Five tips for successful innovation with low-code

Whitepaper by Blis Digital

@blisdigital

Table of contents

Why we wrote this for you...

Low-code is hot. And the Microsoft Power Platform is perhaps even hotter. With the Power Platform, Microsoft delivers a very interesting solution for organizations that already work with Microsoft 365 applications and want to innovate digitally.

Although the Power Platform enables you to build solutions quickly and easily, the solutions and their management are anything but simple, and there are several important considerations you need to think through carefully if you want to truly deploy the platform successfully. As big Power Platform fans and experienced software developers, we like to combine the accessibility and pragmatism of low-code with the principles and best practices of professional software development.

In this whitepaper, we have described the five most important principles and want to provide you with a valuable tool so that you are well-prepared when you start working with the Power Platform. Happy reading!

What is low-code technology?

The Power Platform is Microsoft's low-code platform. Before we zoom in specifically on the Power Platform, let's first say a few words about low-code in general. What is it and why would you use it?

Focus on the solution

No-code is building applications in a graphical interface, without any code involved. With low-code, we refer to a way of building software that is also largely graphical, but additionally provides the space to shape complex logic through easy-to-understand scripts and formulas. This makes low-code development a relatively simple way to build apps and solutions for business challenges. It ensures you can develop quickly, offers a way to use building blocks (comparable to Lego), and is based on a data model. Low-code development is a form of 'declarative programming'.

The counterpart of low-code is called full-code'. Full-code is an imperative form, where turn-by-turn instructions are given to the computer using code.

Citizen developers

Low-code platforms are especially a solution for so-called citizen developers. Citizen developers are users from the business with IT affinity. They are 'officially' not software developers, but thanks to low-code platforms, they can still build solutions themselves without having to involve software developers. Low-code uses simple expressions for creating configurations and formulas, comparable to good old Excel and the well-known Microsoft Access.

Blis Digital

Why do organizations choose low-code?

So it's relatively easy to build apps and applications using low-code. But that's certainly not the only reason why low-code technology is so popular. Organizations typically choose to embrace low-code technology for the following two reasons.

Innovation needs

The need for flexible IT platforms and applications has increased enormously in recent years. With the traditional way of software development, focused on control, stability, and security, you can't always move along with the business's need to innovate quickly. That's exactly where low-code comes in. Whether it's automating business processes, launching digital innovations, or modernizing and SaaS-ifying existing software: low-code is an interesting alternative or complement to existing (legacy) business applications or ERP systems.

Labor market shortage

Low-code also contributes to a solution for the enormous shortage in the labor market in general and that of software developers in particular. Training a citizen developer takes less time than a full-stack software developer. Additionally, you can have these two roles work together nicely; the citizen developer does the simpler 'standard' things, while the full-stack developer can focus on the technically more complex matters. Low-code also offers a solution for labor productivity. By automating work more with 'smart' solutions, an employee can do more in less time.

The benefits of low-code

1. Less code means fewer bugs and shorter development time.
2. Hardly any professional development skills required.
3. Your apps run on the low-code platform, a controlled environment where functionalities are available to arrange things like governance, security, and monitoring.
4. It's easy to integrate with legacy systems and existing data sources.
5. By default, you get a great user interface and your apps can be used directly on various devices, such as PC, tablet, and phone.

Power Platform, the low-code platform within Microsoft 365

Different low-code platforms

Betty Blocks, Appian, Mendix, OutSystems, and Microsoft all have their own capabilities and limitations. We are big fans of the Power Platform, Microsoft's low-code platform, which is available as a cloud-only offering and runs on Microsoft Azure.

Integration with Microsoft 365

The Power Platform is an interesting addition to the existing Microsoft 365 applications that many organizations use, such as Word, Excel, SharePoint, and Teams. This platform makes it possible to easily build apps, automate business processes, and digitize many common manual tasks without too much technical knowledge. As an organization, you completely determine the pace and scale of your growth path. With the Power Platform, you can just as easily create a simple approval flow for expense reports as an advanced virtual assistant for the service desk.

