Low-code development

Thiago Rocha Silva (trsi@mmmi.sdu.dk)
Associate Professor

What's happening now?

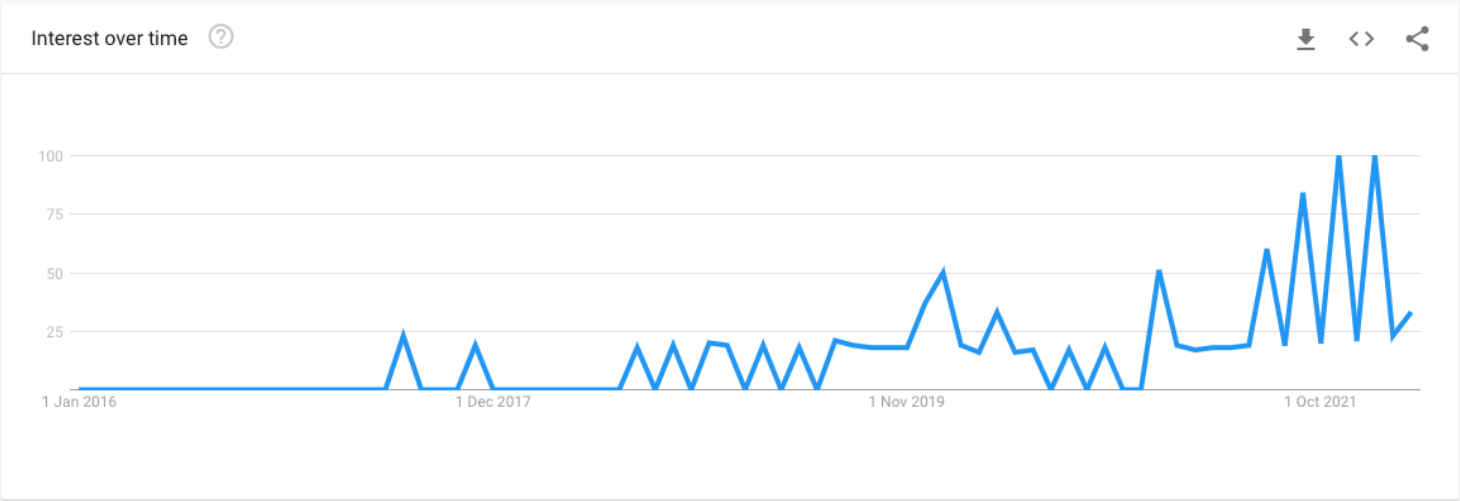
- While the demand for software systems is **exploding**, there is a severe **shortage of software developers** to cope with this demand, and it **won't become any better** in the near future[1].
- There is a **digital native workforce**:
  - **Computer literacy** has **improved dramatically** over the last years.
  - The **basics of computer programming** are now taught in many countries as part of **compulsory education**.
- **Cloud-based services** became a reality (SaaS).
- **Training** is nowadays widely available on YouTube and other online media.
- **Collaborative development** is now supported by a wide range of tools (CSCW).
- Hiring **professional devs** has become **more and more expensive**, especially for SMEs.

[1] Breaux, T., Moritz, J. (2021). The 2021 software developer shortage is coming. Commun. ACM 64(7), 39–41.

What if “citizen developers”[1] could program the software they need?

[1] *The term refers to ordinary citizens, subject-matter experts, etc. who* ***do not know how to program***

Low-Code Development: Google Trends



Low-Code Development

Forbes

Jul 20, 2017, 01:20pm EDT

The Low-Code/No-Code Movement: More Disruptive Than You Realize

Jason Bloomberg Former Contributor

Enterpr

I write a

Subject Matter First

A Manifesto

Subject matter experts, or SMEs, own the knowledge and expertise that is the backbone of software and the foundation of digitalization. But too often this rich expertise is not captured in a structured way and gets lost when translating it for software developers who then analyze, interpret and understand it before writing code. With the rate of change increasing, time-to-market shortening and product variability blooming, this approach is increasingly untenable. It causes delays, quality problems and frustration for everybody involved. We advocate for adopting a mindset that puts subject SMEs directly in control of "their" part of the software and lets developers focus on their core skill, software engineering. Here is how we achieve it:

Empower subject matter experts to capture, understand, reason about data, structures, rules, behaviors and other forms of domain knowledge in a precise and unambiguous form

by providing them with tailored software tools that allow them to directly edit, validate, simulate and test that knowledge

rather than expecting experts to juggle complex subject matter in their minds, Word documents, user stories and other generic, non-optimized tools.

MSCA Marie Skłodowska-Curie Actions

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement n° 813884.

lowcomote

Low-code development platforms allow non-programmers to build full applications by interacting through dynamic graphical user interfaces, visual diagrams and declarative languages.

the potential to

er the different

Springer

Call for Papers

Software and Systems Modeling

Theme Section:

Modeling in Low-Code Development Platforms

IMT Atlantique Bretagne-Pays de la Loire École Mines-Télécom · University of York · UAM Universidad Autónoma de Madrid · Università degli Studi dell'Aquila · JKU Johannes Kepler Universität Linz · BT · intecs Solutions · UGROUND · CLAIS · IncQueryLabs · SPARX Systems · METADEV · The Open Group · aws

Low-Code Development: Projections

- According to Gartner[1], by 2024, low-code application development will be responsible for **more than 65%** of the application development activity.
- Currently, estimates[2] point out that the low-code app development market is roughly around **$10 billion** globally.
- Microsoft PowerApps sees at least **7.4 million new builds** on its platform **every single month**.

