Designing Adaptive AI Assistance for Block-Based Modelling: a Wizard of Oz Study with Domain Experts

Chiara Mannari
National Research Council (CNR)
and University of Pisa
Pisa, Italy
chiara.mannari@isti.cnr.it

Tommaso Turchi
University of Pisa
Pisa, Italy
tommaso.tuchi@unipi.it

Manlio Bacco
National Research Council (CNR)
Pisa, Italy
manlio.bacco@isti.cnr.it

Alessio Ferrari
University College Dublin (UCD)
Dublin, Ireland
National Research Council (CNR)
Pisa, Italy
alessio.ferrari@ucd.ie

Cristina Conati
University of British Columbia
Vancouver, Canada
conati@cs.ubc.ca

Alessio Malizia
University of Pisa
Pisa, Italy
Molde University College
Molde, Norway
alessio.malizia@unipi.it

Abstract

The increasing pervasiveness of software-intensive systems requires involving domain experts more directly in technological development. Visual models, expressed in semi-formal notations, can act as shared artefacts that support communication and collaboration between developers and domain experts. However, modelling with semi-formal notations can be challenging for novice modellers. This study presents an AI-infused, web-based modelling tool designed to support users in formalising domain knowledge without requiring advanced modelling skills. The tool features a block-based, domain-specific language that automatically transforms user-generated structures into semi-formal diagrams. AI-based functionalities include a diagram reader, contextual hints, natural-language instructions, and interaction logging. We evaluated the tool through a Wizard of Oz experiment with agronomists in digital agriculture, where participants completed an exploratory modelling task while interacting with AI assistance. Results reveal three key design implications: (i) adaptive AI support accommodating diverse modelling strategies, (ii) concise, actionable guidance delivered at moments of difficulty, and (iii) practice-oriented assistance that preserves user agency and supports learning-by-doing.

CCS Concepts

• **Human-centered computing** → **HCI design and evaluation methods**; • **Computing methodologies** → **Model development and analysis**.

Keywords

Artificial Intelligence, Computational Thinking, Visual Models

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
AVI 2026, Venice, Italy
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Chiara Mannari, Tommaso Turchi, Manlio Bacco, Alessio Ferrari, Cristina Conati, and Alessio Malizia. 2026. Designing Adaptive AI Assistance for Block-Based Modelling: a Wizard of Oz Study with Domain Experts. In *International Conference on Advanced Visual Interfaces 2026 (AVI 26) June 08–12, 2026 Venice, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Visual models support human reasoning by making abstract structures more concrete and cognitively accessible [38]. Compared to texts, these representations hold advantages such as leveraging spatial organisation and perceptual cues that enhance comprehension, reveal patterns, and reduce cognitive load [27, 5, 22]. Software development teams extensively rely on semi-formal notations such as UML, SysML, BPMN, URN, iStar, and KAOS throughout all stages of the software development lifecycle [16, 7]. The representations derived from these notations offer the dual advantage of being processable by computational systems, while also serving as tools for analysis and communication with end users and stakeholders [39, 29, 4]. In recent years, the increasing pervasiveness of digital technologies and the growing complexity of software-intensive systems have amplified the need to involve end users more directly in technological development [6]. In light of this, models can be employed in participatory settings, acting as shared artefacts to support understanding among actors with diverse backgrounds and expertise [31]. Research has shown that engagement with semi-formal models by domain experts improves participation in key development phases, such as requirements gathering, design, and coding [29].

However, modelling is challenging, particularly for individuals without computational backgrounds. Challenges are largely due to the cognitive demands associated with understanding notations and tools [39], as well as dealing with abstraction [26], a core Computational Thinking (CT) skill intrinsic to modelling [40, 37].

Our research goal is to engage end users in modelling by lowering the cognitive barriers associated with abstraction and notation learning. To structure our design approach, we adopt Repenning's AAA model [32], which operationalises CT development through three stages: Abstraction, Automation, and Analysis. We propose

ModeLLer[1], an open-source web-based modelling tool that maps AI-based functionalities to these CT stages, providing targeted support while maintaining user agency in the modelling task.

To evaluate whether the tool could benefit novice modellers and inform future design and use of the system, we carried out a formative design study using a Wizard of Oz (WOz) approach [11]. We situated the experiment within digital agriculture, a rapidly expanding field characterised by complex socio-technical transformations that call for the involvement of multiple stakeholder groups for technological development and assessment [34], and we focused on domain models, a type of representation used to capture key entities and relationships of the problem domain. We involved agronomists (n = 8) and asked them to model a drone-based agricultural scenario. We analysed data through three complementary lenses: (i) interaction logs, (ii) screen-recorded modelling sessions, and (iii) a thematic analysis of think-aloud protocols and post-test interviews. The results reveal how participants addressed the modelling task and how AI notifications were followed (or dismissed), informing three design implications for modelling assistance: adaptive support tailored to interaction patterns, concise actionable guidance at moments of difficulty, and on-demand aids preserving user agency.

The contributions of this work are: (1) a design approach that integrates AI support into a block-based modelling environment, mapping functionalities to CT stages while maintaining user agency; (2) an open-source tool platform for investigating human–AI co-creation in modelling activities; (3) empirical findings and design implications from an experimental study, informing intelligent support for novice modellers.

2 Background and Related Work

2.1 Challenges for Novice Modellers

Substantial cognitive effort may be required when dealing with modelling notations [26, 39]. Prior work has shown that understanding graphical representations relies on specific reading skills that are acquired through experience. For example, these include the capability to map visual elements to their underlying semantics and conventions. [30, 26, 39]. From a notational perspective, the cognitive effectiveness of modelling languages depends on properties such as semantic transparency and visual expressiveness, which directly affect the mental effort required to interpret the representations [27]. Empirical evidence from model-driven engineering further confirms that modellers frequently experience specific challenges primarily related to notation complexity and cognitive demand [39]. To mitigate these challenges, a consistent body of literature has applied Domain-specific Visual Languages (DSVLs) to tailor constructs to specific application fields, resulting in representations that are more familiar to domain experts [20].

