JOHANNES KEPLER UNIVERSITÄT LINZ

MANAGEMENTPAPER

LOW CODE

Introduction

Many companies rely on low code applications to give employees without programming skills more options. Applications like Microsoft Power BI follow different principles to make this possible. They suggest that programming skills are not necessary. The use of these applications presents many great challenges. Users with basic programming skills are probably better equipped to implement low code applications. For companies, it is important to know which skills employees need to be able to use low code applications efficiently and effectively.

This project aims to collect and evaluate a catalogue of skills and capabilities, but also to identify barriers and possible reasons for rejection (or non-use).

Principles

The principle of low code is to minimize the use of a textual programming language and instead utilize alternative techniques which are closer to the natural thoughts of users. This type of programming offers several advantages, including the ability to design software quickly and easily, avoiding waiting times for programming services from IT experts, and the potential to find a custom-fit solution. Low-code programming models can be divided into three different techniques.

1. **Programming by Demonstration:** This technique involves the user demonstrating a task using mouse and keyboard. The system records the task executed and generates a program that accomplishes the shown task. The demonstration happens on a stage within the program. Example: CoScripter, Excel Macros
2. **Programming by Natural Language:** When it comes to programming by natural language, the user has to enter a text in their natural language. The system then translates the text to a program. Optionally, this program can be rendered on a code canvas for the user to read. Example: ChatGPT
3. **Visual Programming:** Using visual programming languages, users can create programs by changing their visual representation. To create a program, the user drags components from a palette to the canvas. These components can then be adapted in a config pane. Examples: BPMN, Scratch, Power BI.

Skills and Capabilities

In this paper, attributes of citizen developers have been redefined for low code users. These two groups share many attributes. The only big difference is that citizen developers do not have any programming skills. Low code users may have programming skills. Skills that can be redefined for low code users are curiosity, problem-solving skills, and a natural motivation for continuous learning and upskilling. Also very suitable are people with the ability and passion to lead changes with great business expertise and in-depth knowledge of business processes in their working domain. However soft skills like creativity, communication, and an understanding of how development teams work are essential to use low code applications successfully and effectively. Nevertheless, when working on a bigger project in a low code development team, at least one person should come up with knowledge about technical requirements, programming, and design. The literature research findings also include that digital

Authors:
Maximilian Gumplmayr
Florian Manchen
Florian Offenberger
Georg Rülling
Markus Schwarz

natives are probably more suitable than digital immigrants for operating low code applications. Since digital natives grew up with digital technology, they are more used to the rapid nature of changes in digitalization. Although digital natives become frustrated when restrictions on their use of technology are made.

Applications

Low development platforms (LDCPs) are becoming increasingly popular in many companies due to their ability to simplify software development and make it more accessible to non-technical users. LCDPs provide a visual, drag-and-drop interface to create software applications, reducing the need for coding knowledge.

As the use of LCDPs grows, so does the number of available options, making it important to evaluate and compare different platforms. In a study by Apurvanand Sahay et al., a framework was presented based on five dimensions: usability, scalability, security, integration, and extensibility. The study evaluated five popular LCDPs and found that each had strengths and weaknesses in different dimensions, with no single platform excelling in all areas. Additionally, organizations should consider how the LCDP fits into the overall digital ecosystem and supports the organization's goals and objectives.

The architecture of LCDPs can be broken down into three main components: the development environment, the runtime environment, and the underlying infrastructure. LCDPs can also be separated into different architectural layers and classified into two types: code-centric and model-centric.

Acceptance & Rejection

Low-Code/No-Code-Platforms (LCNC) aim to enable knowledge workers to develop simple solutions for problems in their business processes. LCNC platforms minimize development time, costs, and dependency on professional developers, in addition to avoiding shadow IT. The acceptance of such platforms can vary, as traditional IT departments may show resistance against the usage of said platforms. LCNC platforms are highly flexible toolsets, with their difficulty scaling with the complexity of the application. Ploder et al. (2019) introduce an adapted version of the Technology Acceptance Model (TAM) from Venkatesh and Davis.

TAM describes Subjective Norm, Image, Job Relevance, Output Quality, and Result Demonstrability as factors that play into the so-called Perceived Usefulness. The latter, together with the Perceived Ease of Use determines the Intention of Use.

In contrast to the TAM model, there is no counterpart in technology rejection. Technology rejection in general is not as well researched as technology acceptance however, S. Goode (2005) identifies four different domains of technology rejection, namely the environment, the firm or organisation, the user or person and last but not least the system in which the new technology will be deployed.

Note

This management paper is written based on literature. Since no interviews were done until now there might be some discrepancies between this management paper and the finished seminar paper. Only people who work with Power BI are interviewed. Because low code applications differ greatly, facts gained from the interviews might not meet the skills and capabilities of other low code applications.

Authors:
Maximilian Gumplmayr
Florian Manchen
Florian Offenberger
Georg Rülling
Markus Schwarz