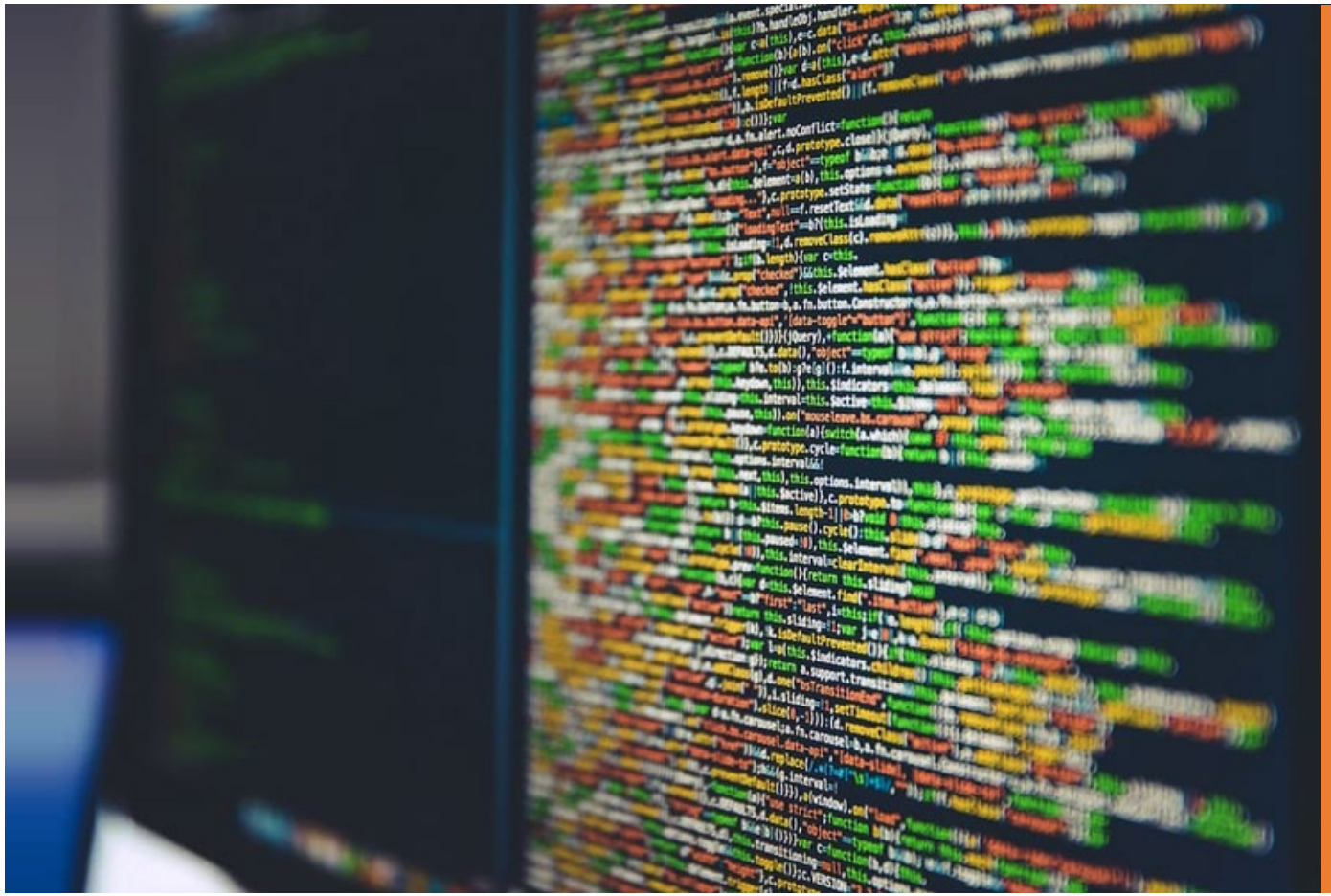
Introduction

Consider this eBook as an architectural interpretation guide for the adventurous modernization explorer.

We almost called it “Raiders of the Lost Apps” because of the many parallels between modernizing applications and archaeology.

To get to the bottom of the biggest obstacles of modernizing our digital estates, we must dig to find the technical debt artifacts and digital foundations of our organizations to transform our applications, reset our culture, and adapt our architecture to meet the challenges of a global, distributed, hybrid IT future.

©2023 Intellyx LLC



By Jason Bloomberg

President &
Principal Analyst,
Intellyx

Eliminating Technical debt: Where to Start?