Among the interaction paradigms grounded in EUD, the jigsaw metaphor, in which domain concepts are represented through visual elements that can be incrementally composed according to specific constraints, has been proven effective in fostering user engagement and lowering the entry barrier for non-technical users [4]. An example is the block-based environment Scratch, which enables users to code through compositional visual elements while minimising the need for prior technical training [33].

[1]ModeLLer is available at: https://unipisa.github.io/blockly-modeller/

2.2 AI-Supported Model Generation

Recent advancements in machine learning and generative AI have led to a growing interest in automated modelling techniques [41, 35]. Most existing studies have mainly focused on generating models from natural language input, using large language models (LLMs) to translate textual descriptions into models [12, 21]. A smaller body of work has explored the opposite direction, namely the use of LLMs to generate textual explanations or descriptions from existing models, to support stakeholders' understanding [19, 24].

Furthermore, commercially available tools such as Miro, draw.io, or DiagramGuru[2] incorporate LLM-based functionalities to allow users to generate diagrams from natural language. However, these tools primarily target users with prior knowledge of notations, and tend to emphasise one-shot diagram generation while offering limited support for subsequent modification and refinement. Moreover, general-purpose diagramming tools typically do not support code generation, which limits interoperability and the transition from models to executable software.

From an end-user perspective, two main concerns emerge. First, non-AI-experts often struggle to effectively formulate prompts, which can limit the usefulness of AI support when relying solely on a text-field input [42]. Second, existing work has highlighted that fully automated modelling may reduce human control and may obscure biases originating from training data, highlighting the need for hybrid human-AI co-creation [1].

2.3 Intelligent Assistants for Modelling

Previous research examined the cognitive strategies used by modellers and conceptualised intelligent modelling environments where modellers collaborate with software agents that adapt dynamically to their behaviour, supporting and structuring their activities [26]. A recent interview study found that even experienced modellers are open to using software assistants. They highlighted needs for help with assessing model completeness, detecting inconsistencies, guidance on best practices, and effective tool usage [36].

Recommender systems (RSs) have been used to provide personalised modelling recommendations, particularly assisting modellers with context-aware suggestions like relevant elements and model fragments [2]. Most existing RSs are integrated into desktop-based tools and offer technical advice specific to certain languages. In contrast, web-based RSs that help create new artefacts from scratch are limited [2].

Rather than relying on predefined workflows or purely artefact-based recommendations, an alternative approach is to provide adaptive hints based on users' observed behaviour [18]. This approach leverages interaction data to infer users' strategies and provide timely, context-sensitive support. Such behaviour-driven assistance has proven effective in exploratory learning environments, where different levels of expertise and heterogeneous interaction strategies are expected [14]. While recent experiments have explored similar methods for acquiring CT skills [28], they have not yet been investigated in the context of modelling tasks.

Contribution. While prior work has advanced AI-based model generation and intelligent assistants, existing approaches often

[2]https://miro.com; https://www.drawio.com; https://www.diagramguru.com

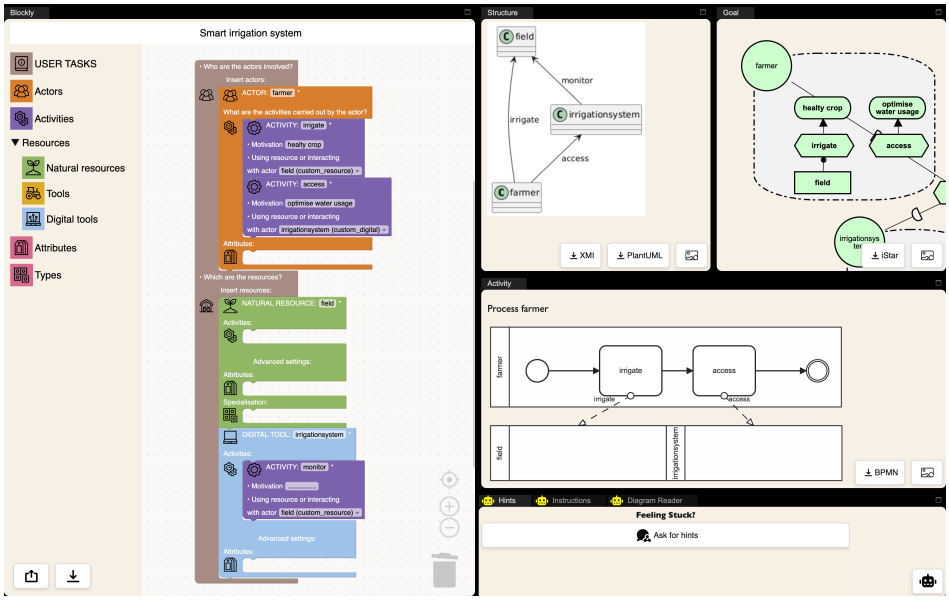


Figure 1: Interface of ModeLLer based on multiple, rearrangeable windows.

assume prior users' expertise. Additionally, generic LLM-based diagram generators focus on generating outputs from prompts rather than facilitating model refinements. In contrast, this work aims to assist novice users during the modelling process through an AI-assisted, block-based environment. Furthermore, behaviour-driven assistance tailored to modellers' interactions—particularly for novices and in web-based environments—remains underexplored. This contribution addresses this gap by exploring how adaptive support aligned with CT stages can better support novices' modelling strategies.

3 The AI-based Modelling Tool

ModeLLer aims to minimise notational burden while preserving the interaction with the semi-formal notations. To achieve this goal, the tool combines a DSVL with a block-based programming approach. A preliminary version without AI features was presented in [25]. Fig. 1 outlines the tool interface, which is organised into two main areas. The *Input area*, located on the left, contains the workspace with a block-based visual editor and the DSVL. On the right, the *Output area*, shows diagrams generated in real time based on the user-created block structures transformed into different modelling representations, including UML, BPMN, and iStar. The layout is single-page, featuring several resizable and rearrangeable windows. This modular structure allows users to adapt the view according to their modelling needs and makes it easy to manage future extension of the tool, such as adding new DSVLs, enabling or disabling output languages, and including new layout components for modelling support. The current version focuses on digital agriculture, using a DSVL derived from the research conducted within the EU-funded Horizon Europe project CODECS (Maximising the CO-benefits of agricultural Digitalisation through conducive digital ECoSystems) [10]. Within the project, a methodology based on the collaborative development of semi-formal models was applied in the context of living labs, i.e., networks of researchers, practitioners and policymakers, for the collaborative assessment of the impacts of agricultural digitalisation [23]. In a similar context, the tool can contribute by empowering stakeholders to create models.

