Reinventing Low-Code: Value-Driven and Learning-Oriented Low-Code Development with SLLM-Integrated Approach

Gayane Sedrakyan[1], Stephan Braams[1,2], Cosmin Ghiauru[1], Anton Tsankov[1], Stijn Schuurman[1], Matthijs Jansen op de Haar[1], Valeri Andreev[1] and Jos van Hillegersberg[1]

[1]*Department High-Tech Business and Entrepreneurship (HBE) Section, Industrial Engineering and Business Information Systems (IEBIS), University of Twente, Enschede, Overijssel, Netherlands*

[2]*Cape Groep, Enschede, Overijssel, Netherlands*

Keywords: Low-Code, Model-Based System Development, Business Models, Business Model Canvas, AI-Enabled Modeling, Requirements Engineering for Low-Code, Citizen Development.

Abstract: Low-code development platforms (LCDPs) are transforming business practices by shifting the focus from traditional, code-intensive approaches to business-centered modeling. These platforms enable citizen developers - non-technical employees within organizations - to build and manage applications that address specific business needs. This democratization accelerates time-to-market and encourages agile, co-participatory development. However, the rise of citizen development also introduces challenges, such as risks to quality, security, and governance, due to limited technical expertise among some users. This paper investigates ways to enhance current low-code practices by integrating AI-based support for text-to-model generation and established business frameworks, such as the Business Model Canvas (BMC). Incorporating BMC into low-code platforms reinforces their core strengths - minimizing code dependency while grounding development in business models. This integration can offer a structured pathway for citizen developers to engage in meaningful learning while ensuring their projects align with organizational objectives. This approach positions low-code not only as a productivity tool aiming faster time to market, but as platforms for continuous learning and strategic alignment with business. The proposed integrations build on a novel feedback-inclusive approach, which received the innovative feedback nomination at the University of Leuven, Belgium[1], and was informed by evidence-based learning experiences at the University of Twente, Netherlands.

1 INTRODUCTION

The rapid evolution of digital transformation in business has paved the way for innovative approaches in software development, among which low-code development platforms (LCDPs) have emerged as transformative tools. By reducing reliance on traditional, code-intensive development, LCDPs enable a shift toward business-centered modeling, making application creation more accessible. These platforms enable re-profiling business developers into application developers and empower *"citizen developers" - non-technical employees* – to actively participate directly in the process of building and managing software solutions and prototypes that meet specific business needs. This democratization of development not only accelerates time-to-market but also fosters collaborative, agile processes that engage a broader range of stakeholders.

Low-code development is a *model-based system development approach* that emphasizes creating applications through *enhanced business modeling* and *reduced reliance on extensive hand-coding*.

In its current form, these platforms offer limited capabilities for understanding business contexts, focusing primarily on modeling from requirements and generating prototypes that can assist in improving domain knowledge, modeling and modeling language knowledge that are involved in the development process with a specific platform. Nevertheless, a

[1] https://www.kuleuven.be/onderwijs/prijs-onderwijsraad/2014-2015/Process-oriented-feedback

Sedrakyan, G., Braams, S., Ghiauru, C., Tsankov, A., Schuurman, S., op de Haar, M. J., Andreev, V. and van Hillegersberg, J.
Reinventing Low-Code: Value-Driven and Learning-Oriented Low-Code Development with SLLM-Integrated Approach.
DOI: 10.5220/0013348200003896
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 13th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2025)*, pages 420-431
ISBN: 978-989-758-729-0; ISSN: 2184-4348
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

substantial amount of tacit knowledge often remains uncaptured during the transformation of requirements into models. Enhancing LCDPs with capabilities such as Text-To-Model Assistance and integration of Business Model Canvas (BMC) into the LC modeling phase offers a structured framework to effectively capture and integrate tacit business insights. This ensures a more comprehensive representation of business needs while fostering stronger alignment between development outputs and strategic business objectives, ultimately delivering unique / refined business values. Additionally, this approach aligns with the model-based development nature of the low-code development approach. The integration of AI-based LLMs further advances these capabilities by enabling more dynamic and context-aware modeling processes.

In this work, we examine the capabilities and advantages of incorporating the BMC into the LCD framework and analyze their impact through comparative studies. Additionally, integrating overall modeling support such as text-to-model assistance is analyzed within the LCD context.

By leveraging the synergy between data, behavior models and BMC, this research *redefines LCDPs as more than mere productivity tools for rapid development*. It positions them as *platforms for continuous learning and strategic alignment, enabling organisations to derive sustained business value and expand development capacity through citizen-led initiatives*. This integration creates a structured, value-driven and learning-oriented pathway that enhances the transformative potential of LCDPs by leveraging their core strengths - business-centric modeling and the involvement non-technical developers, such as citizen developers. Additionally, this approach enriches the learning context for novice and junior (business) developers, fostering a deeper understanding of the principles underlying low-code development and its capacity to generate business value, while simultaneously enabling practical experience and skill development through LCDPs.

This research aims to answer the questions:

- *RQ1: If and how an AI-based text-to-model assistance can contribute to the knowledge enhancement for LCD?*
- *RQ2: If and how the integration of Business Model Canvas into the Low-Code modeling cycle enhances knowledge on capturing added business value ?*

This paper is organized as follows: Chapter 2 provides an overview of LCDPs, including their theoretical background, typologies, commonly used models, and an analysis of their benefits and limitations. Chapter 3 examines the integration of learning support within the low-code development lifecycle, incorporating a text-to-model approach and the concept of business value into the modeling cycle through generative AI. Chapter 4 presents the results of case studies comparing the quality of low-code models and applications developed with unassisted and AI-assisted cycles. It also discusses the findings and highlights the limitations of the study. Finally, Chapter 5 concludes by evaluating the impact of the integration of AI-based support described in the study on enhancing low-code models to support for improved knowledge of LCD developers with particular focus on novice and citizen developers, along with recommendations for future research.