This article is the first in a four-part series. In part two, we'll explore the challenges in navigating the cultural change of modernization. Part three will delve into sustaining the resulting transformation. We'll wrap up in part four, as we discuss evolving toward modern-day technology goals.

©2023 Intellyx LLC

At its most basic, technical debt represents some kind of technology mess that someone has to clean up. In many cases, technical debt results from poorly written code, but more often than not, is more a result of evolving requirements that existing technology simply cannot keep up with.

Technical debt can accrue across the technology landscape – and we could even argue, extends beyond technology altogether in other areas of the business, for example, process debt.

Within the technology arena, technical debt breaks down into two basic categories: infrastructure and applications.

Compared to application technical debt, reducing infrastructure technical debt is the more straightforward. Application technical debt, in contrast, is a knottier problem, because there are so many places that technical debt can hide in existing applications.

Simply identifying this debt is challenge enough. Prioritizing its reduction is also difficult. Eliminating the debt once and for all can be a task of Herculean proportions.

Here's how to start on this journey.

©2023 Intellyx LLC

Assessing Your Current State

The first step in an effective assessment of technical debt is to ensure you start with the big picture. Application technical debt may be where the most urgent problems like, but even so, it's important to place these issues into the proper context.

The first consideration here is the business and its requirements. Where is existing technical debt causing the business or your customers the most pain?

> It's also important to include operational and cost factors in any assessment. In many cases, for example, older applications require older hardware – and thus the plan for infrastructure technical debt and the corresponding application plan are interdependent.

A recent survey by Wakefield Research of enterprise software architects and developers indicated that the most difficult step in application modernization was securing the budget and resources for the project followed by "knowing what to modernize" and "building a business case." The only way to address these challenges is to accurately calculate the current technical debt in those applications up front with a data-driven plan.

Cost considerations are also important to any consideration of technical debt reduction planning. Some debt reduction projects will inevitably be more expensive than others – but won't necessarily deliver more value. You're looking for projects with the most 'bang for the buck.'

©2023 Intellyx LLC

Rationalize Your Applications

Refactoring: Applications in this category are more or less meeting their requirements, but some internal issue with the code is bogging them down. In this situation, refactoring is the best approach – reworking the code in place without necessarily changing the overall functionality of the application.

Deprecating: Sometimes it's simply not worth the cost and trouble of dealing with particular instances of technical debt. While it may provide value to clean up these situations, your assessment has determined that your time and money are best spent elsewhere.

It's important, however, to flag such code as something you've intentionally decided to leave alone (at least for the time being). Hence the notion of deprecation – an application you can still use, with the understanding that if you have a choice, use an alternative instead.

Replatforming: In some situations, the technical debt has more to do with the platform than the application itself. For example, an application might be running on an older version of .NET or Java EE.

The main goal of replatforming is typically to move an application to the cloud – without changing anything about the application that isn't necessary to achieve this goal.

While replatforming is occasionally the best approach to dealing with technical debt, many organizations overuse it – putting debt-laden applications in the cloud without resolving that debt.

©2023 Intellyx LLC

Rearchitecting: Whenever replatforming falls short (as it often does), then rearchitecting is typically the approach that best resolves application technical debt issues – but also tends to be the most expensive option.

While it's possible to replatform an application in the cloud without rearchitecting it, such rearchitecture is necessary in order to take advantage of many of the core benefits of the cloud, including scalability, elasticity, and automated provisioning.

When moving to a cloud native platform (typically Kubernetes), rearchitecture is an absolute must.

Rearchitecting to Reduce Technical Risk

While cloud native computing is all the rage today, it's important to note that taking a cloud native approach may include a variety of architecture and technology options depending upon the business need.

Rearchitecting, therefore, requires a careful consideration of such needs, as well as available resources (human as well as technology), realistic timeframes for modernization, and the overall cost of the initiative.

Rearchitecture also never works within a vacuum. Any rearchitecture project must take into account the existing application and management landscape in order to address issues of security, governance, and whatever integration is necessary to connect the rearchitected applications to other applications and services.

Because rearchitecture initiatives also include replatforming, it's also essential to plan ahead for any cloud migration requirements as part of the overall effort.

©2023 Intellyx LLC

At some point in every rearchitecture effort, of course, it will be necessary to modernize the code as well as the architecture. Fortunately, there are approaches to modernizing code without the onerous responsibility of rewriting it line by line – even in situations where the architecture of the software is changing.