From simple dashboard to mission-critical app

Whether it's a simple dashboard for financial insights or a business-critical application, the Power Platform can handle it all. It's even possible to decide somewhere along your growth path to convert an application from Power Platform low-code to a full-stack app that is further developed by professional developers.

The four components of the Power Platform

The Power Platform is a standard component of most Microsoft 365 licenses and integrates seamlessly with applications like Teams, Dynamics 365, SharePoint, and Excel. The Power Platform consists of four separate applications:

1. **Power BI** for creating reports, interactive dashboards, and data analyses
2. **Power Apps** for creating apps and forms
3. **Power Automate** for automating processes
4. **Power Virtual Agents** for creating chatbots

In addition to these applications, Power Platform uses various Azure services in the background for data connectors, authentication, logging, and AI functions.

Each of the Power Platform applications has a different approach, but all are based on the low-code principle: you can start easily and without much technical knowledge.

You can connect the applications to each other, but also to data from Microsoft 365, Dynamics 365, and Azure or non-Microsoft sources, by using data connectors.

Tip 1: more low-code than you think!

Forrester introduced the term low-code in 2014 to indicate the new thinking about app development. But anyone who thinks low-code is something completely new is mistaken.

This is also low-code

Since 1993, it has been possible to program macros in Excel using Visual Basic for Applications (VBA). And what about all those smart solutions found in Access databases, SharePoint lists and workflows, and InfoPath forms. As far as we're concerned, these are all forms of low-code too.

More mission-critical than you think

This form of low-code has been a delight for many business users for years, but for many companies today it's a huge pain point. Many functionalities have been created this way that the business now relies on, but which are not managed by IT professionals, where no one knows who created them, how they were made, and where they are documented.

Lifecycle management of these functionalities was never considered, but when something stops working, it suddenly becomes the IT department's problem. In short, you'll have to do something about this.

It starts with insight

Making visible what already exists in your environment is an important first step. Now that you know which low-code solutions exist, you can determine per solution what is needed to regain 'control'. And since many of the previously mentioned low-code forms are now outdated, it's probably a good idea to rebuild these solutions using the Power Platform.

Blis Digital

Tip 2: You must also set up ALM for low-code

Just like any other IT solution, low-code solutions must also be managed. After all, you want to maintain grip and control over the solutions in your organization. Moreover, you want to ensure quality and availability of the solutions and keep business-critical data secure. By including the Power Platform as a full-fledged component in your IT strategy, you prevent unwanted surprises. Application Lifecycle Management (ALM) is (part of) the answer to this quest.

What is ALM?

Application Lifecycle Management (ALM) is about the combination of people, policy, and technology needed to control a software solution throughout its entire lifecycle, from concept to retirement. There are various models for setting up ALM, but they all have one thing in common: they are designed to capture the complex steps of software development in a clear and repeatable process. This doesn't just consider the technology, but also how you as an organization handle the management and use of the application. ALM principles are also essential for solutions made with the Power Platform. After all, you want to deploy the Power Platform sustainably.

ALM within the Power Platform

If you want to deploy Power Platform solutions sustainably, it's important to apply ALM principles. ALM has several different phases, each with their own characteristics. By following these phases, you're able to make the right choices and always know where your solution stands in terms of lifespan. Let's take a closer look at each phase.

Plan

Before you start creating a solution with the Power Platform, it's important to think carefully about the licenses you need. Within the Power Platform, various license(s) (models) are available, and costs can vary significantly.

To prevent unwanted surprises, start by mapping the required functionalities and mapping them to the different license options. The Power Platform Licensing Overview is a handy tool to determine which components from the Power Platform are most suitable, possibly supplemented with services from Azure. As part of this, you must also think about the use of the solution. How much data will be stored? How many APIs will you call? All things that affect costs.

Develop

Once you start building, you naturally consider naming conventions and make choices about how you'll arrange access to the application later. The development itself happens in an iterative process that you go through together with end users, and is connected to testing and rolling out your solution. Also think about deploying different Power Platform environments to support a deployment strategy with separate development, test, acceptance, and production environments (DTAP).

