How does low-code development correspond with best practice in software development?

Benny Petersson

William Evans

Computer Science
Bachelor degree
15 credits
VT 2023
Supervisor: Jesper Larsson

Abstract

Low-code development is an alternative to regular software development where the developer is able to build software and websites using a drag and drop interface instead of writing code. Low-code development has gained popularity in recent years, one of the reasons for this is the broader targeted audience which is citizen developers, people with no previous knowledge and experience in programming and web development. However it is unclear if low-code development can uphold the standards set by software development best practices. The purpose of this study is to investigate how low-code development corresponds to best practices in software development. Maintainability, flexibility, portability, reusability, readability, and testability are the six best practices that are specifically investigated in this study. Design and creation was used as the methodology to investigate whether these best practices can be used in low-code development. A system and a website was built in partnership with CGI, using Microsoft Power Platform as the low-code development tool.

The findings of the study indicate that low-code development has a number of limitations when compared to best practices in software development. The limited ability to customize code makes it difficult to accomplish maintainability, flexibility, and testability in low-code development. Additionally, it might be challenging to accomplish portability because low-code development platforms are not always interoperable, and reusability is limited due to the differences in implementation between platforms. Furthermore, it is often difficult to follow the logic flow of a low-code system which makes readability a challenge in low-code development.

While some best practices can be adapted to low-code development, the study draws the conclusion that low-code development still has a long way to go before it can fully live up to best practices in software development. In summary, this study highlights the limitations of low-code development in comparison to best practices for software development and proposes that additional research is required to overcome these limitations.

Key words

Low-code development, best practices for software development, Power Platform.

Table of content

Acknowledgments

We would like to express our sincere gratitude to CGI for their invaluable support in setting up the project for our Bachelor's degree. We extend our heartfelt thanks to Kevin, Patrik, and Tony from CGI for their guidance and expertise throughout the project. We would also like to thank our supervisor, Jesper Larsson for guiding us through the research project, for his mentorship and contributions. We are also grateful to our families, friends, and all who supported us. Without the collective efforts of CGI, Kevin, Patrik, Tony, and Jesper, this project would not have been possible.

Introduction

Low-code development is becoming an increasingly popular trend for companies around the world. Low-code development is a way to create products, build systems and websites without previous knowledge in software development. Low-code is advertised to everyone, since anybody is supposed to be able to learn low-code. The number of low-code development platforms is growing as they have been highly requested by citizen developers [1]. Citizen developers are described as employees who are primarily based in business departments and show IT affinity as well as technical expertise [2]. In other words, citizen developers are domain experts with no programming knowledge [3]. Even though citizen developers have no programming knowledge, they have a major role in both system development and testing in low-code development. But the lack of technical IT knowledge leads to the emergence of new requirements and challenges in the context of low-code. Even though low-code development is advertised as a way to create products, build systems and websites without previous knowledge in software development. Is it really true that a beginner, a so-called citizen developer without programming experience can perform advanced coding operations just using a drag and drop interface? If this is the case, does it follow best practices for software development?

1 Background

1.1 Best practices

Best practice is a term that means different things in different lines of work and there is a wide range of best practices. Best practices are formed through shared experience and research. The lessons learned after each project will over time help new projects to avoid the mistakes that have already been made. By learning from previous mistakes, best practices are created, not only to avoid the mistakes that have been made in previous projects, but also to create a better structure for companies of how they should be working in future projects. In software development, there are several best practices that should be followed to ensure that the quality of the software is as good as possible. The source code should be easy to read, with clear variable and function names, and follow standard coding practices. Clean, modular code is easier to maintain, test, and extend. The name of a variable, function, or class should answer all the big questions. It should tell you why it exists, what it does, and how it is used. If a name requires a comment, then the name does not reveal its intent [4].

It's important to have a clear understanding of what needs to be developed before diving into coding. This includes planning the architecture, identifying the key features and requirements, and creating a design that

meets the needs of the end-users [4]. To ensure the quality of the software, testing is one of the most important things to conduct [5]. Therefore, developers must write code which can be unit-tested and system-tested to verify that the system meets its requirements. Testing should be integrated into the development process from the start, with automated test suites on each code change. This helps catch bugs early and ensures the codebase remains stable. Furthermore, clear documentation helps other developers understand the code and makes it easier to onboard new team members. Since software is constantly changing and developing it is important to maintain the code, improve performance or correct defects. In many cases the system will be used for more purposes than it was designed for, therefore the system must be modified for different environments where the code needs to be flexible and portable. The code should be reusable, saving time and lowering complexity with small functions that can be used multiple times. Since software systems often can lead to large and complex code bases, the ease in which you can read and understand the source code of a system is crucial [5].

This study will focus on the internal best practice characteristics of maintainability, flexibility, portability, reusability, readability and testability and comparing the findings from working mainly with Power Pages which is a part of the low code development tool Microsoft Power Platform. A system will have difficulties to fulfill all the requirements at the same time.

> "The attempt to maximize certain characteristics inevitably conflicts with the attempt to maximize others. Finding an optimal solution from a set of competing objectives is one activity that makes software development a true engineering discipline" [3].

Maintainability	Change or add capabilities, improve performance, or correct defects
Flexibility	How well can the system be used for more cases that it was designed for
Portability	The possibility to use the same software in different environments
Reusability	To use the same code in several parts of the system
Readability	How easily the code or system can be read and understood

Testability	To ensure the quality of the software

1.2 low-code

Low-code is a software development approach, generally where no code writing is required to create a website or a software. To create a website or software using low-code development, a drag and drop user interface is often used. Low-code development is marketed to non-developers, often referred to as citizen developers that have no to little previous skill or experience in programming and software development. Even though low-code development is marketed to citizen developers with no prior software development, previous knowledge in this field is often required to create websites and software using low-code development [6]. Low-code is in many ways being disguised as high code. Even though low-code development is not suitable for everyone in every situation, there are clear advantages to low-code. For example, it enables quick distribution of powerful computerized functionality without requiring developers to have a deep understanding of the underlying platform or computer science.

Low-code development platforms have versatile use-cases across industries:

- Rapid prototyping: Producing prototypes quickly to test ideas.
- Business process automation: Increasing productivity and streamlining processes.
- Giving citizen developers the ability to create applications.
- Cross-platform mobile app development is being accelerated.
- Modernizing and integrating legacy systems.
- IoT Solutions: Creating software that connects and manages IoT devices.

1.3 Microsoft Power Platform

Power Platform dynamics 365 from microsoft is the low-code development platform that is used in this study to act as a sample to understand general conclusions about low-code platforms. Within Microsoft Power Platform, there are five different modules which all have a specific use-case, for example to create websites and mobile apps without writing code. Workflows can be automated for repetitive tasks or cross platform management. With the click of a button, all predetermined tasks are done by the computer. This comes with a simple interface to drop actions in the order they should be done. Power Platform also features a business analytics service that can be used to create reports, visualizations and dashboards. Power Platform also features a chatbot builder with integrated AI that can be built with a drag and drop interface like the rest. Lastly we have Power Pages, the most recent component which is based upon the same platform as the apps but specializes in building websites instead of apps. In Power Pages the user can

build simple and responsive web pages with low or no code. For this project, we will be focusing mainly on Power Pages and Power Apps, which are two of the five modules within Microsoft Power Platform. Power Platform will be evaluated through the project to discover opportunities and limitations in correlation to best practices in software development.

2 Purpose

The purpose of this study is to investigate low-code development in relation to best practices in software development. By exploring the interplay between low-code development and established best practices, the aim for this research is to investigate the potentials and the threats in low-code development. The primary objective of this study is to investigate whether the identified threats in low-code development can be solved by leveraging the best practices from software engineering. By examining the best practices, such as maintainability, flexibility, portability, reusability, readability, and testability, this research seeks to determine if low-code development can align with the rigorous standards and quality expectations of traditional software development.

Additionally, this study aims to evaluate the suitability of low-code development for a broad range of individuals with no previous knowledge of programming and web development, often referred to as citizen developers. By exploring the accessibility, usability, and learning curve associated with low-code development platforms, the research seeks to find an answer to whether low-code development is suitable for citizen developers.

2.1 Research questions

RQ 1:

How does low-code development correspond with best practice in software development ?

RQ 1.1:

How suitable is low-code development for citizen developers ?

RQ 1.2:

How can low-code adopt best practices from software development ?

2.3 Limitations of the study

In this study, the primary focus is to explore the capabilities and limitations of low-code development with a focus on Power Platform. However, it is essential to acknowledge that this narrow focus on a single low-code

platform, presents a potential limitation in terms of drawing broader conclusions about low-code platforms in general. While Power Platform is a widely used and representative low-code platform, it is important to recognize that each low-code platform may have its unique features, strengths, and limitations.

