Integrating Machine Learning and Model-Driven Engineering for User Interfaces: A Systematic Literature Review

Pedro Aguilar-Encarnacion[1], **Carlos Iñiguez-Jarrín**[1], **Julio Sandobalín**[1]

[1]Departamento de Informática y Ciencias de la Computación –
Escuela Politécnica Nacional (EPN)
Av. Ladrón de Guevara E11-253, Quito – Ecuador

{pedro.aguilar,carlos.iniguez,julio.sandobalin}@epn.edu.ec

Abstract. *User Interfaces (UI) operate in heterogeneous contexts and require adaptation and, in some cases, intelligent behavior. Machine Learning (ML) and Model-Driven Engineering (MDE) offer complementary capabilities to address this challenge; however, the evidence on their combined use in UI development remains fragmented. The literature still lacks an integrated synthesis of how ML and MDE are combined in this domain. This study aims to synthesize this evidence through a Systematic Literature Review conducted according to the guidelines of Kitchenham, Budgen, and Klotins. The findings cover UI typologies, ML roles and model families, MDE artefacts and paradigms, ML–MDE integration workflows, and coupling, and identify research gaps.*

1. Introduction

User interfaces (UI) mediate communication between users and systems [Criado et al. 2010]. Because users, tasks, and contexts of use are heterogeneous, UI development increasingly requires mechanisms that support personalization and adaptation in interface structure, appearance, and behavior [Abrahão et al. 2021]. These demands are technically challenging because they require coordinating context-sensitive decisions, interface variability, and executable implementation artefacts within a consistent engineering process [Gaspar-Figueiredo et al. 2024].

In this context, integrating Machine Learning (ML) and Model-Driven Engineering (MDE) offers a promising approach. MDE focuses on using high-level models as the main artifacts in software development, allowing developers to automatically generate a software product from abstract representations such as class diagrams [Almonte et al. 2022]. ML enables systems to learn from user data and adapt their behavior over time. In graphical user interfaces, ML can be used to personalize content, predict user actions, and dynamically adjust layouts based on user preferences and interaction patterns [Omar and Gómez 2025]. When combined, MDE and ML provide a powerful solution: MDE defines the interface's structure and design, while ML enhances it with intelligent, adaptive features. Without MDE, ML may still support adaptive or intelligent behavior; however, the resulting UI development process becomes more difficult to specify, verify, reproduce, and maintain[Amershi et al. 2019].

Within UI development, Intelligent User Interfaces (IUI) are a key category for studying ML–MDE integration because they are characterized by capabilities for representation, reasoning, and action that improve the efficiency, effectiveness, and naturalness of interaction [Maybury 1999]. In this setting, ML models can support intent inference,

structured output generation, code generation, and interface adaptation. Among them, Large Language Models (LLM) are especially useful because they can generate code or structured artefacts aligned with UI specifications. However, these outputs do not by themselves ensure a valid UI structure, compliance with design constraints, or consistency with the intended interface model. In this regard, MDE can be used for IUI creation by providing the models, constraints, and transformations needed to define the interface structure, organize its behaviour, and guide its implementation in a systematic and maintainable way.

Despite this relevance, the literature still lacks an integrated and comparable account of how ML and MDE are combined in UI development. Several primary studies report ML–MDE combinations for developing UI [Ahmed et al. 2025], [Planas et al. 2021], [Cardona-Reyes et al. 2021]. Existing secondary studies have addressed this topic from different perspectives [Da Silva et al. 2022], [Kristić et al. 2025], [Naveed et al. 2024]; however, they do not consolidate the evidence on how ML and MDE are combined to design, generate, and adapt UI.

This lack of consolidation directly affects developers and research teams, who require structured knowledge to implement UI solutions in a reproducible and maintainable manner [Amershi et al. 2019]. Therefore, it is necessary to collect, synthesize, and validate the available evidence in order to build a body of knowledge that can guide researchers and practitioners. A suitable strategy is a Systematic Literature Review (SLR), as it employs an explicit and traceable protocol for searching, selecting, assessing, and extracting data [Kitchenham 2007].

This study presents an SLR on ML–MDE integration for the design, generation, and adaptation of UI. It contributes: *(i)* a taxonomy of UI types; *(ii)* a characterization of the role of ML and the reported families/models; *(iii)* an analysis of MDE artefacts and paradigms; and *(iv)* a synthesis of integration workflows and coupling points. The review aims to consolidate and structure the available empirical evidence, identify transferable integration patterns, and determine research trends and gaps.