But using different environments can also be a solution to meet specific data security requirements.

Test

We'll return to testing low-code applications later, but that they need to be tested is certain. It's important to go through all three test phases. Requirements are often established in the Plan phase. Already in that phase, it's important to involve a software tester to identify any ambiguities or contradictions in those requirements early. Then it's the software tester's task to test integrations and functionality of the process or solution. The software tester also helps with user testing. If these steps are successfully completed, you can use the solution with confidence.

Release

Rolling out low-code solutions often happens by giving people direct access to an app. This way, it's no longer possible to easily make changes – and test them – at a later stage without bothering users. Every change is immediately 'available' to your users. That's why it's always wise to roll out at least separate test and production versions of a solution. The Power Platform offers the possibility to create logical separations with different environments. Using Power Platform Solutions, you can easily bundle your solution and roll it out to another environment. And with Power Platform Build Tools for Azure DevOps, you can even ensure that things are automatically rolled out through a so-called ‘deployment pipeline’.

Operate

Once a solution is operational, you naturally want to know how the use and status of that solution are doing. The Operate phase is mainly about insight. What’s happening and what is the solution being used for? Does it do what it should do? Are there issues, or is usage growing exponentially and do you need to anticipate this?

The Power Platform Center of Excellence Starter Kit gives you insight and information about many of these components. The Power BI dashboard makes all Power Platform solutions in your tenant visible at a glance. Including those solutions built without your knowledge.

Learn

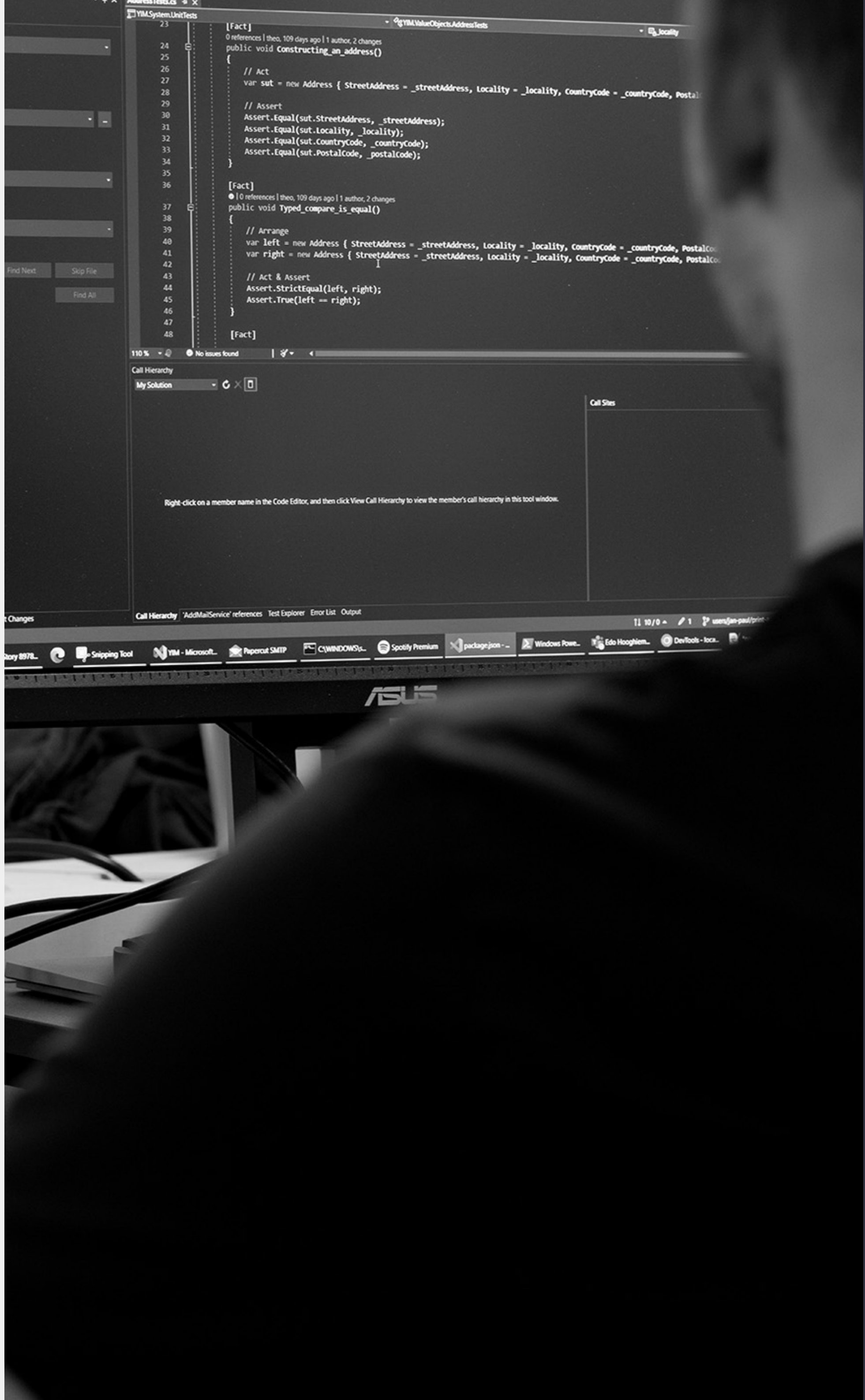
The insights and information you get from the Operate phase, combined with usage reports from the various solutions in your environment, help you maintain overview. This way you move from reactive management (break/fix) to proactive management (in control) to anticipate problems and prevent disruptions. In consultation with users, you continuously develop solutions further and are able to respond to all new questions from the business.

