mrc

10

Must-Have Features of Enterprise Low-Code Software

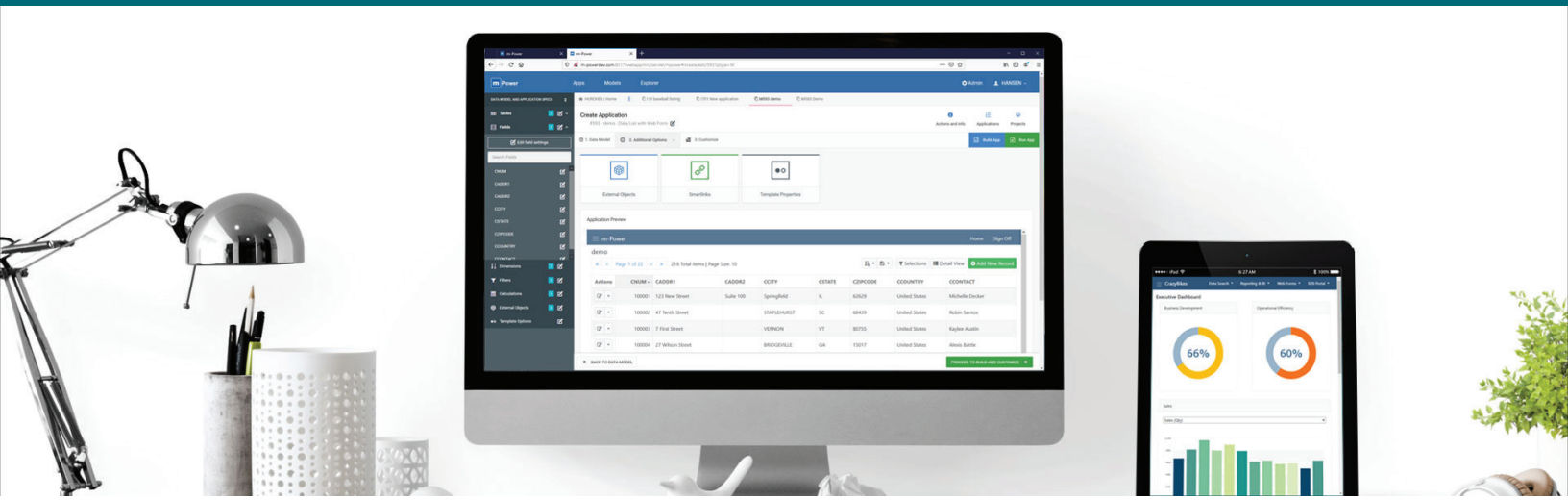
How to cut through the low-code confusion and separate basic tools from the enterprise-class

Table of Contents

About the Author

Joe Stangarone is a 40+ year veteran of the IT industry and president of mrc. Joe shares his leadership and technology insights through the Cup of Joe Blog, which you can find here: https://www.mrc-productivity.com/blog/.

Introduction



I remember back when I first heard the term, “Low-Code.” It was used to describe a growing class of development software tools that simplified enterprise application development.

At the time, I loved the term. Before low-code, all web application development tools were lumped into one big pot. Some were built for enterprise development. Others were very basic developer tools. But, they were all thrown into the “web application development tools” (or even RAD tools) category. How could you find the right development tool when they were all so different?

The “Low-Code” term solved this problem. It separated the enterprise-class development tools from the basic ones. At least…that’s how it started.

