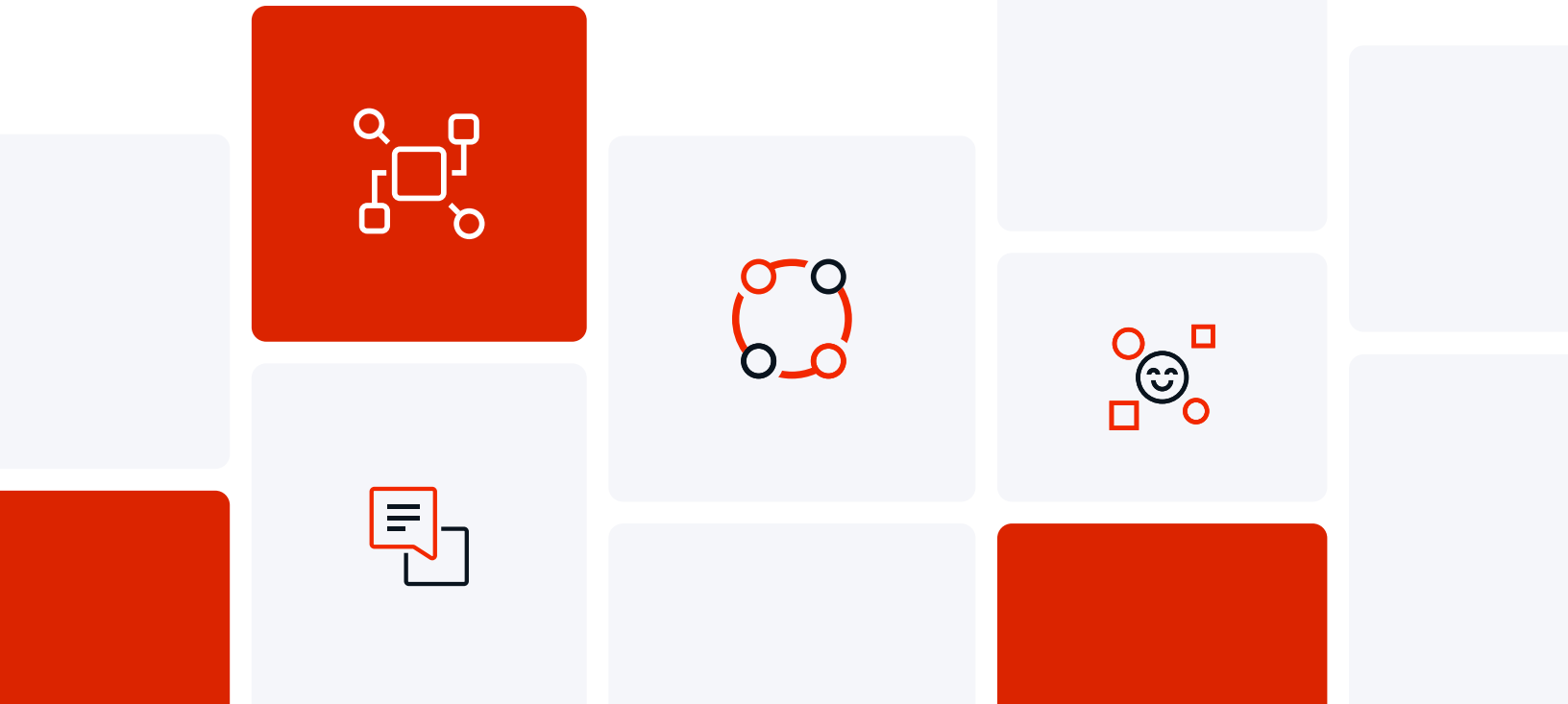
outsystems Intellyx | SWARAN SOFT - Tech Simplified -

WHITE PAPER

Human-Centric Integration

Integration in the Low-Code, Cloud-Native World

Authored by Jason Bloomberg, President at Intellyx



Integration has been the bane of IT ever since the rise of networks enabled applications to talk directly to each other.

Over the years, many hard-coded integration approaches and technologies have enabled enterprises to connect their applications and systems – addressing the business need for integration but leading to a rat's nest of complexity and technical debt.

Rising to this challenge is the new category of low-code integration platforms that bring the ease of use, flexibility, and collaboration benefits of low-code to the integration space.

To tackle the more complex enterprise integration scenarios, however, integration teams must change the way they work in order to leverage low-code across the full lifecycle – dealing with changing requirements and technology capabilities on an ongoing basis.

Why Is Integration So Difficult?