The article is organized as follows. Section 2 describes the SLR methodology. Section 3 reports the results and answers the research questions. Section 4 discusses the findings, implications, and the main gaps identified. Finally, Section 5 presents the conclusions and directions for future work.

2. Method

This SLR follows the guidelines proposed in [Klotins et al. 2015], [Budgen et al. 2008] and [Kitchenham 2007]. The study was organized into three stages: planning, execution, and reporting. This section describes the planning and execution stages, including protocol definition, search strategy, study selection, quality assessment, and snowballing. The reporting stage is presented in the Results section. Figure 1 summarizes the complete review process.

2.1. Research Question

This work is guided by the following research question: ***How are Machine Learning (ML) and Model-Driven Engineering (MDE) integrated to design, generate, and adapt user interfaces?***

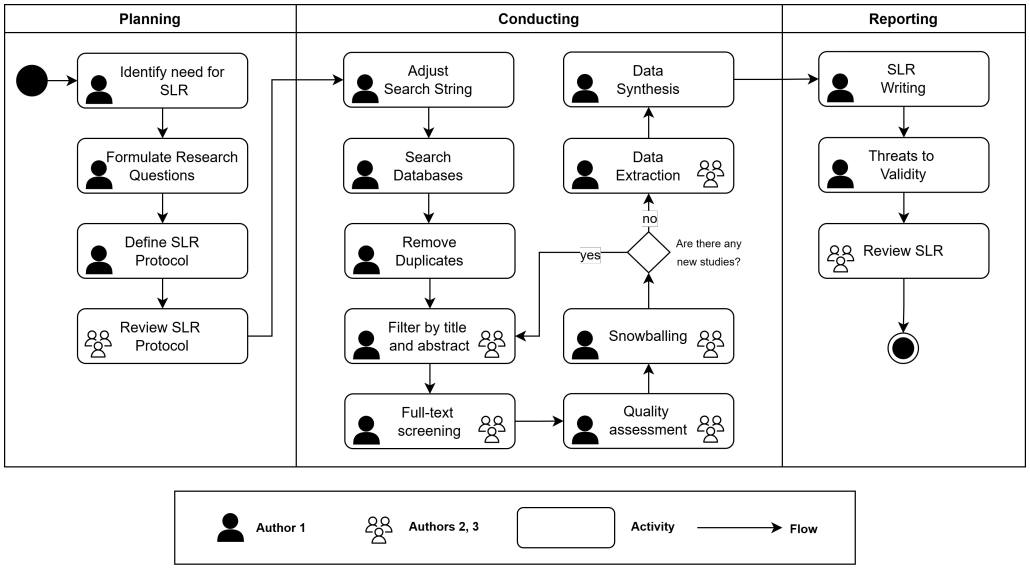


Figure 1. Systematic literature review process.

The main research question was operationalized into eight sub-questions, as shown in Table 1. This structure enables consistent evidence extraction and facilitates cross-study comparison using homogeneous criteria. RQ1 classifies the type of UI according to adaptation and intelligence. RQ2–RQ4 address the ML dimension. RQ5–RQ6 characterize the MDE dimension. RQ7–RQ8 describe ML–MDE integration as a process, capturing the coupling workflow between components and the point at which it occurs.

2.2. Search Process

The search was conducted in Scopus, Web of Science, IEEE Xplore, and the ACM Digital Library. These digital libraries were selected for their strong coverage of software engineering and computing and for indexing peer-reviewed literature [Kitchenham et al. 2015]. The search string was developed based on the key terms of the main research question. To identify these key terms, we used the PCC framework (Population, Concept, Context), as it is suitable for both qualitative and quantitative topics and is commonly applied in scoping reviews [Peters et al. 2020]. The terms identified through this process and their operationalization in the search string are presented in Table 2.

The syntax of the research string was adapted to each search engine to restrict the query to the title, abstract, and keywords, while preserving semantic equivalence across databases and avoiding losses due to syntactic incompatibilities. The string was validated using four seed studies previously selected for their relevance and was iteratively refined until their retrieval in the search results was ensured [Mezhoudi and Vanderdonckt 2021] and [Alti and Lakehal 2025]. The review period was defined as 2014–2025, considering the consolidation of MDE-based approaches applied to interfaces during this timeframe [Mejias et al. 2022] and the practical adoption of ML over the same interval [Akiki et al. 2014].