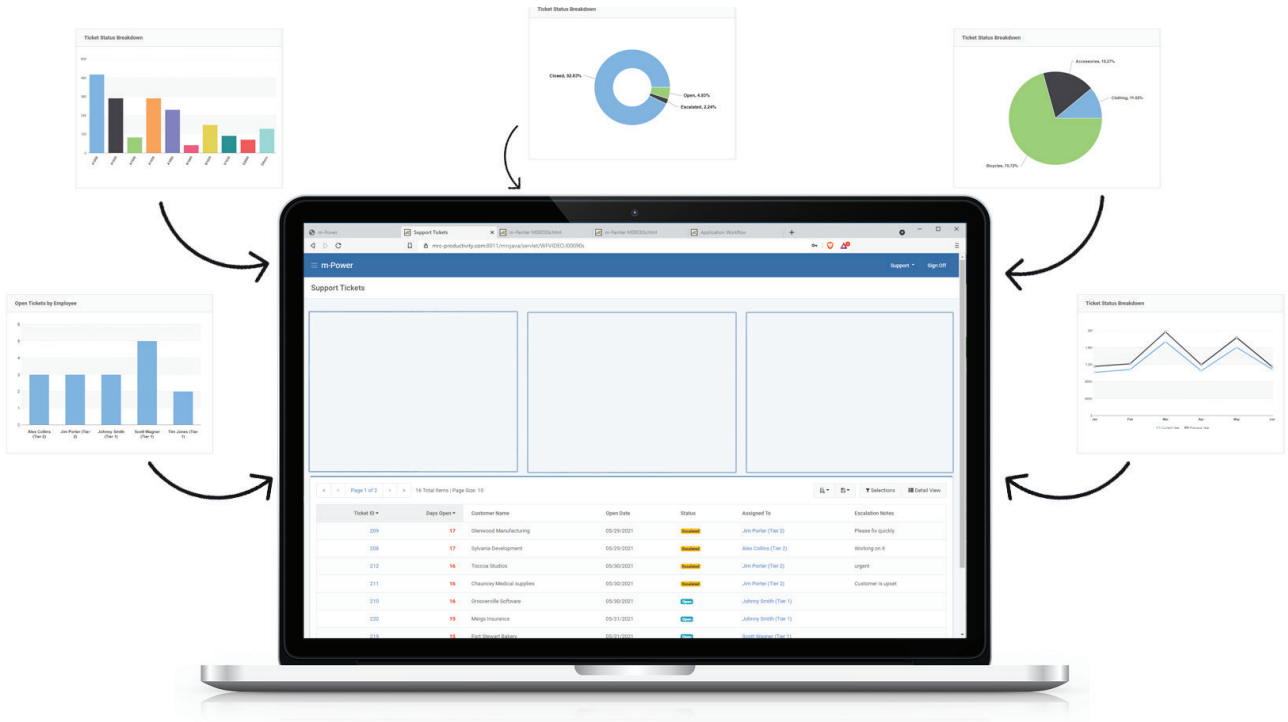
Now, we’re dealing with the same problem. Software vendors have jumped on board and slapped the term on anything remotely related to low-code.

The result: Confusion. If you look at different low-code options available today, you’ll find that they have wildly different features. Comparing low-code software is nearly impossible since they’re all over the map. We’re back to where we started.

This begs the question: How do you separate the basic tools from the enterprise-class tools? What features should you look for in an enterprise low-code platform?

In this paper, I’ll answer these questions. Now, I’m not including obvious features like enterprise-class security, point-and-click interface, or responsive design. These should be standard elements of any low-code tool. Rather, let’s explore some less-obvious features that any enterprise low-code platform should include.

1. Reusability



In the past, business applications were largely monolithic. In other words, every application had all of its necessary functions built in. These monolithic applications were unwieldy and difficult to maintain.

These days, things are different. Modern coding standards call for reusability. They rely on reusable code blocks and web services that you can call from many applications. It's a more logical approach that delivers maintainable applications.

Enterprise low-code platforms should take this 'building block' approach to application development. Enterprise-class tools let you reuse functions and applications (or parts of applications) in other applications. They create basic building blocks and let you combine them however you wish.

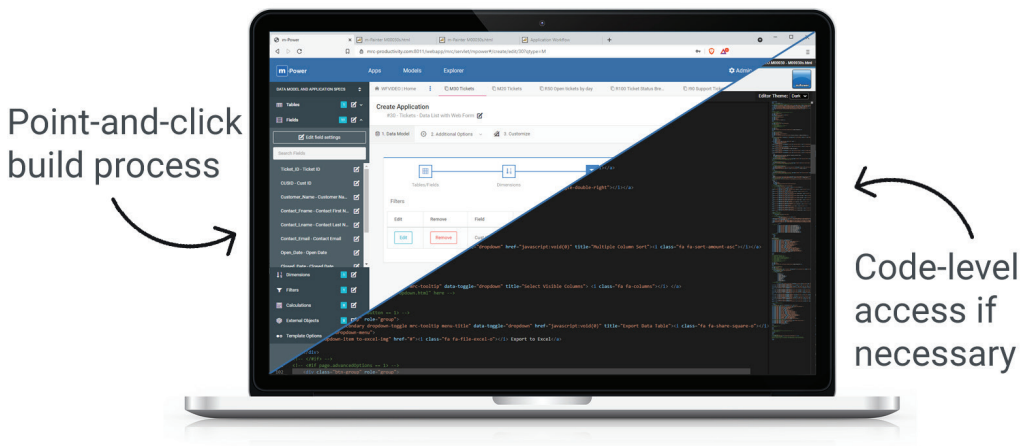
This approach offers a few advantages:

Maintenance: The 'building block' approach is more maintainable. Rather than modifying every application, you can modify a single block which reflects in many applications.

Speed: This creates a development process that becomes faster over time. As you create more and more reusable "blocks", development time improves.

Power: The 'building block' approach lets you create more powerful applications. Since you can combine blocks in different applications, there's no limit to what you can build.

2. Hybrid Options for Developers and End-Users



What's the difference between low-code and no-code?

Low-code development tools are generally aimed at developers. They help developers deliver web applications quickly, but they require some level of coding in a typical build process.

No-code development tools are generally aimed at citizen developers. They require no coding and are easy enough for end-users. Generally speaking, no-code tools create more basic applications.

Each one has its own set of advantages and disadvantages. But, do you really want to purchase two separate development tools for end-users and developers? Probably not.

What should you look for? Look for low-code tools that take more of a hybrid approach. In other words, they can be used by both developers and end-users. I've seen this accomplished in a couple of different ways:

- Two different build processes: Some low-code tools provide a developer interface and an end-user interface. The developer interface offers more options but also requires a more technical skillset. The end-user side is a drag-and-drop interface that requires no coding.

- A single, step-by-step build process: Other low-code tools provide a single, wizard-like build process. Code can be injected into the application if needed, or edited at the code level after the build process. In other words, no coding is needed but the option exists if needed.

The main point: Enterprise low-code tools should also offer no-code features. This delivers the best of both worlds. Developers can deliver applications faster and end-users can create their own apps–all from a single platform.

3. Citizen Developer Controls

Let's take the last point one step further. If you're letting end-users create apps or if generated applications are accessed by all types of employees, the tool must include granular permission controls.

What do I mean by "permission controls"? Well, do you want end users to have full access to all capabilities within the software? Should everyone in your company have access to the same data? Probably not. Any enterprise-class low-code platform should let you manage user permissions on a few levels:

- Application-level: Application-level permissions let you control application access on a per-user role or per-user basis. This typically includes a role-based menuing system, which displays different menu options to different users based on their role.

- Row-level: A critical aspect of B2B, BI, and reporting applications, row-level permissions (or multi-tenant) controls data access within a single application at the row level. Multiple users can access the same application, but only view the data they're authorized to see.

- User privileges: User privilege parameters are used to personalize features and security to individual users or user roles. They can control an application's look and feel, add or hide user options, limit user capabilities, and more.

- Developer permissions: This lets you control what data and features individual developers can access within the development environment. For instance, you probably wouldn't want end users having full access to every software feature or database table.

"The most important features of low-code are for it to be easy-to-learn and easy-to-use, and that it should provide guardrails for citizen developers, so code is not implemented that goes against policy, security or compliance regulations," explains Gaurav Dhillon, CEO of SnapLogic.

4. Workflow Automation

A recent study found that inefficient processes are the biggest source of wasted time in the average workday.

Why do so many businesses struggle in this area? It usually boils down to the fact that their applications and processes weren't built for a digital era. Their applications don't integrate with each other. They don't use data changes to trigger workflows.

The result: Employees must fill in the gaps manually.

That's why automation is such a critical part of modern low-code tools. They must be able to pass data between other applications and services. They must be able to trigger workflows when data is added or changed.

Most importantly, they must include a workflow automation designer. Workflow designers let users automate business processes using a drag and drop interface. They let users set workflow triggers that fire when data is added or changed. These triggers can start all types of workflows, ranging from email notifications to API calls and everything in between.

5. BI/Reporting/Dashboarding

In my experience, most low-code tools don't offer business intelligence capabilities. They provide ways to connect to your BI software but don't offer analytics as a feature.

But, I think that low-code platforms that also provide business intelligence, reporting, and dashboards offer a huge advantage. Let me explain:

Generally speaking, standalone business intelligence software provides read-only data access. In other words, they display data but can't write back to the database.