One of the boundaries encountered during the development of this study was the limitation in language options within Power Platform. As a result, the figures presented in this study are in Swedish. However, the figures primarily serve as illustrative examples and an english explanation to each figure is provided to the reader.

Licenses for the low-code development Microsoft Power Platform has been given as a part of the collaboration with CGI where they wanted research about the new module called Power Pages. The research done for CGI on this subject has not been the same as the research conducted in this study. CGI did not expect anything in return regarding the research for this study and neither them or the researchers will be considered biased. The free licenses provided by CGI is also the reason as to why this study focuses on Microsoft Power Platform as its main low code platform.

3 Method

To answer the research questions, information gathering from previous studies and research within similar fields have been collected. Furthermore, to extend and complete previous studies and research within similar fields, design and creation is the methodology for this study. The reason for using design and creation as a methodology is to evaluate a low-code development tool in regards to the research questions. With the collaboration with CGI, they have enabled the opportunity for a thorough research on the low-code development tool Power Platform.

3.1 Information gathering

Information from previous studies and research about low-code in general and research about best practices in development have been collected. When searching for articles in the same research area, the keywords “CRM, low-code, no-code, best practice, software development and software engineering” have been used. The platforms when searching for previous articles in the same research area are ACM and EBSCO. These two websites have been used because they have access to related peer reviewed research for students at Malmö university. The result from the information gathering gave a good foundation to begin this research, including the following topics:

low-code platforms in general, possibilities and limitations with low-code development, how much they are used and questions about how citizen developers can use low-code development. Furthermore, articles and books about best practices for software development have been found. We are continuing to explore the discourse that those studies use but from the perspective of best practices with low-code.

In addition to information gathering from previous studies and research, information has been gathered by building a website in the low-code platform Power Pages. The purpose with this method is to gain information and knowledge about how the code is structured and written in a low-code platform and how that code corresponds with best practices in software development. This also ties in to the purpose of the study where the meaning is to get a bigger picture about how well low-code corresponds to best practice in software development and how meaningful it is to use a low-code platform like Power Platform when writing code according to best practices.

3.2 Methodology

The methodology used in this study is called design and creation. This is a research strategy that is beneficial when developing a product, either to measure other programs, test out research questions, improve something the market needs or improve on something that is already existing. In the case of this study, design and creations is used as the methodology when it comes to developing a product to test out and learn more about the research questions. “In this way, the researcher using a design and creation strategy learns through making” [7]. Design and creation is a relatively new research field suitable for computer scientists. The benefits of this research strategy is that you can as a researcher practically understand an area by producing the measuring tool itself or by producing something that can help society, this in comparison to other research fields where you might propose changes or help create a view on how something should be done.

> “The design and creation strategy might be the only research strategy employed, where the IT artifact is itself the main contribution to knowledge where you have something tangible to show for your efforts - some kind of IT artifact - rather than just abstract theories or other knowledge” [7].

The disadvantages with design and creation is that it can be hard to distinguish between research and companies evolving regular products. Research is generally about making a contribution that others can understand and generalize. With design and creation it can be hard to generalize what you have made for the greater population even if it is possible to make it replicable [5].

3.3 Design and creation in this study

The study started with defining the best practices found in previous research as follows: Maintainability, flexibility, portability, reusability, readability and testability as explained in section 1.1 best practices.

In order to find out how these six best practices correspond with low code development, we have created a website using the low code system Power Platform. More specifically, the module called Power Pages has been used to create the front-end design elements as well as the forms and views on the website that are connected to the backend database. The backend is made in the module called Power Apps. The website is created in a manner where it would be suitable for an non-governmental organization (NGO) to use. The NGO that we have worked with in collaboration with CGI requested a website with an aim to manage football players and clubs.

The first step was designing the database in Power Apps where necessary tables were set up. The table user_posts was created to gather posts from users, the table content_post to gather news from the administrators, the table content_post_offers to keep track of discounts, the table user_contract to keep track of existing contracts, the table named feedback to collect feedback from users on the site. Finally the table User was created to store contacts with contact information. You can see the tables below in picture 3.0

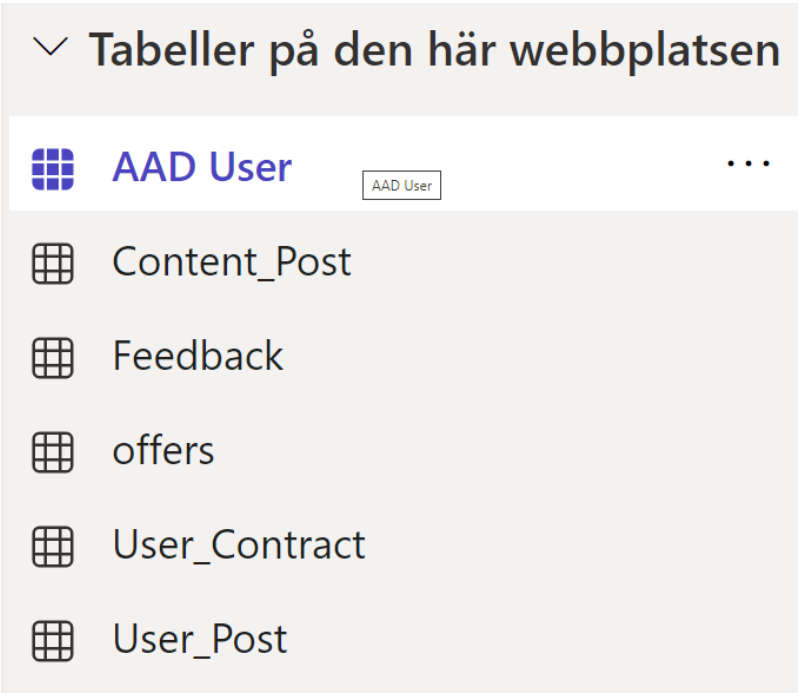


Figure 1. Tables created in Power Platform.

After setting up tables a website was created in power pages using the tables created. A standard theme was set up with green and white on different sections on the website. To fully test the principles from best practices the website was created with full functionality in mind. The website was supposed to function properly for the NGO it was designed for. Content was added on the mainpage along with forms and views of data. Subpages were created and connected to a header with a drop-down menu to enable navigation to the different subpages. Each table that was created got a meaning on a corresponding webpage. The table content_post and the table content_post_offers were used on the main site as seen on figure 2. And the table User_Post is used in the site where the user's personal diary is as you can see in figure 3. The table Feedback is used on the site where you can contact the site owner as you can see in figure 4. The table User_Contract is used on the site where you store your personal contracts and uploaded files as you can see in figure 5. Everything is developed in the module Power Platform. The reason for this is to really examine how the environment works without adding in other modules or API's.

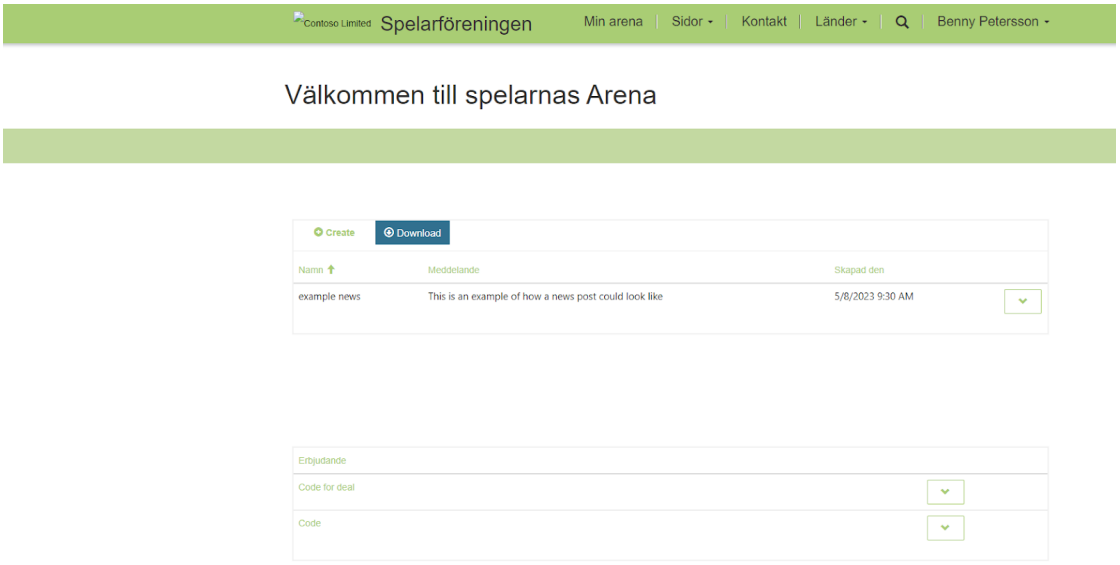


Figure 2. Website created in Power Pages using the tables content_post and content_post_offers.

