Don’t Confuse Low-Code with No-Code

By Jason Bloomberg

As with most emerging markets, the exploding low-code and no-code segments are fraught with confusion. Even coming up with useful names for the offerings in these categories is a challenge.

The big analyst firms aren’t much help. Forrester lumps low-code and no-code together into the Low-Code Development Platform and Mobile Low-Code Development platform market categories, emphasizing mobile vs. some nebulous not-mobile (would that be stationary, perhaps?), rather than breaking out no-code into a different segment.

Gartner isn’t much better. It forgoes the low-code and no-code terminology altogether, instead touting the Application Platform-as-a-Service and the High Productivity Application Platform-as-a-Service market categories.

For Gartner, therefore, on-premises alternatives aren’t salient to the discussion, as both alternatives are strictly cloud-based. Furthermore, products that fall outside the High Productivity aPaaS segment are, what? Low productivity? Why would anyone buy one of those?

Enterprise buyers are left scratching their heads, as the low-code and no-code terminology remains remarkably persistent, in spite of Gartner’s exhortations to the contrary. Even so, these more popular terms are themselves laden with confusion of their own.

It’s time to clear things up.

Low-Code vs. No-Code: It’s not about ‘Code’

As a boutique analyst firm, Intellyx doesn’t deign to name market categories. Instead, we listen to our enterprise audience and seek to gain insight into the direction emerging markets are headed.

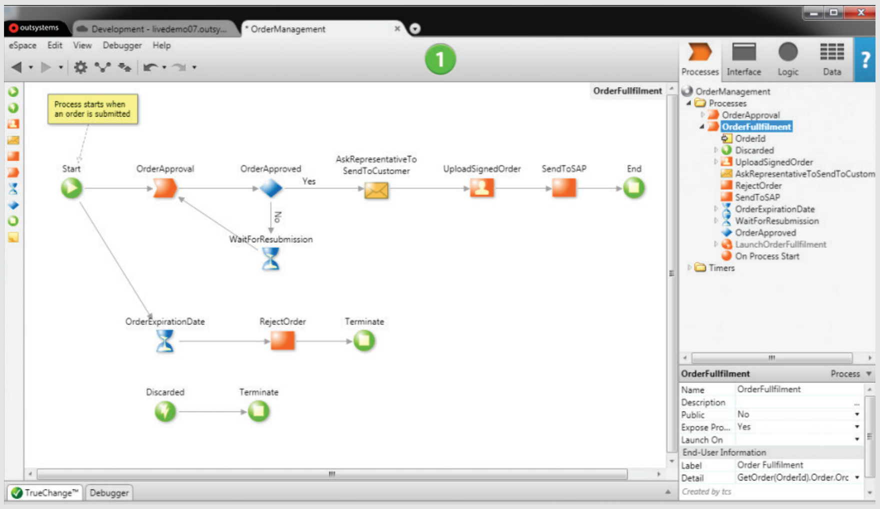
Based on many such conversations, we find that the following definitions have moved to the fore:

Low-Code – Next-generation rapid application development that accelerates and streamlines the work of professional developers.

No-Code – Self-service application assembly for business users who become ‘citizen developers.’

In spite of the word code appearing in both labels, therefore, the distinction between these market segments isn’t about whether someone is writing code or not. The more important distinction is the intended user.

Typical OutSystems interface - For the pro, but not about the code



In fact, the question of whether you need to write any code or not is more of a red herring than anything else. In fact, it's possible for developers to build sophisticated applications with low-code platforms like OutSystems without writing a line of code – although there are certainly reasons to write code on occasion as well.

Furthermore, most no-code platforms allow citizen developers to write code if they like, although coding is certainly optional for any platform that belongs in this category.

Additionally, whenever a citizen developer is creating an application on a no-code platform that requires integration with existing enterprise apps, they usually have to call someone in IT to help with such integrations. That techie will likely have to write some code to implement such an integration.