We'll be covering application modernization in more depth in the rest of this four-part article series. Stay tuned!

The Intellyx Take

In many cases, organizations tackle modernization projects without thinking specifically about resolving technical debt. However, framing such initiatives in terms of such debt is a good way to improve their chances of success.

Today, it's possible to measure technical debt with a reasonable amount of accuracy using solutions like vFunction Assessment Hub. Such measurements give you a heat map across your application landscape, pointing out those areas that have particularly knotty messes to clean up and can even pinpoint where to start refactoring or rearchitecting in those apps including the top app component contributors to technical debt and the resulting ROI.

Without such a heat map, modernization efforts tend to go off the rails. Therefore, while technical debt is generally a bad thing, it does serve certain purposes – including directing the modernization team to the highest priority projects.

©2023 Intellyx LLC

By Jason English

Principal Analyst,
Intellyx

Navigating the Cultural Change of Modernization

Part 2 in the Uncovering Technical Debt series

©2023 Intellyx LLC

Decompose, analyze, break into services, refactor. Whatever methods and tools you use to bring legacy code and architecture up to modern standards, there's a science to enterprise application modernization.

But what if that science is archaeology?

If so, we want to spend less time manually digging and dusting and more time studying the artifacts and culture. We'd much rather be *Indiana Jones*, on an exciting journey to find hidden gems, than an unpaid grad student piecing together shards of pottery.

One thing is for certain. Many have tried modernization, and many have failed. According to a 2022 study by Wakefield Research, 92% of surveyed professionals responsible for a core system surveyed have started, or are planning to start, an app modernization project – but 4 out of 5 reported already failing on one or more such projects.

©2023 Intellyx LLC

Monoliths were built for a reason

To some, a monolith would be any system that was already in place before they were an employee of the organization.

That's a gross generalization, but it's not that far off the mark. Just about any company that has grown over a period of a few years is maintaining at least one monolith, if not more. Add in an acquired company, and add in another business monolith.

The people who originally architected these monoliths have likely moved on or retired. The legacy code inside them was written well before there was a concept of cloud, or microservices, or auto-scaling clusters to meet demand.

According to the survey, 99% of the professionals reported serious challenges in maintaining monolithic apps. Executives lamented keeping up with business requirements and growing technical debt as leading challenges, while IT architects cited a shortage of skilled maintainers, followed by technical debt.

Even as it accrues technical debt, a monolith handles an extremely critical operation for the business. Much like a sacred artifact, removing the monolith from its perch would bring about certain doom. That's why whole departments have sometimes been appointed as 'temple guardians' – installing security perimeters, failsafes, and draconian change control request boards around the monolith to ensure its operations are never interrupted.

Most companies put off modernizing monoliths longer than they should due to fear and uncertainty – but there is always a moment when the business can't afford to not change. Time for our intrepid explorers to swap out the production system with an upgraded one, hopefully without setting off any traps.

©2023 Intellyx LLC

If you can move the org, you can move the monolith

Given enough time and dynamite, people can excavate mountains. Monoliths are not immovable objects by comparison. They are man-made structures in our business topology, built by monolithic teams, which can also disperse like sand in the wind.

Recent events such as the global pandemic have accelerated the arrival of a more distributed organizational structure and remote work culture. Even in companies that retain centralized IT functions, the ubiquitous use of cloud computing means systems and workloads could be running anywhere, and new application topologies are making way for new team topologies to match.

So, why is the organization still holding us back? Fifty percent of both executives and architects agreed that securing adequate budget and resources is the most difficult step in modernization, compared to several other limiting factors including planning, staffing and training concerns.

> Executives are becoming comfortable with a more distributed workforce to manage modern distributed applications. Now they need to step up to the plate to push through resource and staffing constraints.

©2023 Intellyx LLC

One successful approach that has worked for other company-wide initiatives such as cybersecurity and compliance is to implement a shared responsibility model. Just like a security help desk making everyone understand how to protect the company and report phishing attacks and social engineering attempts, application modernization should be everyone's job to some extent.

Not everyone in the organization is a digital native, so companies should prioritize upskilling the team by training them in the complexities of refactoring applications for a distributed, cloudy future, as well as equipping them with intelligent modernization platforms like vFunction.

From there, leadership can look to practices proven at the best-performing companies, such as establishing a Center of Excellence (or CoE) around modernization. This CoE would consist of a matrixed group of execs and IT leaders from across the organization who can effectively take in organizational and customer feedback, then identify and evaluate modernization projects, in order to procure necessary funding and resource allocation targets for success.