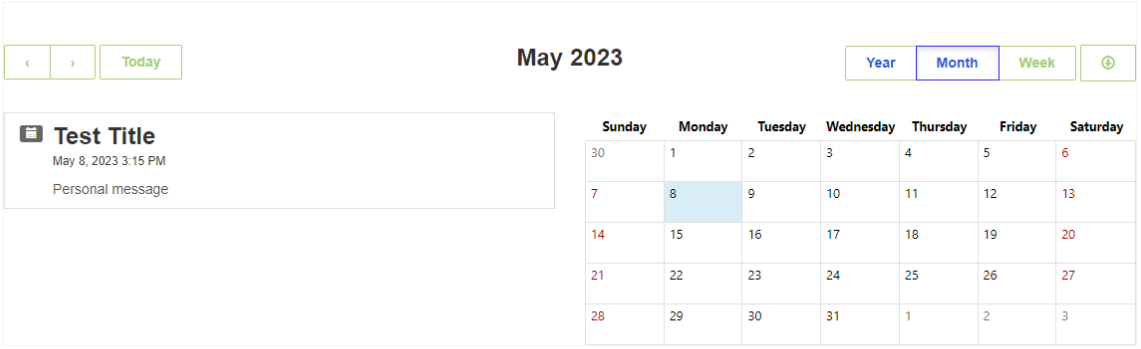


Figure 3. Website created in Power Pages using the table User_Post.

Figure 4. Website created in Power Pages using the table Feedback.

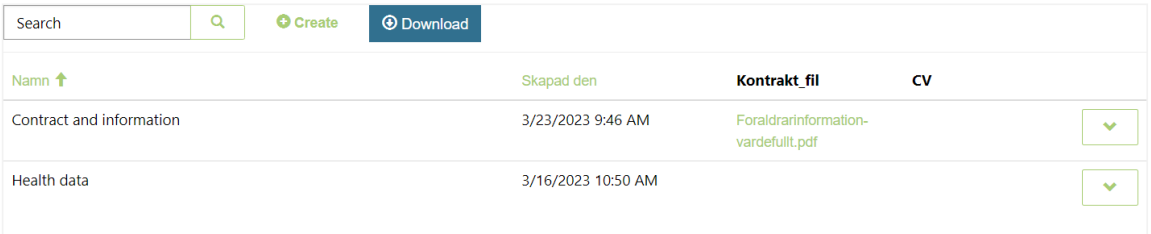


Figure 5. Website created in Power Pages using the table User_Contract.

4 Previous research in the field

In this chapter, various methods for best practice will be presented. To support the methods with empiric data, they will be presented with an

explanation and reasons as to why they are considered best practice from experts in the software development industry. Worth noting, most of the methods presented are aimed at traditional software development, therefore, this research will adapt the methods with a focus on low-code development.

4.1 Low-code

Although the term “low-code” was first coined in 2014, to date, there is no clear definition of “low-code” in either academia or industry. Practitioners use a variety of terms to describe low-code related practices [1]. Andre calcada et al describes low-code development as a technology that facilitates programming by diminishing the handwritten code and allowing nonprogrammers to build and have a more active presence in the development process of the application [8]. Low-code is a collection of tools that enables developers to avoid hand-coding and reduces the development effort of having an application ready for production. He states that low-code applications are developed using model-driven engineering principles and that they are taking advantage of cloud infrastructures, automatic code generation, high level and graphical abstractions to develop entirely functioning applications, meaning that these applications are mostly made through drag and drop of objects.

4.2 Pros and cons with low-code

As per the literature, Low-Code development platforms offer several benefits, such as rapid development and cost reduction through intuitive management. They simplify maintenance and enable the inclusion of business profiles, while minimizing unstable or inconsistent requirements. Low-code development can accelerate digital transformation and improve business responsiveness, and also reduce dependency on hard-to-find expertise. Additionally, low-code development can help lower the barriers to application development in Small and Medium-sized Enterprises (SMEs), thereby enabling them to drive digital innovation according to their own criteria. By using modular building blocks, low-code development can also help software developers reduce repetitive tasks and increase efficiency [2].

A survey and expert interviews revealed that small and medium-sized businesses are primarily interested in utilizing low-code development to address their internal processes. Almost all domain experts reported that their customer inquiries primarily focused on optimizing internal processes using low-code applications [2].

There are clear advantages with low-code, as it enables quick distribution of powerful computerized functionality without requiring developers to have a deep understanding of the underlying platform or computer science [4]. When it comes to basic usage of low-code tools, only an understanding of the "model" in the platform, such as the tables in the database, and some

sense of how to extend and configure the system with declarative constructions are required. Despite not having the same high demands on developers as using high-code tools, the distribution of applications is not much different [6]. This means in practice that a developer with limited knowledge of engineering practice can create and distribute applications, which in turn can lead to issues with version management, automated testing, and easily maintainable code. Another issue with low-code is the amount of code for the applications that are distributed, unlike traditional programming where developers constantly work to refactor their code, contributing to a smaller codebase. This problem leads to developers working with low-code tools with limited knowledge having to work with a large and hard-to-read codebase, which becomes counterproductive when it comes to using low-code tools.

To further emphasize the limitations and challenges of low-code development in correlation to software engineering practices, Y. Luo, et al extracted the following data from Stack Overflow and Reddit which are online discussion forums [1]. The data extracted from Stack Overflow and Reddit by Y. Luo, et al are topics that have been written and discussed on the online discussion forums.

- No access to source code which means that the code can not be reused.
- Vendor lock-in, the downtime or rebuild and possibly lost data is a risk which compromises the reliability.
- Difficulty of maintenance and debugging.
- Difficulty of integration which means that there is a problem with the UI, data store and calculations working together.

Low-code applications often suffer from inadequate documentation both in terms of documents from the developers and embeddable documentation within the code, even if attempts are made to provide it. Separation of concerns and reusability also present challenges for low-code applications. While plugins are a prominent feature, the ability to create modular code using low-code languages, or to reuse such code across multiple applications without creating plugins, is often lacking [4].

4.3 Can low-code replace software developers and normal code?

The adoption of citizen development may call for a shift away from traditional roles and a greater IT skill set across the organization. Additionally, expert interviews indicate that this strategy might result in a decline in the demand for software engineers and developers, which would address the present IT skills shortage [2]. The experts do, however, agree that citizen developers are not intended to completely take over the role of software engineers. R. C. Martin draws a comparison with mathematicians who want to find a math that doesn't need formalization are akin to those who think that code will someday become obsolete [9]. Instead, software developers might be able to focus on more challenging programming tasks

while citizen developers take care of easier applications. This change in roles could lead to a more effective use of resources inside IT departments.

Software developers are required when creating applications that go beyond the implicit framework of low-code development platforms [7]. However, the availability of low-code development also provides engineers and software developers with a way to increase the efficiency in development of custom software. Therefore, low-code development alone cannot completely resolve the lack of IT specialists, as IT professionals are still required for application development. However, in small and medium-sized businesses, low-code development can help to mitigate the urgent demand for IT specialists and lessen the workload of IT managers.

4.5 Best practices:

In software development there are certain best practices that should be followed. Since most of the best practices are aimed for traditional software development, there is a potential threat to low-code development, partly since they are not aimed at low-code development and since some low-code platforms are not created to follow all of the best practices. One of the biggest challenges with low-code testing is limited testing capabilities. Unlike traditional software development, low-code does not provide full access to the underlying code, which can make it difficult to perform comprehensive testing. This can result in a lack of visibility into application performance, which can lead to security vulnerabilities, bugs, and other issues. Another challenge with low-code testing is the difficulty in customizing applications. With low-code, the underlying code is generated automatically, making it difficult to modify the code to meet specific needs or requirements. This can limit the functionality of the application and make it more difficult to integrate with other systems [3].

5 Results

In section 1.1 and 1.2 low-code is described, and the best practices for software development are defined. In this chapter, the findings from developing an artifact in the low-code system Power Platform will be presented and follow the structure of the internal best practice characteristics: 5.1 Maintainability, 5.2 Flexibility, 5.3 Portability, 5.4 Reusability, 5.5 Readability and 5.6 Testability.