3.1 AI-based functionalities

We grounded the design of the AI-based support in Repenning's AAA model, which provided a conceptual foundation for the design of our system within a three-stage CT process. Within our EUD modelling tool, we operationalised the CT stages as follows:

Abstraction, a set of blocks, accessible through a toolbox panel, supporting users in identifying relevant domain concepts. In the context of digital agriculture, the core entities may be represented through actors, activities, natural resources, digital tools, etc.

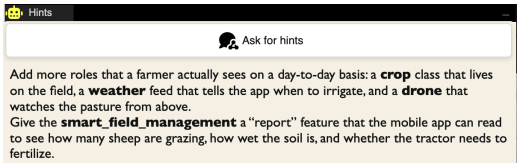
Automation, the jigsaw-based mechanism of dragging-and-dropping blocks on the workspace, enabling the users to arrange custom block structures under certain constraints.

Analysis, the output diagrams that are displayed simultaneously and support evaluation of the resulting models.

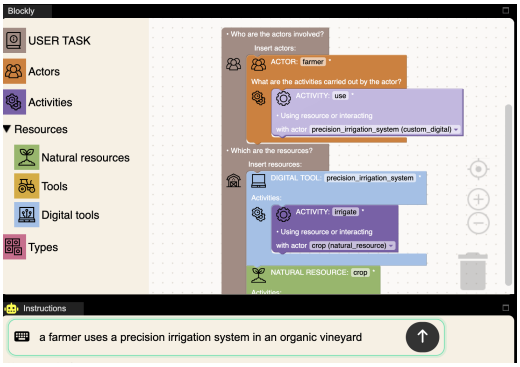
In integrating the AI-based functionalities, we followed principles for developing AI-infused systems [3] by opting for components to support users' cognitive activities without replacing their role in shaping the model. Four AI-based components were introduced, each supporting a CT stage while preserving user control:

(a) Contextual Hints (Abstraction stage) A component with a call-to-action for asking suggestions for model extension based on the current state of the model. An LLM receives as input a piece of code generated by the tool, e.g., a PlantUML describing a class diagram, and returns suggestions for extension (Fig. 2a).

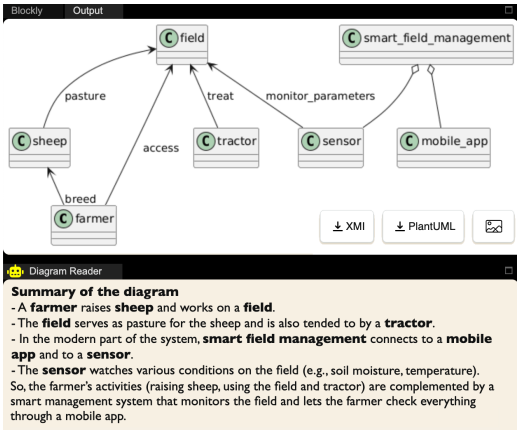
(b) Textual Instructions (Automation stage) A natural-language input field enables users to build or edit models by providing instructions in plain text. The input text is interpreted by an LLM and translated into corresponding *create*, *delete*, and *edit* operations,



(a) Contextual Hints



(b) Textual Instructions



(c) Diagram Reader

Figure 2: AI-based functionalities in ModeLLer: (a) Contextual Hints provide suggestions for model extension; (b) Textual Instructions enable natural-language model manipulation; (c) Diagram Reader generates non-technical descriptions of models.

which are executed through the API of the block-based modelling language to update the view accordingly (Fig. 2b).

(c) Diagram Reader (Analysis stage) An LLM performs a model-to-text transformation of the underlying code at each update of the workspace. The resulting textual description presents the generated models in non-technical terms, supporting users in reflecting on whether the representation aligns with their intended meaning and domain knowledge (Fig. 2c).

(d) Pop-up Notifications (All stages) An RS assists the modellers by providing notifications aligned with each stage of the CT process, based on the actions performed. The system relies on a client-side logging mechanism and a server component that collects and processes interaction data during modelling sessions. These data are used to trigger rule-based, context-sensitive pop-up messages, enabling recommender behaviour during user sessions.

3.2 User-AI interaction flows

Fig. 3 presents the AI-infused system developed illustrating its architecture, organised around CT stages, and possible User-AI interaction flows.

On the left, the end user interacts with a **component-based interface** structured into the three CT-aligned phases. In the **Abstraction** phase, users explore blocks, supported by AI-generated contextual hints. In the **Automation** phase, users construct the model within a drag-and-drop workspace, either manually manipulating blocks or optionally providing textual instructions. These instructions can be sent to generative AI to be converted into blocks, supporting partial automation of modelling actions. In the **Analysis** phase, the resulting diagram is rendered, and an AI reader component supports the user in interpreting the model.

At the centre, the **blocks-to-diagrams** module translates block structures into code (e.g., UML/PlantUML, BPMN, iStar), then a code-to-diagram module transforms code into diagrams.

On the right, **generative AI** provides three optional forms of assistance, accessed via dedicated prompts: (i) converting textual instructions into blocks, (ii) providing hints to extend or refine the diagram, and (iii) explaining the diagram to support interpretation and validation. At the bottom, a telemetry component records **interaction logs** generated during modelling. These logs feed an RS, which analyses user behaviour and triggers pop-up notifications within the interface. Interaction flows are represented through connections: solid lines depict human-controlled actions, while AI-mediated interactions are shown as dashed flows.