From unmanaged to managed

Many solutions in the Power Platform start as a workaround for a certain process or manual tasks. Often without further consequences or choices around a more formal process. However, if you want to deploy the Power Platform without the environment becoming polluted, or losing sensitive data through careless use, you need to think about managing those solutions. Leaving everything open is not an option, as that encourages chaos. But closing everything is also not convenient, because then users will go ‘shopping’ elsewhere.

Summary

Step one is gaining insight: what's already happening? Step two is consciously making choices about whether or not to manage solutions, and if they are managed, to use a more formal process. Without conscious choices and considerations, you'll run into problems sooner or later. Make sure you stay ahead of them!



```
[Fact]
0 references | theo, 109 days ago | 1 author, 2 changes
public void Constructing_an_address()
{
    // Act
    var sut = new Address { StreetAddress = _streetAddress, Locality = _locality, CountryCode = _countryCode, Postal

    // Assert
    Assert.Equal(sut.StreetAddress, _streetAddress);
    Assert.Equal(sut.Locality, _locality);
    Assert.Equal(sut.CountryCode, _countryCode);
    Assert.Equal(sut.PostalCode, _postalCode);
}

[Fact]
| 0 references | theo, 109 days ago | 1 author, 2 changes
public void Typed_compare_is_equal()
{
    // Arrange
    var left = new Address { StreetAddress = _streetAddress, Locality = _locality, CountryCode = _countryCode, PostalCo
    var right = new Address { StreetAddress = _streetAddress, Locality = _locality, CountryCode = _countryCode, PostalCo

    // Act & Assert
    Assert.StrictEqual(left, right);
    Assert.True(left == right);
}

[Fact]
```

YIM.System.UnitTests
YIM.ValueObjects.AddressTests
110 %
No issues found
Call Hierarchy
My Solution
Call Sites
Right-click on a member name in the Code Editor, and then click View Call Hierarchy to view the member's call hierarchy in this tool window.
Call Hierarchy
Test Explorer
Error List
Output
Snipping Tool
Spotify Premium
ASUS

Tip 3: Bridge the gap with Fusion Development

Technical details are shielded as much as possible in low-code platforms. This ensures that so-called citizen developers don't have to worry about such complexity and results in apps being ready faster and able to be used. Unfortunately, you can't build everything with low-code. Sometimes you're missing something. For example, you need a connection with an external system to retrieve data. That's where fusion development comes around the corner.

Fusion what?

