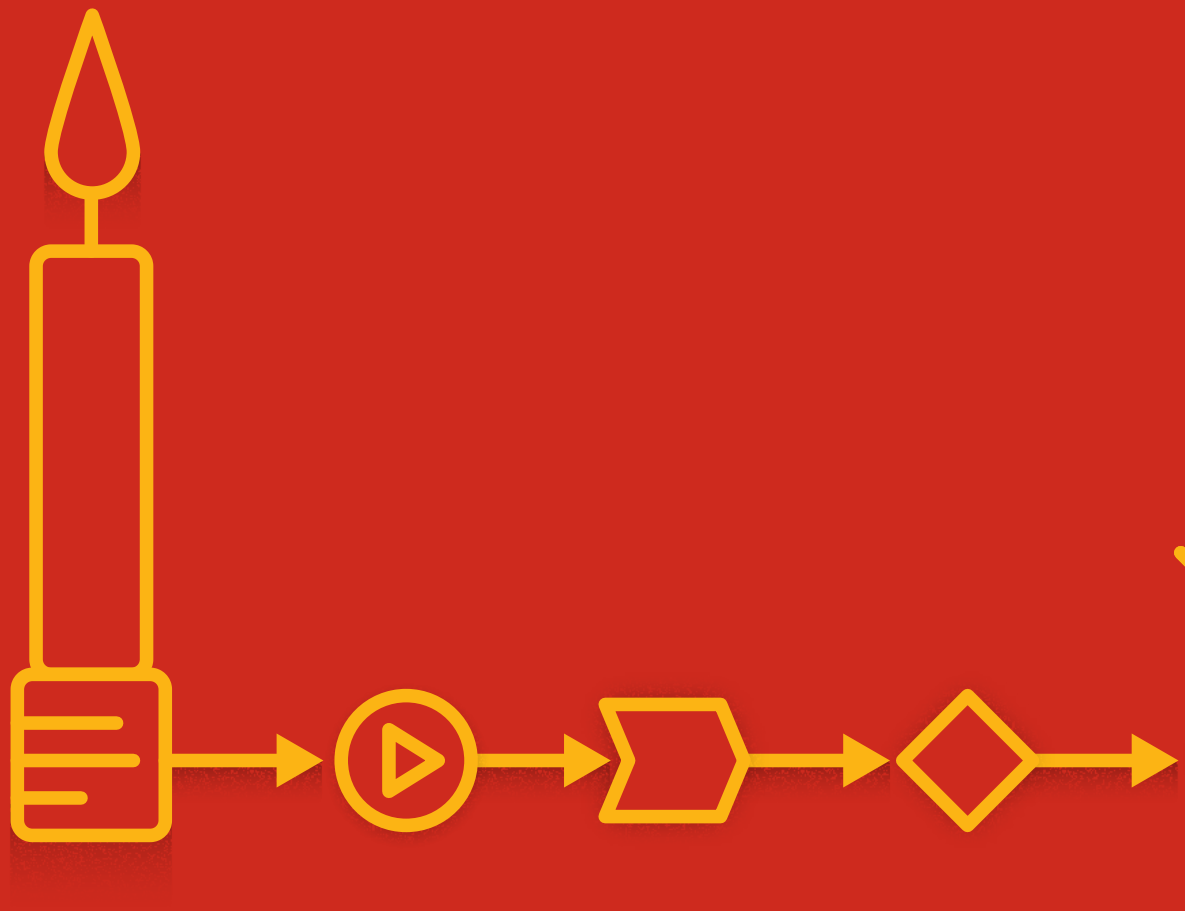
outsystems

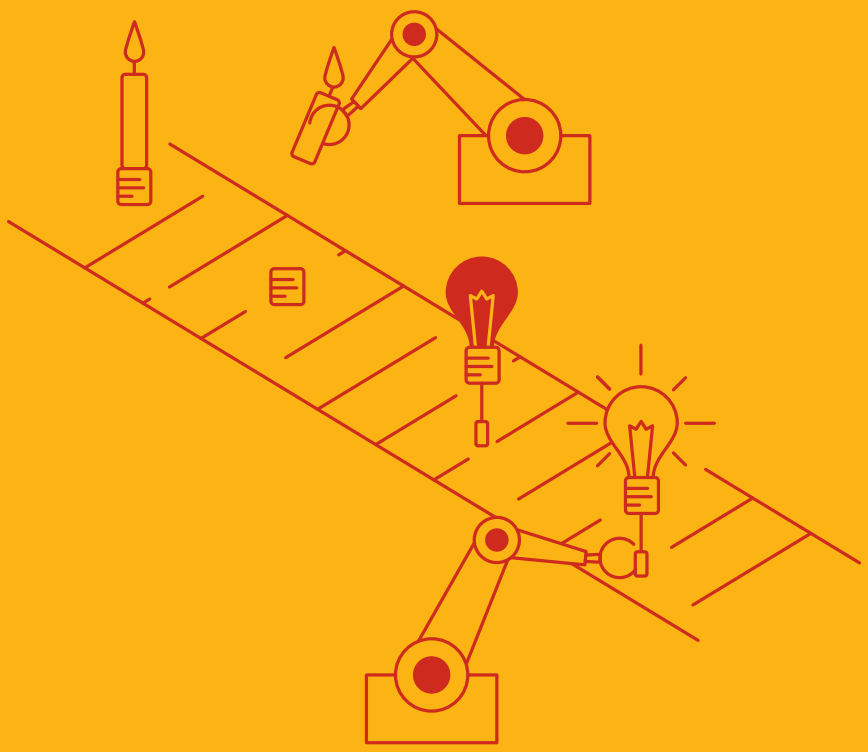


Legacy Modernization:

Finding Your Way With Low-Code

As of June 2018, **93 percent of organizations have plans to pursue digital business transformation,** but 59 percent of organizations are in stage 2 or 3 (out of 5 stages). They are running digital projects and making progress, but not generating the headway required to achieve the larger goal of digital transformation.[1]

Aging legacy systems are crippling IT's ability to innovate and bring new value to the business. And still, IT is expected to deliver new apps for customers or partners. But how can they do that when saddled with squeaky-wheel systems that consume massive amounts of time, budget, and resources? It's time to rethink the notion that enterprises must limp along, babying these behemoths as part of business as usual.