Changing team topologies can be profoundly powerful – in the survey, 62% of respondents cited obtaining proper budget for resources and intelligent tools as the top reason for their successful modernization projects, followed by having the right organizational structure in place.

©2023 Intellyx LLC

The Intellyx Take

The counterintuitive secret of application modernization success?

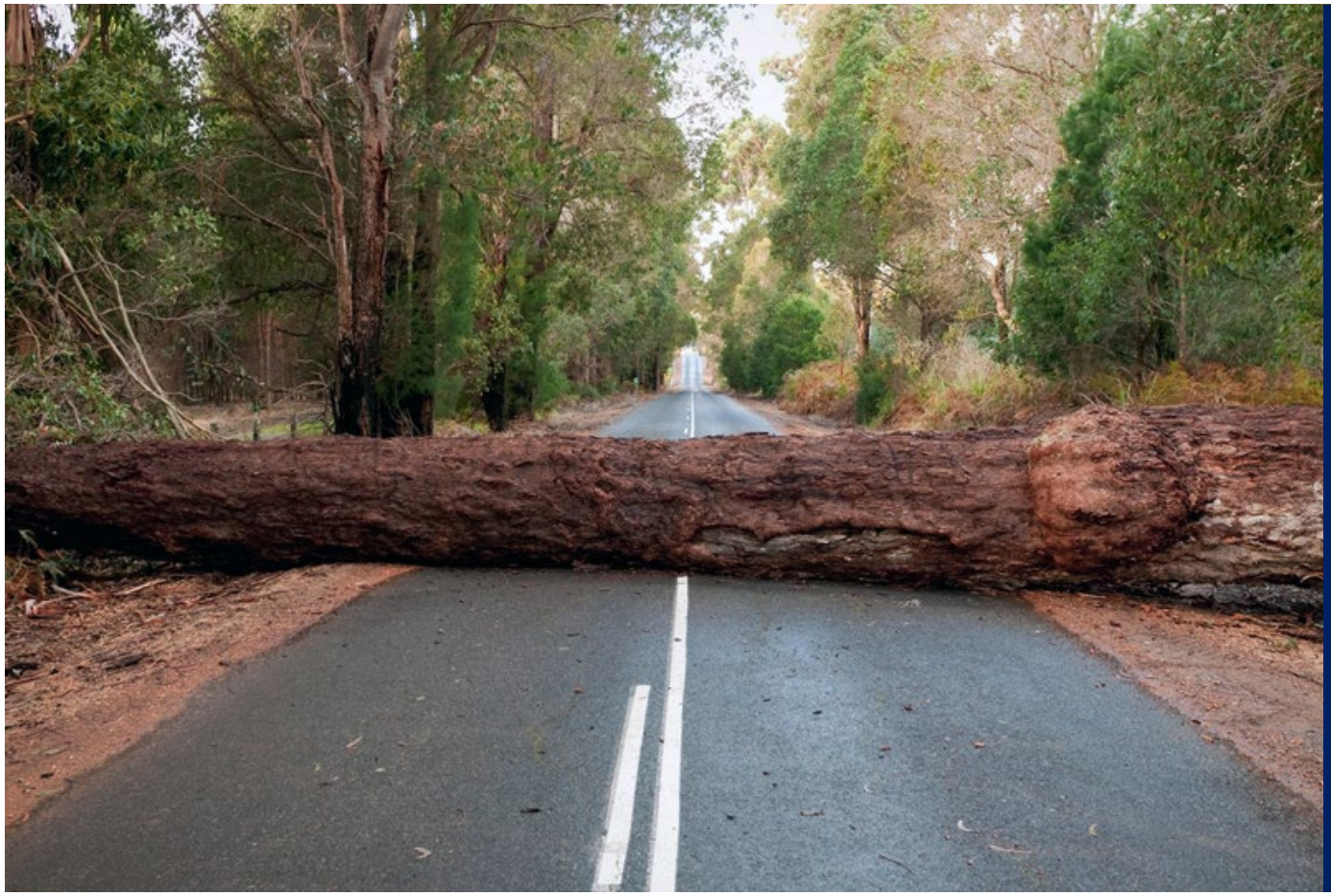
In archeology, scholarly explorers seek to uncover artifacts that tell us about the cultures of the past, so as to better understand how we arrived at our present-day culture.

In application modernization, monolithic artifacts and modern architectures are a reflection of the organization's culture as it changes over time.