If only enterprise applications were like LEGO blocks. We could simply snap them together to build large applications that would work seamlessly. And then when requirements changed, we could unsnap, rearrange, and resnap them, well, in a snap.

Integration, unfortunately, has never really been much at all like LEGOs. The more realistic analogy: you have a box with a hundred different pieces of toys from a hundred different manufacturers. None of them snap onto any of the others, and all you have to stick them together are a dozen sticks of chewing gum.

Not so easy, is it?

Such is the life of enterprise integration professionals. Ask any of them, and they will tell you: integration is hard. In fact, it's always been difficult, in spite of the fact that over the years, professional integration tools have become easier to use.

There are simply too many different kinds of integrations. Integrations are expensive to implement and support, and it's difficult to scale them. Yet, while today's techie-focused integration tooling has helped simplify integration somewhat, it remains a complex, often costly challenge.

You have a box with a hundred different pieces of toys from a hundred different manufacturers. None of them snap onto any of the others, and all you have to stick them together are a dozen sticks of chewing gum

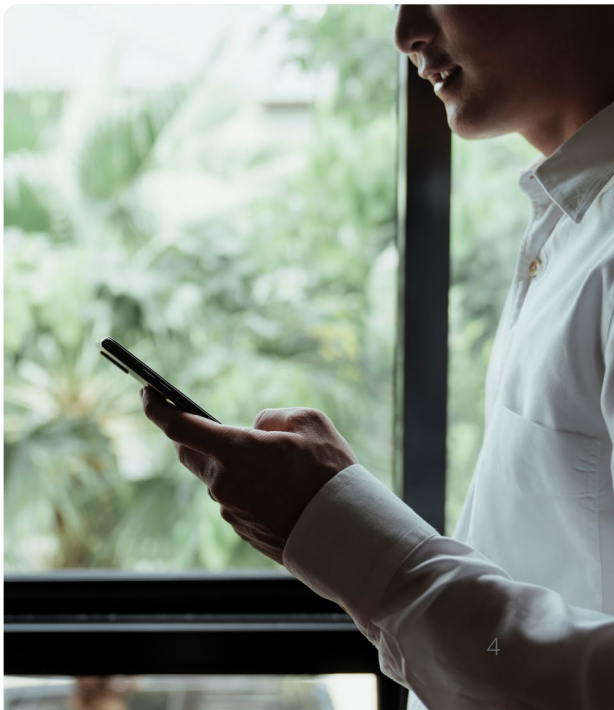
The reason: integration consists of a multifaceted set of challenges, as the diversity among endpoints, protocols, and technologies continues to advance, just like our box of mismatched toys.

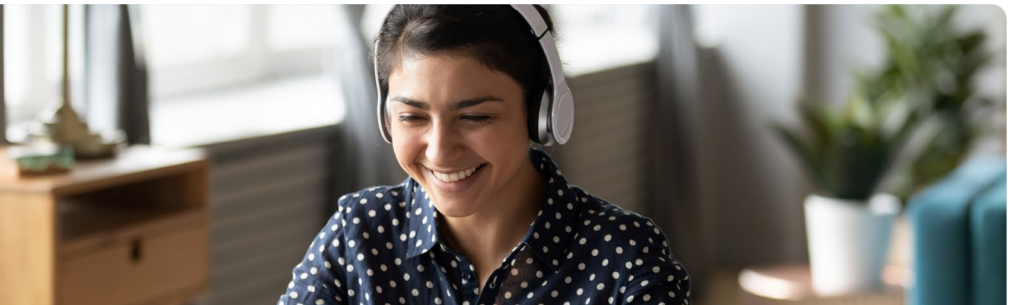
Furthermore, each combination of these elements requires a distinct set of best practices, integration patterns, and even tribal knowledge in order to meet the performance and security requirements for each integration.

Adding to this complexity is the fact we must also add cloud and container-specific patterns to the mix. It's no wonder, therefore, that most of today's enterprise integration tools require a high level of technical expertise, even as their user interfaces have matured.

Furthermore, building integrations represents only a fraction of the challenge. Managing and operating them adds significantly to the burden on staff as problems can happen at any time. Endpoints go down, process errors occur, and duplicate or out of sequence data are commonplace.

The bottom line: to deliver the power and flexibility of enterprise-class integration, an expert must understand how and when to apply each of dozens of integration patterns and consider all the possible error conditions that could make the integration break, regardless of the ease of use of the tooling at their disposal.





The Missing Element: Business Context

Nevertheless, over the years, software vendors have rolled out one generation of product after another in the hopes of simplifying the integration challenge. Included in this trend: 'low-code' integration tools that simplify and streamline the work of professional integration specialists, giving them drag-and-drop integration capabilities that speed their work.