[1] Murray, Sarah. "58% Of Organizations Are at a Digital Deadlock - IDC and the CIO Executive Council's Guidant Supports Executives on the Digital Transformation Journey." IDC: The Premier Global Market Intelligence Company, 20 June 2018, www.idc.com/getdoc.jsp?containerId=prUS44014818.

outsystems

Index

Succeeding in Today's Modern Business

outsystems

Barring financial constraints, there are two main things that IT teams need for success:

- **Agility:** The ability to adapt every part of your business to the changing needs of the market, technology, and your customers. This includes everything from back-end operations to front-end user experiences.
- **Omnipresence:** Meeting your audience wherever they are. For many organizations, this is a combination of physical locations plus digital experiences that look and feel seamless regardless of medium.

If these two things are all it takes to be successful, what is preventing organizations from doing them? Two words: legacy gridlock.

Legacy gridlock is hampering digital innovation in organizations around the world. Here's why.

Legacy Systems Slow Development

The negative effects of aging legacy systems include ratcheting up the backlog count by reducing development speed. A typical app on iTunes has about 50,000 lines of code, and data analytics firm CAST estimates the average application has approximately 300,000 lines of code.[2] And most companies have 10 or more of these on their "to-do" lists. Development efforts are hampered by the massive amount of time it takes to design apps to integrate with legacy systems and research all the dependencies involved. According to the OutSystems State of Application Development 2018 report, legacy system integration and the development of APIs slow down app development for 60 percent of IT organizations.

[2] "Technical Debt Estimation." Risk Management in Software Development and Software Engineering Projects, www.castsoftware.com/research-labs/technical-debt-estimation.

The Legacy System Talent Pool Is Shrinking

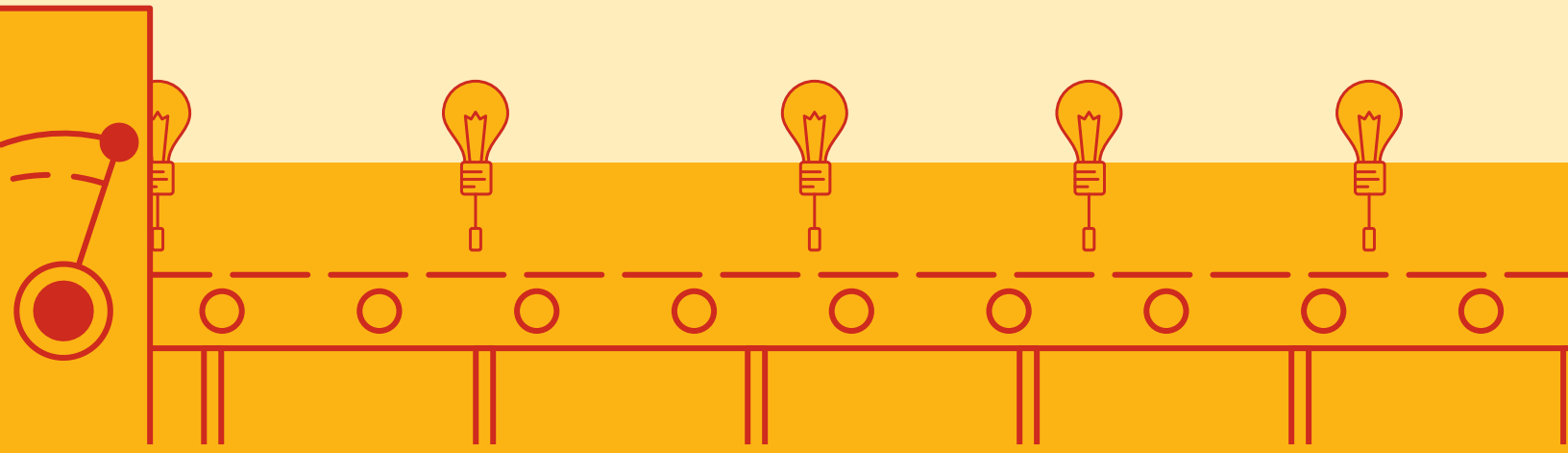
Many studies agree: the talent who can maintain legacy systems and build apps for them is a rapidly shrinking pool. The resources and knowledge it takes to care for and write code for older systems are disappearing due to age or attrition. How many COBOL, PL/I, ABAP, and Lotus Notes programmers do you know?

Then there's the fact that app development isn't a learn-a-technology-once-deliver-forever initiative. It's an ongoing upgrade, requiring regular acquisition specialty skills covering UI/UX design, security, platform architecture, and more. Those who can use these skills to build modern apps aren't going to be able to easily write code that integrates them into a 15-year-old ERP. Even if organizations are lucky to find one of these one-in-a-million "specialists," they come with a high price tag. And, agility and delivery can still suffer from delays and oversight.

The Costs of Maintaining Legacy Systems Is Sky High

Another statistic from the State of Application Development report is that 70–80 percent of an organization's development efforts are spent on maintenance of existing systems rather than on innovative new apps and features. The CAST study assigns a average of $3.61 of technical debt for each released line of code. The math on that amounts to more than $1 million in carry over costs associated with fixing problems that remain in released software in an average enterprise application.

Maintaining older systems focuses budget and resources on solutions designed for yesterday's business environment, not today's or tomorrow's. New integration opportunities, new designs and layouts, and new talent hires all get sidelined in favor of keeping the lights on. According to the U.S. Government Accountability Office, 74 percent of all IT budgets in government use "all of their funds on operations and maintenance."[3]



[3] Office, U.S. Government Accountability. "Information Technology: Federal Agencies Need to Address Aging Legacy Systems." U.S. Government Accountability Office (U.S. GAO), 25 May 2016, www.gao.gov/products/GAO-16-468.

Legacy Systems Add to Fear, Uncertainty, and Doubt

Here is a startling fact: 14 percent of all IT projects fail completely.[4] CIOs are natural-born risk-takers; it comes with the territory. That doesn't mean it's easy. Digital transformation efforts—essentially moving away from what's tried and true and doing something very different with no benchmarks—require lots of supporting data. Ultimately, it's an educated leap of faith. But nothing is guaranteed, and word travels fast. Per Gartner, 75 percent of all ERP projects fail to meet deadline goals, and this is the year that enterprises will insist on post-modern ERP project deployments that deliver proven value in less than two years.