4 WOz Study

To explore whether the AI-assisted functionalities could be beneficial for novices, we conducted a user study leveraging a WOz methodology [11]. WOz approaches have proven effective in AI research as a way to prototype future intelligent capabilities before full automation, enabling researchers to study Human-AI dynamics under realistic conditions [9, 15]. First, we implemented the LLM-based functionalities—contextual hints, text-based instructions, and the diagram reader—using simple prompts that can be iteratively refined based on the study results. Next, we added a WOz workflow to simulate the RS, complementing the tool with notification-based assistance. This choice was based on the fact that RS notifications are highly contingent on users' actions, making it difficult to anticipate a priori which suggestions to issue and when. We framed our investigation around the research questions (RQs):

- RQ1: To what extent do users follow the suggestions provided by the system?
- RQ2: How do users perceive the AI-based assistance offered by the tool?

To answer RQ1, we analysed interaction logs and screen recordings to examine users' uptake of suggestions during modelling. To answer RQ2, we conducted a think-aloud protocol and post-test interviews to capture participants' perceptions and assistance needs. Transcripts were analysed through thematic analysis, following

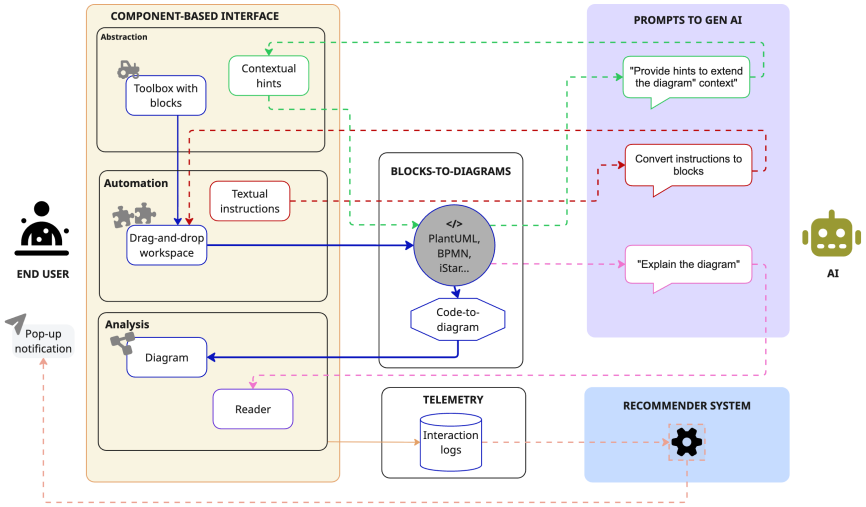


Figure 3: User–AI interaction flows within the tool. Dotted lines indicate interactions where AI intervention occurs.

Braun and Clarke's approach [8]. The first author carried out the coding, and the second author reviewed the resulting codes for clarity and consistency.

4.1 Participants

The study involved eight domain experts (n = 8). We selected this sample size in line with usability-testing guidelines [17], suggesting that approximately 10±2 participants are typically sufficient for identifying the majority of usability issues in think-aloud–style evaluations, while still enabling an in-depth and manageable analysis of behavioural and qualitative data. Participants were adults spanning from young to mid-career, with a majority of females. Most were researchers or practitioners with a background in agronomy. Participants reported low to moderate prior experience with modelling notations and tools, while none identified as an expert modeller. These characteristics were collected through a demographic questionnaire administered after the test session. Data were managed anonymously, in compliance with data protection regulations and with respect for participant privacy. All participants were informed about the nature and purpose of the study and provided their informed consent prior to participation. Following the WOz protocol, participants were informed after completing the test about the simulated nature of the study.

4.2 Procedure

The study was conducted over several days, with participants completing the test during individual sessions held remotely via Teams. The sessions were facilitated by the first author. Each test session lasted approximately 35 minutes and consisted of three main steps:

1) *Training*: participants were introduced to the modelling activity and the tool through a six-minute video.

2) *Interactive session*: this session, lasting 15 to 20 minutes, involved participants completing a modelling task based on a digital agriculture scenario while verbalising their thoughts using a think-aloud protocol. During this phase, a WOz setup was employed, in which the participantes shared their screen and the moderator simulated the behaviour of the RS by providing suggestions when participants encountered difficulties. After each session, the moderator collected the system logs for subsequent analysis.

3) *Interview*: After completing the task, participants joined the moderator in a wrap-up session. During this phase, the WOz setup was revealed, and two open-ended questions were posed: (a) *How did you find the AI-enabled features while you were modelling?* and (b) *What kinds of support—AI-based or otherwise—would make modelling easier for you?* Participants were also invited to provide any further comments or reflections on their experience.

4.3 Task

The task was designed to preserve the open-ended nature of modelling and consisted of an exploration of the tool framed within a scenario of agricultural digitalisation. The scenario was based on a real-world context, using data extracted from materials developed within the framework of the CODECS project. Participants were tasked with creating a model capturing the key elements and relationships of the scenario, drawing on their own background knowledge, without a predefined solution. The scenario concerned the use of drone technology for crop management, where a remote sensing system collects agronomic data through drones and processes it on a digital platform.

4.4 WOz Notifications

Hints were delivered through a chat controlled by the moderator, resulting in contextual pop-up notifications appearing on the user's screen. This mechanism followed a WOz approach, in which the moderator simulated the RS. Hints were triggered upon the detection of a stuck state, inferred both through think-aloud verbalisations and observed interaction patterns, such as repeated execution of the same actions, frequent undo/redo cycles, or prolonged inactivity. The purpose was to support the user's intended actions and guide participants toward modelling goals, while preserving exploratory behaviour. Hints were provided at three CT stages:

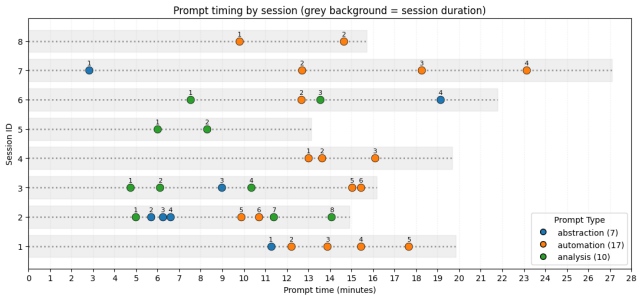


Figure 4: WOz notification distribution

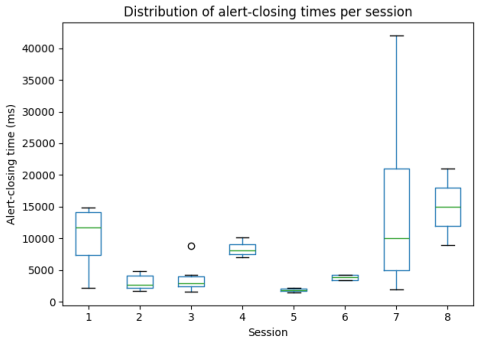


Figure 5: Alert closing time

Abstraction: aimed at eliciting model entities, e.g., encouraging exploration of the available blocks.