5.1 Maintainability

In Microsoft Power Platform, there is a system for maintainability in place. The maintainability mainly works in two different ways in Power Platform. The two systems are drastically different when it comes to maintainability, therefore, they will be divided into two parts. The first part of maintainability is for building simple websites and apps where no scripts in the form of Javascript or the C# .net library are required. This means that you are only using the low-code environment. This part will be referred to as

maintainability for the user interface. The other part of maintainability is when building a system that has advanced functionality currently not available in the low-code environment. Implementing your own code makes a big difference when it comes to maintaining the code and the system. This part will be referred to as maintainability for plug in modules.

5.1.1 Maintainability for user interface

As you can see in figure 6, the user interface that the developers meet is a drag and drop interface. In this picture, the database is set up by pressing the button in the left upper corner and then selecting a new column. The base of the backend is built up by using this database where you can add, modify and delete tables. You can connect the tables to each other using lookup fields which makes it act as a relational database. In the case of maintainability, only small errors can be made since the user does not in fact write any code. The user can simply use the modules that are already there. Most of the things can be deleted, changed or rebuilt. However, deleting tables in the database that are connected through secondary keys, like any other database, is not possible. When creating the database this way, no written code is required for creating the tables, rows, columns, or when modifying something in the database. When using Power Platforms interface to create a database, the source code is hidden in the background. Instead of writing code for creating the database, Power Platform offers a user interface with buttons, such as search, sort, add and delete.

Maintainability for developing the frontend of the app works in the same way as creating databases, by using modules that are available in the low-code environment. New components can be added to the website by choosing to add a module that is either connected to the database, or a plain text/image. Maintaining this part mainly requires that the user presses the buttons for adding, deleting and editing. All of these components can be modified by using the interface and the source code is generated behind the scenes with HTML code and predefined CSS styles.

Another aspect of maintainability is that the platform goes through massive changes twice a year where functions might go missing and new functions arise. This calls for updating every system connected to Power Platform to work with the new updates which can be a massive maintainability project.

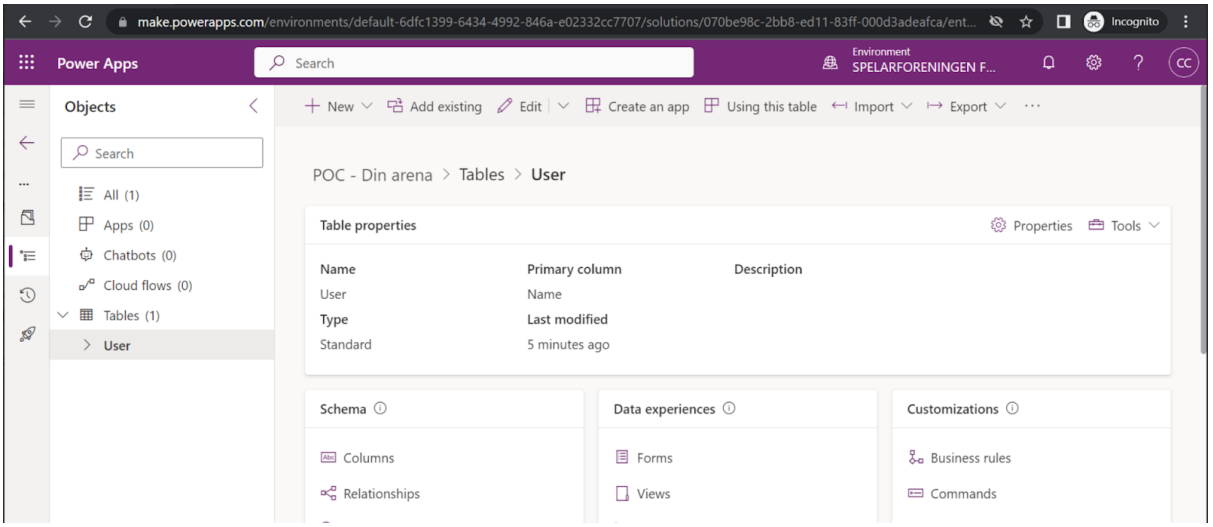


Figure 6. User interface for Power Apps.

5.1.2 Maintainability for plug in modules

Power Platform does not have all the functionality required to fully support advanced programming and is somewhat limited to using the pre-built modules. However, since Power Platform runs on C# and .Net it allows for making your own scripts with additional functions. These additional functions run on standard code which requires previous knowledge in programming from the developer. As you can see in Figure 7, developers are required to ensure that their own code is divided into functions and built in a way that facilitates the testing of the components. The figure shows C# code that does not yet have added functionality. The code gets the information needed from the low-code system. Once the information is retrieved, changes that interact with the low-code system can be made.

As mentioned in section 5.1.1, the functions can be subjected to change twice a year and the code that has been added to the plug-ins might not work with the low-code systems anymore.

```C#
// Obtain the tracing service
ITracingService tracingService =
(ITracingService)serviceProvider.GetService(typeof(ITracingService));

// Obtain the execution context from the service provider.
IPluginExecutionContext context = (IPluginExecutionContext)
    serviceProvider.GetService(typeof(IPluginExecutionContext));

// The InputParameters collection contains all the data passed in the message re
if (context.InputParameters.Contains("Target") &&
    context.InputParameters["Target"] is Entity)
{
    // Obtain the target entity from the input parameters.
    Entity entity = (Entity)context.InputParameters["Target"];

    // Obtain the organization service reference which you will need for
    // web service calls.
    IOrganizationServiceFactory serviceFactory =
        (IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganiza
    IOrganizationService service = serviceFactory.CreateOrganizationService(cont

    try
    {
        // Plug-in business logic goes here.
    }

    catch (FaultException<OrganizationServiceFault> ex)
    {
        throw new InvalidPluginExecutionException("An error occurred in FollowUp
    }
```

Figure 7. Back-end code with no added functionality.

5.2 Flexibility

The flexibility for Power Platforms is somewhat limited. There is a framework for building websites, apps and databases. Furthermore, databases can be imported from other places. However, there are limitations when importing databases from other places since the style of build will be adapted to the same style of build conformed to the entities that exist, the ones you can choose from within the low-code interface. Therefore low-code systems only allow for flexibility in the field that it is designed for. It does not allow for full customization as you can see in figure 8. The figure shows how to create new components for the website in Power Pages. To create a new component, the developer will need to choose what type of component that should be added. For Power Pages, there are several options to choose between, for example text, button, image, video or forms. All the componentes will end up beneath each other. For developing basic websites the developer adds components that are either plain text/image or that are connected to the database or other API's. This is in contrast to development in languages like Java where you can directly implement different frameworks in the code to cater for your needs.

Experienced developers can access the HTML and CSS code that the low-code platform generates to manually modify the styling of the website or add functionality that the low-code interface does not offer. For an experienced developer, low-code development is flexible. However, to reach a high flexibility in low-code development, it is arguably more difficult than traditional web development.

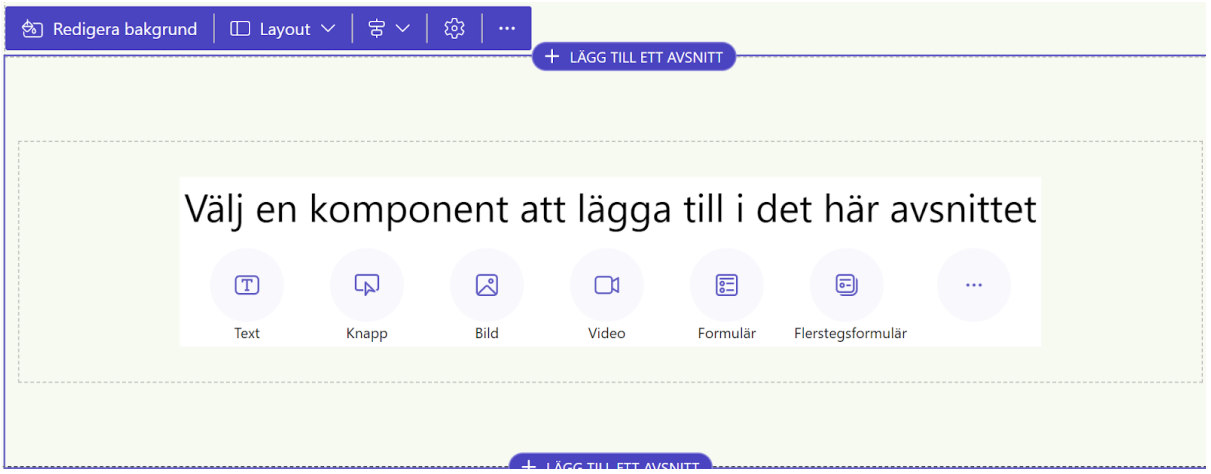


Figure 8. User interface for creating a new component.