Any one of these four reasons associated with legacy gridlock could serve as a pointing finger for why an organization hasn't taken advantage of the benefits of modernization. Together, they paint a bleak picture for established companies and portend big possibilities for smaller, more nimble entrants not bogged down in legacy systems and processes.

[4] Florentine, Sharon. "IT Project Success Rates Finally Improving." CIO, 27 Feb. 2017, www.cio.com/article/3174516/project-management/it-project-success-rates-finally-improving.html.

Is Your Organization Ripe for a Revamp?

outsystems

Knowing now what others are experiencing, how can you tell if it's time to consider a digital modernization effort in your organization?

- ☑ Your business is being disrupted by more nimble competitors.
- ☑ Your backlog is increasing despite increases in staff.
- ☑ Development times are long and getting longer (more than three months).
- ☑ Innovative development efforts comprise a very small portion of total activities.
- ☑ Finding talent with skills to support existing systems and help develop new apps is getting more difficult.
- ☑ Agile and DevOps returns have stalled despite continued investment.

Your Options

Facing a mandate to upgrade parts or all of your application portfolio, what are your options?

Forrester lists seven modernization options for legacy applications in its aptly named, "7 Options to Modernize Legacy Systems" guide.[5] But in the end, it really comes down to three options, each incorporating parts of the other four.

Replace with a commercial, off-the-shelf solution

Rearchitect

Rebuild

outsystems

Replacing Legacy Apps and Systems

There are passionate arguments for and against purchasing a customizable, off-the-shelf solution as a method for modernizing. If this is the situation your organization is in now, the question becomes, "Why get back into it with another vendor?" While most modern ERP applications are customizable, that customization comes with added costs.

If you are pondering this option, here are things to consider:

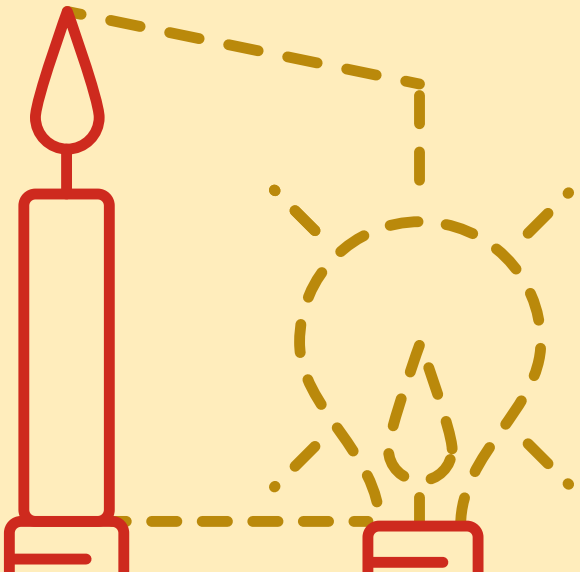
- **Redesigned processes:** In many cases, a new commercial solution will require significant changes to how you do business. These applications have a specific design and process flow, so change usually means that you change, rather than the commercial app you're buying.

- **Expertise:** Once the system is up and running, how much of the day-to-day will your existing team be able to manage versus additional support costs paid back to the vendor? What is the long-term ROI on vendor support costs, not just how badly it impacts this year's OPEX?

- **Customization:** Most commercial software is not industry-specific. But most companies have very specific needs and requirements their business must meet. Every custom deviation from the original source code comes with a cost and adds layers of complexity both now and down the road.

- **Uniqueness:** Does your business have at least one differentiation that makes you unique? Will moving to a new commercial application eat away at that uniqueness?

[5] Moore, Susan. "7 Options to Modernize Legacy Systems." Gartner IT Glossary, Gartner, Inc., www.gartner.com/smarterwithgartner/7-options-to-modernize-legacy-systems/.



Rearchitecting Legacy Apps and Systems

Whether the apps you need to modernize are part of a commercial solution or fully developed in-house, sometimes it makes sense to keep what you can and modernize the rest. This is where rearchitecting comes in. You can keep the core of an app or system that still works for your business and add new functionality without the cost and time of a full rebuild.

If rearchitecting appeals to you, consider these points:

- **Familiarity:** Nobody knows your apps and code like you. When it comes to dependency mapping, skills, and processes, if it is simple to upgrade the parts that can be upgraded without downtime and without breaking other parts of your app portfolio, rearchitecting can be a solid option.

- **Risk:** Rearchitecting isn't without risk, but it can be more palatable than the prospect of shutting off apps and services and switching over to something entirely new.

- **Procrastination:** At the speed technology changes, updating existing code for new functionality, as long as you aren't sacrificing business value, can be a strong reason for rearchitecting in the right circumstances.

Rebuilding Legacy Apps and Systems

For all the reasons outlined in favor of and against replacing or rearchitecting, there are equally valid reasons to start from scratch. And in the process of doing so, it's possible to implement some or all of the same functionality and re-architectured benefits to boot.

Consider these aspects of this option, for example:

- **Proprietary:** Specialized businesses have specialized needs that are often irreplaceable off-the-shelf. In these situations, where there is a very real concern or need to maintain the confidentiality of not just the data on the systems but the source code, rebuilding may be your only viable option.

- **Slow and steady wins the race:** Rare is the circumstance that requires a massive rip-and-replacement of existing systems all at once. Most organizations approach rebuilding in phases with the first phase having nothing to do with actually rewriting code. With a fully rebuilt, customizable solution, you can start small and add additional features and functions as time and resources permit.

- **Future-proofing:** When you are in control of every aspect of your app's design and deployment, you–not a third-party software vendor–control how and where it can be used.