The Differentiated Value of Low-Code

If the point of using a Low-Code platform for professional developers isn't about the code-writing bit, then what is it?

Spend a few days with such an engineer, and you'll soon find out. Even when they're building applications using more traditional tools, only a part of their focus is on cranking out lines of code. These pros' broader role includes working within the overall context of the application architecture, understanding and modeling requirements, and ensuring applications are secure and high quality.

There is also an additional enterprise context to the work of software developers. The applications they build are rarely stand-alone, greenfield apps. In the far more likely scenario, any application they build or modify must integrate with many other applications, as well as middleware and other infrastructure for on-premises deployments and the self-service cloud environment when the apps go into the cloud – or both.

Layered on top of this enterprise architectural context is the broader business context, including multifaceted compliance considerations. Seasoned software engineers must navigate the hazards of PCI, HIPAA, or other industry-specific regulatory contexts, the enterprise's security and privacy rules, as well as established policies and procedures for building, testing, and deploying software in their organization.

Enterprise low-code platforms facilitate, streamline, and accelerate all of these mundane, time-consuming tasks, freeing up developers to focus on building great applications. True, they have to write less code than traditional application development methods require, which also saves time.

But the real win for developers using low-code platforms is taking care of all the cruft surrounding the code that slows them down.

The Hidden Pitfall of No-Code

The no-code value proposition clearly has appeal: non-technical users are able to rapidly assemble business applications using lightweight, visual, drag-and-drop tools. As such no-code platforms mature, the types of business applications these citizen developers can create become increasingly sophisticated as well.

Just one problem: enterprises have been down this road before, folks. Remember Microsoft Access? A database-centric application creation tool simple enough for even the least technical business users to use – and use it they did. In droves.

In a few short years, some organizations had thousands of Access-based apps running under desks, performing important or even mission-critical tasks, entirely off the radar of IT.

It's fun to pick on Access, but in reality, there have been dozens of such tools over the years, and they have all contributed to Shadow IT.

The causes of Shadow IT are well-known: the IT organization is too slow and process-bound to respond quickly enough to the needs of lines of business, so business users go around IT and buy – or build – their own apps and other technology.

The problems of Shadow IT, unfortunately, are also quite familiar: without proper oversight and coordination, security vulnerabilities and compliance violations can proliferate, with no one at the helm to address such issues. Furthermore, these citizen-built apps can be redundant, obsolete, or otherwise low quality.

For modern no-code platforms, this Shadow IT pitfall looms large. Certainly, some of the more mature no-code platforms tackle the Shadow IT issue head on, providing lightweight ways of ensuring that apps on these platforms are adequately secure, compliant, and address ongoing business needs.

Far more common, however, are less mature no-code tools that don't adequately deal with these pitfalls – common because no-code is a rapidly emerging market, and numerous startups are frantically rolling out their wares, putting Shadow IT considerations on the back burner.

The Intellyx Take

No-code platforms certainly have their place in the enterprise, but any manager considering whether to purchase one should take care to understand the potential pitfalls of such tools, as well as the tradeoffs between no-code and low-code platforms.

For stand-alone apps that don't require sophisticated integration and also don't present security or compliance risks, no-code can be a cost-effective means for opening up application creation to a broad business audience.

Once an application requires a greater level of sophistication, however, no-code platforms tend to fall short. Not necessarily because they lack requisite functionality, but because getting security, compliance, and integration right requires technical skill sets beyond those of a typical citizen developer.

For all but the simplest of applications, therefore, managers should consider the low-code option – not only because the apps developers can build with them can be more varied, powerful, and responsive than no-code apps, but also because enterprises require the expertise of professional developers to build their apps property, even when code-writing itself is at a minimum.

At that point, the decision isn't low-code vs. no-code, but rather low-code vs. traditional, hand-coded development. Given the numerous benefits that low-code platforms like OutSystems offer professional development teams, such a decision is an easy one to make.

Jason Bloomberg

President of Intellyx & Contributor to Forbes