Low-Code vs. Model-Driven Architecture

Markus Reiter

January 12, 2021

supervised by Prof. Dr. Ruth Breu

Outline

- Motivation
- Model-Driven Architecture
- Low-Code Architecture
- Criticisms
- Planned Project Procedure

Motivation

- What are the advantages and disadvantages of low-code tools do?
- Are they a viable alternative to model-driven or traditional development?
- When to choose one approach over the other?

Model-Driven Architecture

- provides a set of guidelines for the structuring of specifications
- standardise on models in a given domain to reduce code duplication and speed up development
- code (fully or partially) generated from models, e.g. from UML diagrams
- aimed at developers who have good understanding of underlying programming languages

Model-Driven Architecture

- Example: Swagger
  - API specification given in OpenAPI format
  - API client is generated for the specified programming language
  - support for new languages/frameworks can be added by implementing a new generator
  - very easy to provide clients for many languages with virtually no development effort

Low-Code Architecture

- provides pre-built application components
- graphical user interface for creating both the application logic as well as the user interface
- typically aimed at end-users rather than developers

Criticisms

- Model-Driven Architecture
  - UML diagrams lack details included in the code itself.
  - “the Code is the design” - Should models be derived from code instead of code from models?
- Low-Code Architecture
  - Unsuitable for implementing scalable and mission-critical applications.
  - Increase in unsupported applications built by shadow IT, i.e. applications which are not controlled by a company’s IT department.
- Do these approaches actually make development easier and cheaper?

Evaluation of Low-Code Tools

- Find low-code tools in the following categories:
  - open-source
  - developed by well-known company
  - developed by unknown company
  - old/well-established platform
  - new/unestablished
- Set up each tool
- Build a test application (TODO List) with each tool

Open Standard Business Platform (OSBP)

- open-source
- plug-in for the Eclipse IDE developed since 2016
- community version of the commercial OS.bee product developed by COMPEX
- latest version over one year old
- does not work with latest version of the Eclipse IDE

Corteza Low Code

- open-source
- part of the Corteza Project initiated by Crust Technology in 2019
- the Corteza Project includes a CRM solution built on top of Corteza Low Code, among other things
- web-based platform
- test by signing up with a GitHub or Google account or by deploying it locally using Docker

Oracle APEX (Application Express)

- commercial
- initially released as Oracle Flows in 2000
- web-based platform
- test by signing up for an Oracle Cloud account or by requesting an APEX workspace

Simplifier

- commercial
- initially released by iTiZZiMO in 2012
- web-based platform
- test by using the Simplifier Playground (data is wiped every day) or by requesting a Simplifier test instance

Mendix

- commercial
- founded in 2005 as a subsidiary of Siemens
- web-based platform (Mendix Studio) and Windows application (Mendix Studio Pro) with advanced features
- test by signing up for a regular account which allows hosting unlimited applications (with 1GB of memory and 0.5GB of storage per application)