

Low-Code vs. Model-Driven Architecture

Markus Reiter

January 12, 2021

supervised by Prof. Dr. Ruth Breu

- Motivation
- Model-Driven Architecture
- Low-Code Architecture
- Criticisms
- Evaluation of Low-Code Tools
- Findings & Future Work
- Conclusion

- 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

- 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

- made for building record-based management applications
- process for building the test application:
 1. create application namespace
 2. create module for TODO List records, specifying the necessary fields (status, title, body)
 3. create page and add a list block linked to the module

Corteza Low Code: Editor

Corteza

Messaging

CRM Suite

+

CRM Modules Pages Charts

Lenny Horstink

Public pages

Create a new page:

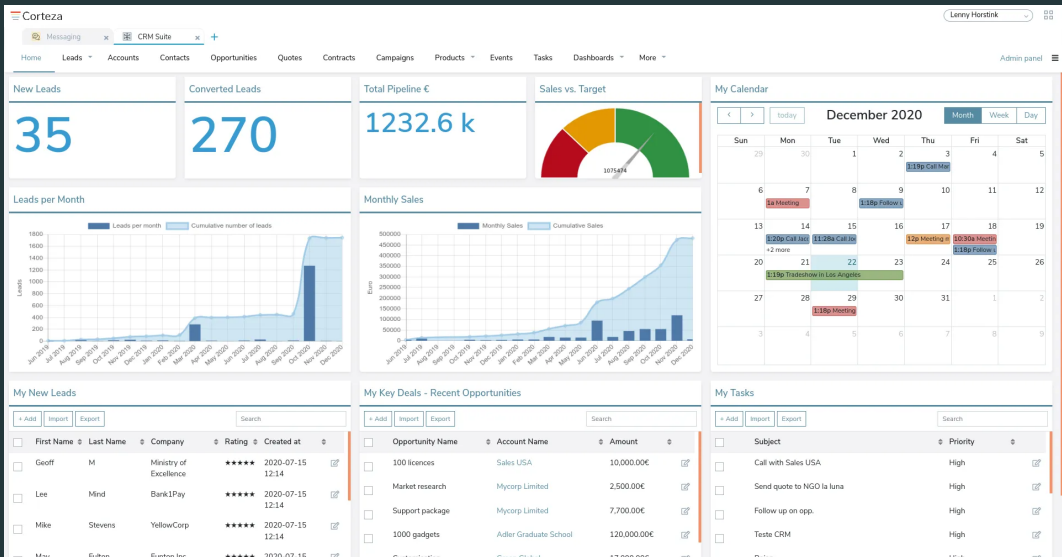
Page title

Create

List of pages

Home	Visible	Page builder	
Leads	Visible	Page builder	
Record page for module "Lead"	Record page for Lead	Page builder	
All New Leads	Visible	Page builder	
My Leads	Visible	Page builder	
Accounts	Visible	Page builder	
Record page for module "Account"	Record page for Account	Page builder	
Record page for module "AccountContactRole"	Record page for AccountContactRole	Page builder	
Case Studies	Not visible	Page builder	
Record page for module "CaseStudies"	Record page for CaseStudies	Page builder	
Contacts	Visible	Page builder	
Record page for module "Contact"	Record page for Contact	Page builder	
Opportunities	Visible	Page builder	
Record page for module "Opportunity"	Record page for Opportunity	Page builder	
Record page for module "OpportunityLineItem"	Record page for OpportunityLineItem	Page builder	
Record page for module "OpportunityCompetitor"	Record page for OpportunityCompetitor	Page builder	
Record page for module "OpportunityContactRole"	Record page for OpportunityContactRole	Page builder	
Record page for module "Offer files"	Record page for	Page builder	

Corteza Low Code: Application



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

Oracle APEX (Application Express)

- process for building the test application:
 1. create database table for TODO List items
(requires basic SQL knowledge)
 2. create new blank application
 3. add list view to home page and select
the corresponding database table
 4. add new form page (dialog style) and select
the corresponding database table
 5. add a button to the home page that opens the form page
 6. add a dynamic action that refreshes the list view
when the form dialog is closed

Oracle APEX: Page Designer

The screenshot displays the Oracle APEX Page Designer interface for 'Application 81611 \ Page Designer'. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'App Gallery'. A green status bar at the top right indicates 'Changes saved' and shows the user 'Markus R.'. The left sidebar shows the page structure for 'Page 1: Home', including 'Pre-Rendering', 'Regions' (Content Body, Columns, Saved Reports, Region Buttons, Dynamic Actions, New task - Dialog closed), and 'Post-Rendering'. The main canvas shows a page layout with a 'Home' header, 'PAGE HEADER', 'PAGE NAVIGATION', 'BREADCRUMB BAR', 'BEFORE CONTENT BODY', 'CONTENT BODY', and 'FOOTER'. The 'CONTENT BODY' contains a 'Todo List' region with 'PREVIOUS', 'ITEMS', and 'REGION CONTENT' sections. A 'CREATE' button is visible in the 'REGION CONTENT' section. The right sidebar shows the 'Button' configuration for the 'CREATE' button, including 'Identification' (Button Name: CREATE, Label: New task), 'Layout' (Sequence: 10, Region: Todo List, Button Position: Right of Interactive Report Search Bar), and 'Appearance' (Button Template: Text, Hot: Yes, Template Options: Use Template Defaults, CSS Classes, Icon). The bottom of the interface shows a 'Regions' tab with a grid of available components: Breadcrumb, Calendar, Cards, Chart, Classic Report, Column Toggle Report, Faceted Search, Form, Help Text, Interactive Grid, Interactive Report, List, List View, PL/SQL Dynamic Content, Reflow Report, Region Display Selector, Static Content, Tree, and URL.