[1] Vincent, P., Iijima, K., Driver, M., Jason, W., Natis, Y. (2016). Magic Quadrant for Enterprise Low-Code Application Platforms. Gartner.
[2] https://www.infoworld.com/article/3694173/why-generative-ai-will-turbocharge-low-code-and-no-code-development.html

Low-Code Development Platforms

K2
kissflow
TRACKVIA
appian
PRONTOFORMS
ninox
Betty Blocks
outsystems
DWKIT
aura quantic
NEWGEN
kintone
MICROPACT
amazon Honeycode BETA
VINYL
quixy
AppSheet
Bonitasoft
Retool
GeneXus SOFTWARE THAT MAKES SOFTWARE
agilepoint
Creatio
salesforce
PEGA
Claris
mx mendix
Microsoft PowerApps
servicenow
ZOHO
SwiftUI
MATS Easier done than said
LANSA An Idera, Inc. Company
unqork
nintex
wavemaker
caspio
bizagi
quickbase
ORACLE APEX
SDU

Low-Code Development Platforms (LCDPs)

- **Definition**:

*"Platforms that enable **rapid delivery** of **business applications** with a **minimum of hand-coding** and minimal upfront investment in setup, training, and deployment"*

Forrester (2014)

Richardson, C., Rymer, J.: New Development Platforms Emerge for Customer-Facing Applications. Forrester Research, Cambridge (2014).

Low-Code Development Platforms (LCDPs)

- **Definition**:

*"Products and/or cloud services for **application development** that employ **visual, declarative techniques instead of programming** and are available to customers at low- or no-cost in money and training time to begin, with costs rising in proportion of the business value of the platforms"*

Forrester (2017)

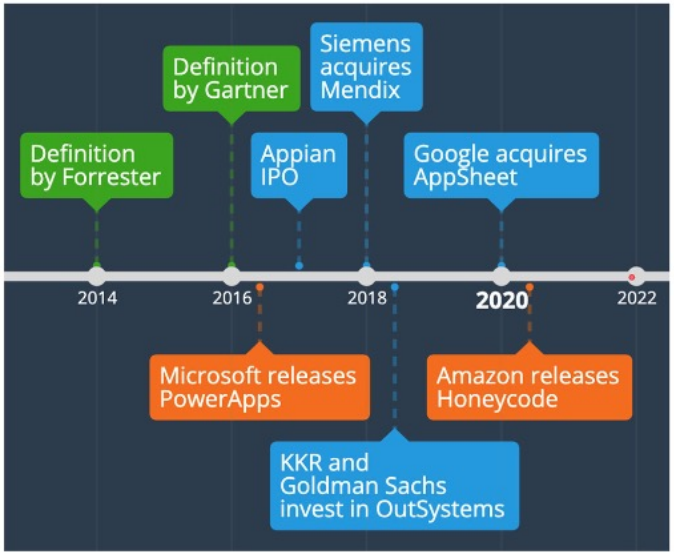
Rymer, J.: The Forrester Wave: Low-Code Development Platforms For AD&D Pros, Q4 2017. Forrester Research, Cambridge (2016)

Low-Code *vs* No-Code

- **No-code** development platform (NCDP) is a related term used for platforms that **eliminate the need for programming** using **visual languages**, **graphical user interfaces**, and **configuration**.
- The term is widely used in marketing, but **do not** clearly represent an specific market segment[1].

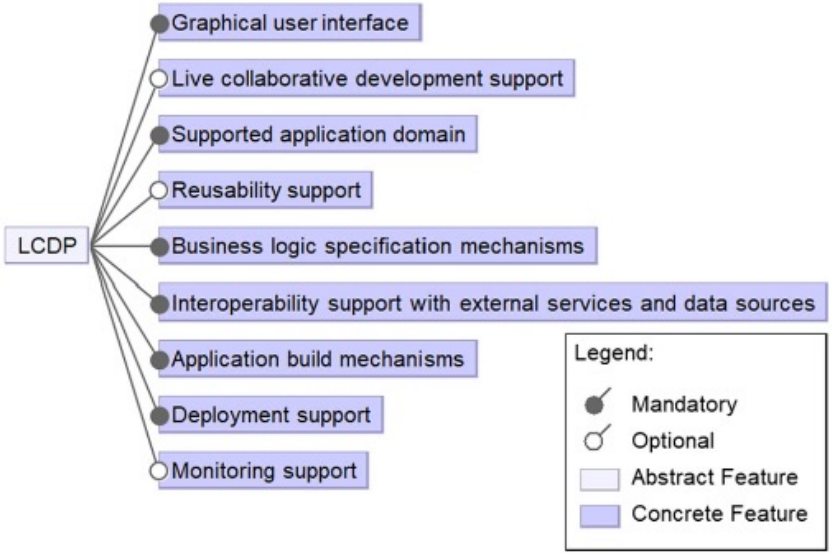
[1]Rymer, J., Koplowitz, R.: Now Tech: Rapid App Delivery, Q1 2019. Forrester Research, Cambridge (2019)

Low-Code Development Platforms (LCDPs)



Major events in low-code history (Di Ruscio et al., 2022).