In that context, modernization is a continuous journey of uncovering and excavating technical debt, in order to explore new opportunities for higher business agility and application performance.

Image source: Movie still, Alamy licensed image.

©2023 Intellyx LLC



By Jason Bloomberg

President &
Principal Analyst,
Intellyx

Don’t Let Technical Debt Stymie Your Java EE Modernization

Part 3 in the Uncovering Technical Debt series

©2023 Intellyx LLC

When Java Enterprise Edition (Java EE) hit the scene in the late 1990s, it was a welcome enterprise-class extension of the explosively popular Java language. J2EE (Java 2 Enterprise Edition, as it was called then) extended Java's 'write once, run anywhere' promise to n-tier architectures, offering Session and Enterprise JavaBeans (EJBs) on the back end, Servlets on the web server, and Java Server Pages (JSPs) for dynamically building HTML-based web pages.

Today more than two decades later, massive quantities of Java EE code remain in production – only now it is all legacy, burdened with technical debt as technologies and best practices advance over time.

The encapsulated, modular object orientation of Java broke up the monolithic procedural code of languages that preceded it. Today, it's the Java EE applications themselves that we consider monolithic, fraught with internal dependencies and complex inheritance hierarchies that add to their technical debt.

Modernizing these legacy Java EE monoliths, however, is a greater challenge than people expected. Simply getting their heads around the internal complexity of such applications is a Herculean task, let alone refactoring them.

For many organizations, throwing time, human effort, and money at the problem shows little to no progress, as they reach a point where some aspect of the modernization project stymies them, and progress grinds to a halt.

Don't let technical debt stymie your Java EE modernization initiative. Here's how to overcome the roadblocks.

©2023 Intellyx LLC

Two Examples of Java EE Technical Debt Roadblocks

A Fortune 100 government-sponsored bank struggled with several legacy Java EE applications, the largest of which was a 20-year-old monolith that contained over 10,000 classes representing 8 million lines of code.

Replacing – or even temporarily turning off – this mission-critical app was impossible. Furthermore, years of effort on analysis in attempts to untangle the complex internal interdependencies went basically nowhere.

The second example, a Fortune 500 financial information and ratings firm, faced the modernization of many legacy Java EE applications. The company made progress with their modernization initiative, shifting from Oracle WebLogic to Tomcat, eliminating EJBs, and upgrading to Java 8.

What stymied this company, however, was its dependence on Apache Struts 1, an open-source web application framework that reached end-of-life in 2013.

This aging framework supported most of their Java EE applications, despite introducing potential compatibility, security, and maintenance issues for the company's legacy applications.

Boiling Down the Problem

In both situations, the core roadblock to progress with these respective modernization initiatives was complexity – either the complexity inherent in a massive monolithic application or in the complex interdependencies among numerous applications that depended on an obsolete framework.

©2023 Intellyx LLC

Obscurity, however, wasn't the problem: both organizations had all the data they required about the inner workings of their Java EE applications. Both companies had their respective source code, and Java's built-in introspection capabilities gave them all the data they required about how the applications would run in production.

In both cases, there was simply too much information for people to understand how best to modernize their respective applications. They needed a better approach to making decisions based upon large quantities of data.

The answer: artificial intelligence (AI).

Breaking Down the Roadblocks

When such data sets are available, AI is able to discern patterns where humans get lost in the noise. By leveraging AI-based analysis tooling from vFunction, both organizations got a handle on their respective complexity, giving them a clear roadmap for resolving interdependencies and refactoring legacy Java EE code.

The Fortune 100 bank's multi-phase approach to Java EE modernization included automated complexity analysis, AI-driven static and dynamic analysis of running code, and refactoring recommendations that included the automated extraction of services into human-readable JSON-formatted specification files.

The Fortune 500 financial information firm leveraged vFunction to define new service boundaries and a common shared library. It then merged and consolidated several services, removing the legacy Struts 1 dependency in favor of a modern Spring REST controller. It also converted numerous Java EE dependencies to Spring Boot, a modern, cloud native Java framework.

©2023 Intellyx LLC

The Business Challenges of Technical Debt

Both organizations were in a 'for want of a nail, the kingdom is lost' situation – what seems like a relatively straightforward issue stymied their respective strategic modernization efforts.

When such a roadblock presents itself, then all estimates about how long the modernization will take and how much it will cost go out the window. Progress may stop, but the modernization meter keeps running, as the initiative shows less and less value to the organization as the team continues to beat their head against a wall.