2 RELATED WORK

Low-code development platforms enable users to create applications with minimal hand-coding (Sedrakyan & Snoeck, 2013), relying on visual modeling, drag-and-drop interfaces, and pre-built components. This aligns with the principles of **Model-Based Systems Engineering (MBSE)**, where models represent system architecture and behavior (Di Ruscio et. al, 2022). In the context of low-code, the visual models serve as both the blueprint and executable logic, offering a unified framework for system design. This chapter is structured as follows: (1) a summary of the theoretical foundations of low-code development, (2) an exploration of low-code model types and notations commonly used in development processes, and (3) support for learning context, and (4) a description and a discussion of the Business Model Canvas and its capabilities.

2.1 LCD Definitions and Typology

LCD is characterized by its use of visual, declarative techniques and minimal hand-coding, offering two primary approaches:

- Descriptive low-code;
- Prescriptive (composable) low-code;
- No-Code development;
- AI-Enhanced Low-Code;
- Hybrid approaches.

The **descriptive low-code** approach allows developers to visually design the structure and functionality of applications, often leveraging graphical interfaces and drag-and-drop components to represent software elements. Mendix is an example of such descriptive LC platform (Henkel & Stirna, 2010). In contrast, the **prescriptive (or composable)**

low-code approach focuses on rapidly assembling applications using pre-built components and templates, reducing the need for custom coding. Novulo is an example of a composable LCDP[2]. **No-code development** approach uses a subset of LCD designed exclusively for non-technical users, enabling application development without any coding. These platforms rely entirely on pre-defined templates, visual interfaces, drag-and-drop and pre-built components with automated workflows, making them ideal for citizen development initiatives. AI-enabled LC combines the principles of low-code with AI-based tools, such as large language models (LLMs), to support more dynamic development processes. These platforms can assist in generating models, code, or application logic based on natural language descriptions or domain-specific inputs, significantly reducing the cognitive load on developers. **Hybrid approaches** combine the strengths of low-code platforms with traditional high-code (HC) approaches, enabling developers to leverage the best potential of both domains. This hybrid approach is becoming increasingly popular as it addresses the limitations of pure low-code or high-code systems, allowing benefiting from the speed and simplicity of low-code platforms while retaining the flexibility and precision of high-code development. There are two main approaches for LC and HC integrations:

- ***Extending Low-Code with High-Code***: Low-code platforms are often limited in their ability to handle highly customized features or integrations with complex systems. By integrating high-code, developers can extend the functionality of low-code applications, such as implementing custom algorithms, integrating advanced APIs, or designing unique user interfaces that go beyond the capabilities of visual tools. As an example, a low-code CRM system could be extended with high-code to include a custom AI-based recommendation engine for personalized sales strategies.
- ***Using Low-Code in High-Code Applications***: Conversely, low-code components can be embedded within high-code applications to accelerate development cycles for less critical or repetitive modules. For instance, low-code tools can be used to quickly prototype dashboards, automate workflows, or build internal tools, reducing the burden on high-code developers. As an example, a high-code e-commerce platform could incorporate a low-code module to manage and automate inventory processes, allowing business analysts to make changes without requiring in-depth programming expertise.

The user base for low-code platforms is diverse, including **business analysts / developers**, **experienced software engineers**, **citizen developers**. Among typical business use cases for low-code applications are:

- Rapid Prototyping: Quickly testing ideas and concepts to validate them before significant investment.
- Legacy System Modernization: Addressing outdated systems with scalable, flexible solutions.
- Process Automation: Streamlining repetitive tasks and workflows.
- Citizen Development Initiatives: Empowering non-technical users to participate in and/or build applications tailored to departmental needs, reducing IT workloads.

2.2 Theoretical Background of LCD

The theoretical foundation of low-code traditionally includes **model-driven engineering (MDE)**, providing a framework for generating software applications based on highly abstracted models and (meta)model-to-(meta)model transformations (Sedrakyan & Snoeck, 2014). MDE principles guide the creation of reusable models and transformations that capture domain-specific knowledge and facilitate code generation with minimal manual effort. Descriptive low code leverages MDE to generate software applications based on modeled data, behavior, and user interfaces visually. Prescriptive low code, focused on modular design and component reuse, aligns with MDE principles through domain-specific modeling languages (DSMLs) for modeling LC applications. Rather than relying on structural and behavioral models of a system, this approach focuses on integrating reusable components through models of reusable components that allow "stitching" these components together to enable the rapid assembly of applications with enhanced consistency, reusability, and scalability.

2.3 Common Model Types and Notations Used in LCD

The development lifecycle of the current descriptive low-code (DLC) approach, which will be the focus of

[2] https://www.novulo.com/

this research, is depicted in Figure 1. It typically begins with capturing the data requirements of the information system, followed by designing workflows and defining detailed functionalities.

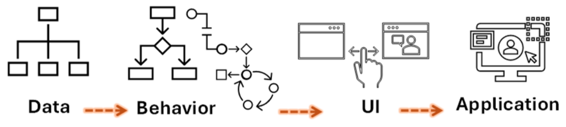

Figure 1: Common Modeling Workflow in Descriptive Low-Code Development Platforms.