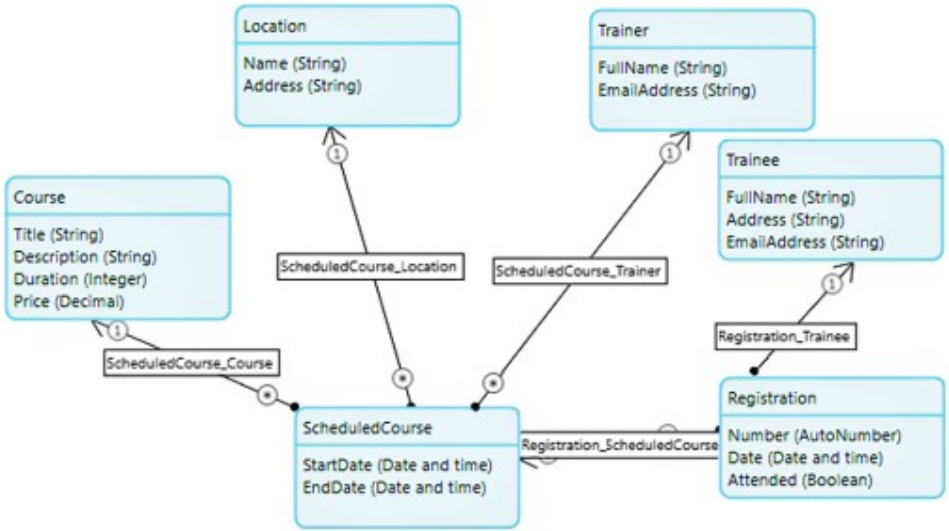
Top-level features of LCDPs



Source: Di Ruscio et al. (2022)

LCDPs: Tool-supported steps

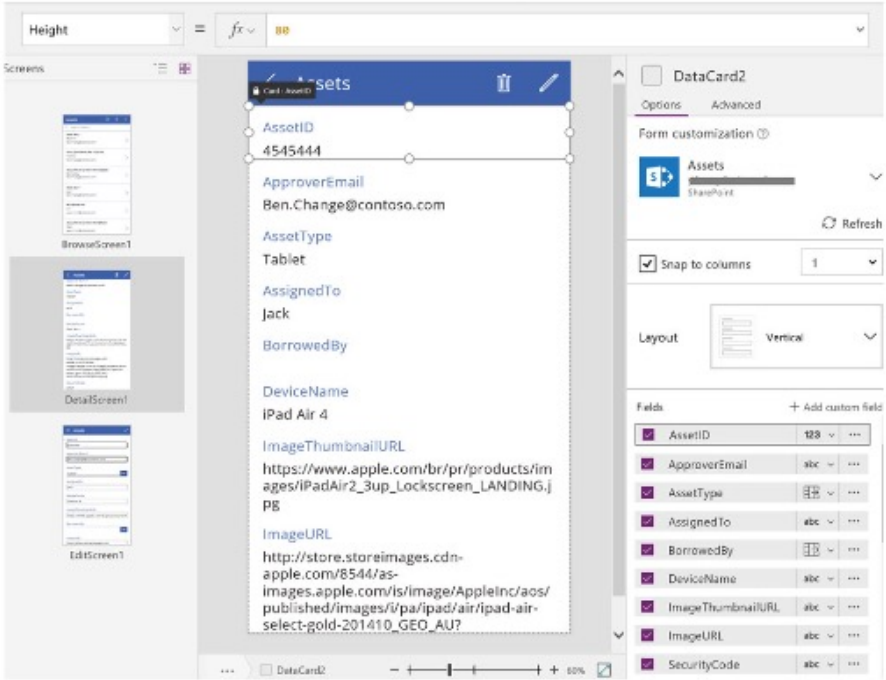
- **Domain modelling**:
  - ❖ Users are provided with modelling constructs to represent **concepts and relationships** underpinning the application being developed.



A simple domain model specified in **Mendix** (Sahay et al., 2020).

LCDPs: Tool-supported steps

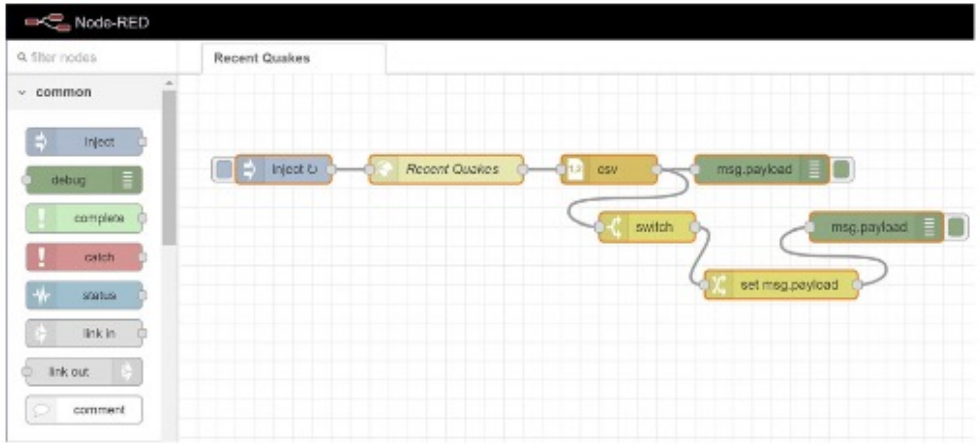
- **User interface definition**:
  - ❖ Users define **data forms and pages** to create, edit, and visualize data that the application under development will manage.