For other vendors, however, low-code isn't simple enough – so they're rolling out what we might call 'no-code' integration tools suitable for non-technical business people to build a range of integrations, from straightforward to moderately complex. These tools eschew all but the most basic of integration patterns and leverage pre-built integrations whenever possible.

The presumption is that citizen integrators want to be able to connect A to B simply and without any technical expertise, for any value of A or B relevant to the enterprise – in other words, as simple as working with LEGO blocks.

> **In reality, there is no reason why a business person would want to become a 'citizen integrator.' And there likely never will be.**

However, this line of reasoning is incorrect. In reality, there is no reason why a business person would want to become a 'citizen integrator.' And there likely never will be.

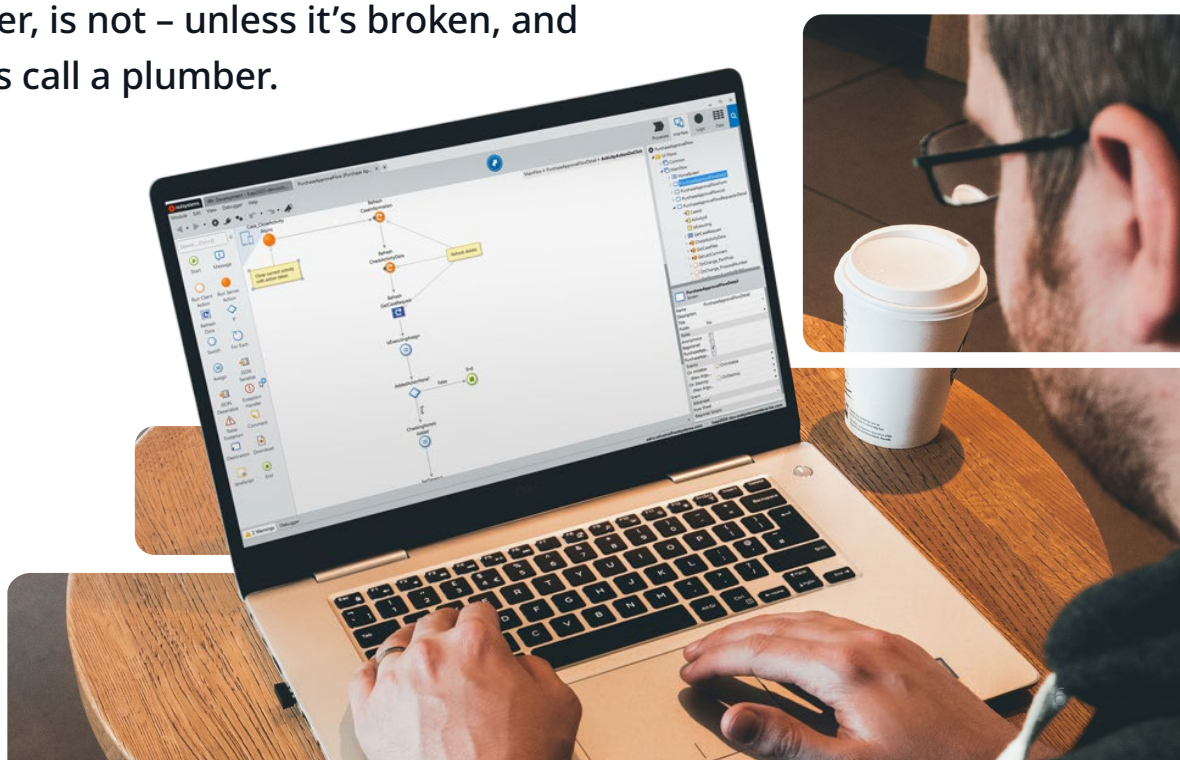
I'm not trying to say that there are no business users performing integration tasks. Examples of such activities are increasingly common. However, you're not likely to go into a business meeting and have a discussion about connecting A to B.

Business users may want insights into their business, or smoothly running processes, or lowered business risk, or any number of other business priorities, but connecting A to B isn't on the list. It's just plumbing.

Fundamentally, integration is a means to a business end, but the business doesn't really care about integration, as long as it's working properly. After all, you don't think about the water pipes when you brush your teeth – unless there's a problem, and then you call a plumber.

Here's the disconnect: as vendors think about what they might want a citizen integrator to do, they take plumbing tasks and simplify them. After all, there are plenty of situations where business users might want a hand in building business applications.