Why Rebuilding Makes the Most Sense

It's called a conundrum for a reason. How do you completely overhaul core systems and all associated connected apps and services while also keeping up with business as usual? Keep these three things in mind.

outsystems

Migration

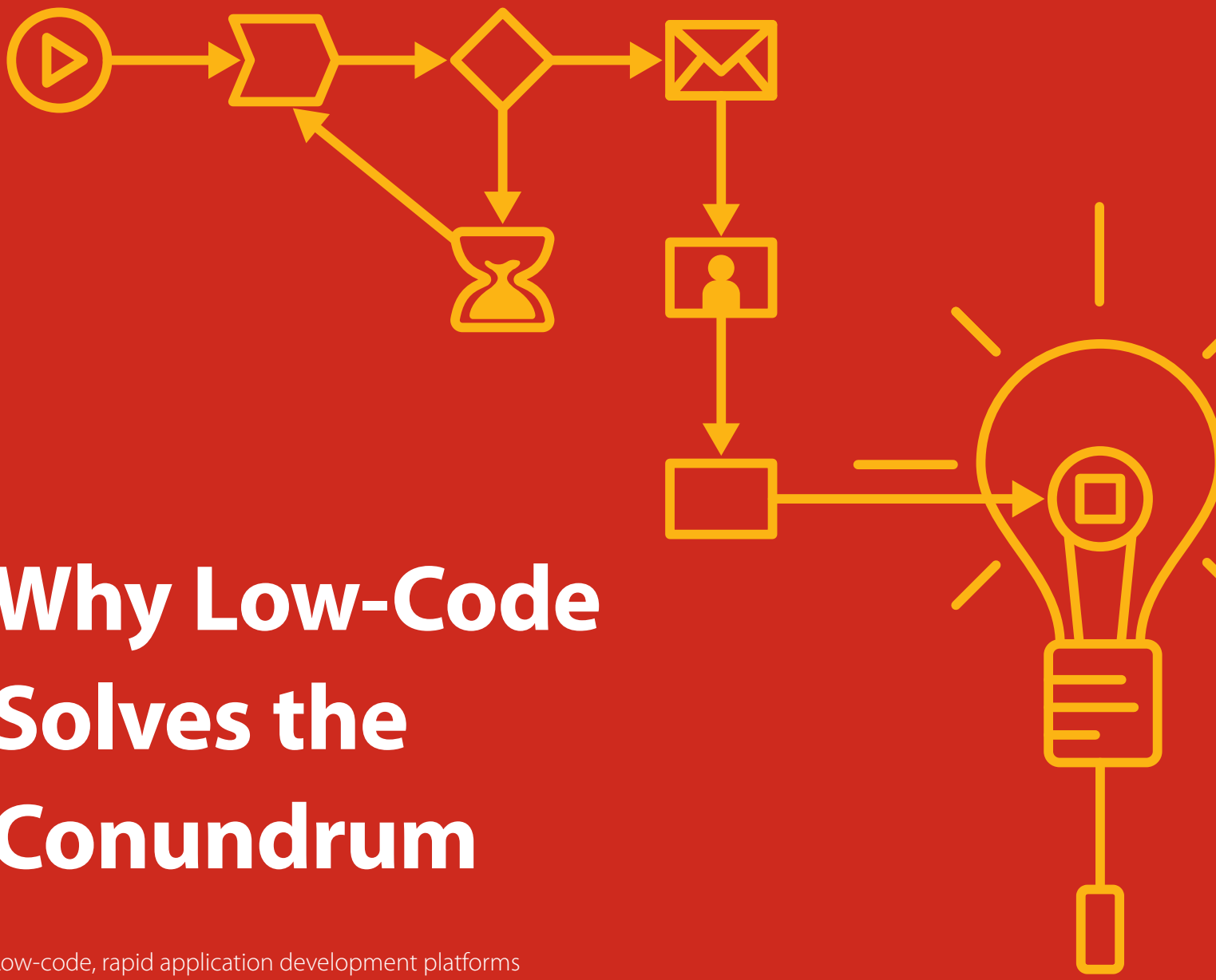
The older a system gets, the more obvious its limitations become. One (temporary) patch for these limitations is the use of APIs to extend the functionality of the system. But if we're being honest, that's simply kicking the can a bit further down the road. Sooner or later, you have to address the fact that you cannot keep limping old systems along at the pace you need and with the resources you have available.

End Users

Most SaaS (software-as-a-service) offerings come with the vendor's own ideas about accessibility and UX/UI. This may or may not meet your users' needs—internal or external. For organizations with SaaS accessed by many, the solution's limitations are likely to create an adverse effect. They could cause the organization to fail to meet the ever-changing data requirements they have for the business, for example. Or, unstructured data could end up at the bottom of a data lake. Either situation can result in lost revenue opportunities at best and unhappy and disenchanted customers at worst.

Agility

Everyone knows what being Agile means, but the nuances go deeper than any methodology. Particularly when you're talking about buying SaaS, questions about customization, accessibility, future-proofing, and support all come to mind. It's hard to imagine that a platform designed by someone else will enable you to use new kinds of technology like containers and microservices and be something that your developer teams understand from top to bottom.



Why Low-Code Solves the Conundrum

Low-code, rapid application development platforms are the best of both worlds in the build vs. buy discussion. Yes, you're purchasing a platform, but you can build what you need, how you need it, using the latest architectures and best practices. Your other option, a standard, non-low-code development center, would require additional skills in highly specialized focus areas such as security and user experience, driving up costs and slowing down development.

Once "bought," a low-code platform addresses the "build" part of the equation as follows:

- **Your business is not off-the-shelf:** Not now and not in the future. Your organization has particular needs for data access and control, unique business requirements, specific integration, and more. With low-code, you can address all your special needs.