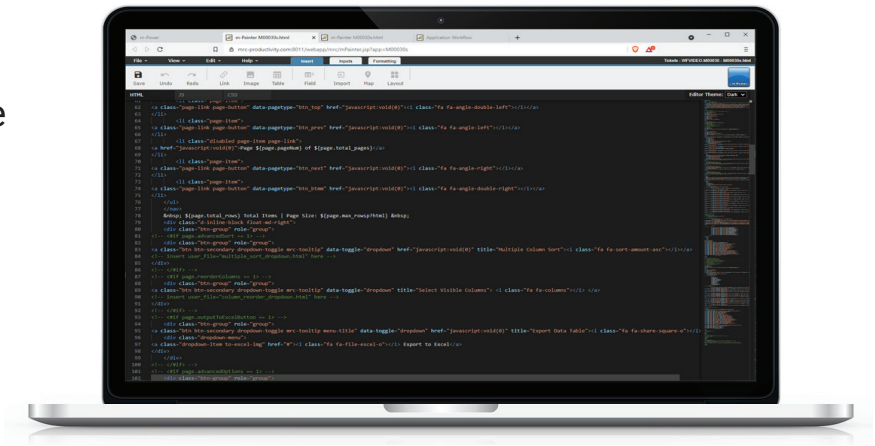
Low-code tools typically provide the opposite. They create applications that can read and write to the database, but usually don't offer analytics.

Some low-code platforms provide the best of both worlds. They offer read/write capabilities and also create reports, BI apps, and dashboards. This is huge.

Why is that so important?

Data writeback adds another level to your analytics. Rather than simply displaying data, your BI apps and reports can take action. This gives users the ability to send alerts, perform actions based on data, and trigger workflows. In other words, it gives you proactive analytics that can automatically take actions based on data changes.

6. Customization



Customization (or a lack thereof) is a big concern with low-code development tools in general. Many (wrongly) assume that you can't customize the generated business applications. In fact, customization is often listed as a drawback to low-code.

The idea that low-code tools offer limited customization is a myth. It stems from the assumption that low-code platforms are the same as 4GL and RAD tools of the past. Those tools were limited, but things have changed. These days, most enterprise-class won't have those restrictions.

That being said, I can't speak for every tool with regards to customization (as I haven't used them all). Some are more limited than others. To be sure you choose a tool that doesn't limit your options, ask the vendor questions like:

- How do you customize the generated applications? Does the tool include a graphical editor? Does it let you modify the HTML/CSS if necessary?