When Microsoft talks about fusion development, they refer to 'bridging the gap' between user and software builder. Fusion development brings together end users, citizen developers, full-stack developers, and IT administrators. This creates a multidisciplinary team with just one goal: quickly and accessibly creating solutions for business challenges. Within this team, everyone benefits from the advantages of low-code.

Fusion development in practice

What does such collaboration within a fusion development team look like? Citizen developers get to work with the Power Platform. If the standard functionalities, components, or (data) connections are not sufficient, they bring in a full-stack developer. The IT Pro ensures that apps and data integrations are rolled out properly, and that everything can be used safely and controlled. This way you expand the functionality of the Power Platform together, exchange knowledge efficiently, everyone does what they're good at, and all your development and IT capacity is optimally deployed.

Unleash
the power of
technology

Scale up when needed

And is it really threatening to become a success? Then you can easily scale up in Azure and plug in other services to ensure your app remains scalable and future-proof.

Extending Power Platform itself

Fusion development helps you achieve more with the Power Platform by allowing you to add connectors and functionality. But it would be a shame to use them in just one project, so you can roll out connectors within your entire organization. The data you've unlocked this way is then available to all other Power Platform developers. Many developers also share their extensions externally in Microsoft's Independent Publisher Connector Program. This allows other organizations to use your connector too.

Management without code

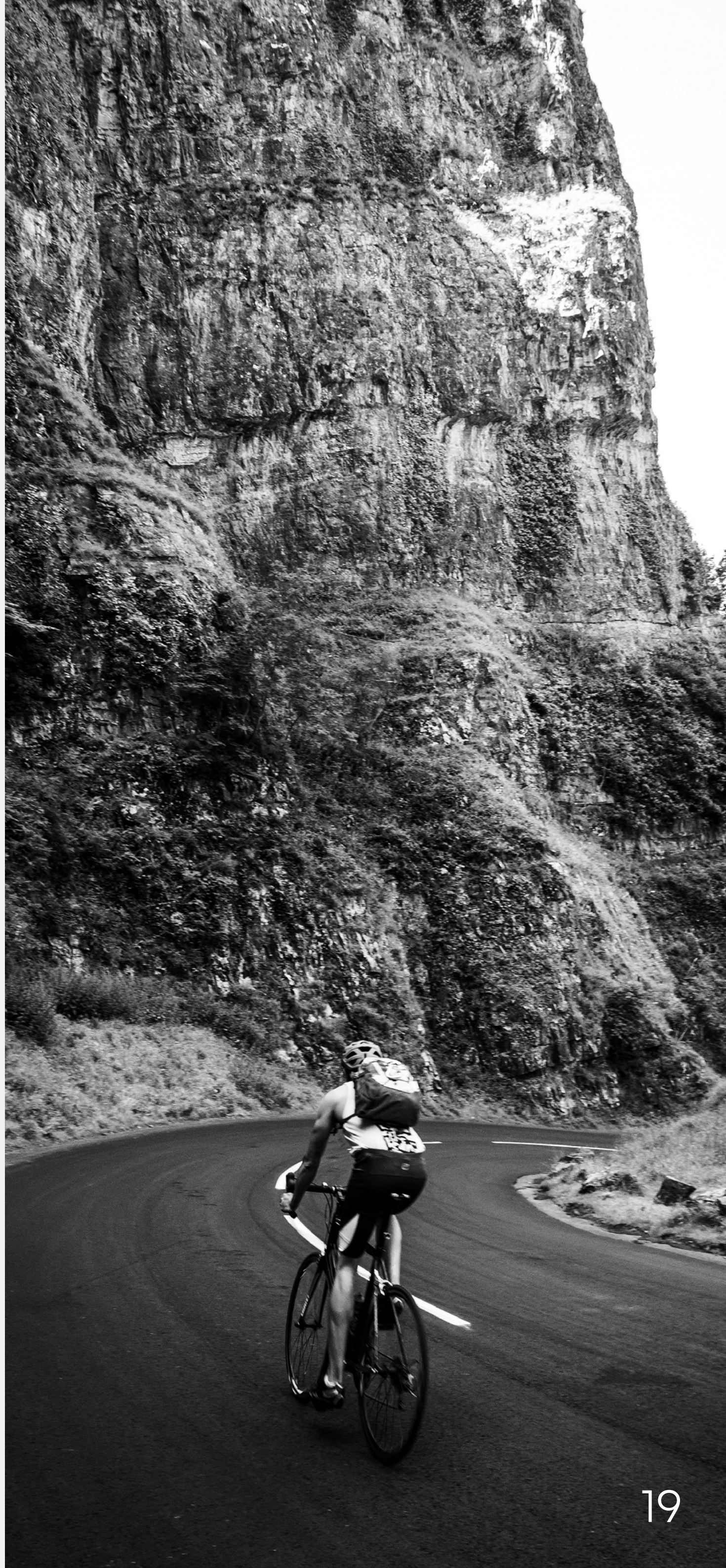
However, fusion development doesn't stop at developing apps with the Power Platform. You can also increasingly leave application management to citizen developers. By packaging an app in a 'solution', they can manage it themselves, share it with other environments or departments, or reuse it as part of a new solution. All without writing code. Azure DevOps provides these solutions with version history and gives developers the ability to set up development, test, acceptance, and production environments. Such a streamlined process helps an organization deal professionally with the Power Platform and ensures that Fusion development acts as an accelerator in conceiving, testing, building, and rolling out digital innovations.