5.3 Portability

The possibility to use the same software in different environments is limited. Generally, the coding is done within the platform and you connect external components to that low-code platform. The low-code platform is then the main platform that is used which means that the possibility to move and reuse the software created in a low-code application are limited. However, some low-code platforms enable portability within their own platform. Power Platform is built up by five different applications which work together. This means that you can create the database in one of Power Platforms applications and can then use the database in another Power Platform application. For example the database can be created in Power Apps which then can be used in Power Pages which is the application for creating the graphical interface for a website.

5.4 Reusability

Low-code platforms mainly achieve reusability within their own platforms. Power Platform has reusability between all their platforms when using the dataverse-database which lies underneath all the different applications as seen in figure 9. Figure 9 shows a system overview of the different low-code systems Microsoft Power Platform consists of. At the bottom dataverse is the common platform that spans through all the systems. This means that the user can use the same database in every system as long as they create and develop everything within the Power Platform ecosystem. The user also has the possibility to use data from another platform as the base for all the apps. In the ecosystem of Microsoft Power Pages, the code and the modules that are built are almost fully reusable. When connecting to other systems,

the low-code modules do not function nor is the code behind them as it is made for low-code logic.

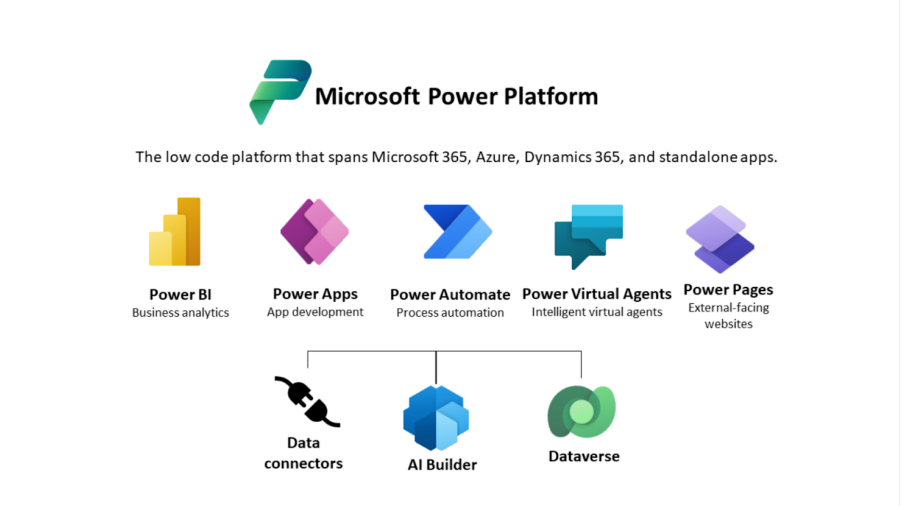


Figure 9. The different low-code systems in Microsoft Power Platform.

5.5 Readability

Readability for Power Platform and low-code platforms are a bit different than traditional code because of the low-code interface that you mainly see. Readability in Power Platform can be divided into three categories. One category with the user interface, one category with connected code through plug-ins and one category for the underlying source code.

5.5.1 Readability for user interface

In the user interface for Power Platform there is an overview of different functions and implemented tables. In this overview the code is not visible to the user, only the functions. Therefore the readability in this view is unobstructed. It is always the same no matter who the developer is. In figure 10 you can see the overview of data types for incoming data in the middle. To the right you can see the functions coming in from javascript code as an event.

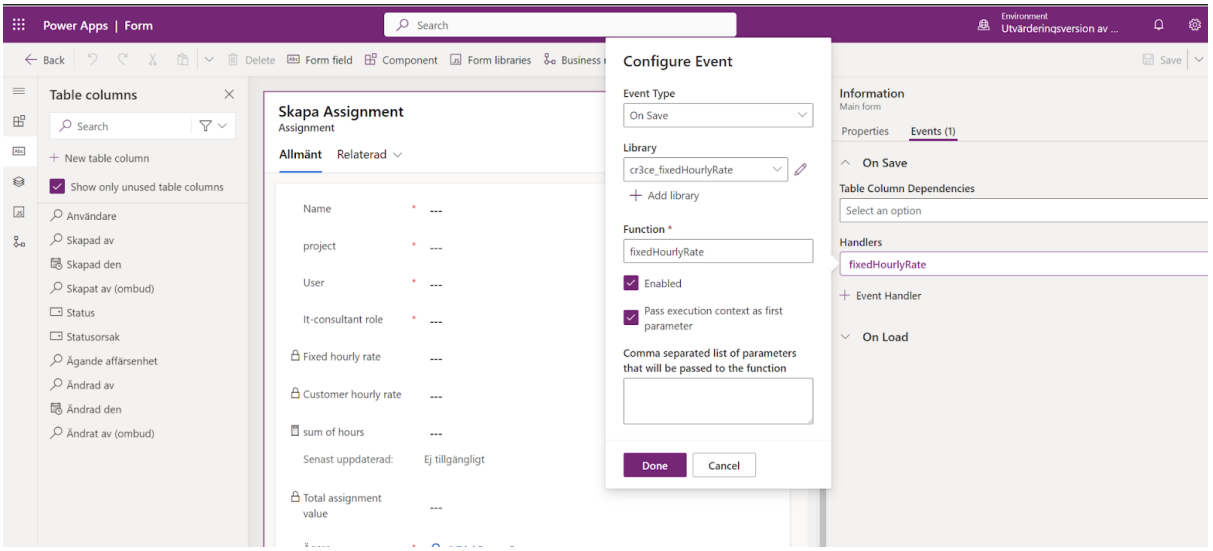


Figure 10. User interface for Power Apps.

5.5.2 Readability for plug in modules

The readability in the plug in modules differs from the overview in the user interface. Each plug in module has their own file of code which means that there is a separation between functions depending on where you are in the program. As you can see in figure 11, there is a single function to the event from figure 10 and the readability is dependent on how the developer writes the code.

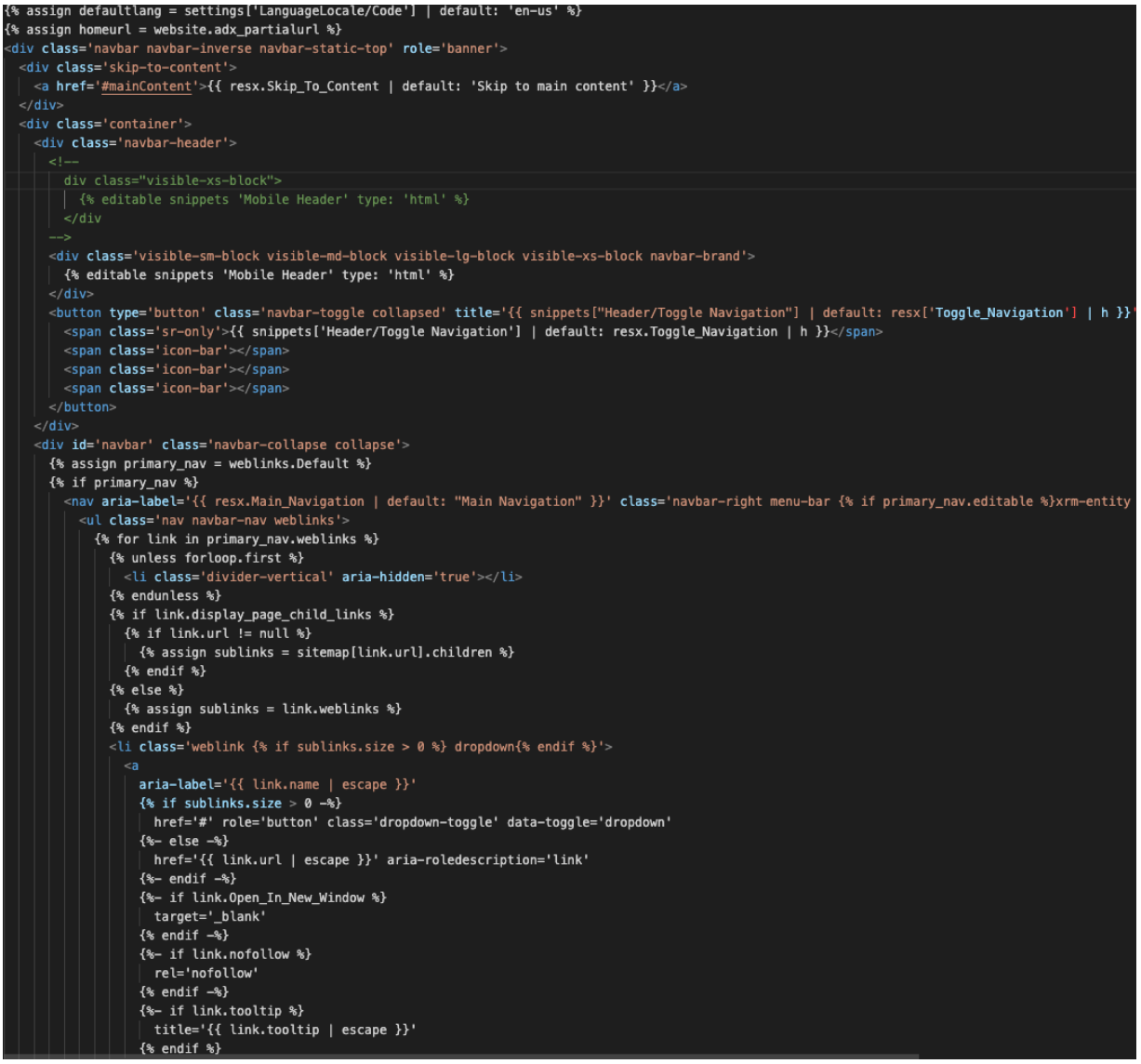
```
var sdk = window.sdk || {};

(function () {
  this.attributeOnChange = function (executionContext) {
    var formContext = executionContext.getformContext();
    var consultantrole = formContext
      .getAttribute("bp_itconsultantrole")
      .getValue();
    if ((consultantrole = "Projekt Manager")) {
      formContext.getAttribute("bp_itconsultantrole").setValue(1500);
    } else if ((consultantrole = "Architect")) {
      formContext.getAttribute("bp_itconsultantrole").setValue(1000);
    } else if ((consultantrole = "Developer")) {
      formContext.getAttribute("bp_itconsultantrole").setValue(800);
    }
  };
}.call(sdk));
```