Next, the user interface design and interactions are established, culminating in the generation of a fully functional application.

LCDPs can employ a variety of model types and notations to facilitate visual application design.

These include models for structured representations of application components, behaviors, and interactions:

- **Structural Models** define the underlying data structures and relationships within the application. Commonly used notations include:
  - *UML Class Diagrams* to represent *Domain Models* through data structure representing entities, attributes, and their relationships (e.g. database schema).
  - *Entity-Relationship Diagrams* depicting data relationships in database-centric designs.
- **Behavioral Models** focus on the dynamic aspects of the application, such as workflows and event-driven processes. Notations often include:
  - *BPMN (Business Process Model and Notation)* visualizing business workflows and decision points.
  - *Activity Diagrams* allowing visualize tasks, conditions, and parallel processes to define user interactions and business logic visually, e.g. handling data validation or conditional routing.
  - *UML State Diagrams (or Statecharts)* capturing state transitions based on user actions or system events, e.g. handling state changes for a ticket system (e.g., "Open", "In Progress", "Closed").
  - *Use Case Diagrams* defining user roles and interactions to support user stories, e.g. mapping out user roles like "Admin" and "Customer" and their interactions with the app.
- **User Interface Models** describe the design and interaction patterns of the application's front-end, often including:
  - *UI Flow Diagrams* illustrating user navigation paths with a high-level overview of how users navigate through the application, including pathways between pages, screens, or modal windows.
  - *UML State Diagrams* depict how the application responds to user input by transitioning between different states. For instance, a login form might transition to a "logged in" state upon successful authentication.
  - *Wireframes and Mockups* allowing sketching the layout and navigation.
  - *Drag-and-Drop Interfaces* are widely adopted in LCDPs, often with "what-you-see-is-what-you-get" (WYSIWYG) editors that allow LC developers to design interfaces visually, reducing the need for manual coding expertise.
  - *Hybrid approaches* combine models are often used in combination to map out the flow of interactions between the user and the application, aiming to capture how different elements of the interface respond to user actions.
- **Component Models** are used in prescriptive low-code approaches to define reusable building blocks using module libraries encapsuling pre-built functionalities and services.

2.4 Transformative Benefits and Drawbacks of LCD

LCDPs represent a paradigm shift in software development, offering significant opportunities for organizations to enhance efficiency, reduce costs, and increase adaptability. By abstracting the complexity of traditional development processes through models, LCDPs enable broader participation in software creation and facilitate rapid prototyping and deployment. However, their adoption introduces certain technical and organizational challenges that necessitate critical evaluation (Rokis & Kirikova, 2022). This section outlines the benefits and challenges of LCD, emphasizing their implications for business value and operational dynamics.

LCD offers ***transformative benefits*** for software development by enabling faster, more adaptive processes that align with evolving business needs.

Although not exhaustive, the following benefits highlight key advantages:

- **Cost Efficiency and Democratized Development:** LC approach empower non-technical personnel, or citizen developers, to participate in application development, reducing reliance on specialized expertise. This democratization not only lowers development costs but also expands organizational capacity for addressing diverse business needs through software solutions.
- **Accelerated Time-to-Market:** LC approach leverages visual modeling and reusable components to significantly reduce development timelines. This capability supports rapid prototyping and the iterative refinement of Minimum Viable Products (MVPs), fostering innovation within tight timeframes.
- **Testing Ideas with Uncertain Requirements:** The visual design tools and rapid development cycles of LCDPs facilitate continuous feedback loops, enabling early and frequent incorporation of stakeholder insights. In dynamic environments, where requirements are frequently incomplete or subject to change, LCDPs provide the flexibility needed for iterative development. This adaptability reduces the risks of developing misaligned or obsolete solutions. With this capability LCDPs align with the "Mode 2" of the framework called "BimodalIT" [3], which emphasizes experimentation and responsiveness in contexts where requirements remain fluid (Horlach et al., 2022). By providing flexibility for iterative development, LCDPs are well-suited for dynamic environments where requirements are incomplete or subject to change, supporting organizations in testing ideas and responding to shifting market demands while maintaining stability balanced with experimentation and adaptability.
- **Agile Development and Enhanced Collaboration Across Teams and with End-users / Customers:** By offering a faster development cycles to deliver MVPs, LCD promotes shared understanding between technical teams and business stakeholders. This collaboration bridges the traditional divide between IT and business functions, enabling more precise translation of strategic objectives into functional software, while also integrating end-users into iterative prototyping cycles. This collaborative and agile approach not only improves development efficiency but also enhances customer satisfaction through their involvement in earlier phases of development, continuous feedback and refinement.
- **Legacy System Modernization and fast pluggable services:** Organizations often face challenges in updating legacy systems that are integral to their operations. LCDPs provide scalable solutions that extend the functionality of outdated systems, integrating them with modern software infrastructures and reducing the need for costly overhauls.
- **Maintainability:** LCDPs simplify updates through visual modeling and reusable components, facilitating rapid adaptation to changing requirements with centralized management. For example, modifying a model and re-deploying is typically sufficient to introduce changes to the LC applications. Integrated Continousu Integration and Delivery (CI/CD) pipelines (Humble & Farley, 2010) optimize deployment processes through reduced downtime and enhanced operational agility.
- **Cross-platform Deployment:** LCDPs support responsive designs and native mobile capabilities, allowing applications to function across devices and platforms.