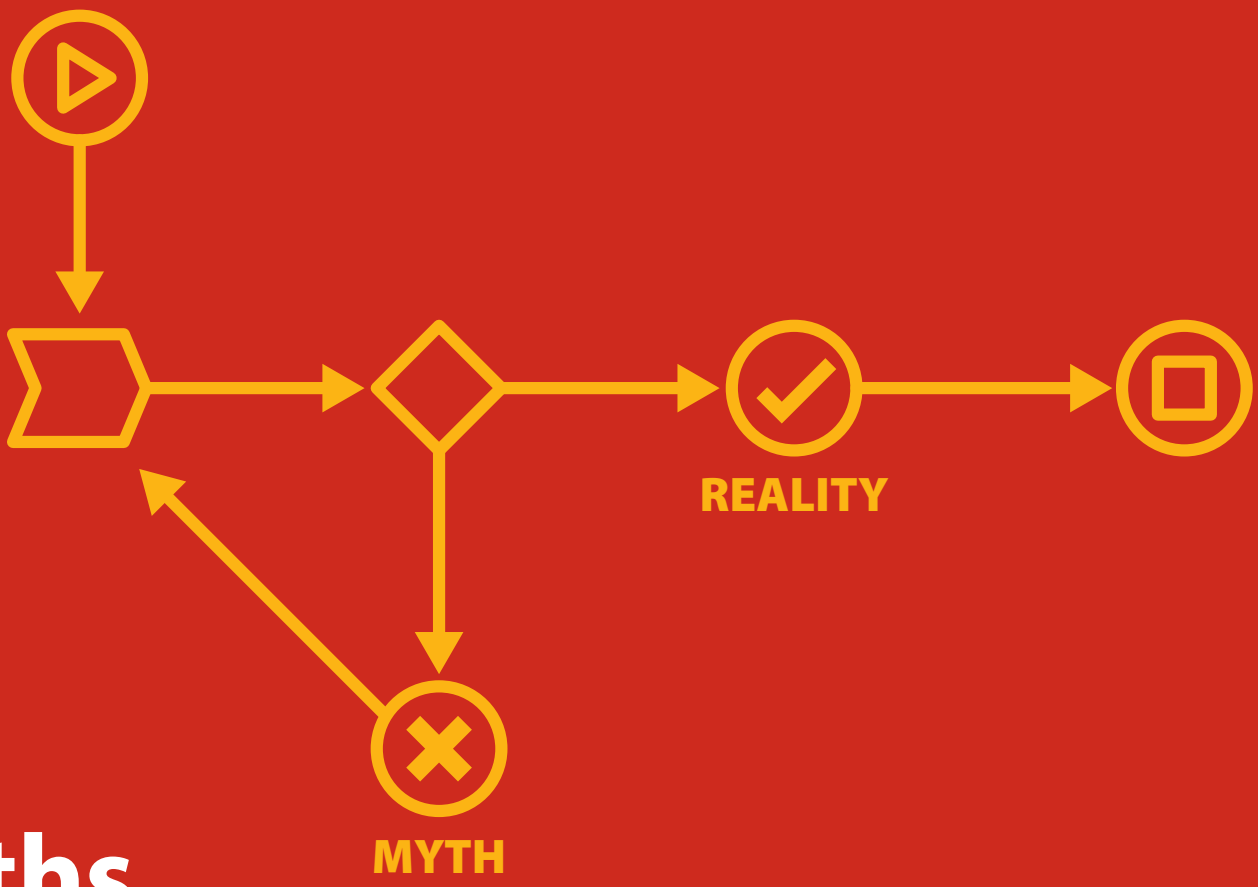
- **You avoid vendor lock-in:** While there is something to be said about buying a SaaS platform from an established vendor, businesses change. In a decade, you've customized your on-premises or SaaS platform so much and you have so much proprietary code and data in it, that trying to back out now with your data and business intact is nearly impossible. Low-code frees you from all that.

- **Your build integrates with everything:** Software is designed based on current existing technology. But technology changes and there is no guarantee that software you buy today will be compatible with something your business needs in the future. With low-code, you create the building blocks and APIs that will be compatible no matter where your business takes you.

- **Do you need more? Build more:** With a low-code platform, backlogs aren't an argument for not doing something anymore. Need an internal app to support inspections? No problem. These are just as easy to build as a new front-line mobile app.

- **Future-proof your architecture:** Development today is all about speed. Legacy architectures don't allow for speed and delivery improvements like containers and microservices. A low-code platform is ready for whatever technology improvements come your way.

- **Agility is no longer an issue:** There will always be two camps in IT. One camp needs to continually experiment and push the boundaries of what can be done with existing systems (and upgrading when necessary). The other camp is more focused on ensuring everything keeps running without any business disruption. Doing both of these requires agility, and this is where low-code shines. Plus, it facilitates key DevOps practices around automation, integration, and continuous delivery.



Myths and Realities of Low-Code and Legacy Modernization

As with most new technology, low-code had humble beginnings. But low-code has matured and is now considered by industry experts to offer 70–80 percent of IT application requirements.[6] Let's dispel some myths.

[6] "High-Productivity Application Platform-as-a-Service, Gartner Magic Quadrant." Lead With Speed, OutSystems, 2018, www.outsystems.com/1/high-productivity-apaas-gartner-18

Myth 1

Low-code isn't for hardcore developers.

Why it's a myth: Does using lasers and computer-guided surgical techniques make a surgeon any less of an expert? No, and neither does a low-code platform make a full-stack developer any less of an expert. Low-code isn't doing the work for the developer; it's just making what the developer does already faster and more intuitive.

Myth 2

Low-code forces me to use templates, so I can't customize anything or use custom code to build features I want.

Why it's a myth: Low-code is not "no-code." Sure, it's possible to deliver apps that require little to no customization, but that's hardly going to satisfy that 70–80 percent of enterprise app requirements that Gartner talks about. Not everything you need can be built into a no-code platform. Enterprise-grade low-code platforms enable you to develop what you need or get it from a repository after someone else builds it.

Myth 3

There's no real difference between low-code and no-code.

Why it's a myth: Without getting into the weeds, the difference between the two is similar to the difference between a calculator and a laptop computer. With a calculator, just about anyone can use it to do some very useful things, but you're limited to what the device can inherently do. You can't reprogram it. You can't download games to it. It is a purpose-built product. A laptop, on the other hand, can do everything the calculator can do. But, add a little bit more expertise, and its usability goes far beyond simple mathematical calculations. Similarly, a low-code platform allows almost anyone to develop apps faster. But the ceiling for professional developers using a low-code platform is practically limitless.

> *While adding new digital customer or employee experiences can drive significant business improvement, transformation must go deeper. Digital transformation requires new digital business capabilities to help the business deliver customer outcomes in entirely new ways.*
>
> — Forrester, Five Myths of Digital Transformation[7]

Myth 4

Large applications take more than 18 months no matter how many resources you throw at them.

Why it's a myth: This is a self-fulfilling prophecy and nothing more. Anytime a project goes on this long, the more likely it is to suffer from delays, budget cuts, and plain-old changing requirements-itis. Agile development practices, coupled with rapid application development (à la low-code) and cloud-based hosting, have proven that even large apps shouldn't take much longer than 12 months–tops! And even if requirements change in the middle of the project, IT teams can quickly go in a different direction as necessary with little impact to timelines.

Myth 5

A bimodal IT strategy will help us modernize without having to completely rebuild.

Why it's a myth: The idea behind bimodal IT is that you can have your cake and eat it, too. IT teams can keep some of the slower and safer practices associated with development and rollout for mode 1 systems of record (such as back-ends), while enabling those who build mode 2 systems of engagement (front-ends and mobile apps) to be more nimble and entrepreneurial.