Figure 11. Plug in modules for Power Platform.

5.5.3 Readability for the source code

In some cases the developer will need to write and change the source code to be able to customize the application. As you can see in figure 12, the readability is not optimized. Figure 12 shows a small part of the source code

for the header of the website created in Power Pages. To create the same header in traditional web development, the source code should be structured in a way which is easy to read, preferably with comments to increase the readability. In this case, with low-code development, the source code makes it look more advanced than it might be because of the low readability. This can intimidate a citizen developer, meaning that they rather use the limited styling options that are available in the user interface, instead of customizing it the way they want.



```
{% assign defaultlang = settings['LanguageLocale/Code'] | default: 'en-us' %}
{% assign homeurl = website.adx_partialurl %}
<div class='navbar navbar-inverse navbar-static-top' role='banner'>
  <div class='skip-to-content'>
    <a href='#mainContent'>{{ resx.Skip_To_Content | default: 'Skip to main content' }}</a>
  </div>
  <div class='container'>
    <div class='navbar-header'>
      <!--
        div class="visible-xs-block">
        | {% editable snippets 'Mobile Header' type: 'html' %}
        </div
      -->
      <div class='visible-sm-block visible-md-block visible-lg-block visible-xs-block navbar-brand'>
        {% editable snippets 'Mobile Header' type: 'html' %}
      </div>
      <button type='button' class='navbar-toggle collapsed' title='{{ snippets["Header/Toggle Navigation"] | default: resx['Toggle_Navigation'] | h }}'
        <span class='sr-only'>{{ snippets['Header/Toggle Navigation'] | default: resx.Toggle_Navigation | h }}</span>
        <span class='icon-bar'></span>
        <span class='icon-bar'></span>
        <span class='icon-bar'></span>
      </button>
    </div>
    <div id='navbar' class='navbar-collapse collapse'>
      {% assign primary_nav = weblinks.Default %}
      {% if primary_nav %}
        <nav aria-label='{{ resx.Main_Navigation | default: "Main Navigation" }}' class='navbar-right menu-bar {% if primary_nav.editable %}xrm-entity
          <ul class='nav navbar-nav weblinks'>
            {% for link in primary_nav.weblinks %}
              {% unless forloop.first %}
                <li class='divider-vertical' aria-hidden='true'></li>
              {% endunless %}
              {% if link.display_page_child_links %}
                {% if link.url != null %}
                  {% assign sublinks = sitemap[link.url].children %}
                {% endif %}
              {% else %}
                {% assign sublinks = link.weblinks %}
              {% endif %}
              <li class='weblink {% if sublinks.size > 0 %} dropdown{% endif %}'>
                <a
                  aria-label='{{ link.name | escape }}'
                  {% if sublinks.size > 0 -%}
                    href='#' role='button' class='dropdown-toggle' data-toggle='dropdown'
                  {%- else -%}
                    href='{{ link.url | escape }}' aria-roledescription='link'
                  {%- endif -%}
                  {%- if link.Open_In_New_Window %}
                    target='_blank'
                  {% endif -%}
                  {%- if link.nofollow %}
                    rel='nofollow'
                  {% endif -%}
                  {%- if link.tooltip %}
                    title='{{ link.tooltip | escape }}'
                  {% endif %}
```

Figure 12. Source code created by Power Platform.

5.6 Testability

When publishing a Power Pages website, there is a checklist created for developers to ensure that the site is working as it should, see figure 13. There are seven points to complete to ensure a fully functional website and within these seven points, there are several additional points which must be completed. As seen on figure 13, the first area which is “kör webbplatskontrollen” (run the site checker), 17 points were approved, 1

warning and 0 errors. Moving on to figure 14, more information is given to each point. To make it as user friendly as possible, there are solution tips for the points which are not approved.

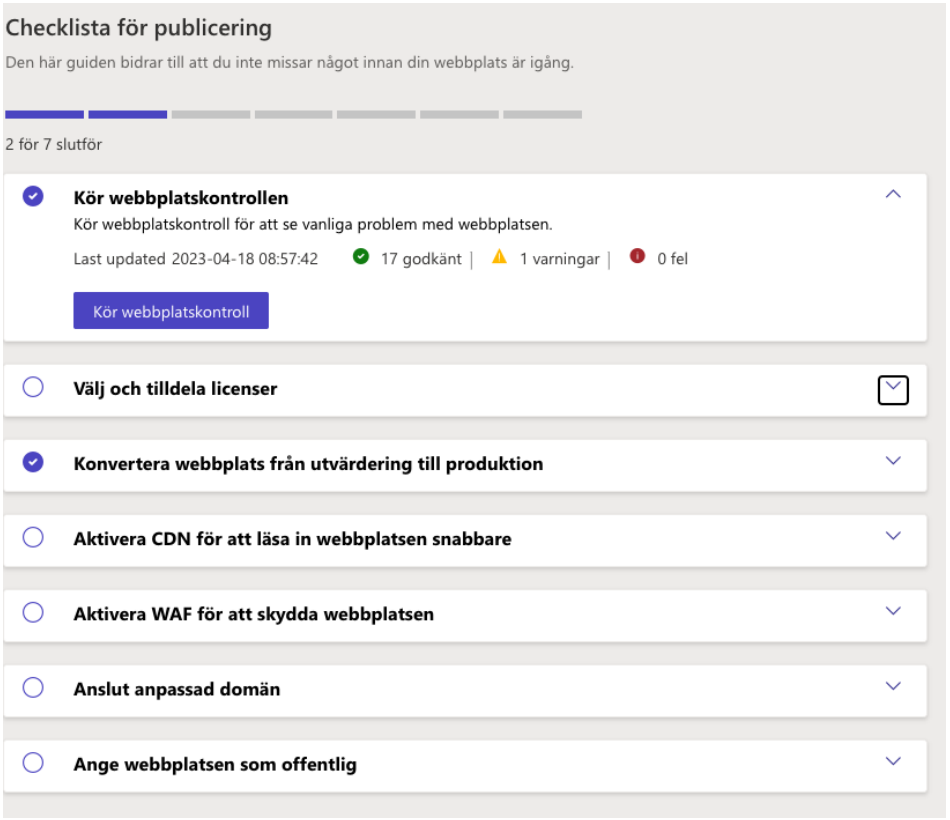


Figure 13. Checklist for publishing the website.