In the digital era, applications are business concerns. Integration, however, is not – unless it's broken, and then business users call a plumber.



The Challenge of Low-Code Integration

Any homeowner can clear a simple toilet clog with a plunger, but what do they do when the problem is more complicated? Buying a plunger that can handle such difficult tasks is not likely to be the answer. It's time to call a pro.

The same situation applies to integration. Simple integration problems – say, connecting a CRM app to an email app to automate sending of emails – may have simple solutions. True to form, there are a number of lightweight, no-code integration products on the market that handle these simple cases.

The more complex the integration scenario, however, the more difficult the integration – or at least, that's the conventional wisdom. The question is: can low-code overturn this wisdom? Is it possible to simplify the solutions to complex integration problems with better tools?

The answer depends on what we mean by solving a complex integration problem, given the multifaceted set of challenges facing modern integration today.

Considering the diversity of endpoints, the dynamic nature of business requirements, and the horizontally scalable nature of today's IT infrastructures, **we cannot simply jump to the conclusion that complex integration problems can have simple solutions.**

This error in judgment, in fact, has created numerous headaches for IT organizations over the years. Expedient integration solutions that gloss over the complexity of underlying applications may work in the short term, but long term add to the organization's technical debt.

> **Can low-code overturn this wisdom? Is it possible to simplify the solutions to complex integration problems with better tools?**

Technical debt, in fact, describes the primary pitfall to integration generally. We can address today's business requirements by connecting A to B, hardcoding the connection instead of building in any flexibility for future requirements change.

Such a course of action may be the quickest, least expensive way to deliver on current requirements, but creates a rat's nest of inflexible complexity that someone will have to clean up later.

The last thing we want is for our newfangled low-code integration tool to deliver the same false promises. 'Solving a complex integration problem' no longer means implementing some expedient hard-coded connection. Our organizations and our customers expect better.

Instead, we must free ourselves from the shackles of hard-coded integrations and allow for the fact that some integrations must be inherently dynamic – *after deployment*.

The goal of being able to change integrations dynamically at runtime was a dream of the ESBs and Web Services days. Early implementations of Web Services supported automatic discovery and integration, where endpoints could negotiate with each other automatically at runtime.

In practice, however, this type of dynamic integration proved impractical – and Web Services became yet another type of endpoint for hard-coding integrations.

Even today, the use cases for fully automated integrations that can change dynamically at runtime are limited. The sheer variety of different kinds of changes are simply too much for automated integrations to handle.

The Human in the Loop

To balance these priorities of changing integrations at scale, we must consider integration to be a *full-lifecycle activity*.

In other words, we must break free from the waterfall-centric belief that the pre-deployment and post-deployment phases for software are somehow immutable. Instead, we must take a page out of the Agile playbook, and approach the integration lifecycle in an iterative manner.

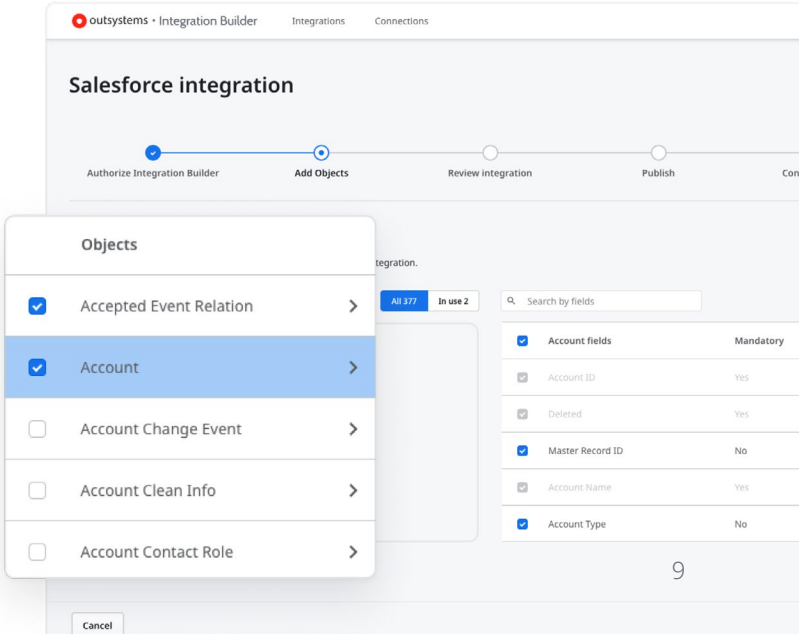
We're expecting constant change without slowing down the deployment of new and updated software, the theory goes, so we allow and plan for changes in the production environment.