Not only does morale suffer under such circumstances, but the technical debt continues to accrue as well. In both situations, the legacy apps were mission-critical, and thus had to keep working. Even though the modernization efforts had stalled, the respective teams were still responsible for maintaining the legacy apps – thus making the problem worse over time.

©2023 Intellyx LLC

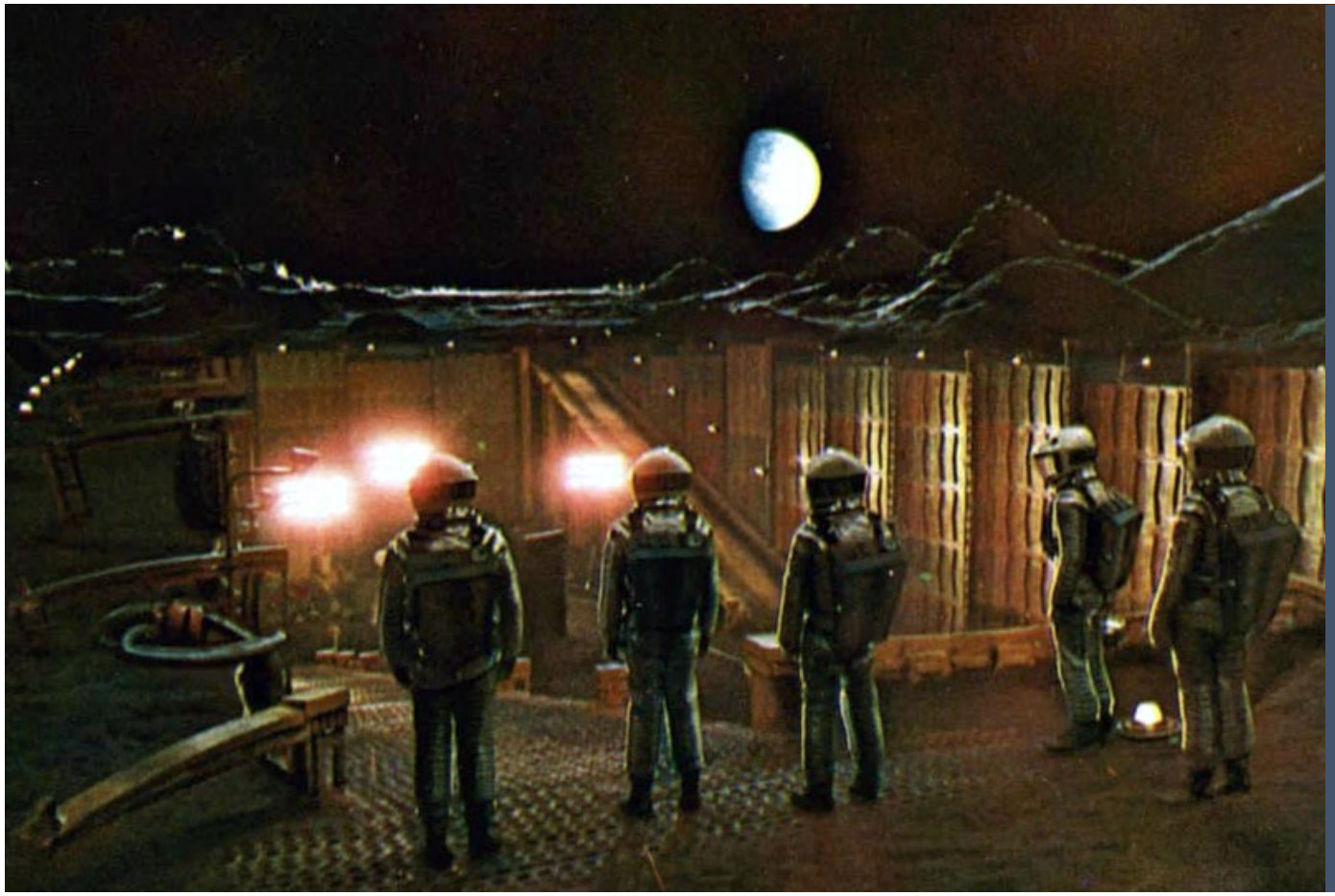
The Intellyx Take

During the planning stages of any modernization initiative, the teams had hammered out reasonable estimates for cost, time, and resource requirements. Such estimates were invariably on the low side – and when a roadblock stops progress, the management team must discard such estimates entirely.