2.3. Inclusion and Exclusion Criteria

The inclusion criteria were as follows: **IC1** Studies that explicitly integrate ML–MDE in the context of UI development (design, generation, adaptation, or support). **IC2** Studies

Table 1. Research sub-questions and motivations.

ID	Research sub-question	Motivation
RQ1	What type of UI does the study propose or analyze in terms of its adaptivity and intelligence?	To classify the type of interface addressed in each study (e.g., traditional, adaptive, intelligent, or other categories).
RQ2	What role does ML play in the development of the UI in each study?	To identify the role of ML in the UI design, generation, and adaptation process.
RQ3	Which families of ML models are employed for the construction of UI?	To determine the ML model family involved in the design, generation, and/or adaptation of the UI.
RQ4	What learning paradigm is used by the ML model(s) to design, generate, and/or adapt the UI?	To identify the learning paradigm (supervised, unsupervised, reinforcement learning, or others) used to design, generate, and/or adapt UI.
RQ5	Which MDE artefacts are used to represent and/or generate UI?	To identify the core MDE artefacts used to design, generate, and/or adapt the UI (e.g., metamodels, DSL, M2M/M2T, others).
RQ6	Which MDE approaches or paradigms are adopted to organize UI development?	To identify the MDE paradigm structuring the UI design, generation, and/or adaptation process (e.g., MDD, MDA, or others).
RQ7	What workflow is used to integrate ML with MDE in UI development?	To characterize the integration workflow for generating UI with ML and MDE.
RQ8	At what stage is ML integrated with MDE to generate and/or adapt UI?	To identify the phase (design-time, deployment-time, or run-time) at which ML– MDE integration occurs to generate UI.

in which the UI is a relevant artifact of the study, rather than an incidental mention. **IC3** Publications in English or Spanish only.

The exclusion criteria were: **EC1** Studies that use ML or MDE in isolation. **EC2** Studies that refer to "AI/intelligent" without identifiable ML, or that rely solely on rule-based approaches. **EC3** Abstracts, posters, presentations, tutorials, patents, or grey literature, as well as secondary/tertiary reviews. **EC4** Studies that are unavailable or do not allow at least minimal data extraction. **EC5** Duplicate studies.

2.4. Selection of Primary Studies

The initial search retrieved 141 records. After removing duplicates (**EC5**), 124 records remained. Subsequently, **EC3** was applied, reducing the set to 102 candidate studies. These articles then underwent the screening and refinement phases, as described in the following subsections.

Study selection was conducted by three researchers. In each phase, the studies were distributed evenly among the reviewers for an initial assessment. After that, the reviewers exchanged the articles for a second independent assessment, ensuring that each

Table 2. Search terms and query strings.

Term	Query string
UI	(”user interface*” OR ”graphical user interface*” OR GUI OR frontend OR ”front-end” OR ”user-facing” OR ”intelligent user interface*” OR ”adaptive user interface*” OR ”user interface adaptation” OR ”context-aware interface*” OR ”mixed-initiative” OR ”task-driven” OR ”user experience” OR ”human-computer interaction”) AND
ML	(”machine learning” OR ”deep learning” OR ”reinforcement learning” OR ”multi-agent reinforcement learning” OR ”neural network*” OR ”generative AI” OR ”generative artificial intelligence” OR ”large language model*” OR ”foundation model*” OR ”transformer*” OR ”generative adversarial network*”) AND
MDE	(”model-driven engineering” OR ”model-driven development” OR ”model-based engineering” OR ”model-based development” OR ”model driven architecture” OR metamodel* OR ”meta-model*” OR ”domain-specific language*” OR ”domain specific language*” OR ”domain-specific modeling language*” OR ”domain specific modeling language*” OR ”model transformation*” OR ”model-to-model transformation*” OR ”model-to-text transformation*” OR ”models@runtime” OR ”model@runtime” OR ”models at runtime” OR ”runtime model*” OR ”model-driven UI” OR ”model driven UI” OR ”intelligent model-driven UI” OR ”model-driven UI adaptation” OR ”intelligent model-driven UI adaptation” OR ”model-driven user interface*” OR ”model-based user interface*” OR ”model-based UI”)

study was evaluated at least twice. At the end of each phase, disagreement cases were subjected to a third assessment by the researcher who had not previously evaluated the study; this reviewer issued the final inclusion or exclusion decision.