Automation: addressing practical interaction, e.g., drag-and-drop or zooming, and mainly delivered in moments of user difficulty.

Analysis: feedback related to the output model or block structure, including suggestions to leverage tool functionalities such as the diagram reader.

5 Results

We report results from three complementary analyses: interaction logs, recorded sessions of participants, and a thematic analysis of the think-aloud and post-test feedback.

5.1 Adoption of System Suggestions (RQ1)

In the exploratory sessions, which lasted an average of 18 minutes and 32 seconds, the moderator sent a total of 34 WOz notifications, averaging 4.25 notifications per session.

Fig. 4 illustrates the distribution of the notifications. Each row represents a session, the x-axis contains elapsed time, and the grey background bars indicate session duration. Markers indicate individual notifications, with colors corresponding to the type of notification classified into one of the CT stages: Abstraction (blue), Automation (orange), and Analysis (green). Automation-related hints are the most frequent and appear throughout the sessions, often occurring in the second part. These hints mainly addressed operational issues, such as managing aggregations, associations or drag-and-drop (e.g., "use the selector *is part of* within the sensors block to connect the sensor block to the drone"). Analysis-related hints were delivered in half of the sessions, suggesting revision of the elements created (e.g., "check advisor's connections"). Abstraction-related hints are less frequent and tend to appear earlier in the sessions, supporting users in identifying model elements through interaction with the toolbox (e.g., "explore blocks on the menu").

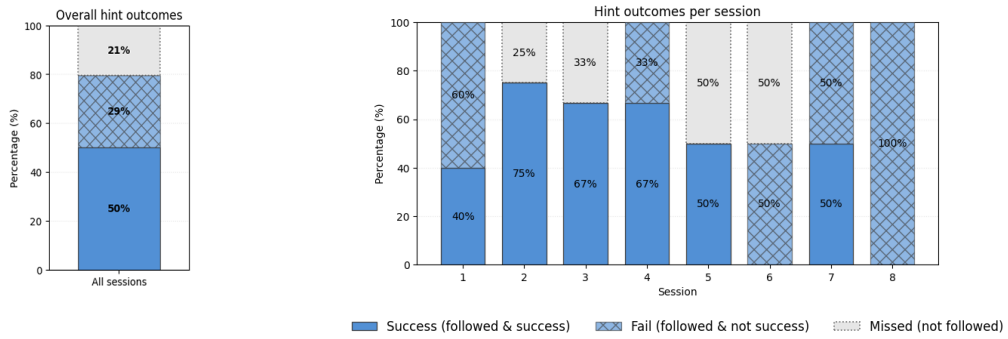
To assess users' reaction to WOz notifications, we first examined how quickly participants reacted to assistance by closing the alert appearing on the screen. Fig.5 shows the distribution of alert-closing times across the sessions. Each boxplot represents one session, illustrating the variability in how long participants took to dismiss system alerts. Substantial between-session variability emerges. Sessions 2, 3, 5, and 6 show short and relatively stable closing times, suggesting that alerts were dismissed in less than 5 seconds. Sessions 1, 4, and 8 exhibit longer median closing times, indicating that participants spent more time reading the alert content. Session 7 presents highly heterogeneous behaviour, with some alerts dismissed rapidly, while others were kept open for extended periods. Overall, the figure highlights that alert engagement was not uniform, varying both across participants and within sessions.

Finally, through observation of the recorded sessions, we assessed whether the received suggestions were successful. We identified three outcomes: *success*, when the user successfully performed the suggested action; *fail*, when the user attempted the action without achieving the suggested result; and *missed*, when a suggestion was ignored by the user. Fig.6a summarises hint outcomes across all sessions. This distribution shows that nearly 80% of system suggestions were followed, while only a small proportion were ignored. Among all provided suggestions, half successfully supported users, whereas the remaining half did not lead to a successful outcome.

We investigated the variability of suggestion success across sessions. Fig.6b highlights session variability. Sessions 2, 3, and 4 show high success rates. In contrast, sessions 6 and 8 exhibit null success, with hints either failing or not being followed at all. Remaining sessions display a more balanced distribution. Overall, there is significant variability across sessions. To assess whether variability persists at the level of suggestion type, we analysed hint success rate based on the CT stage. The results presented in Fig.6c indicate that suggestions related to the automation stage achieve the highest success rate (53%), with minimal dismissal by users. In contrast, suggestions associated with abstraction and analysis exhibit a more balanced distribution of outcomes, with abstraction hints showing higher dismissal rates.

5.2 Feedback on AI Assistance (RQ2)

Several themes emerged from the feedback, concerning both the perceived usefulness of the support and its limitations. Table 1 summarises these themes. Participants' feedback revealed *a mixed experience with WOz notifications*. The majority perceived the suggestions received as useful aids, especially when they addressed practical issues: "Oh, oops. I'm writing too much. Ok, I should use shorter names for activities" (P2). However, other participants reported that such hints were misaligned with workflow: "The pop-up works if you take the time to read it. I basically didn't read them and



(a) Success rate aggregated **(b) Success rate per session** **(c) Success rate per CT stage**

Figure 6: Pop-up notifications success rates

just kept going" (P6). Some hints were hard to interpret: "I didn't understand in the beginning when it tells me to using *is part of*" (P1), and others were perceived as constraining: "At the beginning I'd like to enter longer information without being prompted to shorten it, and then streamline it later, so I can check whether the generated text reflects what I meant to include in the diagram" (P5).