User interface definition with **Microsoft PowerApps**.

LCDPs: Tool-supported steps

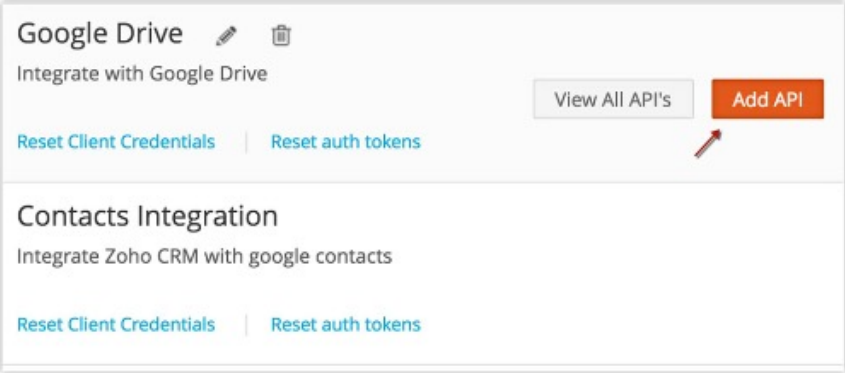
- **Business logic specification**:
  - ❖ Users define the **control and data flows** of the system under development through business logic specification mechanisms such as **graphical workflows** and **textual business rules**.



Business logic specification with **Node-RED**.

LCDPs: Tool-supported steps

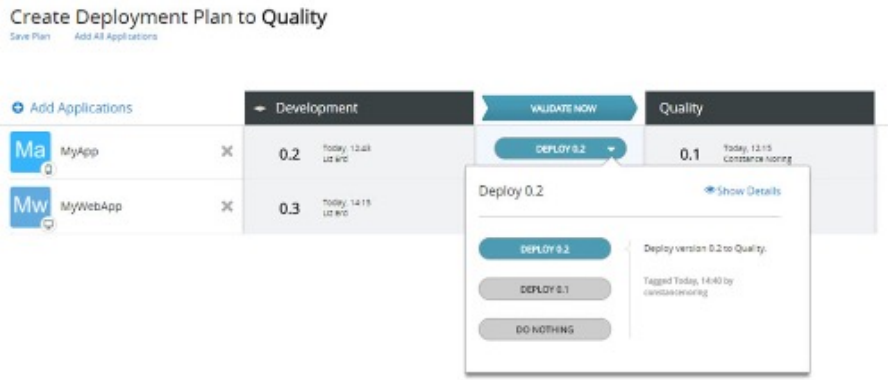
- **Integration with external services**:
  - ❖ LCDPs typically provide **interoperability support** with external services and data sources to **use services or consume data** provided by third-party systems, e.g., using dedicated APIs.



Configuring the Google Drive connector in **Zoho Creator**.

LCDPs: Tool-supported steps

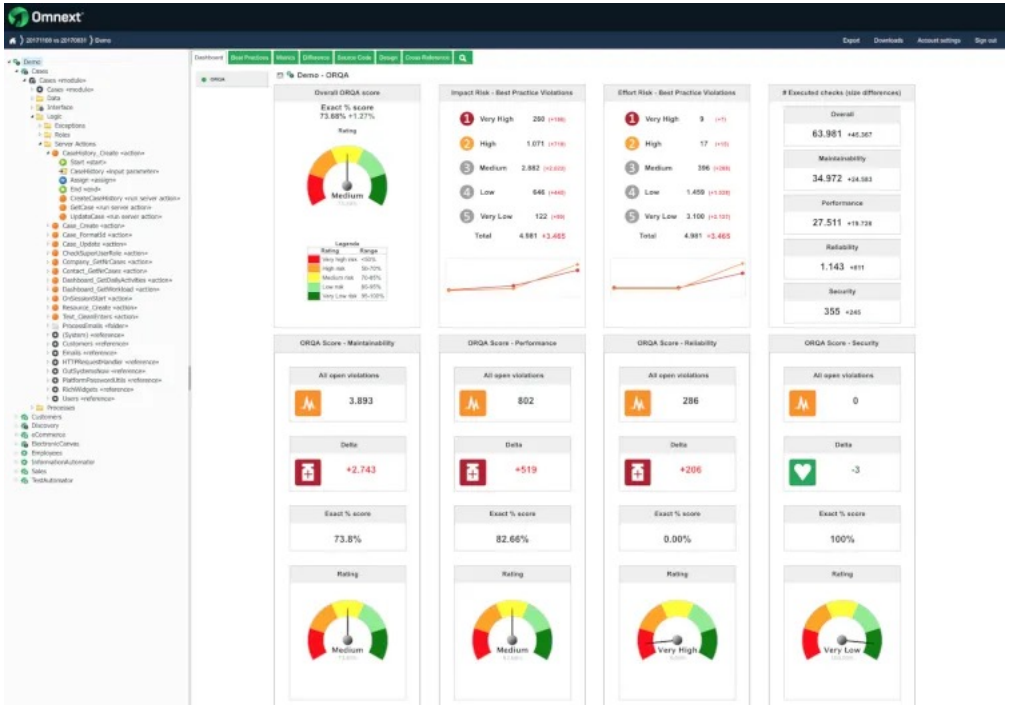
- **Application generation and deployment**:
  - ❖ LCDPs **generates and deploys** the modelled application by means of provided application build mechanisms. Deployments are typically done on **cloud infrastructures**.