Tip 4: Low-code solutions must also be tested

The Microsoft Power Platform makes software development accessible to many more people. From handy email hack to enterprise-level business process: it can often be realized without writing code. And if you don't write code, you don't need to test either, right?

You click an app together and voila, it can go to production. That's a misunderstanding. Whether it's low-code or traditional coding, putting something into production without testing is not a safe and manageable way of software development.

Do you need a team of twenty testers for your low-code apps? Probably not. Yet you must seriously test before a Power Platform application is put into use. After all, you want to deliver quality. But why do you need to test software that's already half finished before you start?



Noise on the line

The answer lies in communication. When you want to create a solution, it starts with an idea in your head. We all know how difficult it is to translate an image in your head to paper. This is also how it works with building low-code solutions. Say: a team of nine people thinks a certain process could be smarter. Only one of them actually gets to work on the solution. Somewhere in that process there's a translation step, and that's where noise occurs. Simply because people sometimes misunderstand each other. That's why it's always good to agree that when the app is finished, you'll take another fresh look at it. Even with low-code.

The balance in software testing

Software testers are often accused of only seeing problems, but that's not really how it works. A software tester is mainly someone who asks with every piece of business logic: What does this actually mean? Fully thinking through the functionality of your app and working out what's needed to make all your logic work under all circumstances is a specialty. So yes, a tester finds things that could go wrong. That's the negative side. On the positive side: a tester ensures quality. You make things. A tester helps you make them really good. This creates confidence in the product and ensures problem-free rollout and productive use. When done right, these two things are in balance.

Three test phases

Testing low-code applications also takes place on three levels:

1. Requirements testing

Before you start, you need to have a plan. Good requirements are very important even with low-code. As we already wrote in the chapter on Application Lifecycle Management, it's smart to involve a tester in this phase. They're good at recognizing gaps in requirements that could cause problems later.

2. Integration and functional testing

The most detailed, technical test level (the 'unit test') can usually be skipped with low-code. But testing the functionality of the app and integrations with other systems is naturally important. Do the modules talk to each other? Does the data arrive? Making assumptions in this area is dangerous and can cause serious problems and delays in the next test round, or in production. In a functional test, a tester behaves as a user and performs all actions as worked out in the requirements. This way you know if the app really does what the user asked for.

3. User testing

In the last test round, you also let the user judge whether the app does what they asked for. Never skip this phase! A user looks at software with a completely different perspective than the techies. So they'll certainly still have comments.

The more important the app, the more test rounds needed

A common misconception is that user testing is enough for low-code. But users test software from their own perspective and usually not very systematically. This means you don't always surface dangerous assumptions from the development process. A user consciously or unconsciously assumes that a developer has thought of everything. But 'everything' is quite a lot, even for an experienced developer, and that assumption doesn't have to be correct.