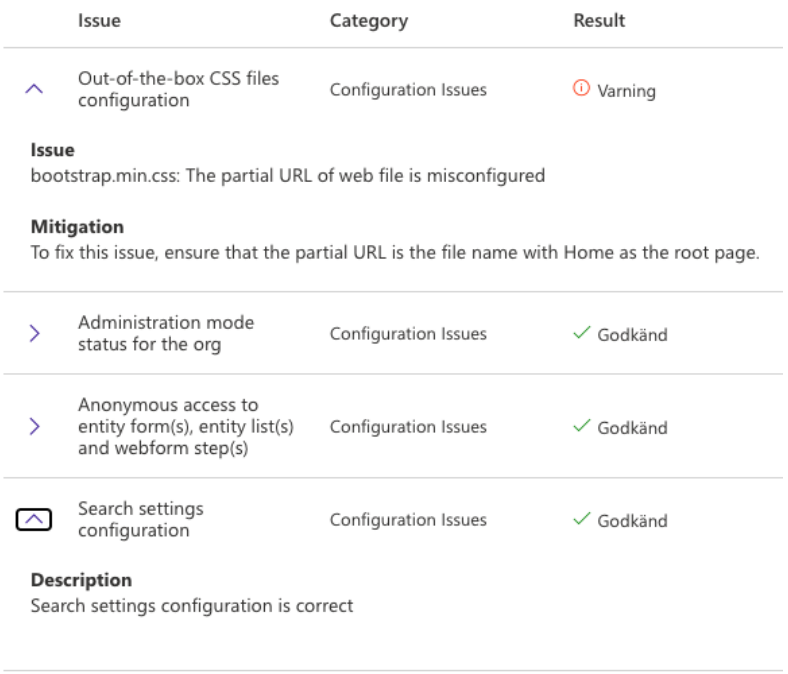


Figure 14. Further information for each checkpoint.

When using Power Pages, there is no specific tool for testing the application or website beyond the “checklist for publishing”. However, if the developer decides to create a canvas app in Power Apps, there is a system for testing which is called Power Apps Test Studio (PATS). A canvas app is much the same as Power Pages, the biggest difference is that in canvas apps the developer will start with a blank app/page whether as in Power Pages there are templates to follow and the database is created first. PATS is a low-code solution to write, organize, and automate tests for canvas apps [10]. There are two options for writing tests in Test Studio: either utilize Power Apps expressions or a recorder that records your interactions with the app and generates the expressions for you. Then, you can utilize Test Studio to replay these tests and ensure that your app is operating properly. In order to automate testing, you can also run the tests using a web browser and incorporate them into your app deployment procedure.

6 Analysis and discussion

6.1 How suitable is low-code development for citizen developers?

A citizen developer, as mentioned in the introduction, are people without a classical IT background, who are primarily based in business departments and show IT affinity as well as technical expertise [2]. So how suitable are low code systems and especially Power Platform for citizen developers?

Y. Lou et al says that low code development allows users to create systems faster with minimal effort. It also makes development more agile since the platforms are equipped with ready-to-use implementations [1]. A. Calçada et al shows that developing in low code is on average 1,74 times faster than developing in java swing and 1.10 times faster than javascript [8]. As for this study, low-code is proven to indeed be a fast way to create a website with basic functionality without previous programming or web development knowledge. However, low-code development still requires knowledge about common architectures for building more advanced systems and databases.

When working with the low-code development interface, the results show that the readability is good and easy to understand since the interface of the different pages have a consistent user interface. The maintainability is good as long as the developer only uses the basic functions and keeps a tab on new updates. As long as the developer keeps to the low-code interface in their development the results have shown that the analyzed Power Platform has full reusability between the components. In summary, low-code systems and Power Platform seem to share the positive aspects of developing faster than traditional coding and keeping ready-to-use modules with a built in architecture. Power Pages in particular support good readability and reusability. This seems to imply that low code development is fitting for a citizen developer.

Low-code also comes with disadvantages, both for a citizen developer and for senior developers since low code systems can be slower than writing code because of their generated larger code size.

> “In terms of execution runtime, Java Swing applications are on average, 2.72 times faster and JavaScript applications 36.61 times faster, when compared to Low Code applications“ [8].

Low-code systems also face challenges such as slow loading and publishing times, lack of access to source code, and being locked into a specific vendor [1]. The consequence of this can be too high costs for switching systems. Developing in Power Platform and analyzing the work has shown that the platform is under constant change. Functions and implementations change and new functions get added. This creates problems where the modules you have connected with traditional code or the modules you have built in the system might not work. It was shown that Power Platform comes with limited capability when developing traditional websites. In order to create fully functional websites and design capabilities, previous knowledge in programming and web development are required. The reason for this is because the functionality and design options in the low-code interface are limited. The study has shown that the styling capabilities of Power Platform are limited to changing theme colors and text sizes, as well as padding and margin. Therefore the developer will need to add their own source code. In order to add all the functionality and styling to the website, there are several things that are required from the citizen developer. Firstly, the citizen developer will need to find the right file for the source code. Secondly, they need to find a way to add their own source code to the file which may be the biggest challenge. The source code created by the low-code tool has a low readability, mixes different programming languages and they more than often use an external framework. This means that the citizen developer will need to learn HTML, CSS, the framework and in some cases JavaScript just to be able to change the styling of the website. Furthermore, to add functionality, the citizen developer will need to learn back-end programming. For example when working with Power Platform, they will need to learn C# and .Net but this may vary between different low-code development tools. The same knowledge is needed to maintain the written code. As seen in figure 7 and figure 8, this is no simple task for a citizen developer. When a citizen developer creates these extra modules and adds their own source code, the readability may even decrease further.

In summary, Power Platform and especially Power Pages is suitable for a citizen developer when creating a simple application in a fast way using the drag and drop interface. Worth noting is that even if the development is fast, the execution time is not [8]. As soon as the application or website requires functions or designs beyond what is offered in the low-code user interface, previous knowledge and experience in programming and web development is required. This seems to imply that citizen developers should stick to creating the simpler modules in the drag and drop interface. For the more advanced parts of low-code development, cooperating with more experienced developers is advised. It also seems to apply that a citizen developer needs a low code system that suits what they are creating.

6.2 How can low-code adopt best practices from software development

Best practices was first described in the background as a number of ways to plan and develop in a good manner. With an emphasis on the best coding principles like planning the architecture and identifying the key features and requirements to create a design that meets the needs of the end users.

The method of this study was design and creation by building a full frontend and backend website in a low-code environment to see how well low-code systems corresponded with best practices in software development. As mentioned by T. C. Lethbridge, only basic understanding of the model seems to be enough when it comes to planning the architecture and identifying the requirements that lead to a good product for the end user [6]. However, through the work of this paper, we have discovered additional best practice principles in software development. In the background it was mentioned that clean, modular code is easier to maintain, test, and extend. S. McConnell stated that since software is constantly changing and developing, the requirement for modifying a software system to change or add capabilities, improve performance, or correct defects are important [11].

By using design and creation as the method for this study, we have been able to investigate and analyze the low-code environment Microsoft Power Platform in regards to best practices in software development. Maintainability in low-code development was made in a way that it is easily maintainable since the low-code environment is divided into modules that connect to each other. When using the low-code interface, developing the system in a maintainable way should be easy, as long as the system only consists of the modules from the low code environment. T. C. Lethbridge challenges the findings in the result about maintainability by saying that this in practice means that a developer with limited knowledge of engineering practice can create and distribute applications, which in turn can lead to issues with version management, automated testing, and easily maintainable code [4]. This leads us to the downside of Power Platforms maintainability, which is maintaining the source code that the developer creates in C# or Javascript. This code has no version management that connects to the low code and might not work properly with updates to the low-code system.

In summary, developing in Power Pages leads to easily maintainable code as long as the developer only uses predefined functions. As soon as the developer wants additional functions it needs to be done by adding source code. However, there is no version management and the files connected to Power Platform are all separated. Maintainability has both benefits and downsides. Maintainability seems to be at its best when only using the predefined modules in Power Platform.

The best practice subcategory flexibility is described by S McConnell who states that the system in many cases will be used for more cases that it was originally designed for [11]. As for the results from the study the conclusion might be drawn that low-code systems mainly act in their own area and mainly allow for flexibility in the field that the system is designed for. It seems to be difficult for a user of the system to use the flexibility of the system since they don't control the source code. Experienced developers can access the source code that the low-code platform creates and add their own source code to modify the design and add functionality. For an experienced developer, low-code development is more flexible than for a citizen developer. However, we would argue that reaching flexibility for a system or a website, traditional web development enables this in an easier way. In summary it seems that the flexibility for Power Platform is lower than traditional web development and the conclusion can be drawn that this might be the same for other low-code platforms. We suggest that developers seeking high flexibility might find this easier to achieve through traditional web development compared to low-code development.