[7] Fenwick, Nigel. "Five Myths and Misunderstandings Of Digital Transformation." Forrester, 26 Sept. 2018, go.forrester.com/blogs/five-myths-digital-transformation

Here are questions to ask yourself before suggesting this in a board meeting:

1. How big is my IT team? Do we have the headcount, and more specifically, the diverse skills to pull off two independent and divergent workstreams?

2. New systems of engagement products, features, and functionality still need to integrate into my older systems of record. If my new agile team is even two times faster than my legacy team, am I just creating a new backlog stream and how will that affect my agile team's efforts?

3. At what point does Moore's law begin favoring our truly digitally transformative competitors, leaving us with a (still) legacy core system of record while others are (still) faster and more innovative?

There are valid use cases for improving legacy "things," but most of them boil down to attachments: what we're used to, what we're comfortable with, what we know. These are not justifiable reasons for being "safe" in business. Being innovative is rarely comfortable and often requires a journey down the road less traveled. John Shedd of Marshall Fields said, "A ship in the harbor is safe, but that's not what ships are made for."

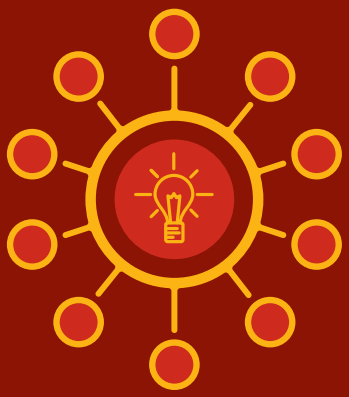
What is your business made for?

The Right Low-Code Platform for Legacy Modernization

In its last Magic Quadrant for Enterprise High-Productivity Application Platform as a Service report, Gartner noted 85 vendors all claiming to fit into the low-code category. But all low-code platforms are not created (or extended) equally. A low-code platform that is suited for complex undertakings like rebuilding aging legacy systems should embrace microservices, maintain team agility, enable visibility, enforce governance, standardize operations and portability, and offer modern user experiences.

This is where OutSystems outpaces other low-code vendors.

outsystems



Embracing Microservices

A service-oriented architecture (SOA) is the opposite of a monolithic one. Yes, it is difficult to consider abandoning the "known" processes that built the organization. But, unless the goal is to construct a clone of what you have now (which defeats the purpose of modernizing), then a micro-SOA, aka microservice architecture, is in your future.

What microservices support in OutSystems lets you do better:

- **Decouple the monolith** so that small changes do not turn into multi-day research and development dependency projects.
- **Isolate** those things that do need to change frequently so the change process takes hours, not days.
- **Reduce** dependencies between services by splitting up objects that have too many responsibilities so that they run faster, more independently, and require less upkeep as other parts of the app change.

Maintain Team Agility

Just as dependencies with apps and systems grow over time, so do dependencies between IT teams. As a result, the complexity of builds increases and slows down release cycles. And microservices, by themselves, aren't a silver bullet for reducing team dependency.

Only OutSystems improves team agility by enabling them to:

- **Automatically analyze** new service deployments before launch to ensure no impact to existing services.
- **Identify a minimum set of impacted apps** in cases where the change isn't limited to a particular set of services.
- **Have advance warning** of any issues so builds don't break.
- **Spend less time troubleshooting** thanks to early warning and thorough impact analysis pre-launch.

Enabling Visibility

As your app portfolio increases in size and complexity, it can put stress on certain services. On average, each app in a large portfolio has as many as 10 interconnected services, which makes investigation into why an app is performing poorly (or even failing) much more difficult.

Improving visibility into your app portfolio is enabled in OutSystems as follows:

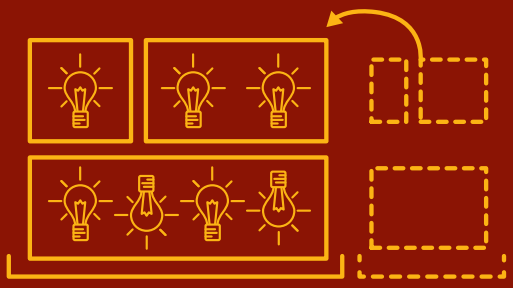
- **Embedded monitoring** tracks health and usage statistics so you can see how your entire app portfolio is doing at-a-glance.
- **Out-of-the-box analytics** continuously track key performance metrics so you can identify choke points and know when and why a given app or use case is not performing as designed.

Enforcing Governance

In many cases, enterprises using low-code grow their IT teams thanks to the ability to build more apps and the need to support them. With this growth comes the need to control which teams can create, manage, and consume services.

OutSystems helps organizations establish and enforce governance with:

- **A built-in entitlement model** that extends to all apps and services keeps the delivery pipeline flowing while eliminating worry about teams accessing restricted data.
- **Cross-functional team organization** models that align with business objectives allow for better information sharing.
- **Playbooks and helpful guidance,** along with recommendations for how and when organizations should introduce governance principles and the roles involved, help those who are just getting started with governance.

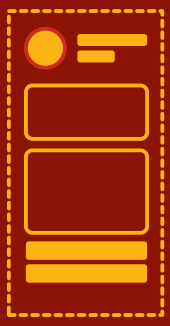


Standardizing Operations and Portability

In addition to microservices, containers are a way to encapsulate entire pieces of executable code so that they can be updated, moved, or replaced without having to also modify connected apps or services. This creates operational efficiency that benefits the entire software development lifecycle.

OutSystems lets you operationalize containers so you can:

- **Take advantage of existing container management platforms** such as Docker with Kubernetes and achieve more flexibility in scaling apps and better portability in deployment.
- **Still support non-containerized apps** now while allowing your IT teams to replace existing code with new container-based apps as business needs require.
- **Deploy container-based apps** to leading CaaS and PaaS providers, including Amazon ECS, Azure Container Service, Pivotal PAS, and other, on-premise Docker container environments managed by Kubernetes, Docker Swarm, and others.



Modernizing User Experiences

Enterprise development platforms need to satisfy UX requirements of internal and external users, customers, and business partners. Low-code development platforms can simplify UX design and help organizations standardize the look and feel of their apps across touchpoints.

Modern UI design in OutSystems:

- **Creates** a coordinated, modern look and feel across your entire app portfolio.
- **Offers more than 200 pre-designed templates** that satisfy approximately 80 percent of digital operations use cases.
- **Lets you customize** as little or as much as you need with the OutSystems UI Framework—from simple branding updates in your apps to complete layout changes using you own code.