Low-code feeds such assumptions because the development process is less technical and it's therefore easy to assume that 'everything goes automatically'. How many test rounds you need naturally depends on the complexity of the app and the number of users. Have you made an app for yourself? Then you can indeed click critically through your app once and conclude that it does pretty much what you want. But if more users are going to work with the app, then more is needed. Even an app used by just five people can reveal unexpected issues you never anticipated.

The more important and extensive the app and the more users, the greater the chance of unintended effects. The stricter your quality requirements should therefore be. Additionally, you also need other things that you undoubtedly recognize from full-stack development processes: documentation, manuals, and governance agreements about naming, rights, access, and data cleanup. You must also establish deployment and update processes if you want to broadly roll out a low-code app.

Specific tests for low-code

The beauty of Power Platform is that many things are already ready. Microsoft ensures your app always starts and the infrastructure is in order. You don't need to worry about that. Standard integrations and connections you've used in your app will also just work. However, users themselves are responsible for data accessibility. That can go wrong. What if you use a Teams team in your app and that Team gets deleted? What if certain folders aren't accessible?

These are exceptions you need to test. Data processing and presentation must also be tested. With low-code, you usually don't have full visibility into what functionality is supported. Platforms change every day and you can't know everything. For

example, you might think of having a rain emoji in your app's title during bad weather, but then test whether that works on all devices and whether unexpected effects occur. It would be a shame if emails don't arrive because of that. Also keep in mind that app usage can grow. Three users storing all their files in one folder might still work fine. With 3,000 users, that's really a different story.

Summary

Testing is part of the development process. Even with low-code. The more important and extensive the app and the more users, the more important the test process is. Fortunately, much is already arranged within the Power Platform, but even with low-code applications – just like with full-stack software development – the quality of your solution is the result of a thoughtful test process.

Tip 5: You must stimulate low-code adoption

Using low-code platforms, people from the business with IT affinity can easily build apps themselves without much technical knowledge. But however accessible a low-code platform like Microsoft Power Platform may be, to ensure it's truly embraced, you must also pay attention to guiding, supporting, and training citizen developers.

The Power Platform Adoption Maturity Model is a useful tool for this. It gives you a phased approach to start small and gradually grow toward organization-wide rollout of the Power Platform. Also see this article on Microsoft.com "Repeatable patterns for successful Power Platform adoption".

Power Platform as a growth model

Before you start promoting the Power Platform, you need to make visible which low-code applications are already being used in your organization. Then determine per application what the (technical) quality is and how future-proof the solution is. You'll probably conclude that rebuilding the application using the Power Platform is wise.

Based on the Power Platform Adoption Maturity Model, you can properly determine what you need to arrange as an organization to make Power Platform usage successful.

Don't try to jump straight to Level 500, that's not necessary. By growing in a manageable way, you prevent unpleasant growing pains.

Want to read more? These resources are worth a visit:

Gartner Insights: Low-Code Application Platforms 2022

Forrester Wave ranks top platforms for developers

Transform your business apps with fusion development

Ebook: Fusion development approach to building apps

Microsoft Power Platform Center of Excellence Starter Kit

Power Platform Build Tools for Azure DevOps

Microsoft Power Platform CLI

Power Platform PowerShell Modules

Repeatable patterns for successful adoption

We are Blis Digital!

Microsoft's Power Platform allows you to easily build solutions. It presents plenty of opportunities. However, there are several things you should consider before you start using the Power Platform. Realize that it's not something you simply 'add on'. It's not just a business tool for making fun applications. If you use the Power Platform, it must be a strategic part of your IT landscape. Simply 'activating' a low-code platform is bound to cause misery, loss of control, and potentially even severe risks in terms of security, data leaks, and disruptions.

To successfully use the Power Platform, you need to think about several essential issues in advance, including governance, security, documentation, testing, support, and ALM. In short, think before you act. We hope that this white paper has helped you do just that.

Blis Digital

@blisdigital