Application deployment with **OutSystems**.

LCDPs: Tool-supported steps

- **Application maintenance**:
  - ❖ LCDPs provide mechanisms to **monitor and maintain** the developed system by means of dedicated features, e.g., to react in case of **unforeseen requirements** that need to be addressed or **fix issues** that might occur during the operation of the system.



Dashboard with **OutSystems**.

Typical features in LCDPs

Source: Sahay et al. (2020)

Feature	OutSystems	Mendix	Zoho Creator	MS PowerApp	Google App Maker	Kissflow	Salesforce App Cloud	Appian
Graphical user interface								
Drag-and-drop designer	✓	✓	✓		✓	✓	✓	✓
Point and click approach				✓				
Pre-built forms/reports	✓	✓	✓	✓	✓	✓	✓	✓
Pre-built dashboards	✓		✓	✓		✓	✓	
Forms			✓	✓				
Progress tracking	✓	✓	✓	✓	✓	✓	✓	✓
Advanced reporting						✓		
Built-in workflows			✓			✓	✓	
Configurable workflows			✓			✓	✓	
Interoperability support								
Interoperability with external service	✓	✓	✓	✓		✓	✓	✓
Connection with data sources	✓	✓	✓	✓	✓	✓	✓	✓
Security Support								
Application security	✓	✓	✓	✓	✓	✓	✓	✓
Platform security	✓	✓	✓	✓	✓	✓	✓	✓
Collaborative development support								
Off-line collaboration	✓	✓	✓	✓	✓	✓	✓	✓
On-line collaboration	✓	✓			✓	✓	✓	✓
Reusability support								
Built-in workflows			✓			✓	✓	
Pre-built forms/reports	✓	✓	✓	✓	✓	✓	✓	✓
Pre-built dashboards	✓		✓	✓		✓	✓	
Scalability								
Scalability on number of users	✓	✓	✓	✓	✓	✓	✓	✓
Scalability on data traffic	✓	✓	✓	✓	✓	✓	✓	
Scalability on data storage	✓	✓	✓	✓	✓	✓	✓	
Business logic specification mechanisms								
Business rules engine	✓	✓	✓	✓	✓	✓	✓	✓
Graphical workflow editor	✓	✓				✓	✓	
AI enabled business logic	✓					✓	✓	✓
Application build mechanisms								
Code generation	✓							
Models at run-time		✓	✓	✓	✓	✓	✓	✓
Deployment support								
Deployment on cloud	✓	✓	✓	✓	✓	✓	✓	✓
Deployment on local infrastructures	✓	✓					✓	✓
Kinds of supported applications								
Event monitoring	✓	✓	✓	✓	✓	✓	✓	✓
Process automation	✓		✓	✓	✓	✓		✓
Approval process control					✓			
Escalation management						✓		
Inventory management	✓	✓	✓	✓	✓	✓	✓	✓
Quality management		✓	✓	✓	✓	✓	✓	✓
Workflow management	✓	✓	✓	✓	✓	✓	✓	✓

Typical features in LCDPs

Source: Bock & Frank (2021)