While not exhaustive, the following challenges highlight ***key drawbacks*** with LCD approach:

- **Potential for Errors by non-IT developers:** Citizen developers may lack technical skills, the domain-specific and modeling and modeling language expertise to fully understand or translate complex requirements into accurate models, leading to inconsistencies and errors.
- **Security Vulnerabilities:** The abstraction inherent to LCDPs and lack of technical insight of non-IT developers can obscure critical security considerations, increasing the risk of vulnerabilities in the resulting applications (Sedrakyan at al., 2024).
- **Scalability and Customization Constraints:** While LCDPs excel in rapid and standardized application development, their capacity to address highly customized or large-scale enterprise requirements is often limited,

[3] https://www.gartner.com/en/information-technology/glossary/bimodal

necessitating integration with high-code solutions, increasing complexity.

- **Dependence on Platform Ecosystems:** The. reliance on proprietary tools and environments introduces significant risks, including vendor lock-in, platform discontinuity, or deprecation of features. These risks can restrict long-term strategic flexibility and make migration to alternative solutions costly and challenging.
- **Reliance on Third-party Tools and Cloud Servers:** Many LCDPs depend on third-party integrations, which may use outdated or unsupported versions of libraries, APIs, or plugins. This dependency can introduce vulnerabilities, compatibility issues, or maintenance challenges over time. Additionally, reliance on cloud-based servers can create privacy and security concerns, especially when sensitive data is stored or processed in environments that may not fully comply with regulatory or organizational standards. This can increase the risk of data breaches, unauthorized access, or compliance violations.

3 LEARNING- AND VALUE-ORIENTED ENHANCEMENTS

This chapter delves into the enhancement of the LCD lifecycle with a learning support. It investigates if and how the integration of AI-driven advancements can influence knowledge gaps. It explores how feedback-enabled LCD cycles can refine iterative design processes, ensuring adaptability and continuous improvement. Additionally, the chapter examines the possibilities for transformation of textual inputs into structured data and process models and the incorporation of business value frameworks into the modeling process, demonstrating the potential for AI to bridge knowledge gaps of novices in the context of LC application development and output quality.

3.1 Model-Based Feedback Integration

As models serve as the central engine of LCDPs, the quality of these models is critical for generating effective applications. Modeling, however, is a multifaceted skill that extends beyond theoretical understanding, requiring practical expertise. While LCDPs simplify the learning curve through visual and intuitive interfaces, making them better suited for business and citizen developer profiles, learning processes can be significantly enhanced by feedback-enabled mechanisms. Such mechanisms not only improve the transformation of requirements into functional applications but also foster iterative refinement, enabling trial-and-error cycles that promote deeper learning and improved model accuracy. Since modeling forms the foundation of low-code approaches, embedding learning aids early in the modeling process ensures users develop a stronger grasp of the concepts and techniques.

Existing research underscores the value of ***model-based automated feedback*** in advancing novice modelers' skills across various stages of system design. Feedback mechanisms linking the execution outcomes in a resulted application to the parts of models that cause them have proven effective in refining structural models (Sedrakyan & Snoeck, 2013), e.g., understanding and improving the semantic quality of UML class diagrams (Sedrakyan & Snoeck, 2017), improving behavioral models, e.g., UML statecharts (Sedrakyan et al., 2017), to address event failures, and iteratively guiding user interface design (Ruiz et al., 2015). Furthermore, feedback-driven testing assists in defect detection and validates requirements against model specifications, mitigating risks associated with limited domain or technical knowledge (Sedrakyan & Snoeck, 2014). Additionally, ***modeling behavior patterns*** were found to be linked with modeling process outcomes, suggesting that incorporating ***feedforward*** mechanisms to stimulate effective modeling behaviors has the potential to enhance model quality (Sedrakyan & Snoeck, 2017).

Given the model-centric nature of LCDPs, integrating model-driven feedback mechanisms aligns well with their core objectives and design principles. Such integration has the potential to enhance learning outcomes and development efficiency, expanding the scope of LCDPs to leverage citizen development more effectively in areas previously considered better suited for IT expertise to better respond to the IT talent deficit. Research in educational contexts further supports the effectiveness of model-driven feedback for learners with profiles similar to citizen developers, emphasizing its potential to empower these users and improve their ability to contribute meaningfully to development processes.

3.2 LLM-Based Text-to-Model (T2M) Assistance

In this study, we employ an integrated descriptive Low-Code (LC) and AI-based approach. Descriptive

LC was selected due to its effective alignment with Information Systems data structures and business processes within a modeling context. The Text-to-Model (T2M) approach was implemented using the TeToMo framework (Sedrakyan et al., 2022). Integration was achieved through a ChatGPT (OpenAI, 2024) API connector for Mendix[4]. The query used to interact with ChatGPT is as follows:

```
'Generate MermaidJS code for a '
+replaceAll(toLowerCase(toString($Ch
atGPTRequest/Dropdown)),'_','') + '
+ $ChatGPTRequest/Prompt
```

This query utilizes the selected diagram type from a dropdown menu and constructs a prompt for generating appropriate MermaidJS code. The output diagram is derived from the input text specified in `$ChatGPTRequest/Prompt`. Java code is then used to format the ChatGPT response into usable MermaidJS code. Afterwards a microflow designed in Mendix Modeler, is called to processes user input and format it. Subsequently, a second microflow is invoked to utilize the ChatGPT API and retrieve a response. The resulting response, in JSON format, is processed by Java code to generate the final diagram. This diagram is stored as a new "Diagram" entity and displayed on the user page.