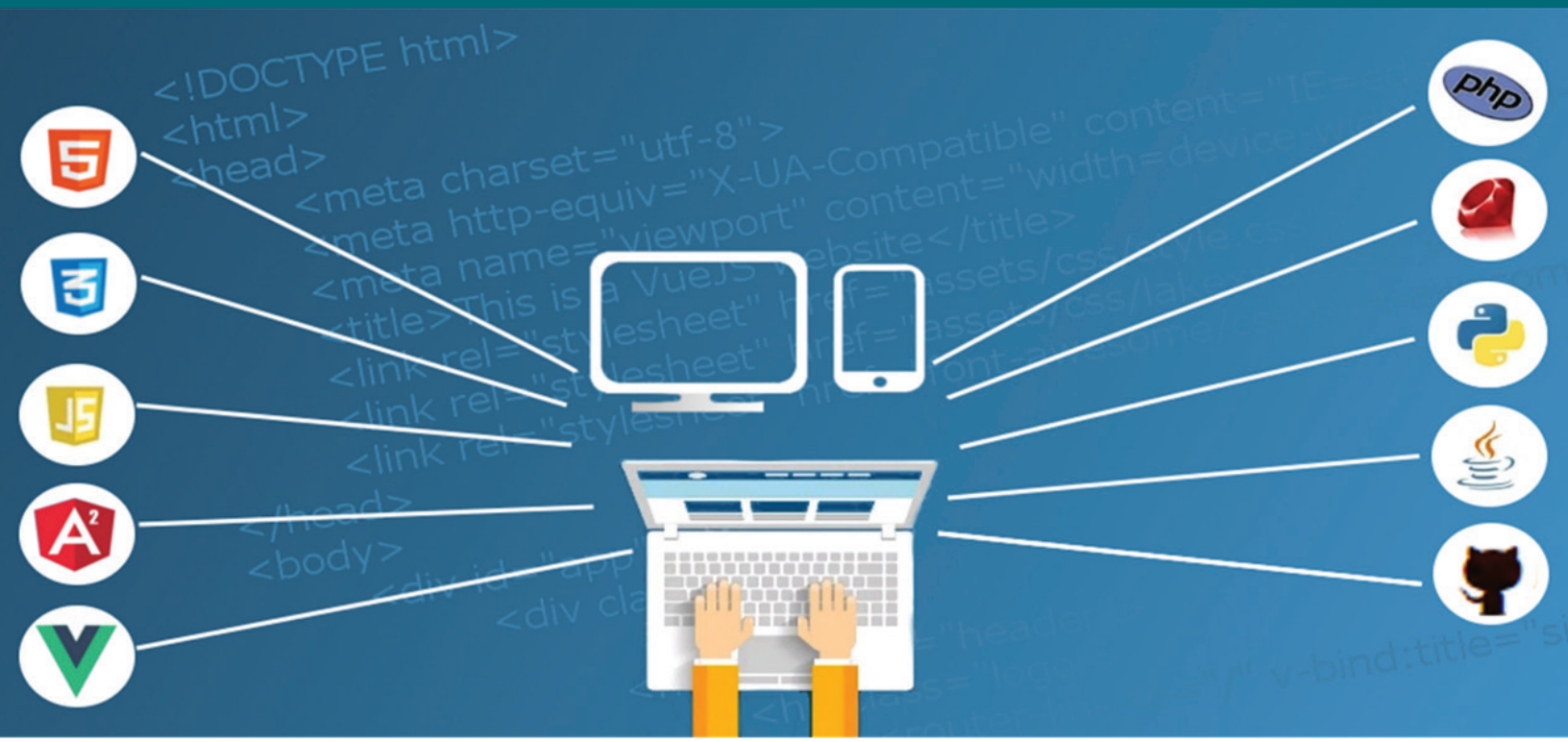
- Do you allow white-labeling? Can you customize the output to match your company's look and feel?

- Can we add custom business logic when we build applications? Is there a way to add external code to the applications? Can we create rules/logic so the applications fit our business processes?

- Can you create/modify application templates? This is important because it lets you customize the look/feel of every generated application without manually changing each one.

"I believe that software is truly low-code when it allows for a reasonable level of customization," says Rebeca Sena, Architectural Marketing Consultant at GetSpace.digital. "For instance, many drag-and-drop programs such as analytics dashboarding solutions or website builders don't allow to personalize many basic functionalities. In fact, that makes their low-code feature become a teaser to lock customers in with their technology."

7. Standard Languages



The business case for low-code software usually boils down to time and money savings. They let you deliver applications in less time, using fewer resources. They reduce the need for coding knowledge in the development process.

However, these benefits only apply if the platform has a low learning curve and doesn't require additional technical skills or proprietary programming languages. The problem is, some tools still require that you learn their unique language. Avoid those like the plague.

Here are a few questions you should ask before you buy: What skills are needed to use the platform? How long will it take to get up and running? What technical knowledge will our developers (or citizen developers) need? These are all important questions to consider when calculating the ROI of the software.

"If you have to learn the tool's own unique language to get things done, you're halfway to making the low-code platform meaningless," says Nate Tsang, Founder & CEO at WallStreetZen. "I'd rather just use Javascript or Python or a language I already know rather than a new one, even if it is intuitive or simple."

8. Open Architecture

One of the most important “features” of low-code software is also one you can’t see: The architecture. While application architecture might be invisible to the users, its impact is enormous.

Why should you care so much about architecture? It plays a major role in your overall experience with low-code platforms. It impacts portability, security, flexibility, and more.

Most importantly, architecture plays a massive role in a low-code tool’s integration and extendability. After all, what if your low-code software can’t do everything you need? How easily can you extend that software to meet your needs? You’ll need to answer two questions:

- **How easily can we integrate this with our existing systems?**
- **How easily can we integrate this with third-party services?**

Before you license any low-code platform, understand its integration capabilities. Does it support RESTful web services? Is it built using open technologies? What kind of code does it generate? These questions will save you from many integration headaches down the road.