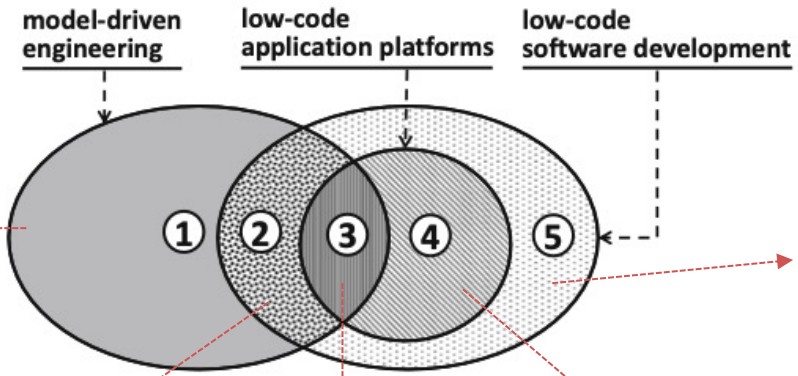
Criterion	LC_1	LC_2	LC_3	LC_4	LC_5	LC_6	LC_7	Overall
Static Perspective								
Mechanisms for data structure definitions	●●●	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Data modeling component	●●●	●●●	●●●	●●●	●●●	●●○	○○○	●●○
Internal databases and persistence mechanisms	●●●	●●●	●●●	●●●	●●●	●●●	○○○	●●●
Access to external data sources (APIs)	●●●	●●●	●●●	●●●	●●●	●●●	●●●	●●●
Data reference models	●●○	●○○	●○○	○○○	●○○	●○○	○○○	●○○
Adaptation mechanisms for data (reference) models	●○○	●○○	●○○	○○○	●○○	●○○	○○○	●○○
Dynamic Perspective								
Mechanisms for program flow specifications	●●○	●●○	●●○	●●○	●●○	●○○	●●●	●●○
Process modeling component	●●○	●●○	●●○	○○○	●●○	●○○	●●●	●●○
Integration with static and functional components and artifacts	●●●	●○○	●●○	●●○	●●○	○○○	●●●	○○○
Process reference models	●○○	●○○	○○○	○○○	●○○	○○○	○○○	●○○
Adaptation mechanisms for process (reference) models	●○○	●○○	●○○	●○○	●○○	●○○	●○○	●○○
Functional Perspective								
Mechanisms for functional specifications	●●○	●●○	●●○	●●○	●●○	○○○	●●○	●●○
Functional modeling component	○○○	○○○	○○○	○○○	○○○	○○○	○○○	○○○
Generic functional reference specifications	●●●	●●●	●●●	●●○	●●●	●○○	●●○	●●○
Domain-specific functional reference specifications	●●○	●○○	●○○	○○○	●○○	○○○	○○○	●○○
Adaptation mechanisms for functional (reference) specifications	●○○	●○○	●○○	●○○	●○○	●○○	●○○	●○○
GUI Design								
GUI design component	●●●	●●●	●●●	●●●	●●●	●○○	●●●	●●●
Graphical GUI editor	●●●	●●●	●●●	●●●	●●●	●○○	●●●	●●●
Automatic generation of GUIs from data structures	○○○	●○○	●●○	○○○	●●●	●○○	●●○	●●○
GUI reference models	●●○	●○○	●○○	●○○	●○○	●○○	●○○	●○○
Roles and Users								
Specification mechanisms for roles and users	●●●	●●●	●●●	●○○	●●●	●●○	●●○	●●○
Modeling component for roles and users	○○○	●○○	○○○	○○○	○○○	○○○	○○○	○○○
Artificial Intelligence								
Internal artificial intelligence components	●●○	●○○	●●○	○○○	●●○	○○○	○○○	●○○
Integrability of external artificial intelligence services	●●○	●○○	●●○	○○○	●○○	○○○	○○○	●○○

Explanation: ○○○ = not or weakly addressed; ●○○ = partly addressed; ●●○ = well addressed; ●●● = extensively addressed.

LC1: Microsoft PowerApps, **LC2**: Mendix, **LC3**: Appian, **LC4**: Wavemaker, **LC5**: Pega, **LC6**: Quickbase, **LC7**: Bonita Platform

Low-Code *vs* Model-Driven Engineering

Source: Adapted from Di Ruscio et al. (2022).



Approaches that use **models** as machine-processable artefacts but **do not aim at reducing the amount of code**.

Approaches that **cannot be considered model-driven**, but **aim to reduce the amount of code** required to implement a system **without offering deployment or lifecycle management capabilities**.

Approaches that use **models** as machine-processable artefacts and **aim to reduce the amount of code** required to implement a system (e.g. via code generation or interpretation) but **without offering deployment or life-cycle management capabilities**.

Platforms that use **models** to facilitate the development of software applications **with reduced code** and offer **built-in deployment and lifecycle management features**.

Platforms that **cannot be considered model-driven**, but facilitate the development of software applications **with reduced code** and **built-in deployment and lifecycle management features**.

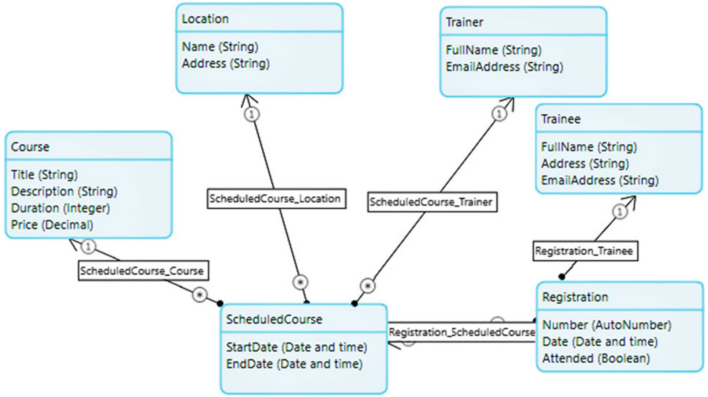
Low-code development:
Challenges and Opportunities