3.3 Integrating Business Model Canvas Using an SLLM Approach

While the quality of models is central for the quality of LCD applications, in the modern competitive landscape, success is not solely measured by technical accuracy between requirements and application design artefacts (models). The ability to create and capture business value has become a critical differentiator. Enhancing LC platforms with feedback systems that can assist with capturing and modeling business value can bridge this gap, empowering citizen developers and novice users to align application development with strategic business objectives. This dual focus on learning and value creation has the potential to position LC platforms as not only productivity tools but also as enablers of innovation and competitive advantage. Enhancing LC models with capabilities for capturing business value in addition can further amplify its learning context, offering developers the tools to bridge technical and business perspectives. This integration could significantly expand the LC platform's role not only as a productivity but also as an educational environment.

The Business Model Canvas (BMC) is a strategic management framework used to visualize, design, and refine business models to capture business value and align operations around value creation. It consists of nine interconnected building blocks: Customer Segments, Value Propositions, Channels, Customer Relationships, Revenue Streams, Key Resources, Key Activities, Key Partnerships, and Cost Structure. These elements collectively represent the logic of how an organization creates, delivers, and captures value. As LCD platforms focus on enhancing business models and reduce manual coding for accelerated application development by empowering citizen developers, integrating business-focused frameworks is crucial to align with this goals. The Business Model Canvas (BMC), with its emphasis on value creation and strategic alignment between organisational objectives and market demands, is particularly well-suited to this integration.

This study introduces the term Specific Large Language Model (SLLM) to describe LLMs tailored for specific domains or tasks. The integrated descriptive modeling with AI-assisted BMC is achieved through a Specific Language Model (SLLM) chatbot designed for integration with the Canvas Learning Management System (CLMS)[5] of the University of Twente (UTwente). The chatbot aims to improve accuracy by also learning from CLMS courses and offer specialised support to students for their tasks that link to teacher-specified resources. The integration is using RAG (Retrieval-Augmented Generation), a technique that combines retrieval of relevant documents from a knowledge base with the generation of responses, enhancing the quality of output in conversational AI systems (Cai et al., 2022).

To ensure that sensitive information, such as student and teacher data, is not transmitted to external AI providers like OpenAI or Meta via internet APIs, the SLLM is hosted in a local Docker container. The system uses the Llama3.1-8b-instruct-q8 (Ollama, n.d.) which is a 8-bit quantized Llama. While more powerful model versions are available, they demand substantially more system resources to improve response precision and accuracy. We have chosen the 8-bit quantized model for its resource efficiency and adequate performance. Python server in FastAPI (Rodríguez, n.d.) is used for routing the requests of the conversational chatbot to the appropriate services.

[4] https://marketplace.mendix.com/link/component/206616

[5] https://www.utwente.nl/en/educational-systems/about-the-applications/canvas/

the */load request* that utilizes the UTwente's Canvas API to gather the required modules, documents and pages from the given course, converts it to text, chunks and embeds the pieces of texts. These embedded chunks are then stored inside the Weaviate vector database [6]. The */status request* is then used to verify whether a database entry exists for a given course, ensuring compatibility of the extension for student use. Subsequently, the */chat_stream request* is employed to handle questions posed by students within a course. This request processes the user's query and its history, reformulating the query into a contextual format to perform a similarity search within the Weaviate database. After receiving the response from Weaviate, the query, along with the retrieved documents, is forwarded to the LLM for further processing and response generation. Many course files on Canvas are in PDF format, and text extraction is natively supported by the Python code on the server. However, for non-native formats such as Microsoft Office files (e.g., Word and PowerPoint), LibreOffice [7], an open-source office software, is employed to ensure compatibility.

Teachers can utilize domain-specific prompts to narrow down the guidance provided by the system, ensuring alignment with course objectives and desired learning outcomes. The resulting SLLM enables students to receive answers not only sourced from the internet but also refined with tailored knowledge, course-specific resources, and contextual information derived from historical learner data, thus supporting more accurate and personalized learning assistance.

3.4 Learning Effects of T2M and BMC Interventions

This section explores the outcomes of two quasi-experimental studies designed to evaluate the impact of different intervention methods on novice learners' ability to develop Low-Code (LC) applications. Unlike traditional experimental designs that typically involve the controlled manipulation of multiple variables to mitigate the influence of unknown factors, the quasi-experimental approach adopted here reflected a more natural and flexible design. This approach aligns with real-world scenarios where controlling all variables is challenging, as in our case, requiring incremental development cycles spread over the course timeframe. Yet, without strict manipulation of multiple variables, the findings provide corrections for which the effects of unknown variables, although important in traditional experimental research, were considered less critical to the outcomes of these studies.

It is important to note that the tools described in Sections 3.2 and 3.3 were not utilized during these quasi-experiments, as they were still under development. Instead, basic AI-based assistance using ChatGPT was employed. The subsequent tool development described earlier was informed by the findings of these quasi-experiments, aligning with the interventions applied and aimed at advancing toward a specialized, integrated, all-in-one environment. This development also incorporated evidence-based perspectives from learning sciences, such as feedback typology and formats, as well as adherence to technological standards, including the safeguarding of learner data privacy.

The quasi-experiments employed randomized student samples and course project topics, examining interventions across two cycles: an unassisted cycle (referred to as the *1st cycle*) and an assisted cycle using ChatGPT-based tools (referred to as the *2nd cycle*). In the first quasi-experiment (referred to as the *1st experiment*), text-to-model assistance was introduced during the 2nd cycle to support the modeling of system structure and behavior within the Mendix modeler. The second quasi-experiment (referred to as the *2nd experiment*) implemented AI-based BMC feedback during the 2nd cycle to enhance application refinement.

