Low-Code vs. Model-Driven Architecture

Markus Reiter

University of Innsbruck, Austria

Abstract. In this paper we present a survey conducted by comparing low-code tools with model-driven tools for developing software in order to get a better understanding of the advantages and drawbacks of both approaches and whether they are a viable alternative to traditional software development.

Keywords: Low-Code · Model-Driven · Development

1 Introduction

In this paper, we compare Low-Code Platforms to Model-Driven Architecture and try to determine when and where to use them over Model-Driven Architecture or vice-versa.

In order to make clear to the reader the difference between the two approaches we firstly provide definitions and characteristics of each paradigm in the following sections.

1.1 What is Model-Driven Architecture?

With Model-Driven Architecture or Model-Driven Engineering, the goal is to standardise on models in a given domain. This is done in order to reduce code duplication and speed up development. Especially when bootstrapping a new project, it is helpful keep boilerplate code to a minimum. In addition to reducing development time, Model-Driven Engineering also tries to increase productivity by maximising compatibility between systems by using the aforementioned standardised models. [1] Generally, Model-Driven Architectures are geared towards developers, i.e. people who also have a good understanding of the underlying programming language(s) of the respective architecture. One common way Model-Driven Engineering is used in practice is code generation. Using code generation can speed up many processes: For example, given an API specification (e.g. using the industry standard OpenAPI [2]), one can generate all code that is necessary for a client for this API. Swagger [3] is one such tool which can generate API clients for many different programming languages. There are also tools which can generate a complete class hierarchy from an UML diagram; many IDEs offer such functionality either natively or via plug-ins. These are only a few examples of what falls into the category of Model-Driven Architecture, the main point is that all of them help developers reach their goal faster than doing the same tasks manually.

1.2 What is a Low-Code Platform?

A low-code platform, as the name implies, requires a minimal amount of code to be written. Similarly to conventional programming (i.e. using a programming language), an IDE is used for creating programs using low-code platforms. Often, the IDE is part of the low-code platform itself instead of a standalone program as is the case for conventional programming. In most cases, the IDE of the platform and the low-code platform cannot be used separately so the term low-code platform is also commonly used to refer to the IDE interface itself. Low-code platforms are targeted towards users with little to no programming experience, which means the actual programming of a low-code system is done mainly via visual building blocks. These building blocks contain pre-built templates and functionality and can be combined to form a coherent user interface. Oftentimes, custom functionality can be implemented using a click-and-drag pattern between building blocks and by assigning an action accordingly. The available building blocks vary by platform which makes them less portable than a system written using a programming language. This is one common point of critique on low-code platforms.

Another sub-category of low-code platforms are so called no-code platforms [4]. Where to draw the line between low-code and no-code is not clearly defined, especially since low-code platforms can sometimes be used in a way that cannot be differentiated from no-code platforms. For the purposes of this paper, low-code platforms are general purpose visual programming languages [5] aimed at both developer and non-developer users while no-code platforms are closer to software as a service (SaaS) platforms targeted mostly (or only) towards non-developers. Another way to look at it is that low-code platforms still give developers the freedom to add custom building blocks for their specific use case and no-code solutions are usually purpose-made for a specific line of business or a specific type of department. Some examples include monday.com [6], a no-code platform for project management tasks; FileMaker [7], which provides a database engine and a corresponding no-code platform for creating graphical user interfaces for interacting with databases; and Airtable [8], a no-code platform for creating databases connected to interactive spreadsheets.

2 Evaluation of Low-Code Tools

2.1 Selection of Low-Code Tools

Before evaluating low-code tools, we had to first determine which tools to evaluate. Given the ever evolving landscape of low-code platforms, there are countless companies offering low-code solutions. The following is an incomplete list of low-code platforms we found:

– Oracle APEX (Application Express) (originally released as Oracle Flows in 2000) by Oracle (founded in 1977)
– OutSystems (founded in 2001)
– Mendix (founded in 2005), a subsidiary of Siemens
– Simplifier (founded in 2012 as iTiZZiMO, renamed 2019 to Simplifier, the same name as its low-code platform)
– OSBP (Open Standard Business Platform), an open-source low-code platform (developed since 2016) under the umbrella of the Eclipse Foundation
– Corteza Low Code, part of the open-source Corteza Project (released in 2019), originally developed by Crust Technology

We wanted our evaluation to contain at least one open-source platform (e.g. OSBP), one platform backed by a well-known corporation (e.g. Oracle), one platform by a relatively new/unknown corporation (e.g. Simplifier), one relatively old/well-established platform (e.g. Oracle APEX or OutSystems) and one relatively new platform (e.g. Simplifier or Corteza). With the given list of low-code platforms, we could satisfy this requirement.