Limitations and challenges of low-code

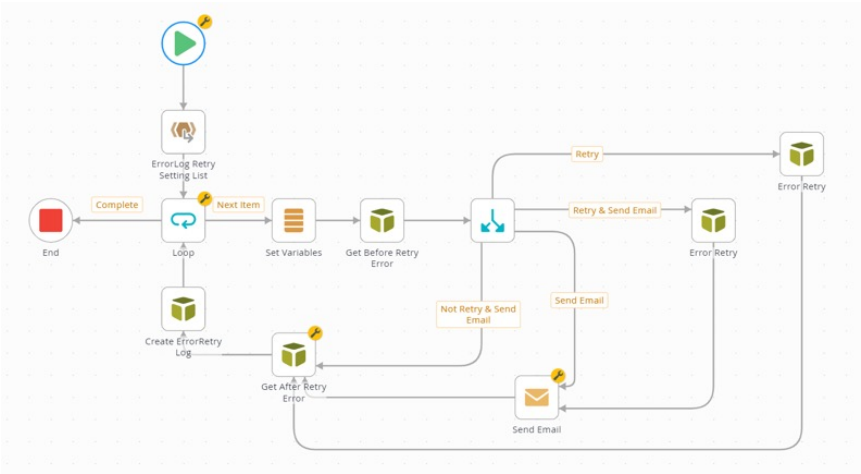
Practitioners' Perspective on Stack Overflow and Reddit (Luo et al., 2021).

Limitations and challenges of LCD	Example	Count
High learning curve	*you need to learn a lot about how this tool works to do the thing you're trying to do*	21
High pricing	*These larger vendors can get expensive, because they charge you for every user and you have to buy packages of 50 or 100 users*	13
Lack of customization	*Restrictive customisation on design and layouts*	11
Slow loading and publishing	*Loading speeds can be slow*	9
Less powerful than programming	*A full-fledged programming language will always have more power than a "no-code/low-code" solution such as PowerApps*	6
High complexity	*they're often too convoluted to use*	6
Complex issues still need coding	*If you go further and having a complex issue that can only be solved with invoking code or creating custom activities, you really need to code*	5
No access of source code	*Therefore you cannot take the code and use it elsewhere*	4
Not really ease of use	*No code is great, but not as easy as picking an app that's already written*	4
Limitation to experienced developers	*Most no-code tools are designed more like a prototyping tool and also targeted for non-developers which makes it very difficult for someone with development background to use*	4
Vendor lock-in	*Then there's the issue of vendor lock in. If you build using a nocode tool and they host etc. then if they raise their prices or shut down, that's going to a huge cost in downtime or rebuild and possibly lost data*	3
Difficulty of maintenance and debugging	*An additional risk is the continued support and maintenance of the low-code platform*	3
Difficulty of integration	*it looks to be a hard problem to make the UI, data store and calculations work together*	3
Unfriendly user experience	*it has a steeper and at times user unfriendly UX*	2
Need of basic programming knowledge	*most of them do require code at some point*	2

Issue: Model-based development



Domain model specification in Mendix



Workflow specification for error handling in Nintex K2

- **Proprietary notations** that are **not well-suited** for end users.
- Vendor **lock-in**.

What if we could use a language that's more familiar to "citizen developers"?

Scenario-Based Specifications

- **Scenarios in natural language** have been used for a long time in software development to specify user requirements.
- Scenarios also sound **natural** to ordinary people since we’re used to explain situations using **examples** in real life.
- Specifications in **free natural language**, however, have **many issues** such as ambiguity, verbosity, low testability, low maintainability, etc.
- **Behaviour-Driven Development (BDD)** scenarios allow for a good **balance** between **natural language** and **formalism**.

A Domain-Specific Language (DSL) for BDD

- BDD scenarios specified as **state machines**.
- Both **domain** and **behavioural** specifications.
- Support for **different abstraction levels** (declarative and imperative).

```
model <model name>

entity <entity name> {
  actions: <list of actions>
  states: <list of states>
  properties: <list of properties>
}
```

```
Scenario: <scenario name>
Given <initial state>
    which means
        Given <initial state>
        When <action>
        Then <resultant state>
When <action>
    which means
        Given <initial state>
        When <action>
        Then <resultant state>
Then <resultant state>
    which means
        Given <initial state>
        When <action>
        Then <resultant state>
```

BDD Scenarios: Domain-Specific Language (DSL)

*sample.bdd

```
model sample using widgets

entity account {
    actions: withdraw, transfer
    states: void
    properties: balance
}

Scenario: Transferring money - Transaction succeeded
Given the balance of the account "1787461" is 1000
    which means
        Given the BrowserWindow "Account Balance" is displayed
        When I click on "See the balance"
        Then the value of the TextField "Balance" is "1000"
When I transfer 1000 from the account "1787461"
    which means
        Given the BrowserWindow "SEPA Transfer" is displayed
        When I type "1000" into the field "Amount"
        Then the Notification "Your transfer has been successfully processed" is displayed
Then the account "1787461" is void
    which means
        Given the BrowserWindow "Account Balance" is_
```