Among the LLM-based components, the *diagram reader emerged as the most valued AI feature*, helping users to reflect on the model under construction: "the AI is kind of trying to help me to describe this system in this diagram reader" (P7). The theme *limits of AI-generated textual support* emerged from cases in which the generated description introduced interactions not represented in the diagram, thus producing hallucinated content. This could foster reliance on the textual output rather than on the underlying model structure. As P8 noted: "For me, the system was working because what I had in mind was written there, so from my point of view I was doing relatively well." *Text-based instructions and contextual hints functionalities remained peripheral*, as many participants declared having noticed them sparingly: "I focused on the central area, where I was supposed to work, and only at a certain point I realised that there were other elements" (P7) and would require time to explore the tool entirely: "I'll just need to learn more, like get so immersed" (P2). In contrast, for some participants, these features proved effective. For example, P4, reflecting on their experience with textual instructions, remarked: "When you ask to build an activity or an actor, it works very well, I think. And then I asked him to link some features, so Ok" (P4). However, there was generally a *preference for block-based interaction over text-based interaction*: "Instinctively, I would not use it. I would prefer building my actor, my activity, etc. But maybe then, when you're used to the tool, maybe it's faster to ask AI" (P4).

Regarding support needs, participants generally did not request additional theory. Instead, they expressed *appreciation for a learning-by-doing approach*. As P6 remarked, "I learn by trying it out". A clear *need for concrete examples* also emerged, as one participant noted, "Maybe it would make sense to include an example, like showing an already completed diagram" (P3). Moreover, participants expressed a *need for on-demand validation*, which they could actively request at moments of uncertainty: "I was looking how the diagram worked, I don't know if it made sense or not. Probably then I would have to ask someone who tell me ok, this is the diagram is well conceptualised or not. I don't know if the AI can do it" (P1). Finally, several participants suggested *conversational support*, enabling them to ask questions in natural language "I think it's nice to have the possibility to ask some questions. For example, how can I link my drone to my server?" (P4).

Table 1: Thematic analysis results

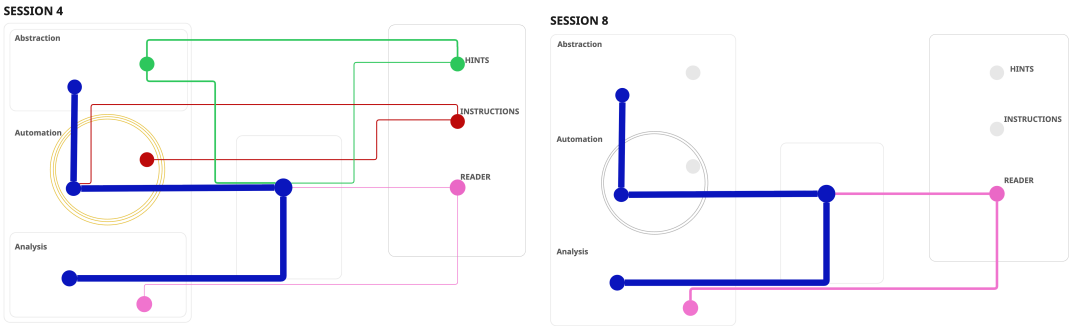
1	Mixed experience with WOz notifications
2	Diagram reader as the most valued AI feature
3	Limits of AI-generated textual support
4	Peripheral role of text-based instructions and contextual hints
5	Preference for block-based interaction over text-based interaction
6	Appreciation for a learning-by-doing approach
7	Need for concrete examples
8	Need for on-demand validation
9	Interest in conversational AI assistance

6 Post-Hoc Analysis and Lessons Learnt

The study provided insights into how domain experts explored and perceived an AI-based modelling tool in terms of system functionalities and interaction strategies. We discuss them in relation to our goal of informing the design of an intelligent assistant for novices.

Findings from RQ1 suggest that AI notifications were effective in assisting novice modellers. We observed a process of co-adaptation [13] between the system and the users: the system issued notifications in response to users' ongoing actions, while users adjusted their behaviour to the suggestions and incorporated them into their workflow. The higher success rate of suggestions provided at the automation stage suggests the value of prompts that help users resolve impasses with minimal interpretation effort, indicating the effectiveness of operational guidance. At the same time, unsuccessful suggestions highlight design space for improving how assistance is delivered and integrated into the workflow. Future work should better scaffold co-adaptation by examining why suggestions fail while also assessing users' satisfaction.

User feedback (RQ2) indicates that participants appreciated the block-based interaction and the practice-oriented approach to modelling. However, despite having similar backgrounds, they employed



(a) Session 4 **(b) Session 8**

Figure 7: Session patterns

different modelling strategies and perceived AI assistance in different ways. This can be further confirmed by the log analysis, revealing multiple patterns of human-AI interaction. Fig.7 reports two representative examples. These figures were generated from logs and recording analysis and provide a summative view of AI-feature usage at the session level. The visualisations match the model of User–AI interaction flow illustrated in Fig. 3, with colours distinguishing the different AI features, while the line thickness reflects the relative extent of its use within the session. Thicker lines indicate more intense interactions, while the circles represent WOz suggestions. Yellow circles denote suggestions that were followed, whereas grey circles indicate dismissed suggestions. These patterns support the identification of user profiles, such as more explorative behaviours, characterised by broader and more frequent engagement with multiple AI features, versus more cautious behaviours, marked by limited or selective use of AI support. A comparison between the pattern resulting from Session 4 (Fig. 7a) and the pattern resulting from Session 8 (Fig. 7b) highlights differences in user behaviour. Pattern 4 reflects a wide-ranging approach, with users actively engaging with all available AI functionalities and following WOz suggestions. By contrast, Pattern 8 shows a more focused workflow centred on the diagram reader, on which users relied more heavily while largely disregarding other AI elements. These patterns confirm two distinct behaviours also emerging from the feedback, with P4 exhibiting a more exploratory approach and P8 being more focused on validating the output through the diagram reader and dismissing system suggestions. These findings suggest that future AI assistance should be based on the identification of interaction patterns and provide adaptive support accordingly.

Building on these insights, we present lessons learnt highlighting design implications for intelligent modelling assistance for novices:

(1) The agronomists approached the task using different strategies and responded to assistance in heterogeneous ways. An intelligent modelling system should therefore adapt to diverse interaction styles and learn from user preferences, rather than being constrained to standard interaction patterns.