2.4.1. Phase 1: Filtering by title and abstract

In this phase, 102 studies were assessed using criteria **IC1**, **IC2**, **IC3**, **EC1** and **EC2**. Each article was classified into one of three categories: include, exclude, or pending decision. Only studies for which the reviewers agreed on exclusion were removed. As a result, 71 studies were retained.

2.4.2. Phase 2: Full-text screening

The full text of the 71 selected studies was assessed. Each article was classified as included or excluded according to the predefined criteria. In this phase, 49 studies were excluded because they did not report explicit evidence of ML–MDE integration in UI development. Consequently, 22 studies were retained. To assess inter-reviewer consistency, Cohen’s kappa was calculated $\kappa = 0.83$. This value states very high agreement and supports the reliability of the selection process [McHugh 2012].

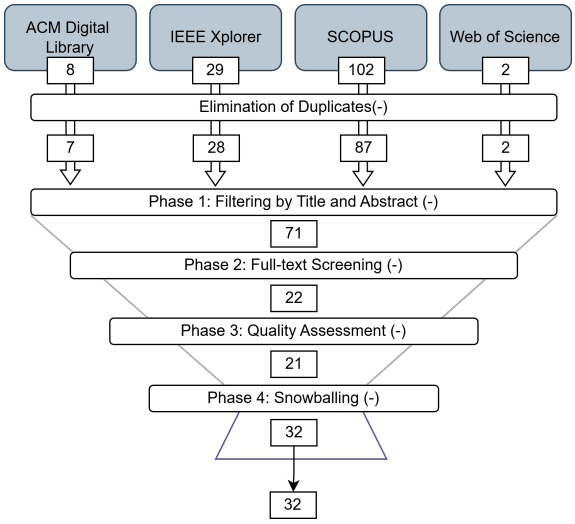


Figure 2. Primary study selection process.

2.4.3. Phase 3: Quality assessment

The quality assessment was defined based on the model proposed by Kitchenham and Charters [Kitchenham 2007]. The quality assessment instrument (criteria and scoring scheme) is provided as part of the study artifacts in the Artifact Availability section. Each criterion was scored using a three-valued scale: $\mathbf{Y} = 1$ (meets), $\mathbf{P} = 0.5$ (partially meets), and $\mathbf{N} = 0$ (does not meet), following common practice in software engineering secondary studies [Kitchenham et al. 2009]. To operationalize study filtering, we used the mean quality score as the cut-off point and excluded studies scoring below this value. After applying this threshold (5.5), one study was excluded and 21 studies remained for synthesis and the subsequent snowballing phase.

2.4.4. Phase 4: Snowballing

This phase applies backward snowballing, whose steps are defined in [Wohlin 2014], to identify additional works aligned with the objective of the review. In this study, snowballing was carried out after the quality assessment to prioritize sources with minimally verifiable evidence. The procedure was recursive: each candidate reference was subjected to the same selection workflow (screening, full-text review, and QA), and the process was repeated until no new eligible studies were identified.

In the first iteration, starting from 21 primary studies, 11 additional articles were retrieved. Consequently, the set increased to 32 studies. The second iteration identified no new primary studies. Therefore, data extraction was performed on 32 studies. Figure 2 presents the primary study selection process, together with the results after each phase, as described in this section.

3. Analysis of the primary Studies

This section presents the results of the analysis of the primary studies. First, we summarize the bibliometric features of the final set. Then, we synthesize the evidence for the

research sub-questions (RQ1–RQ8) defined in Table 1.

3.1. Bibliometric Analysis

Figure 3a summarizes the primary studies by document type and publication year. Of the 32 studies, 19 were published in conference proceedings and 13 in journals. The earliest study, published in 2015, proposes a methodology for intelligent mashup interfaces that combines ML on heterogeneous data with model-based transformations [Fernandez-Garcia et al. 2015]. The remaining output is concentrated between 2019 and 2025, with a marked increase in 2024 and 2025.

