Quick Answer: What Is the Difference Between No-Code and Low-Code Development Tools?

Published 9 March 2021 - ID G00745717 - 5 min read

By Analysts Jason Wong, Paul Vincent, Akash Jain

Initiatives: Software Engineering Strategies; Applications and Software Engineering Leaders; Software Engineering Technologies

The overused terms “low-code” and “no-code” are confusing application and software engineering leaders about which development tools to choose. Leaders should focus on evaluating these tools based on their fit to use cases and skill sets, rather than the labels used to describe the products.

Quick Answer

What is the difference between no-code versus low-code development tools?

- “No-code” is a marketing term, implying the tool is for non-professional developers.
- “Low-code” tools often support scripting for capabilities beyond only a “no-code” approach.
- Invest in tools that offer prebuilt “blocks,” but beware of tools that are highly specialized solutions.
- Ensure the no-code or low-code approach fits the skill sets of the target developers, whether professional or citizen developers or both.
- Regardless of marketing labels, make sure the tool supports your existing integration, automation, and software development life cycle (SDLC) stacks.

More Detail

Application and software engineering leaders are bombarded by vendors promising great benefits, such as speed and ease of use through the use of their low-code or no-code tools. Additionally, business unit leaders are demanding to use more of these tools to enable business technologists (identified as “technology producers” that do not report to IT) to produce apps and automations for their business needs (see Democratize and Distribute Technology Work Across the Entire Enterprise to Accelerate Digital Business).

Ignore vendor hype and promises, and instead focus on the underlying platform architecture and technology approaches of the tools to find the best fit ones — regardless if they call themselves low-code, no-code or both.
“No-Code” Is a Marketing Term, Implying the Tool Is for Non-Professional Developers

No-code development tools abstract the codebase entirely by providing a visual “what you see is what you get” (WYSIWYG) interface to building an application; from the data model to business logic to workflows to the user interface (UI).

Fundamentally there is really no such thing as “no-code.” It’s a marketing label, just like Salesforce’s slogan of “no software.” There is always code and software running somewhere, just hidden.

This complete abstraction of application behavior to avoid code means that any custom feature, and or function needed for an application, is limited to what is already surfaced in the tool. The no-code approach only allows for a certain level of customization through the WYSIWYG tool. Adding new capabilities will usually require the vendor, or its professional services staff, to create and release new features for use within the tool.

“Low-Code” Tools Often Support Scripting for Capabilities Beyond Only a “No-Code” Approach

Low-code development is rapid application development or high-productivity development, with an option to use code or scripting. These tools can use a variety of approaches that both automate and abstract application development activities, such as drag-and-drop editors, code generation, component assembly and model-driven and metadata-driven development.

Low-code development offers a way to add custom features or functions for an application with new code. The code supported can be a proprietary language or standards-based languages. JavaScript is one of the most popular and common languages supported by low-code development tools. This approach allows customers and partners to have a higher degree of customization than no-code tools. However, the added custom code must be carefully managed to prevent accumulation of technical debt and change management issues.

Invest in Tools That Offer Prebuilt “Blocks,” But Beware of Tools That Are Highly Specialized Solutions

Low-code and no-code development approaches typically abstract functionality using the concept of assembling “blocks,” which can be application components, external API references, templates, modules or widgets. Just as with children’s building block sets, each tool has different sized and shaped blocks. Example of blocks include:
Tools that are labeled both low-code and no-code ("low-code/no-code" or "LCNC") are just low-code in approach. Any development tool needs an expression language to describe business logic, regardless of whether it’s no-code or low-code. The complexity and sophistication of these expression languages determines the types of applications that can be constructed. The no-code aspect means that, for certain app use cases, the tool does not require any coding. However, this is also usually true for most low-code marketed tools.

Additionally, the terms low-code and no-code are so commonly used in vendor marketing that in some cases they merely refer to a specific application or solution that is easy to configure or extend. For example, an asset management application can be no-code or low-code if it allows for some extensibility beyond what comes out of the box. This is very different from buying a general purpose low-code or no-code development tool which can build many different types of applications or solutions. Evaluate these tools based not just on one application use case, but against several potential options to ensure it can grow with enterprise needs.

Ensure the No-Code or Low-Code Approach Fits the Skill Sets of the Target Developers, Whether Professional or Citizen Developers or Both

No-code does not necessarily mean no technical skills needed. Tools that are positioned as no-code aim to provide all the building blocks for customers, usually on an ongoing release basis. These vendors try to offer assembly blocks small enough so that developers can assemble a bigger custom application component if needed, but writing code is not allowed. No-code implies no programming language is used, but even some visual modeling tools require technical expertise or understanding programming metaphors.

Low-code requires more developer buy-in and skills alignment. Tools that are positioned as low-code tools allow developers to build new blocks that may be missing in the set, by writing some code. These custom blocks must fit properly with the out-of-the-box blocks, so development must use a specific language, framework, pattern or SDK.

Regardless of Marketing Labels, Make Sure the Tool Supports Your Existing Integration, Automation and SDLC Stacks

If your organization wants to use no-code or low-code tools to compose applications by assembling building blocks from multiple vendors, the pieces need to fit well. Usually this would require glueing the

- Integration connectors to popular SaaS applications or data sources
- UI components such as list views, a signature capture or camera control
- Process models for common business or industry-specific workflows
- Business logic to support business rules, validation and other application features
pieces, which means using APIs and integration code. However, some vendors will have prebuilt integrations (i.e., specific block connectors) and others will have a marketplace where partners can ensure that their blocks are built to the same specification and standards. This may involve creating an OpenAPI definition for a service or supporting webhooks for simple events.

When evaluating no-code and low-code development tools, go beyond the vendor’s marketing message to assess the underlying approach to their development tools or platforms. Just because a vendor is marketing “low-code” may not mean it’s for professional developers, and another marketing “no-code” doesn’t always mean it’s for citizen developers. Determine how code is abstracted and added because this will ultimately determine the extensibility and flexibility of their approach, and how easy the tool is to connect with other technologies.

**Recommended by the Authors**

- Magic Quadrant for Enterprise Low-Code Application Platforms
- Critical Capabilities for Enterprise Low-Code Application Platforms
- Solution Comparison for Low-Code Application Platforms
- Forecast Analysis: Low-Code Development Technologies

© 2021 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. and its affiliates. This publication may not be reproduced or distributed in any form without Gartner’s prior written permission. It consists of the opinions of Gartner’s research organization, which should not be construed as statements of fact. While the information contained in this publication has been obtained from sources believed to be reliable, Gartner disclaims all warranties as to the accuracy, completeness or adequacy of such information. Although Gartner research may address legal and financial issues, Gartner does not provide legal or investment advice and its research should not be construed or used as such. Your access and use of this publication are governed by Gartner’s Usage Policy. Gartner prides itself on its reputation for independence and objectivity. Its research is produced independently by its research organization without input or influence from any third party. For further information, see "Guiding Principles on Independence and Objectivity."