The learning context and task descriptions included five cases of similar complexity, which were offered to students to select from and develop into applications as part of their course project tasks. These cases included requirements descriptions that necessitated further elicitation from end-users (e.g., company use case owners or teachers acting as case owners). Examples of the case descriptions included sustainable food recommender applications, fitness recommender apps, and learning support applications.

A total of 22 student outcomes were examined during the academic year 2022–23 (*1st experiment*), and 19 student outcomes during the academic year 2023–24 (*2nd experiment*). The participants represented diverse backgrounds, spanning both technical and non-technical master-level study programs, and ranged in age from 22 to 35. Performance was evaluated using cumulative rubric points for intermediate and final outcomes.

[6] https://weaviate.io/

[7] https://www.libreoffice.org

Table 1: Corrections observed after the assisted cycle in Experiment 1 (CX1, Column 2) and Experiment 2 (CX2, Column 3).

MVP quality assessment metrics*	**CX1**	**CX2**
Capture of business context	*1*	*3*
Customer Journey and Business Processes	*3*	*4*
Added value for business	*1*	*3*
Addresses customer problem	*1*	*4*
Link between business processes and value	*2*	*5*
Data model quality and design choices	*3*	*1*
Actors & Roles	*2*	*3*
Security risks identified and addressed	*1*	*0*
Overall MVP quality (all features included are essential for the business)	*2*	*5*
Validation with users (end-user score)	*1*	*3*
UI (intuitive navigation)	*0*	*0*
Behavior validation for desired and undesired paths	*1*	*0*

* All criteria scores were rounded to the nearest integer

In the *1st experiment*, students were required to transfer their business requirements into LC applications during an unassisted cycle (*1st cycle*). They subsequently utilized a T2M assistance cycle with a ChatGPT version to enhance their models (*2nd cycle*). In the *2nd experiment*, students developed their chosen cases into LC applications during the unassisted cycle and later refined their applications using feedback from intermediate results in a BMC-integrated cycle.

The intermediate (*unassisted*) and final (*assisted*) quality of the developed applications was evaluated using an assessment rubric covering multiple criteria (Table 1, column 1): capturing business context; customer journey and business processes; added value for business; addressing customer pain points; linking business processes to value; data model quality and design choices; actors and roles; identification and mitigation of security risks; overall Minimally Viable Product (MVP) quality (ensuring all features were essential for the business); validation with end-users (denoting customer feedback scores); UI intuitiveness for navigation; and validation of both desired and undesired paths. Application quality was rated by experts on a 0–10 scale, with 0 indicating an unsatisfactory solution and 10 representing an excellent solution meeting all rubric criteria.

In the *1st experiment*, as model quality was a major objective, specific sub-criteria were used to evaluate the data model and design choices. These included the number of correct/incorrect entities, attributes, associations, actors and roles, and activities; semantic quality in terms of accurately reflecting the requirements; and valid desired/undesired paths (e.g., modeled constraints), application of best practice patterns. Both positive and negative corrections were analyzed after the intervention. The *1st experiment* demonstrated a ***positive correction of 1.5*** after employing AI-based T2M assistance (see the average for each graded criterion in Table 1). No substantial negative corrections were observed in any of the rubric criteria or sub-criteria related to model quality. No substantial negative corrections were observed in any of the rubric criteria or sub-criteria related to model quality for the 2nd experiment either.

Normality testing revealed both normal and non-normal distributions in both experiments, which were appropriately assessed before and after the interventions. These findings aligned with the anticipated corrections resulting from the applied interventions.

4 DISCUSSION

This section synthesizes the findings from the two quasi-experimental studies conducted to evaluate the effects of different intervention methods on novice learners' abilities to develop LC applications. The study aimed to assess the impact of AI-based interventions, specifically Text-to-Model assistance using ChatGPT and AI-based BMC creation assistance - on the quality of resulting LC applications. The findings from both experiments underscore the effectiveness of AI-assisted tools in improving the quality of LC applications developed by novices. The positive corrections observed in both experiments align with existing research suggesting that AI-based assistance can provide valuable guidance within LCDPs as learning environments, particularly for tasks that require complex problem-solving such as requirements elicitation and engineering requiring modeling skills. The results are consistent with studies that highlight the potential of assisted modeling environments to improve learning outcomes by offering personalized and context-aware feedback (Sedrakyan & Snoeck, 2016). The lack of negative corrections further suggests that the intervention was beneficial, without introducing any unintended consequences.

The T2M approach exhibited significant potential in assisting students to identify key model elements, such as entities, attributes, relationships, mandatory and optional associations, events, and their sequences, derived from textual requirements. The second experiment, which incorporated AI-based BMC assistance, further demonstrated the effectiveness of AI in supporting novice developers in business-oriented tasks, bridging the gap between requirements modeling and business strategy to create more meaningful (LC) applications. Applications and learning cycles for novice developers were positively evaluated by domain experts, including educators, LC platform vendors, and consultants with extensive experience in LC development. These findings underscore the potential of integrating AI-driven Text-to-Model and Business Model Canvas integration assistance to enhance the functionality and educational value of LC platforms. This enhancement aligns with the fundamental characteristics of LC focusing on increasing the emphasis on business modeling while reducing reliance on manual coding efforts. Furthermore, it reinforces the objectives of LC development to enhance on citizen development and business value generation.