Geographically, the studies are limited to 18 countries. The analysis is based on the institutional affiliation of the first author. Europe accounts for the largest share of publications (20), followed by Asia (5), Africa (4), and North America (3). Spain leads in productivity (11), while China, Greece, the United Kingdom, and the United States each report two studies. The remaining 14 countries each report a single study.

3.2. Answers to the research questions

The results are organized into four thematic subsections, aligned with the research subquestions, ranging from UI characterization to the ML and MDE dimensions.

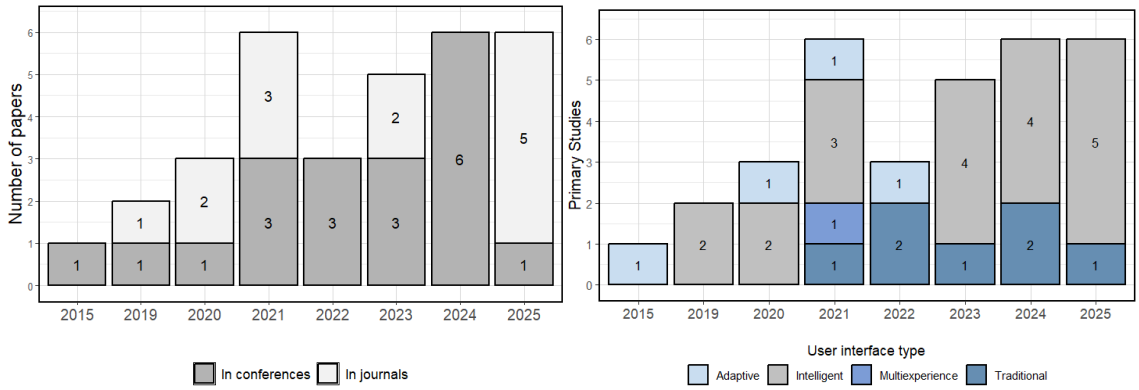
3.2.1. UI Types and Adaptation/Intelligence Levels (RQ1)

For **RQ1**, UI types were classified using operational definitions to ensure comparability. Traditional UI maintains a fixed structure and behavior. Adaptive UI (AUI) automatically adjusts presentation or behavior at runtime based on the user context. Multi-experience UI (MXUI) integrates multiple coordinated modalities and interfaces across devices and applications. IUI incorporates decision-making capabilities that extend adaptation.

Figure 3b shows the distribution of UI types. IUI dominate the evidence base (62.5%). Traditional UI account for 21.9%. AUI represent 12.5%. MXUI are less frequent (3.1%). The temporal distribution shows distinct trends. IUI increase after 2019 and peak in 2023–2025. Traditional UIs concentrate in 2021–2025, but remain less frequent than IUI. AUIs appear intermittently (2015, 2020–2022) and are not reported in 2024–2025. MXUI is observed only in 2021, indicating limited evidence for this UI type in the corpus.

3.2.2. Machine Learning Dimension (RQ2–RQ4)

For **RQ2**, the role of ML was operationalized into eight categories (C1–C8) according to its functional contribution to the UI pipeline: predicting design parameters for UI configuration or generation (C1), adapting the UI at runtime based on context and behavior (C2), recommending configurations or variants with a human-in-the-loop final decision (C3), conversational support (C4), generating formal UI artifacts from text or images (C5), orchestrating tools and modules (C6), confidence/quality-based gating to steer the interactive flow (C7), and domain prediction presented to the user without modifying the UI structure or navigation (C8).



(a) Publications in journals and conferences. **(b) UI types in primary studies.**

Figure 3. Publication trends and UI types.