(2) The log analysis showed that the most followed hints were those delivered at the *Automation* stage, indicating that users are more likely to accept suggestions when they struggle, and they are offered concrete solutions. Accordingly, an intelligent system should detect modelling challenges and provide actionable hints that minimise the effort needed to interpret them and to get unstuck.

(3) Our findings suggest that novice modellers may benefit from a practice-oriented approach to modelling. The interaction fostered engagement by supporting a learning-by-doing workflow, helping users articulate abstraction through block-based construction. In this context, user agency should remain central, while intelligent assistance peripheral and transparent, being offered primarily on demand to avoid disrupting users' exploratory flow.

7 Limitations

We acknowledge several limitations: first, the produced models were not validated at the content level, as the goal was to observe interaction patterns rather than assess artefact correctness. Second, we did not collect standard usability or satisfaction measures, given the prototype stage and the formative nature of the study, which prioritised exploring functionality and interaction over summative assessment. Third, the limited session duration may have constrained exploration and shaped how participants engaged with assistance compared to more realistic, extended use. Fourth, the specific task design and the chosen scenario may potentially limit the generalisability of the findings to other contexts. Finally, the study did not include baseline conditions or comparisons with alternative tools or assistance mechanisms, limiting causal claims about the added value of specific features.

8 Conclusions and Future Work

This paper presented a CT-grounded design approach for AI-assisted modelling that integrates intelligent support into a block-based environment while preserving user agency.

Future work could advance along multiple directions: (i) implementing the RS by detecting user struggle and offering actionable hints and adaptive strategies tailored to different user profiles extracted from interaction patterns; (ii) refining the diagram reader by increasing the accuracy of generated output and by enabling the system to refrain from generating content when the underlying model is inaccurate or incomplete, thereby reinforcing its role as a validator of model quality; (iii) addressing the current limitations through more extensive, comparative, and longitudinal studies, allowing for a more comprehensive evaluation of all AI-supported functionalities. Such studies could also be conducted in domains beyond digital agriculture and extended to assess the impact of these improvements on usability and modelling outcomes.

9 Acknowledgments

Research partly funded by the European Union's Horizon Europe research and innovation programme under grant agreement no. 101060179.

References

[1] Silvia Abrahão, John Grundy, Mauro Pezzè, Margaret-Anne Storey, and Damian A. Tamburri. 2025. Software engineering by and for humans in an ai era. *ACM Trans. Softw. Eng. Methodol.*, 34, 5, Article 129, (May 2025), 46 pages. doi:10.1145/3715111.

[2] Lissette Almonte, Esther Guerra, Iván Cantador, and Juan Lara. 2022. Recommender systems in model-driven engineering: a systematic mapping review. *Software and Systems Modeling*, 21, (Feb. 2022). doi:10.1007/s10270-021-00905-x.

[3] Saleema Amershi et al. 2019. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19). Association for Computing Machinery, Glasgow, Scotland Uk, 1–13. doi:10.1145/3290605.3300233.

[4] Barbara Rita Barricelli, Fabio Cassano, Daniela Fogli, and Antonio Piccinno. 2019. End-user development, end-user programming and end-user software engineering: a systematic mapping study. *Journal of Systems and Software*, 149, 101–137. doi:10.1016/j.jss.2018.11.041.

[5] A. Blake, Gem Stapleton, Peter Rodgers, and Anestis Touloumis. 2021. Evaluating free rides and observational advantages in set visualizations. *Journal of Logic, Language and Information*, 30, (Sept. 2021), 1–44. doi:10.1007/s10849-021-09331-0.

[6] Susanne Bødker and Morten Kyng. 2018. Participatory design that matters—facing the big issues. *ACM Transactions on Computer-Human Interaction*, 25, 1, Article 4, (Feb. 2018), 31 pages. doi:10.1145/3152421.

[7] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2012. *Model-Driven Software Engineering in Practice*. Vol. 1. (Sept. 2012). doi:10.2200/S00441ED1V01Y201208SWE001.

[8] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology*, 3, 2, 77–101.

[9] Zhuoyi Cheng, Pei Chen, Wenzheng Song, Hongbo Zhang, Zhuoshu Li, and Lingyun Sun. 2025. An exploratory study on how ai awareness impacts human-ai design collaboration. In *Proceedings of the 30th International Conference on Intelligent User Interfaces* (IUI '25). Association for Computing Machinery, 157–172. doi:10.1145/3708359.3712162.

[10] 2023. https://www.horizoncodecs.eu/. (2023).

[11] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. 1993. Wizard of oz studies — why and how. *Knowledge-Based Systems*, 6, 4, 258–266. Special Issue: Intelligent User Interfaces. doi:https://doi.org/10.1016/0950-7051(93)90017-N.

[12] Alessio Ferrari, Sallam Abualhaija, and Chetan Arora. [n. d.] Model generation with llms: from requirements to uml sequence diagrams. In *2024 IEEE REW*. doi:10.1109/REW61692.2024.00044.

[13] Gerhard Fischer and Elisa Giaccardi. 2006. Meta-design: a framework for the future of end-user development. *End user development*, 427–457.

[14] Lauren Fratamico, Cristina Conati, Samad Kardan, and Ido Roll. 2017. Applying a framework for student modeling in exploratory learning environments: comparing data representation granularity to handle environment complexity. *International Journal of Artificial Intelligence in Education*, 27, (Jan. 2017). doi:10.1007/s40593-016-0131-y.

[15] Ken Gu, Madeleine Grunde-McLaughlin, Andrew McNutt, Jeffrey Heer, and Tim Althoff. 2024. How do data analysts respond to ai assistance? a wizard-of-oz study. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (CHI '24) Article 1015. Association for Computing Machinery, Honolulu, HI, USA, 22 pages. doi:10.1145/3613904.3641891.

[16] Jennifer Horkoff et al. 2019. Goal-oriented requirements engineering: an extended systematic mapping study. *Requir. Eng.*, 24, 2, (June 2019), 133–160.