How OutSystems Helped Other Companies Like Yours

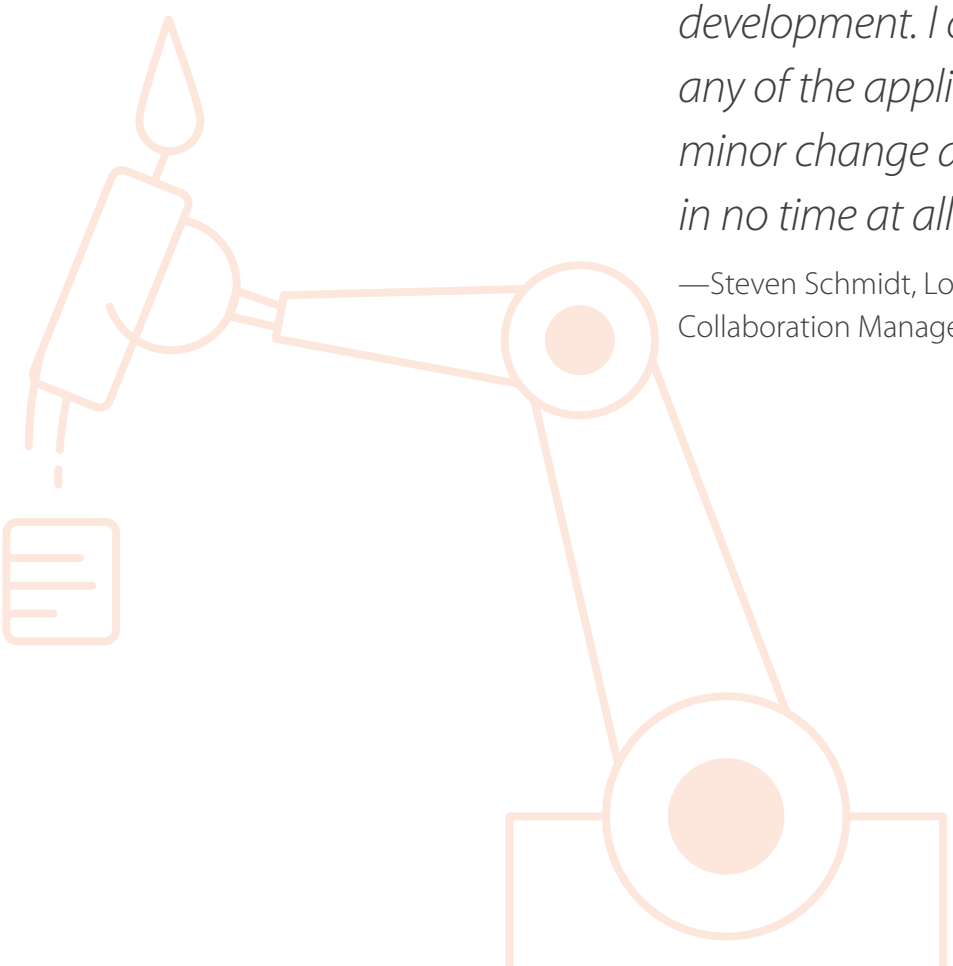
To understand just what OutSystems has to offer your company in terms of legacy modernization, take a look at these real-world examples.

outsystems

Logitech

Logitech used OutSystems to replace its **Lotus Notes** implementation and the thousands of apps supported by it. To-date, Logitech has built **80 apps** with a team of just **four developers**, including developing mobile apps, something the team had not done prior. OutSystems helps Logitech get more work done, with less effort, often developing new apps in just two weeks.

> *Low-code is really amazing when it comes to the ability to speed up our development efforts, but it's not just the development. I can go into pretty much any of the applications and make a minor change and just deploy a hotfix in no time at all.*
>
> —Steven Schmidt, Logitech Enterprise Collaboration Manager

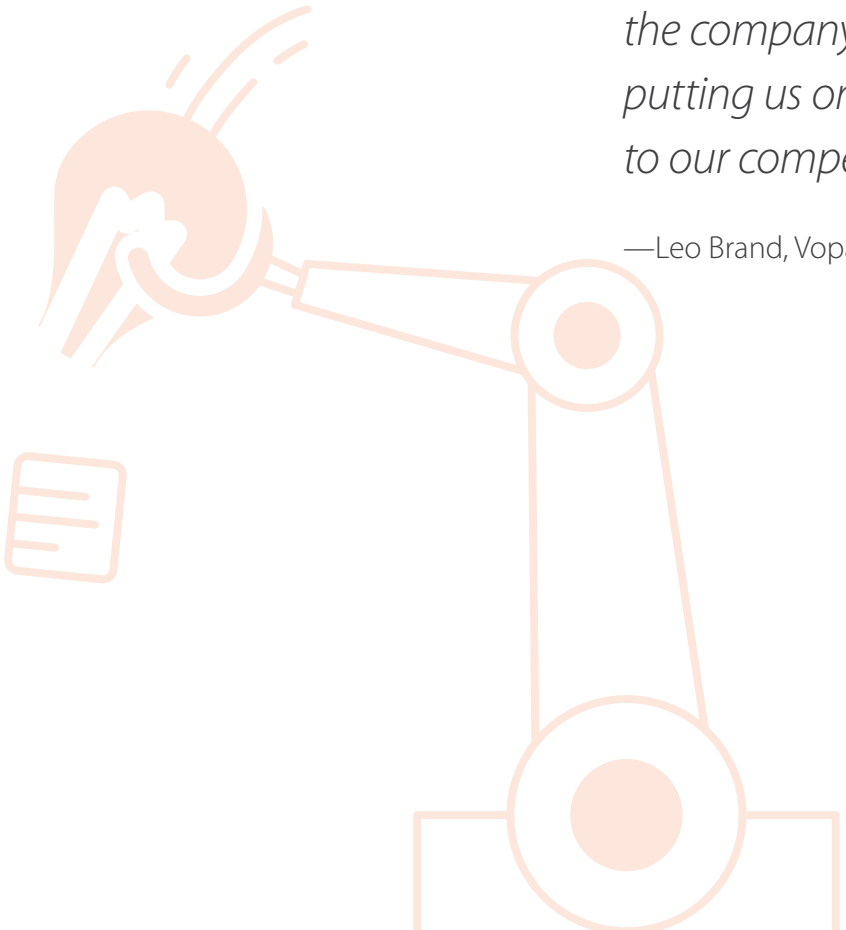


outsystems

Vopak

When the company's **ERP system** was being sunset, Vopak decided to build a **new terminal management** system themselves with a focus on cloud and mobile use. In just **12 months**, the company delivered a new terminal management system that supports its core logistics processes.