This modern, dynamic approach to building software that supports change at scale applies to integration as well. Given the wide variety of integration scenarios and the diversity of changes that might impact them, however, a low-code approach to integration becomes absolutely essential.

Low-code empowers development teams and stakeholders to be more collaborative as well as iterative – not just while building applications, but while running and updating them as well.

Integrations won't always be perfect in the face of changing requirements and dynamic applications, but with low-code, the people responsible for those integrations have the power and control to keep them up to date across the software lifecycle.

In other words, the right low-code integration tools – tools like OutSystems Integration Builder – combine the best of automation and low-code simplicity, where the software automatically creates and updates integrations when possible, while facilitating human integration actions via low-code when necessary.



The Integration Feedback Loop

This low-code vision for integration disrupts the traditional hard-coding 'one and done' mentality that has led to so much technical debt in the past.

Instead, the integration team leverages an ongoing feedback loop, as requirements drive integrations, which in turn drive the user experience, leading to new and changed requirements. This feedback loop exemplifies what we mean by full lifecycle integration.

The full-lifecycle approach to integration reduces technical debt by eliminating integration drift. Integration drift refers to the fact that requirements for particular integrations and the specific capabilities of those integrations tend to diverge over time, both because requirements change and also because the capabilities of the applications may evolve as well.

Furthermore, keeping the requirements central to full-lifecycle integration has the result of focusing the integration effort on the customer experience, rather than the underlying systems of record or other applications the integrations connect to.

While low-code integration tools may be necessary to achieve full-lifecycle integration, they are not sufficient. The integration team must also rise to the challenge with updated skills and a greater willingness to collaborate than integration required before.

That being said, the combination of the right tools, the right people, and the right processes is as one might expect the key to success – even in today's modern, dynamic IT and business environments.

The integration team must also rise to the challenge with updated skills and a greater willingness to collaborate than integration required before.

Solving Modern Problems with Human-Centric Integration

Even though they add to technical debt, hard-coded integrations have met many of the needs of enterprises for decades. If such integrations continue to meet the business need, there may be no good reason to replace them.

In what situations, therefore, is low-code, human-centric integration particularly useful?

Perhaps the most important use cases are when the organization wishes to transform its customer experience (CX), a central part of any digital transformation effort.

Examples of CX transformation include improvements to the customer journey (including the onboarding process), as well as more flexible, customer-centric self-service portals.

Vertical industries that primarily interact with customers via omnichannel experiences are particularly well-suited for full-lifecycle integration, including financial services, healthcare, and insurance.

For other use cases, the employee experience (EX) is the primary driver of low-code integration, for example, when building new employee onboarding applications or applications for field service operations.

Leveraging low-code integration for such process automation situations drives workplace innovation in a variety of industries, including retail, technology, telecommunications, manufacturing, utilities, and logistics.

The final critical use case for low-code, full-lifecycle integration is application modernization – or more specifically, extending existing applications like SAP, Salesforce, and Microsoft Dynamics. Such integration achieves this difficult goal by empowering the application team to build new functionality and a new CX for such applications. All industries can take advantage of such modernization capabilities.

The Intellyx Take

The Covid era has been challenging for most organizations as they doubled down on their digital transformation initiatives.

Critical to such transformations is the agility to change – how companies operate, and in particular, how they interact with customers.

Integration and low-code play a critical role in making all these changes possible on the timeline the business requires.

The reason human-centric, full-lifecycle, low-code integration is well-suited for solving such transformational problems is because the business and technology landscapes are particularly dynamic.

Enterprises can't afford to go slow, layering on expensive technical debt in the interests of expediency. Instead, organizations must rethink integration as part of their overall digital transformation – and low-code is an essential enabler of this change.

Next steps with [Swaran Soft Support Solutions Pvt. Ltd.]

Swaran Soft is a Digital Transformation experienced in Digital Adoption, Omni Channel Commerce, IoT, ML, AI & Big Data. We serve large Global Corporates, Government, Institutions & Innovation Hungry Startups for end to end IT adoption & business continuity services.

We envision longevity in our business relationships for a win-win associations.

Want to know more? Contact Swaran Soft Support Solutions Pvt. Ltd.:

Email Swaran Soft Support Solutions Pvt. Ltd.
info@swaransoft.com

Visit our site to learn more
www.swaransoft.com
www.lowcodedigitalfactory.com

About OutSystems

Thousands of customers worldwide trust OutSystems—the only solution that enables the visual development of entire application portfolios that easily integrate with existing systems.
Learn more at www.outsystems.com