“The most essential feature that a good low-code development tool should have is Extendability,” says Daniel Velez Vasquez, CEO at Home Security. “Low-code tools should ideally be able to accommodate all possible use cases. However, this isn’t always possible. If the functions, data formats, schemas, or endpoints are too complicated or niche, this may be the case. For developers, the most important thing is transparency and extensibility at all levels, with open model specifications and tools for platform extensibility. You must be able to easily enter the code level in order to create additional extensions for the low-code platform.”

9. Platform Control

I don't know if this technically qualifies as a low-code "feature", but platform flexibility is critically important. After all, what happens if you decide to move away from the low-code software? How much vendor lock-in will you deal with?

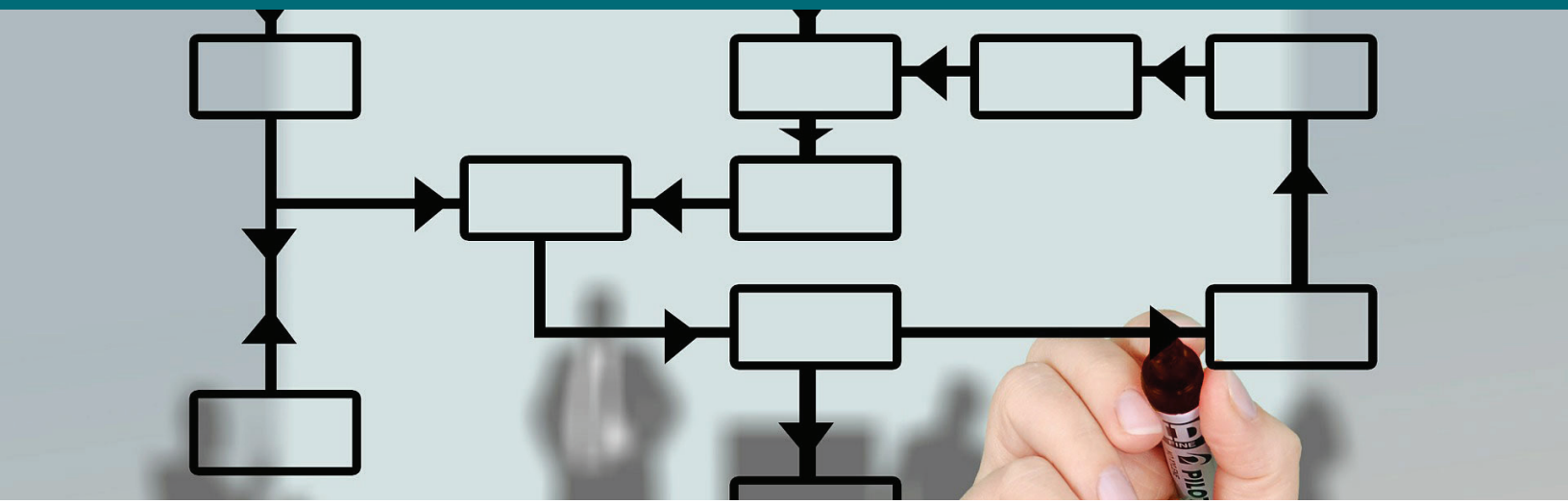
To answer that question, you must look at the software from a few different angles:

1. **Who controls the data?** Is it stored in-house or on their servers? If it's stored on their servers, ensure there's a simple way to export and download that data if necessary. I've known far too many companies who tried to switch software platforms only to realize that they didn't even own their own data.

2. **Does the platform create standalone applications, or do they require an active subscription to run?** Obviously, you want applications that run independently of the development tool. That way, they'll still work if you move away from it.

3. **How easily can you switch platforms?** For instance, what happens if you decide to move your applications from the cloud to in-house? While this may not be a concern now, this flexibility could be very important later on. You don't want to lock yourself into a single platform.

10. Standardized Processes



One often-overlooked advantage to low-code: Many tools are designed to standardize the development process.