Setting the appropriate expectations with stakeholders, therefore, is fraught with challenges, especially when those stakeholders are skeptical to begin with. Unless the modernization team takes an entirely different approach – say, leveraging AI-based analysis to unravel previously intractable complexity – stakeholders are unlikely to support further modernization efforts.

It's important to note the role that vFunction played. It didn't wave a magic wand, converting legacy code to modern microservices. Rather, it processed and interpreted the data each organization had available (both static and dynamic), leveraging AI to discern the important patterns in those data necessary to make the appropriate decisions that resulted in timely modernization results. Considering the deep challenges these customers faced, such results felt like magic in the end.

©2023 Intellyx LLC



By Jason English

Principal Analyst,
Intellyx

Evolving Toward Modern-Day Goals with Continuous Modernization

Part 4 in the Uncovering Technical Debt Series

©2023 Intellyx LLC

We've dug deep into our technology stacks, uncovering all of the legacy artifacts and monoliths that we could find from past incarnations of our organization.

We've cataloged them, rebuilt them to modern coding standards, and decoupled their functionality into object-oriented, service-enabled, API-addressable microservices.

Now what? Are we modernized yet?

Well, mostly. There are always some systems that just aren't worth the time and attention to replace right now, even with intelligent automation and refactoring solutions.

Plus, we acquired one of our partner companies last year, and we haven't had a chance to merge their catalog with our ordering system yet, so they are still sending us EDI dumps and faxes for urgent customer requests…

We're never really done with continuous modernization

We've compared legacy modernization to the discipline of archaeology. But what happens once archaeologists finish their excavation and classification expeditions? Anthropologists can take over the work from here, interpreting societal trends and impacts even as the current culture continues to evolve and generate new artifacts.

Similarly, discovering and eliminating uncovered technical debt isn't a one-time modernization project, it's a continuous expedition of reevaluation. Once an application is refactored, replatformed or rearchitected, it creates concentric ripples, exposing more dependencies and instances of technical debt across the extended ecosystem, including adjacent applications within the organization, third-party services and partner systems.

©2023 Intellyx LLC

Mapping the as-is and to-be state of the codebase with discovery and assessment tools is useful for prioritizing the development teams' targets for each project phase around business value, but business priorities will change along with the application suite.

Development teams also get great utility from conducting modernization projects with the help of AI-driven code assessment and refactoring solutions like vFunction Modernization Hub, but they can realize even greater benefits by retaining the history of what worked (and didn't work) to inform future transformations.

Not every modernization project works out equally well, but when the hard lessons of modernization feed back into the next assessment phase, this virtuous cycle can become part of the muscle memory of the organization, allowing mental energy to be spent on the most important choices that affect the long-term goals of the business.

Putting technical debt to rest: what to expect

No computer science college student or self-taught coder sets out to spend a career finding and fixing bugs in their code, much less someone else's – but on average, developers spend at least 30 to 50 percent of their time on rework, rather than innovation and enablement of new features that are perceived to add business value.

Besides the perceived thanklessness of the effort, developers encounter morale-destroying toil when sifting through legacy code, which usually contains lots of class redundancies, recursive methods, poor documentation and a general lack of traceability, resulting in slow progress.

Continuous modernization offers a way out of this thankless job, by preventing technical debt from collecting during each assessment and refactoring phase.

©2023 Intellyx LLC

Here's some levers teams are pulling for successful long-term improvements:

Continuous assessment. The best performing initiatives are not just conducting initial assessments, they are continuously mapping, measuring and observing modernization efforts before, during, and after each refactoring run.

FinOps practices basically bring financial concerns and tradeoffs to each modernization selection process or option. IT buying executives have been doing ROI analyses for vendor selection and capex computing investments for years. Now, savvy buyers are getting better cost justification for money spent on modernization, with real financial metrics for resources, employee and customer retention, and delivered customer value.

SLO objectives offer positive motivation for time-and-labor savings and incremental delivery of new services, in comparison to the negative contractual penalties enforced through SLA failures. Developers are incentivized to meet goals such as faster refactoring projects, faster automated deployments, and higher value updates – with fewer hitches and developer rework required.

Qualitative business goals are equally as important to success. Better team morale improves productivity and employee retention rates, versus trying to replace high-quality people with new ones that could take months to get up to speed. Developers love working for agile enterprises, where they can test theories and ultimately help the application suite evolve faster in the future to meet changing customer needs.

©2023 Intellyx LLC

Trending toward velocity and morale at Trend Micro

Trend Micro is considered a global leader in cloud workload security, with several successful products underneath the banner of its platform – but that didn't mean their modernization journey started without major headaches.