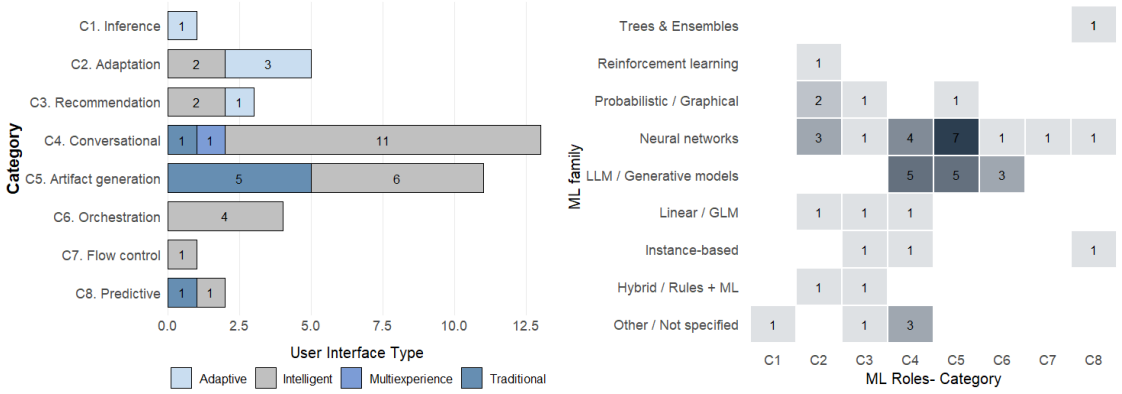
Figure 4a reports 40 role assignments across 32 studies. Conversational control (C4) is the most frequent role (32.5%). Artifact generation (C5) follows (27.5%). Runtime adaptation (C2) accounts for 12.5%, and orchestration (C6) for 10.0%. Human-in-the-loop recommendation (C3) appears in 7.5% of the assignments. Predictive visualization without UI modification (C8) represents 5.0%. Design-parameter prediction (C1) and gating (C7) are marginal (2.5% each). C4 and C6 occur almost exclusively in IUI. C5 appears in both IUI and traditional UI. Conversational (C4) and orchestration (C6) roles are almost entirely concentrated in IUI, whereas artifact generation (C5) appears in both IUI and traditional UI. This suggests that artifact generation functions as a transferable mechanism to automate UI construction, even when the resulting interface does not incorporate "intelligent" runtime capabilities.

For **RQ3**, neural networks are the most frequently reported model family (n = 18), followed by LLM and other language/generative models (n = 13). Probabilistic and graphical models appear in a smaller number of studies (n = 4). Linear/GLM and instance-based methods are rare (n = 3 each), and hybrid rule-based + ML approaches are also uncommon (n = 2). Tree-based and ensemble methods, as well as reinforcement learning, are reported only once (n = 1 each). No margin- or kernel-based methods were identified. In addition, 15.6% of the studies do not specify the ML family (n = 5). Figure 4b summarizes these results.

For **RQ4**, the studies do not consistently report the learning paradigm. The Not specified category is the most frequent (n = 15), indicating that training procedures are documented only to a limited extent. Among studies that do report it, supervised learning is the most common (n = 12), followed by mixed schemes combining supervised and unsupervised learning (n = 2). Reinforcement learning is observed only sporadically (n = 2), and only one case describes a hybrid approach that integrates supervised learning with incremental and transfer learning (n = 1).

3.2.3. Model-Driven Engineering Dimension (RQ5–RQ6)

For **RQ5**, we grouped the reported MDE artifacts into five categories: languages/DSLs (A), models as instances (B), transformations and generation (C), validation/quality (D),



(a) ML roles by UI type

(b) ML model families by role in UI development.

Figure 4. Machine learning roles and model families.

and templates, rules, and assets (E). Languages/DSLs (A) are the most frequent artifacts, followed by models (B) and transformations/generation (C). Templates, rules, and assets (E) provide operational support for UI generation and adaptation.

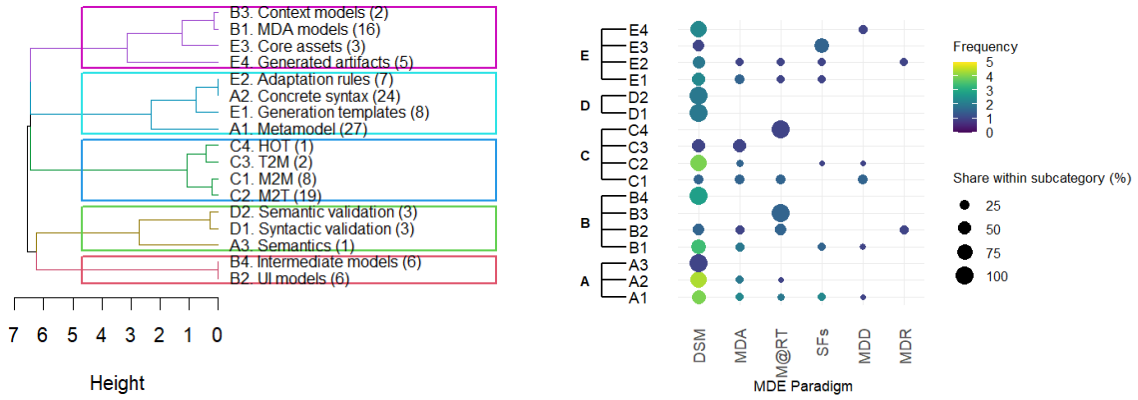
Figure 5a presents a hierarchical clustering of artifact subcategories (A1–E4) using their occurrence frequencies as input. Language specification artifacts (A1 metamodel, A2 concrete syntax) cluster with operationalization artifacts (E1 templates, E2 rules), indicating that DSL definition is commonly paired with explicit mechanisms to instantiate the UI. A second cluster groups transformation artifacts (C2 M2T, C1 M2M), suggesting that automation relies primarily on transformations into code or configuration artifacts and, to a lesser extent, on model-to-model transformations. Validation artifacts (D1 syntactic validation, D2 semantic validation) remain separate and occur at low frequency, indicating that verification is rarely reported in the analyzed pipelines.