- available
- disabled
- displayed
- enabled
- shown
- not

StateName **available**

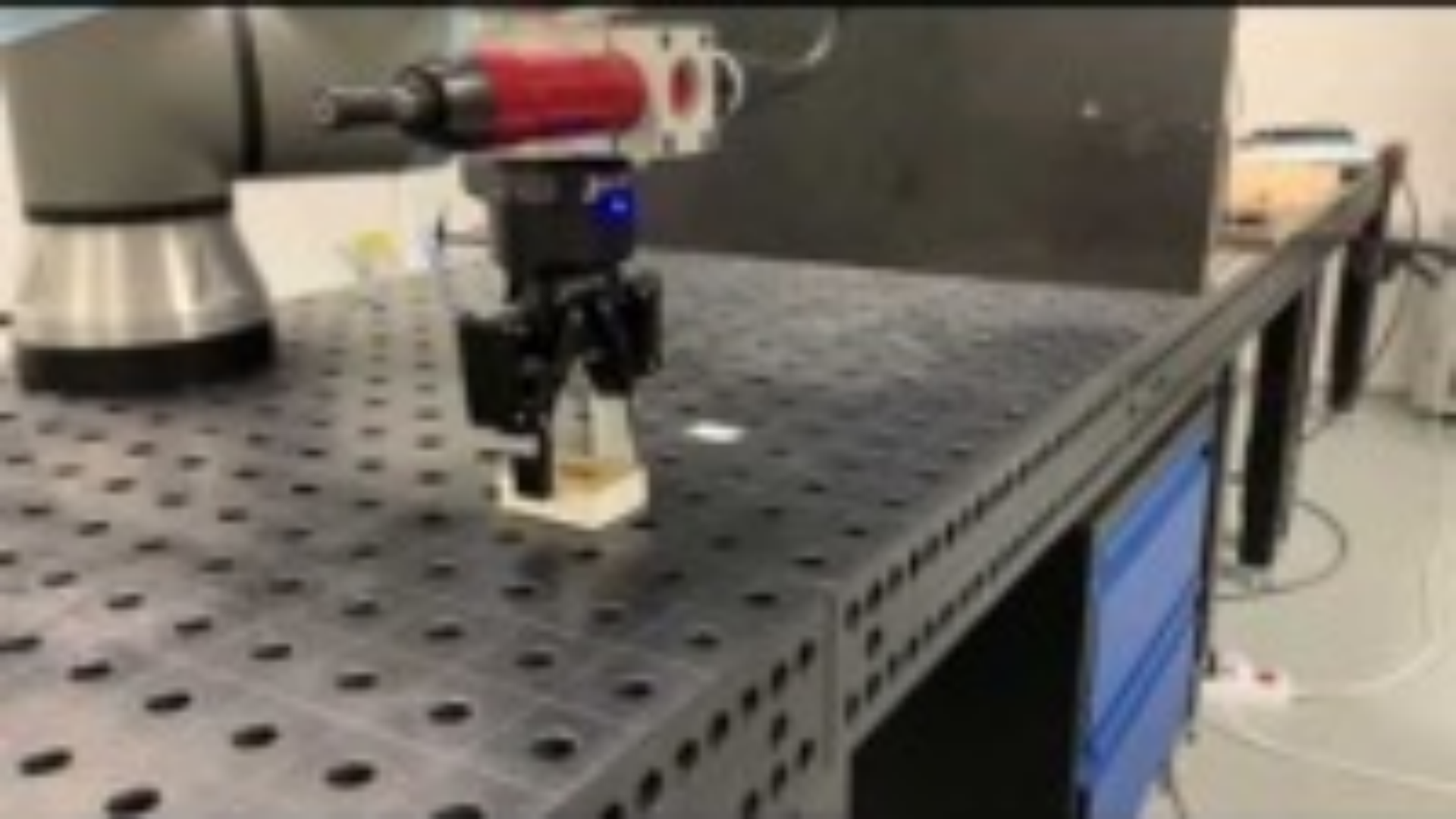
Press 'F2' for focus

Editor for BDD Scenarios

Issue: Focus on application software

- They have been primarily focusing on **application software** for business processes automation of **low-to-moderate complexity**.
- This kind of application is considered *easy* to develop since they consist of **software-only solutions** that basically rely on manipulation of data forms.

Can scenario-based specifications play a key role in low-code development of cyber-physical systems?



RoboTrainer.bdd

```
model sample
using robotrainer_domain

device robotrainer_one is robot{
    initial state: pulling softly
}

entity hand is body_part

Scenario: "Top to medium"
Given the hand is at the "top" position
And the robotrainer_one is pulling softly
When the hand moves down to the "medium" position
Then the robotrainer_one is pulling strongly

Scenario: "Medium to bottom"
Given the hand is at the "medium" position
And the robotrainer_one is pulling strongly
When the hand moves down to the "bottom" position
Then the robotrainer_one is pulling softly

Scenario: "bottom to medium"
Given the hand is at the "bottom" position
And the robotrainer_one is pulling softly
When the hand moves up to the "medium" position
Then the robotrainer_one is pulling strongly

Scenario: "medium to top"
Given the hand is at the "medium" position
And the robotrainer_one is pulling strongly
When the hand moves up to the "top" position
Then the robotrainer_one is pulling softly
```

Synchronize | Git Staging | Git Reflog | Properties | RoboTrainer

Rt at localhost

RoboTrainer

Low-code development

Thiago Rocha Silva (trsi@mmmi.sdu.dk)
Associate Professor