[17] Wonil Hwang and Gavriel Salvendy. 2010. Number of people required for usability evaluation: the 10±2 rule. *Commun. ACM*, 53, 5, (May 2010), 130–133. doi:10.1145/1735223.1735255.

[18] Samad Kardan and Cristina Conati. 2015. Providing adaptive support in an interactive simulation for learning: an experimental evaluation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). Association for Computing Machinery, Seoul, Republic of Korea, 3671–3680. doi:10.1145/2702123.2702424.

[19] Nataliia Klievtsova, Juergen Mangler, Timotheus Kampik, and Stefanie Rinderle-Ma. [n. d.] Utilizing process models in the requirements engineering process through model2text transformation. In *RE 2024*. doi:10.1109/RE59067.2024.00028.

[20] Tomaž Kosar, Sudev Bohra, and Marjan Mernik. 2016. Domain-specific languages: a systematic mapping study. *Information and Software Technology*, 71, 77–91. doi:https://doi.org/10.1016/j.infsof.2015.11.001.

[21] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil MP van der Aalst. 2024. Process modeling with large language models. In *International Conference on Business Process Modeling, Development and Support*. Springer, 229–244.

[22] Jill H. Larkin and Herbert A. Simon. 1987. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 1, 65–100. doi:10.1111/j.1551-6708.1987.tb00863.x.

[23] Chiara Mannari, Manlio Bacco, Giorgio Oronzo Spagnolo, Alessio Malizia, and Alessio Ferrari. 2024. Towards a method for modelling socio-technical process transformation in digital agriculture. In *REW*. IEEE, 306–315.

[24] Chiara Mannari, Tommaso Turchi, Manlio Bacco, and Alessio Malizia. 2025. Assisting stakeholders in class diagram interpretation with llms: a work in progress. In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)*, 298–303. doi:10.1109/REW66121.2025.00045.

[25] Chiara Mannari, Tommaso Turchi, Caterina Frosali, Manlio Bacco, Alessio Ferrari, Cristina Conati, and Alessio Malizia. 2025. Assessing computational thinking skills through artefacts: the case of modeller. In *International Symposium on End User Development*. Springer, 312–321. doi:10.1007/978-3-031-95452-8_19.

[26] Markus Martini, Jakob Pinggera, Manuel Neurauter, Pierre Sachse, Marco R Furtner, and Barbara Weber. 2016. The impact of working memory and the "process of process modelling" on model quality: investigating experienced versus inexperienced modellers. *Scientific reports*, 6, 1, 25561.

[27] Daniel Moody. 2009. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35, 6, 756–779.

[28] Rohit Murali, Sébastien Lallé, and Cristina Conati. 2024. An intelligent pedagogical agent for in-the-wild interaction in an open-ended learning environment for computational thinking. In *Proceedings of the 24th ACM International Conference on Intelligent Virtual Agents* (IVA '24) Article 3. Association for Computing Machinery, GLASGOW, United Kingdom, 9 pages. doi:10.1145/3652988.3673948.

[29] Bashar Nuseibeh and Steve Easterbrook. 2000. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (ICSE '00). Association for Computing Machinery, Limerick, Ireland, 35–46.

[30] Marian Petre. 1995. Why looking isn't always seeing: readership skills and graphical programming. *Commun. ACM*, 38, 6, (June 1995), 33–44. doi:10.1145/203241.203251.

[31] Barbara Quimby and Melissa Beresford. 2023. Participatory modeling: a methodology for engaging stakeholder knowledge and participation in social science research. *Field Methods*, 35, 1, 73–82. eprint: https://doi.org/10.1177/1525822X221076986. doi:10.1177/1525822X221076986.

[32] Alexander Repenning. 2017. Moving beyond syntax: lessons from 20 years of blocks programing in agentsheets. *J. Vis. Lang. Sentient Syst.*, 3, 1, 68–91.

[33] Mitchel Resnick et al. 2009. Scratch: programming for all. *Communications of the ACM*, 52, 11, 60–67.

[34] Mark Ryan, Gohar Isakhanyan, and Bedir Tekinerdogan. 2023. An interdisciplinary approach to artificial intelligence in agriculture. *NJAS*, 25, (Jan. 2023), 1–31.

[35] Rijul Saini, Gunter Mussbacher, Jin L. C. Guo, and Jörg Kienzle. 2022. Automated, interactive, and traceable domain modelling empowered by artificial intelligence. *Softw. Syst. Model.*, 21, 3.

[36] Maxime Savary-Leblanc, Xavier Le Pallec, and Sébastien Gérard. 2024. Understanding the need for assistance in software modeling: interviews with experts. *Software and Systems Modeling*, 23, 1, 103–135.

[37] Annette Upmeier zu Belzen, Dirk Krüger, and Jan van Driel, (Eds.) 2019. *Learning abstraction as a modeling competence. Towards a Competence-Based View on Models and Modeling in Science Education*. Springer International Publishing, Cham, 291–307. doi:10.1007/978-3-030-30255-9_17.

[38] Gem Stapleton. 2018. Reasoning with diagrams: observation, inference and overspecificity (plenary). In *The 24th International DMS Conference on Visualization and Visual Languages, DMSVIVA 2018, Hotel Pullman, Redwood City, San Francisco Bay, USA, June 29 to 30, 2018*. Gem Stapleton and Kang Zhang, (Eds.) KSI Research Inc. and Knowledge Systems Institute Graduate School, 1–7.

[39] Charlotte Verbruggen and Monique Snoeck. 2022. Practitioners' experiences with model-driven engineering: a meta-review. *Software and Systems Modeling*, 22. doi:10.1007/s10270-022-01020-1.

[40] Jeannette M. Wing. 2011. Research notebook: computational thinking–what and why? *Carnegie Mellon School of Computer Science*. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

[41] Mohammad Amin Zadenoori, Jacek Dabrowski, Waad Alhoshan, Liping Zhao, and Alessio Ferrari. 2025. Large language models (LLMs) for requirements engineering (RE): a systematic literature review. *arXiv preprint arXiv:2509.11446*. doi:10.48550/arXiv.2509.11446.

[42] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (CHI '23) Article 437. Association for Computing Machinery, Hamburg, Germany, 21 pages. doi:10.1145/3544548.3581388.