Oracle APEX: Application

≡

TODO List

me@reitermark.us

Q

Go

Actions

New task

Status	Title	Body
Done	Write paper	
Done	Create presentation	
Outstanding	Proof-read paper	

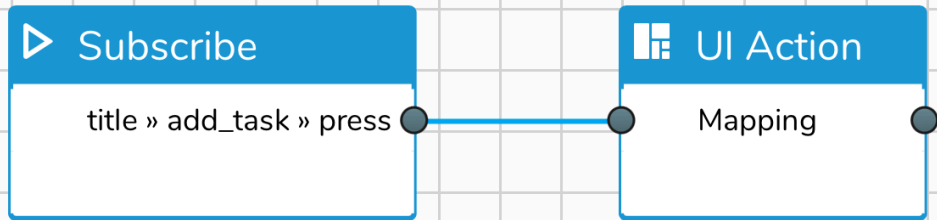
1 - 3



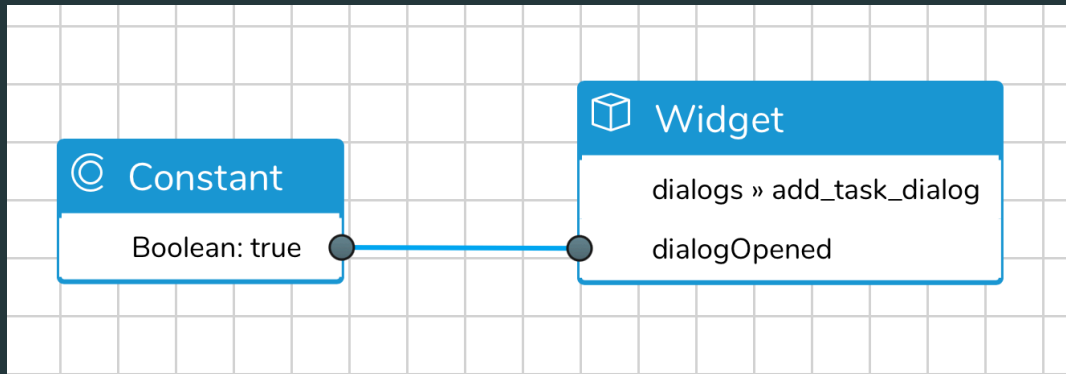
- commercial
- initially released by iTiZZiMO in 2012
- web-based platform
- test by using the Simplifier Playground (data is wiped every day) or by requesting a Simplifier test instance

- process for building the test application:
 1. create database connector (SQLite)
 2. create database schema for TODO List items and deploy to connector
 3. add list view and button to home page
 4. add new page containing a form with input fields and button
 5. create processes for
 - loading items into list view
 - submitting the form page
 - opening the form page with the button

Simplifier: Process



Simplifier: Process



Simplifier: Application Editor

The screenshot displays the Simplifier Application Editor interface for an application named "Application ITIZ_Template_SQL_ShoppingList". The interface is divided into several sections:

- Top Bar:** Includes the Simplifier logo, the title "Application Editor", a user profile "demo demo", and a "Verlassen" (Logout) button.
- Navigation Tabs:** "Designer" (active), "Prozess", "Data Workbench", "Tests", and "Andere".
- Left Panel:**
 - Screens:** A list of screens including "Start Screen", "Splash", "Main" (selected), "Dialogs", and "Messages".
 - Tree View:** A hierarchical structure of the application components, including "custom-Header", "subHeader", "ScreenContent", "Main_Table_Items_Products", "Main_ColumnListItem_Products", "Main_Toolbar_Products", "Main_Column_Product", and "Main_Column_Amount".
- Center Canvas:** A visual representation of the "Main" screen. It features a dark header bar, a search bar, a "Product" dropdown, and a table with columns "Product" and "Amount". The table currently displays "No data".
- Right Panel:** A properties panel for the selected "Main" screen, showing sections for "General", "Appearance", "Data", "Misc", and "Others".

The "Appearance" section on the right includes toggle switches for "showFooter", "showHeader", "showNavBarButton", and "showSubHeader". The "Data" section includes a "title" field. The "Misc" section includes a "navBarButtonTooltip" field. The "Others" section includes a "usedAsTitle" toggle switch.