S. McConnell further describes best practices with the subcategory portability as the possibility to use the same software in different environments [11]. The results of this study shows that the portability is limited because the development is mainly done within the specific low-code environment. External databases and APIs can be brought in, but the possibility to move and reuse the software created in a low-code application are limited. However, Power Platform enables portability within their own platform. If Power Platform with all the modules were to be connected with something else such as a subsystem it would probably carry a massive codebase with it. T. C. Lethbridge comments on the codebase of low-code systems and states that an issue with low-code is the amount of code generated [6]. This is the contrast to software development, where developers constantly work to refactor their code to minimize the size of the codebase. In summary the portability of Power Platform seems to be highly limited when working with other platforms. This seems to imply that Power Platform and other low code platforms should be used when it does not need to be portable with other software outside of the low-code environment. On the contrary it seems that the portability and interconnectivity within the low-code environment is good.

S. McConnell further describes best practices with the subcategory reusability as the possibility to use parts of a system in other systems which contributes to effectiveness since developers will not need to rewrite the same code [11]. The results of this study shows that low-code platforms have some reusability. Foremost they have reusability of their own modules in their own platforms. Power Platform has reusability between all its platforms as long as you use the dataverse-database that lies underneath all the different applications. The entire system is in the cloud and can be accessed everywhere.

> “A low code development platform can be set in a cloud, allowing the development of Low Code applications using minimal code writing” [8].

At the bottom of Power Platform, dataverse is the platform that spans through all the systems as seen in figure 9. In the ecosystem of Microsoft Power Platform, the code and modules that are built are almost fully reusable. However, when connecting to other systems outside of the Power Platform environment, the low-code modules do not function, nor is the source code generated from the low-code development since it is made for low-code logic. In summary it seems that developing for reusability outside of Power Platform modules is difficult since the developer might not be able to access the source code, hence it is difficult to reuse it. On the contrary, the reusability within the low-code platform is excellent and supports reusability on a high level which speeds the development up. This seems to imply that it is beneficial to have much of the development within the Power Platform. This causes vendor lock in, but functions well in itself.

S. McConnell states that since software programs often can lead to large and complex code-bases, the ease with which you can read and understand the source code of a system is crucial [11]. The results of this study shows that readability for Power Platform and low-code platforms differ from traditional software development because of the low-code interface that you mainly see where you have an overview of the modules and what they contain. This overview is easily readable but also takes time to read because there are different subareas of the interface to find what you want to see. In traditional software development on the other hand, the flexibility is much higher in this regard, allowing almost-limitless flexibility in arranging files [4]. The interface leaves no room for commenting your code properly.
The readability inside the code modules is difficult to understand and read as the developer can not see the source code and can not see how the code connects since connections are seen through the interface. Inside the code modules, the generated source code is unstructured with different code sections coming from Power Platform. In summary it seems that the readability within the Power Platform modules is good. However, they are slow without a proper overview of everything. The readability for using the generated source code is lower than traditional software development since they can not be structured into files and folders. This seems to imply that the readability is high when using basic functions but decreases drastically when adding source code. This can lead to frustration amongst developers who want to keep good track of their code.

S. McConnell further describes best practices with the subcategory testability as in order to ensure the quality of the software, testing the software is one of the most important things to conduct [11]. Therefore, developers must write code which can be unit-tested and system-tested to verify that the system meets its requirements. The results of the study shows that for Power Pages, the testing is mainly done in a small test tool to ensure the developer that the website is ready for deployment. It is a small checklist with suggestions if something is wrong. Power Platform on the other hand has a bigger testing suite program called Test Studio. In the test studio there are two options for writing tests. First option is to utilize Power

Apps expressions or a recorder that records your interactions with the app and generates the expressions for you. These tests are written using a low-code language and can even be recorded by clicking on the things you want to test with the mouse. This may be a significant limitation for developers who want to ensure the quality of their code. In summary testing capabilities are limited when developing a website in Power Platform but capable of doing user friendly test-suites for the application development. This seems to imply that testing is a vital part of the Platform and easy to use. It may be difficult to reach a high testability since testing is not fully developed in low-code development yet.

7 Conclusion

The aim of this study was to investigate how suitable low code-development is for citizen developers and how low code can adopt best practices from software development. As for suitability for citizen developers, the conclusions can be drawn that Power Platform is a fast way to develop a website with enough functionality to function properly and can be deployed instantly. For the sake of citizen developers, this platform can be used in that way. The platform has good readability and reusability when using the low-code interface. When the application or website requires more functions than the interface offers, previous web development knowledge is required. The same goes for the limited styling options, where the citizen developer is required to add HTML, CSS and JavaScript to be able to fully customize the styling of the website. To summarize and conclude the answer for the first research question: How suitable is low-code development for citizen developers? The answer is that low-code development is suitable for citizen developers as long as they can manage with the functionality and styling options that are offered within the low-code interface. However, when the citizen developer needs to add further functionality or styling to the website, low-code development may not be suitable for a citizen developer. It is important to note that the conclusions drawn about low code development may not be generalizable and may not be representative of other low-code development tools since this study focuses on a single low code platform.

As for the adoption of best practices from software development in low-code development, the conclusion can be drawn that Power Platform does to some extent correspond with best practices from software development but with limitations. The code is easily maintainable and the reusability is high when using the low-code interface. The same goes for the readability, it is good as long as you are developing in the low-code interface. The testability has more limitations since the testing capabilities are limited to tests on things that are developed in the interface of Power Platform. The conclusion can be drawn that the best practices from software development are limited when developing in the low code system Power Platform. However, most of the best practices are still fulfilled in some way, even if it might not be optimized.

When it comes to more advanced functions and development, Power Platform performs worse. It has no version management when adding source code and the code has low readability since there are several files containing different programming languages which in many cases are mixed together. The flexibility is low since the code is connected to modules in which the code is hidden and unchangeable. The portability is limited and carries a big generated code-base with it. The reusability outside the platform is low since the functions created in the platform stays in the platform and the code generated is inaccessible. The readability is low because of cluttered code with extra bits and pieces of code. When it comes to the CSS part, it is especially hard since that document connects to hidden HTML code.

8 Future work

For future study in the subject of low-code development in regards to best practices in software development, it is recommended to extend the testing and examine how low-code development corresponds with the same best practices utilized in this study across multiple platforms. This will provide a comprehensive understanding of the applicability and effectiveness of these best practices in the low-code development context. Additionally, to enhance the validity and generalizability of the findings, future research should aim to work with a larger sample size by incorporating a broader range of low-code development tools. This will allow for a more robust analysis and a deeper exploration of the impact of low-code platforms in regards to best practices in software development. By conducting these studies, researchers can contribute to the advancement of knowledge in the field of low-code development and provide valuable insights for practitioners seeking to adopt best practices in their low-code projects.

References

[1] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, “Characteristics and Challenges of Low-Code Development,” *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Oct. 2021, doi: https://doi.org/10.1145/3475716.3475782.

[2] L. Bies, M. Weber, T. Greff, and D. Werth, “A Mixed-Methods Study of Low-Code Development Platforms: Drivers of Digital Innovation in SMEs,” *IEEE Xplore*, Nov. 01, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9987920. [Accessed: Feb. 28, 2023]

[3] F. Khorram, J.-M. Mottu, and G. Sunyé, “Challenges & opportunities in low-code testing,” *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, Oct. 2020, doi: https://doi.org/10.1145/3417990.3420204.

[4] Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin

[5] R. Souza, O. Cico, and I. Machado, “A Survey on Software Engineering Practices in Brazilian Startups,” *arXiv:2108.00343 [cs]*, Jul. 2021, Accessed: Mar. 02, 2023. [Online]. Available: https://arxiv.org/abs/2108.00343

[6] T. C. Lethbridge, “Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering,” *rd.springer.com*, Oct. 17, 2021. https://rd.springer.com/chapter/10.1007%2F978-3-030-89159-6_14 (accessed Feb. 07, 2023).

[7] B. J. Oates, M. Griffiths, R. McLean, and B. J. Oates, *Researching Information Systems and Computing*. Los Angeles: SAGE, 2022.

[8] A. Calçada and J. Bernardino, “Experimental Evaluation of low-code development, Java Swing and JavaScript programming,” *International Database Engineered Applications Symposium*, Aug. 2022, doi: https://doi.org/10.1145/3548785.3548792.

[9] R. C. Martin, *Clean code a handbook of agile software craftsmanship.* Upper Saddle River [Etc.] Prentice Hall, 2010.

[10] ChrisBal-MSFT, “Test Studio - Power Apps,” *learn.microsoft.com*. https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/test-studio (accessed Apr. 06, 2023).

[11] S. McConnell, *Code Complete*, 2nd ed. Pearson Education, 2004.