Interestingly, slightly higher corrections were observed among students enrolled in programs lacking prior knowledge of UML and programming, *suggesting that the approach effectively supports the knowledge-building processes of novices.* However, students with technical backgrounds consistently outperformed their non-technical peers even in the AI-assisted cycle. *This observation suggests that, while the LCD approach minimizes the required technical expertise, developers with technical insight still derive additional benefits.*

While the quasi-experimental design provided valuable insights, several limitations should be considered. The study was conducted with a sample of students from various academic backgrounds and years, which, while contributing to the generalizability of the findings, may have influenced the results. Additionally, it is difficult to determine whether the observed improvements were solely due to the interventions or if other factors played a role. Future studies could explore the impact of AI-based interventions in more homogeneous student groups to better isolate the effects of the interventions. Furthermore, experiments involving heterogeneous samples with stricter designs could help measure the effects of compound or unknown variables, thus providing a more comprehensive understanding of the interventions. Testing the effects with larger samples across different academic institutions or even with junior professionals from industry and citizen developers could further enrich the findings.

Furthermore, while the interventions showed positive effects on model quality and application development, it remains unclear whether these improvements translate into long-term learning gains or real-world application success. Longitudinal studies that track the retention of skills and the real-world performance of the applications developed by students would be valuable in assessing the lasting impact of AI-based interventions.

4.1 Considerations

Another important point for discussion is the accuracy of the AI-based assistance used during the experiment, particularly regarding hallucination, a common challenge for generative AI systems. While current results are promising, future implementations can enhance reliability by leveraging Retrieval-Augmented Generation (RAG). RAG reduces hallucinations by grounding responses in course content, ensuring outputs are both accurate and contextually relevant. For instance, prompts within the T2M and BMC systems can be adapted to educational contexts by embedding references to specific course materials or assessment criteria. This approach not only aligns AI assistance with cognitive and socio-cognitive learning theories.

Equally significant is the innovative and sustainable concept of decentralized, which could transform the feedback architecture. This approach envisions deploying SLLMs locally on user devices, leveraging concepts from federated learning. In such a setup, models can be periodically updated and trained locally, with aggregated insights shared back to a central system without transferring raw data. This not only ensures privacy but also minimizes server reliance, promoting scalability and sustainability. Technically, this will involve lightweight, containerized deployments of SLLMs on user machines, with mechanisms for secure synchronization and conflict resolution when aggregating updates. Key considerations include optimizing resource use on user devices, ensuring model coherence across decentralized systems, and maintaining robust encryption to protect sensitive data during synchronization.

5 CONCLUSIONS

The findings of this study demonstrate the potential of AI-based assistance in improving novice learners' abilities to develop Low-Code applications. The Text-to-Model assistance and BMC feedback interventions led to significant improvements in both the technical quality and business aspects of the applications, underscoring the value of AI in supporting a comprehensive perspective of developing with Low-Code platforms in a learning context (RQ1, RQ2). The study found that students with no prior knowledge of modeling and programming showed slightly greater improvements in their assisted cycle, indicating that the approach effectively supports the knowledge-building processes of novices (RQ1, RQ2). However, students with technical backgrounds continued to outperform their non-technical peers, even in the AI-assisted cycle (RQ1, RQ2). This suggests that, while the LCD approach reduces the technical expertise required, developers with technical insights still perform better and are likely to derive more benefits from AI due to their background knowledge (RQ1, RQ2). The findings of this study show that AI-based text-to-model support, is helpful in identifying and justifying potential model elements such as classes, attributes, associations, and activities as evidenced by improvements in application quality (RQ1). The study also highlights the potential of integrating the Business Model Canvas into the low-code (LC) development approach, demonstrating its capacity to enhance the model-based development approach of LC by incorporating business value modeling, as evidenced by improvements in application quality (RQ2). Beyond serving as a productivity tool to accelerate development speed, reduce time-to-market, and lower costs, LC platforms augmented with the BMC also offer a capacity to serve a learning environment. This dual functionality underscores the transformative role of LC platforms in fostering both rapid prototyping and developer skill acquisition through trial-error loops.

Future research directions include integrating the suggested AI-based approaches to refine LC capabilities even further. The findings of the experiments highlight the need for addressing security risks, validation mechanisms, and user interface design in the modeling process. The RAAFT framework proposed by Sedrakyan et al. (2024) provides a foundation for designing security modeling guidelines that can be effectively integrated into low-code (LC) models. Additionally, enhancing modeling support for validation, such as prompts for (assisting) detecting undesired paths with model generation assistance (e.g. microflow within the Mendix modeler), offers a promising avenue for future research.

Furthermore, the integrated BMC and T2M approach outlined in this study has applications both in bridging the LCD loop through a generative text-to-application capability and beyond, demonstrating potential for supporting assessment assistance in educational contexts, such as enabling automated grading systems.

To evaluate the practical impact, other potential future directions for this study include studies with broader participant pools, including both academic and industry settings, with stricter experimental designs that account for effects from other variables (such as learning from the case in the first cycle) and compound unknown variables.

While the quasi-experimental design in this study did not offer strict control over all contributing variables, the positive outcomes suggest that AI-enhanced interventions integrated into low-code platforms are promising. Future research should further investigate the educational role of AI, with an emphasis on broadening the scope of educational interventions and evaluating long-term learning outcomes. Given that model quality scores may not accurately reflect learning processes and skills acquisition, alternative assessment criteria warrant exploration. To further enhance LCDPs, integrating cognitive feedback and behavioral feedforward mechanisms, as suggested by Sedrakyan et al. (2016), could provide enriched learning and development experiences within the LC context. Additionally, enhancing model-based mechanisms and AI-based support for other phases of the LC lifecycle, such as guiding the service and third-party API integrations with models that enable triggering/calling integrations, and offering guidance for platform selection, would enhance the versatility and functionality of LCDPs. These advancements, alongside deeper AI integrations, have the potential to elevate LCD platforms, making them more powerful and adaptable for a broader range of users and use cases. While these developments have the potential to redefine LC platforms, making them more accessible and versatile for a broader range of users, future research must carefully address scenarios where human input and judgment remain indispensable to ensure reliability, accuracy, and alignment with complex business objectives. Additionally, incorporating mechanisms for preventing deskilling due to excessive reliance on AI-driven assistance represents an important avenue for future study.