Simplifier: Application

Main

Product

Amount

Bread

2

Apple Juice

3

Tooth Paste

1

- 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)

- process for building the test application:
 1. create blank application
 2. add list view and button to home page, which prompts the user to select or create a new data source
 3. create data source for TODO List items
 4. add new form page linked to the corresponding data source
 5. add button to home page that opens the form page

Mendix Studio

The screenshot displays the Mendix Studio interface for a project named 'home'. The main canvas shows a 'TODO List' application. At the top left, there is a 'New task' button. Below it, a list of tasks is displayed, each with a placeholder for a title, status, and body. The right sidebar is open to the 'Properties' panel for the selected 'BUTTON' widget. The 'On Click Action' section shows a 'Create Object' action for the 'task' entity. The 'Caption' is set to 'New task'. The 'Icon' is set to 'Select icon'. The 'Render Mode' is set to 'Button'. The 'Style' is set to 'Primary'. The 'Conditional Visibility' section is also visible.

TODO List

Button task

New task

{title}
{status}
{body}

{title}
{status}
{body}

{title}
{status}
{body}

{title}
{status}
{body}

{title}
{status}
{body}

Properties

BUTTON

Events

On Click Action

Nothing Page Microflow More

Create Object

Entity

task (MyFirstModule)

Page

edit_task (MyFirstModule)

General

Caption

New task

Icon

Select icon

Render Mode

Button

Style

Primary

Conditional Visibility

Attribute-Based

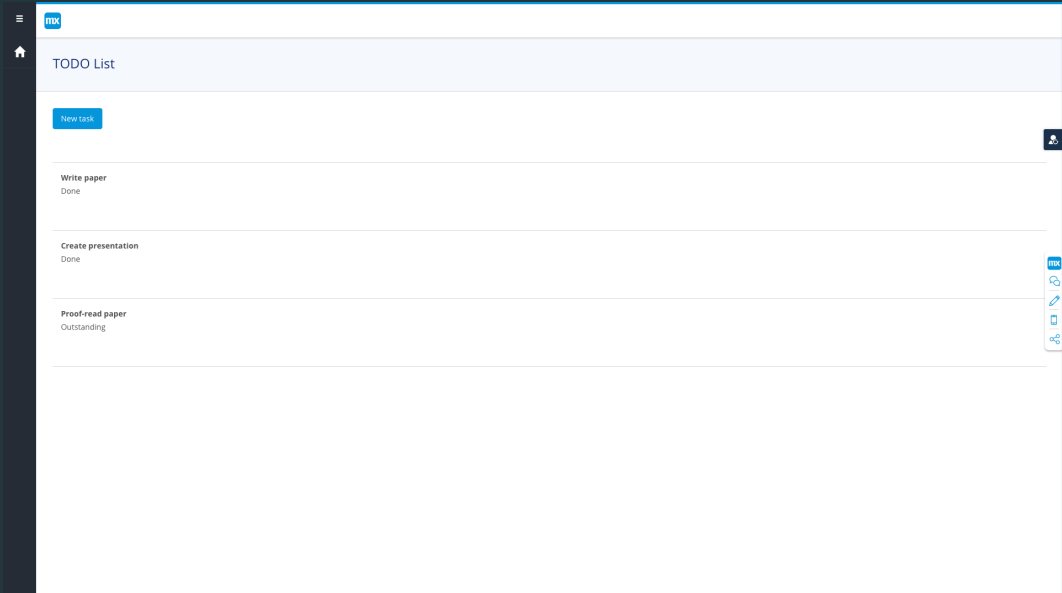
Select attribute

To use an attribute, move the widget into a data view or a list view.

Wrap with a new data view

Delete

Mendix: Application



Unique Features

- **Oracle APEX**
 - Creation of applications from existing data, e.g. CSV files
- **Mendix**
 - AI-assisted wizard for creating custom workflows (microflows)
 - publishing as native mobile applications for iOS and Android

Assessment of Low-Code Tools

- ease of use
- customisability
- portability
- scalability
- suitability for mission-critical applications

Assessment of Low-Code Tools

	OSBP	Corteza	APEX	Simplifier	Mendix
ease of use	N/A	high	medium	medium	high
customisability	N/A	low	medium	medium	high
portability	N/A	medium	low	medium	medium
scalability	N/A	medium ¹	high	high	high
mission-criticality	low	high	high	medium ²	high

¹medium in general, high for certain types of applications, e.g. management applications

²medium only due to the problems encountered, high otherwise

Findings & Future Work

- very small number of open-source low-code tools
 - assumption: open-source community mostly consists of developers, so there is no need/demand for low-code platforms
 - future work: investigate low-code other types of platforms and compare their presence in the open-source vs. the commercial space
- more streamlined user interface in the more mature products like Oracle APEX and Mendix
- lack of portability is a valid concern for low-code tools, stored data may be the only element of a platform that is portable by using

Conclusion

- low-code platforms are a valid alternative to model-driven development
- low-code platforms are not as flexible and limited in the ways they can be extended
- choice between low-code and model-driven architecture:
 - highly dependent on which low-code platform is used, similar to choosing a programming language or framework for a particular task
 - highly dependent on the application requirements, low-code well-suited for management applications (e.g. process management, CRM)

Questions?