To ensure we could properly test all of these tools, we conducted a brief first inspection of each. Therefore, the following section will give a brief explanation of each platform, any problems encountered during the initial setup and the steps necessary before being able to use the low-code platform.

OSBP (Open Standard Business Platform) The Open Standard Business Platform is an open-source development environment which combines no-code and low-code with traditional application development. [9] It is the community version of OS.bee [10], a commercial product developed by COMPEX only available to business users. The interface for the OSBP platform is implemented via a plug-in for the Eclipse IDE. The user first needs to install the Eclipse IDE, after which they can add the link to the plug-in repository and install it directly from within the IDE. Unfortunately, the latest version of OSBP was released more than one year ago and is incompatible with the latest version of the Eclipse IDE. At the time of writing (Jan 2, 2021), the last commit in the OSBP Git repository was on May 9, 2019. The latest version of the Eclipse IDE was 2020-12. The OSBP website recommended downloading Eclipse Neon, a version of the Eclipse IDE released in 2016. A non-functioning IDE plug-in for an outdated version of said IDE which has not been updated in over a year is a red flag. We therefore deemed OSBP unsuitable and decided to exclude it from further evaluation.

Corteza Corteza Low Code is part of the Corteza Project [11] initiated by Crust Technology. The Corteza Project encompasses Corteza CRM Suite, an open-source customer relationship management tool built entirely on Corteza Low Code; Corteza Messaging, a messaging solution similar to the likes of Slack; Corteza One, a unified workspace from which to access and run web applications and finally Corteza Low Code, a low-code platform designed for building record-based applications. It is possible to try out Corteza directly in the browser by creating an account or by using an existing GitHub or Google account to sign up. We chose to set up a local demo deployment using Docker. The Corteza Project provides very good documentation on how to get started, including a ready-made `docker-compose.yml` file containing all necessary services for a full Corteza installation. After a simple call to `docker-compose up -d`, the Corteza server was running locally. The last step before being ready to use was to set up an administrator account in the local Corteza instance.

3 Evaluation of Model-Driven Tools

4 Findings

5 Conclusion

References

[1] Wikipedia contributors. *Model-driven engineering.* 2020. URL: `https://en.wikipedia.org/w/index.php?title=Model-driven_engineering&oldid=995747770` (visited on 12/27/2020).

[2] *OpenAPI Initiative.* 2020. URL: `https://www.openapis.org` (visited on 12/27/2020).

[3] *Swagger.* 2020. URL: `https://swagger.io` (visited on 12/27/2020).

[4] Wikipedia contributors. *No-code development platform.* 2021. URL: `https://en.wikipedia.org/w/index.php?title=No-code_development_platform&oldid=996724223` (visited on 01/01/2020).

[5] Wikipedia contributors. *Visual programming language.* 2021. URL: `https://en.wikipedia.org/w/index.php?title=Visual_programming_language&oldid=991997087` (visited on 01/01/2021).

[6] Wikipedia contributors. *monday.com.* 2021. URL: `https://en.wikipedia.org/w/index.php?title=Monday.com&oldid=993059669` (visited on 01/01/2021).

[7] Wikipedia contributors. *FileMaker.* 2021. URL: `https://en.wikipedia.org/w/index.php?title=FileMaker&oldid=991237143` (visited on 01/01/2021).

[8] Wikipedia contributors. *Airtable.* 2021. URL: `https://en.wikipedia.org/w/index.php?title=Airtable&oldid=992831018` (visited on 01/01/2021).

[9] *OSBP (Open Standard Business Platform).* 2021. URL: `https://www.eclipse.org/osbp/index.html` (visited on 01/02/2020).

[10] *OS.bee.* 2021. URL: `https://www.osbee.org/home/` (visited on 01/02/2020).

[11] *Corteza Project.* 2021. URL: `https://cortezaproject.org` (visited on 01/02/2020).