Much of their existing product suite, with more than 2 million lines of code and 10,000 independent Java classes, was built before secure API connections between cloud infrastructure and microservices were fully sussed out by the development market. Therefore, earlier customers were more inclined to trust on-premises installations and updates of vital virus, spam and spyware prevention software.

As the modern trends of SaaS-based vendors and cloud-based enterprise applications really hit stride over the last decade, Trend Micro started offering a re-hosted version of its suite under their CloudOne™ Platform banner.

Their initial lift-and-shift of one module's code and data store to AWS offered some scalability and cost benefits due to elastic compute resources, but as the user base grew, it was becoming harder and harder for product dev teams to get a handle on inter-product dependencies that hindered future releases and updates to meet customer needs. Morale suffered as the replatforming took about a year.

Trend Micro turned to **vFunction Assessment Hub** to identify and prioritize modernization of their most critical "Heartbeat" integration service – with more than 4000 Java classes that take in data from sensors, event feeds and data services across the product suite.

> Using vFunction for modernization, the team was able to visually understand code complexity, with an applied AI for identifying essential, interconnected and circular dependencies, deprecating dead code that would no longer add any actual value for customers going forward.

©2023 Intellyx LLC

Through refactoring, they were able to decide which classes should be included as part of the new Heartbeat service, and which should be kept in a common library for shared use across other product modules in the future.

This modernization project took less than 3 months – a 4X speed improvement over the previous project, with successive update deployment times decreased by 90%. Best of all, morale on the team has improved by leaps and bounds.

©2023 Intellyx LLC

The Intellyx Take

Continuous modernization offers enterprises a lasting bridge from the monolithic past to a microservices future, but with constant change at enterprise scale, the journeys across this bridge will never really end.

To get to the bottom of the biggest obstacles of modernizing our digital estates, we must first assess and prioritize code refactoring and application architecture efforts around resolving technical debt.

Then, our intrepid teams can venture forth, digging to unearth the artifacts and digital foundations of our organizations, transforming our applications into modular cloud native services, resetting the values of our shared culture, and adapting our architectures to meet the challenges of a global, distributed, hybrid IT future.

Can you dig it?

©2022 Intellyx LLC. Intellyx retains editorial control of this document. At the time of writing, vFunction is an Intellyx customer.
Image credit: Licensed from Alamy "2001: A Space Odyssey" movie still.

About Jason Bloomberg

Jason Bloomberg is a leading IT industry analyst, author, keynote speaker, and globally recognized expert on multiple disruptive trends in enterprise technology and digital transformation.

He is founder and president of Digital Transformation analyst firm Intellyx. He is ranked among the top nine low-code analysts on the Influencer50 Low-Code50 Study for 2019, #5 on Onalytica's list of top Digital Transformation influencers for 2018, and #15 on Jax's list of top DevOps influencers for 2017.

Mr. Bloomberg is the author or coauthor of five books, including *Low-Code for Dummies*, published in October 2019.

About Jason 'JE' English

Jason "JE" English is Principal Analyst and CMO at Intellyx. Drawing on expertise in designing, marketing and selling enterprise software and services, he is focused on covering how agile collaboration between customers, partners and employees accelerates innovation.

A writer and community builder with more than 25 years of experience in software dev/test, cloud computing, security, blockchain and supply chain companies, JE led marketing efforts for the development, testing and virtualization software company ITKO from its bootstrap startup days, through a successful acquisition by CA in 2011. He co-authored the book *Service Virtualization: Reality is Overrated* to capture the then-novel practice of test environment simulation for Agile development. Follow him on Twitter at @bluefug.

©2023 Intellyx LLC

About vFunction

vFunction is the first and only platform for developers and architects that intelligently and automatically transforms complex monolithic Java applications into microservices, restoring engineering velocity and optimizing the benefits of the cloud. Designed to eliminate the time, risk and cost constraints of manually modernizing business applications, vFunction delivers a scalable, repeatable factory model purpose-built for cloud native modernization. With vFunction, leading companies around the world are accelerating the journey to cloud-native architecture and gaining a competitive edge. vFunction is headquartered in Palo Alto, CA, with offices in Israel.

Learn more at **https://vfunction.com**.

©2023 Intellyx LLC. Intellyx retains editorial control over the content of this document. At the time of writing, vFunction is an Intellyx customer. Image sources licensed/acquired by client.