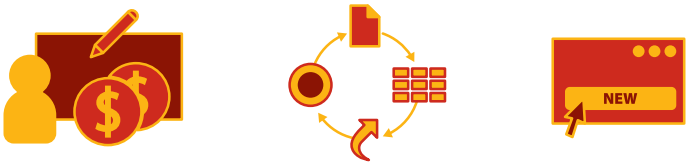
> *OutSystems enables us to build our own custom applications, bringing them to the business as a PaaS and implementing changes extremely fast. This gives Vopak the ability to innovate the company's core processes, really putting us on the forefront compared to our competition.*
>
> —Leo Brand, Vopak CEO



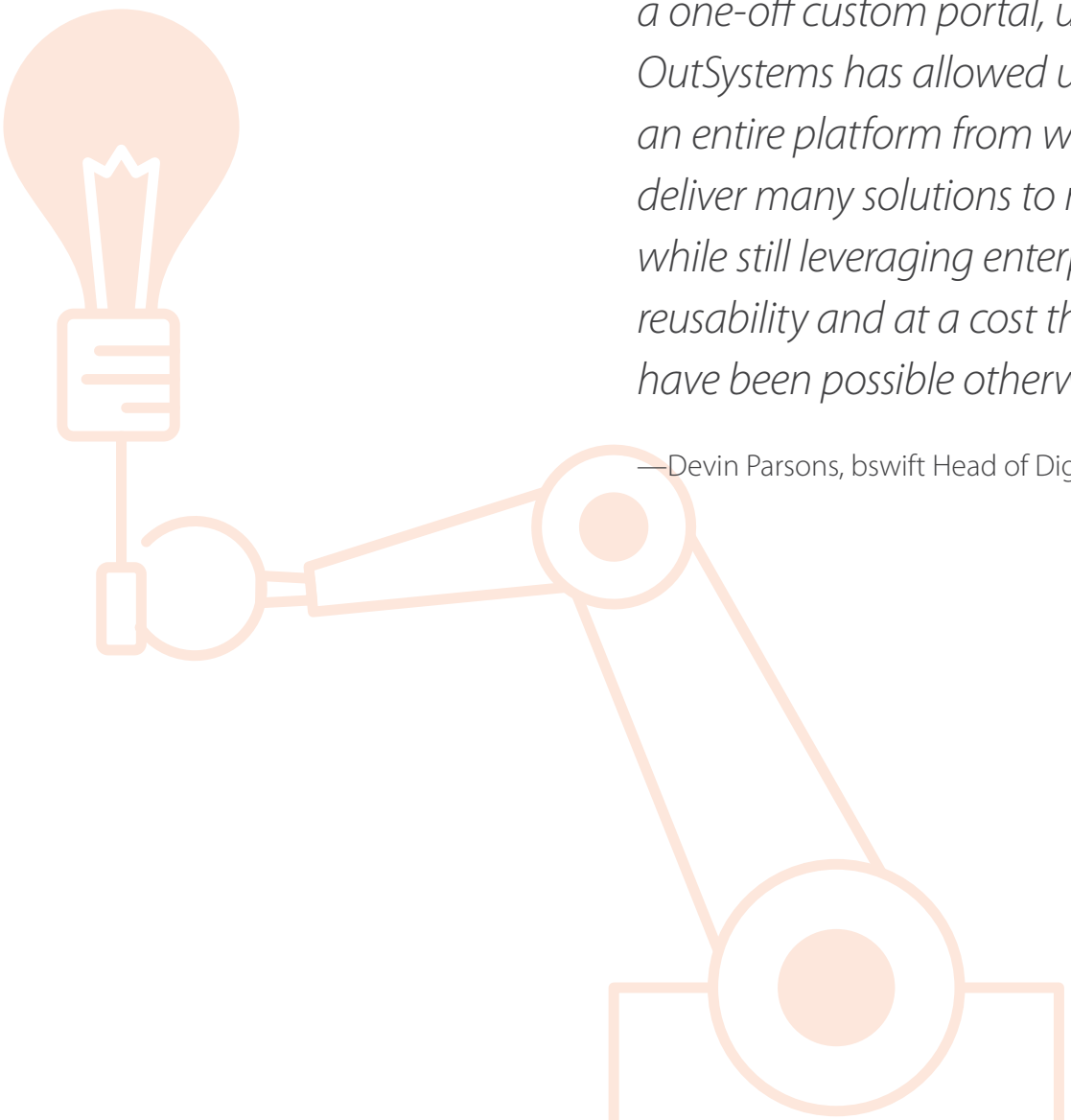
outsystems

bswift

HR and payroll administration company, bswift, is often asked to deliver **new custom portal solutions** for its customers. Wanting a more **repeatable process**, one that could bring additional value to the company, bswift used OutSystems to develop a **new microsite** that lets them deliver specialized solutions for their clients without the "custom" price tag or lengthy timelines.



> *"While we could have simply built a one-off custom portal, using OutSystems has allowed us to create an entire platform from which we can deliver many solutions to many clients, while still leveraging enterprise-class reusability and at a cost that would not have been possible otherwise."*
>
> —Devin Parsons, bswift Head of Digital Transformation



outsystems

Live Happily Ever After With Your Modernization

Whether you're designing solely for mobile and web or you need to deliver apps that satisfy the needs of an enterprise, there's low-code and then there's OutSystems low-code. If you want to modernize, and fast, OutSystems low-code is best suited for your initiatives.

outsystems

About OutSystems

OutSystems is the number one platform for low-code rapid application development. Thousands of customers worldwide trust OutSystems as the only solution that combines the power of low-code development with advanced mobile capabilities, enabling visual development of entire application portfolios that easily integrate with existing systems.

The Fastest Way to Build Enterprise-Grade Applications

- **Visually develop full-stack apps**
- **Integrate with everything**
- **Deploy to any device**
- **No lock-in, no boundaries**

Learn more at outsystems.com

outsystems

www.outsystems.com