ACKNOWLEDGEMENTS

The research is conducted in the context of the course Low Code Application Development: UTwente OSIRIS site at https://utwente.osiris-student.nl/onderwijscatalogus/extern/cursus.

REFERENCES

Sedrakyan, G., & Snoeck, M. (2013, February). A PIM-to-Code requirements engineering framework. In International Conference on Model-Driven Engineering and Software Development (Vol. 2, pp. 163-169). SciTePress.

Sedrakyan, G., & Snoeck, M. (2014). Lightweight semantic prototyper for conceptual modeling. In Advances in Conceptual Modeling: ER 2014 Workshops, ENMO, MoBiD, MReBA, QMMQ, SeCoGIS, WISM, and ER Demos, Atlanta, GA, USA, October 27-29, 2014. Proceedings 33 (pp. 298-302). Springer International Publishing.

Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low code development and model-driven engineering: Two sides of the same coin?. Software and Systems Modeling, 21(2), 437-446.

Henkel, M., & Stirna, J. (2010). Pondering on the key functionality of model driven development tools: The case of mendix. In *Perspectives in Business Informatics Research: 9th International Conference, BIR 2010, Rostock Germany, September 29–October 1, 2010. Proceedings 9* (pp. 146-160). Springer Berlin Heidelberg.

Horlach, B., Drews, P., & Schirmer, I. (2016, March). Bimodal IT: Business-IT Alignment in the Age of Digital Transformation. In *MKWI* (pp. 1417-1428).

Rokis, K., & Kirikova, M. (2022, September). Challenges of low-code/no-code software development: A literature review. In International Conference on Business Informatics Research (pp. 3-17). Cham: Springer International Publishing.

Sedrakyan, G., Iacob, M. E. , & van Hillegersberg, J. 2024. Towards LowDevSecOps Framework for Low-Code Development: Integrating Process-Oriented Recommendations for Security Risk Management. In Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24). Association for Computing Machinery, New York, NY, USA, 886–894. https://doi.org/10.1145/3652620.3688335

Humble, J., & Farley, D. (2010). *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education.

Sedrakyan, G., & Snoeck, M. (2013, June). Feedback-enabled MDA-prototyping effects on modeling knowledge. In International Workshop on Business Process Modeling, Development and Support (pp. 411-425). Berlin, Heidelberg: Springer Berlin Heidelberg.

Sedrakyan, G., & Snoeck, M. (2017). Cognitive feedback and behavioral feedforward automation perspectives for modeling and validation in a learning context. In Model-Driven Engineering and Software Development: 4th International Conference, MODELSWARD 2016, Rome, Italy, February 19-21, 2016, Revised Selected Papers 4 (pp. 70-92). Springer International Publishing.

Sedrakyan, G., Poelmans, S., & Snoeck, M. (2017). Assessing the influence of feedback-inclusive rapid prototyping on understanding the semantics of parallel UML statecharts by novice modellers. Information and Software Technology, 82, 159-172.

Ruiz, J., Sedrakyan, G., & Snoeck, M. (2015, September). Generating user interface from conceptual, presentation and user models with JMermaid in a learning approach. In Proceedings of the XVI International Conference on Human Computer Interaction (pp. 1-8).

Sedrakyan, G., & Snoeck, M. (2014). Do we need to teach testing skills in courses on requirements engineering and modelling. In CEUR workshop proceedings (Vol. 1217, pp. 40-44). CEUR-WS. org.

Sedrakyan, G., Abdi, A., Van Den Berg, S. M., Veldkamp, B. P., & Van Hillegersberg, J. (2022). Text-to-model (tetomo) transformation framework to support requirements analysis and modeling. In *10th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2022* (pp. 129-136). SCITEPRESS.

OpenAI. (2024). *ChatGPT (Nov 2024 version)* Large Language Model. OpenAI. https://chat.openai.com/

Sveidqvist, K., & Jain, A. (2021). *The official guide to Mermaid. js: create complex diagrams and beautiful flowcharts easily using text and code*. Packt Publishing.

Murray, A., & Scuotto, V. (2015). The business model canvas. *Symphonya. Emerging Issues in Management*, 94-109.

Cai, D., Wang, Y., Liu, L., & Shi, S. (2022, July). Recent advances in retrieval-augmented text generation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 3417-3419).

Ollama. (n.d.). *Ollama Library*. Retrieved November 30, 2024, from https://ollama.com/library

Rodríguez, S. (n.d.). *FastAPI documentation*. FastAPI. Retrieved November 30, 2024, from https://fastapi.tiangolo.com/

Sedrakyan, G., & Snoeck, M. (2016). Enriching Model Execution with Feedback to Support Testing of Semantic Conformance between Models and Requirements.

Sedrakyan, G., Järvelä, S., & Kirschner, P. (2016). Conceptual framework for feedback automation and personalization for designing learning analytics dashboards. In Conference EARLI SIG (Vol. 27).