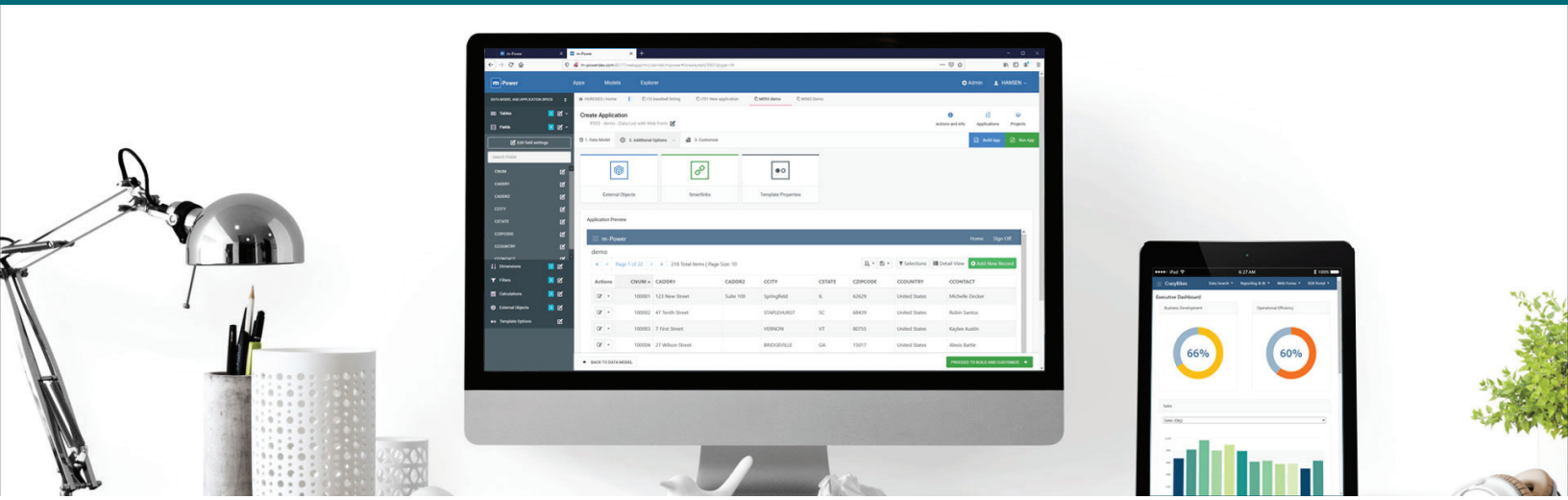
Why is that so important? Think about it for a minute. You'll always get some level of variance in your code if you have multiple developers on staff. Each developer has different habits and coding styles. Some will follow strict standards while others won't. If you have a team of developers, this means that different applications might be developed with different standards.

Anyone who has ever had to sort through another developer's code knows how important this is. Just imagine how nice it would be to jump right in and pick up where they left off because everyone uses the same process.

Now, the level of standardization will vary by tool. It's important that you know how the build process works and how easily developers can document their changes. This will save you many headaches down the road.

*"Automation simplifies things, but you can also tie many automations and systems in knots," says Rory O'Brien, Head of Solutions at Tonkean. "Proper documentation *about* the no-code/low-code creations and automations you've made is as important as engineers who document their code. If I get hit by a bus, who on my team or in the organization is going to have to come and troubleshoot or untangle all my crazy stuff I've created?"*

Summary



The “low-code development” term was first coined back in 2014. Since that time, we’ve seen an explosion of low-code tools. These days, you’ll find well over 200 options.

The problem: They’re all different. Since “low-code” is such a vague term, low-code software is all over the map.

The result: Low-code software buyers must wade through a jumbled mess of options. Different low-code solutions offer different approaches and capabilities. They find that comparing options is like comparing apples and oranges.

In this paper, I’ve tried to help address the issue with these 10 crucial low-code features. These are features that might not be obvious on first glance, but play a critical role in your success with low-code tools. I truly hope you find the information helpful.

If you have any questions for us, please reach out at 630-916-0662 or via email at mrc@mrc-productivity.com. You can visit us on the web at https://www.mrc-productivity.com.

About mrc

michaels, ross & cole, ltd (mrc) is a global software company which specializes in web application development software. Headquartered in Oak Brook, IL, and established in 1981, mrc has offices in the U.S. and the UK. mrc offers award-winning development software, as well as consulting, mentoring, and training services.

mrc’s Oak Brook headquarters handles all product development and maintenance, as well as customer support and training for North America and South America. mrc’s UK office provides customer support and training for Europe, Africa, and Asia.