For **RQ6**, Domain-Specific Modeling (DSM) is the dominant paradigm (62.5%). Model-Driven Architecture (MDA) follows (12.5%). Models@Runtime and Software Factories show comparable presence (9.4% each). Model-Driven Development (MDD) and Model-Driven Reengineering (MDR) are rare (3.1% each).

Figure 5b relates artifact subcategories to MDE paradigms. In DSM, A1, A2, and C2 are the most frequent, indicating a DSL-centered process in which the metamodel and concrete syntax define UI concepts and M2T transformations produce implementations. In MDA, models (B1, B2) and transformations (C1, C2) are the most frequent, reflecting a process based on abstraction levels and their refinement through successive transformations until executable artifacts are produced. In Models@Runtime, occurrences concentrate on B3 and C4, supporting runtime models for context capture and adaptation control. In Software Factories, E1, E3, and E4 predominate, reflecting an asset- and template-driven automation approach.

3.2.4. ML–MDE Integration Workflow and Timing (RQ7–RQ8)

For **RQ7**, we identified five recurring ML–MDE integration patterns. Each pattern specifies the interaction between ML components and MDE artifacts, and the artifact ex-



(a) MDE artefact subcategories. **(b) MDE paradigms by artefact subcategory.**

Figure 5. MDE paradigms and artefact subcategories.

changed at the coupling point. Figure 6 presents a synthesized version of these workflows. The complete figure is available at the link provided in the Artifact Availability section.

1. **P1 – Runtime NLU triggers modeled behavior.** This pattern is typical of conversational UI. MDE defines a DSL/metamodel for intents, entities, and dialog flow at design time. At run time, ML performs NLU inference (intent classification and entity extraction). Inferred intent triggers transitions, actions, or responses encoded in the MDE model.
2. **P2 – MDE generates resources required by ML.** MDE produces inputs for ML components—such as training-data structures, labels, configurations, or feature schemas—derived from models. The core characteristic is model-driven generation of ML inputs, which helps keep modeled concepts synchronized with ML behavior.
3. **P3 – LLM generates structured models; MDE enforces conformance and compiles.** An LLM produces structured artifacts (e.g., DSL code or JSON instances) from natural-language requirements or interaction. MDE specifies constraints and validates the output against the DSL/metamodel. Conformant artifacts are then transformed to code or executable UI. Studies often report iterative correction loops.
4. **P4 – Computer vision infers UI structure; MDE synthesizes and compiles.** ML extracts UI structure from images, sketches, or screenshots by detecting components and relationships or producing token/DSL representations. MDE converts these predictions into models and applies transformations to generate executable artifacts (e.g., HTML/CSS or native code).
5. **P5 – ML selects adaptations; MDE materializes updates.** ML observes context and interaction, and predicts adaptations. MDE applies transformations, variability mechanisms, or rules to produce the updated UI. This pattern emphasizes personalization and adaptation, typically at run time.

For **RQ8**, coupling occurs mainly at design time (47.9%) and run time (41.7%). Deployment-time coupling is rare (10.4%). Figure 7b shows that IUI dominate coupling stages: 14 of 23 cases at design time, 15 of 20 at run time, and 4 of 5 at deployment time. Traditional UIs appear mainly at design time (7 cases) and only marginally at run time

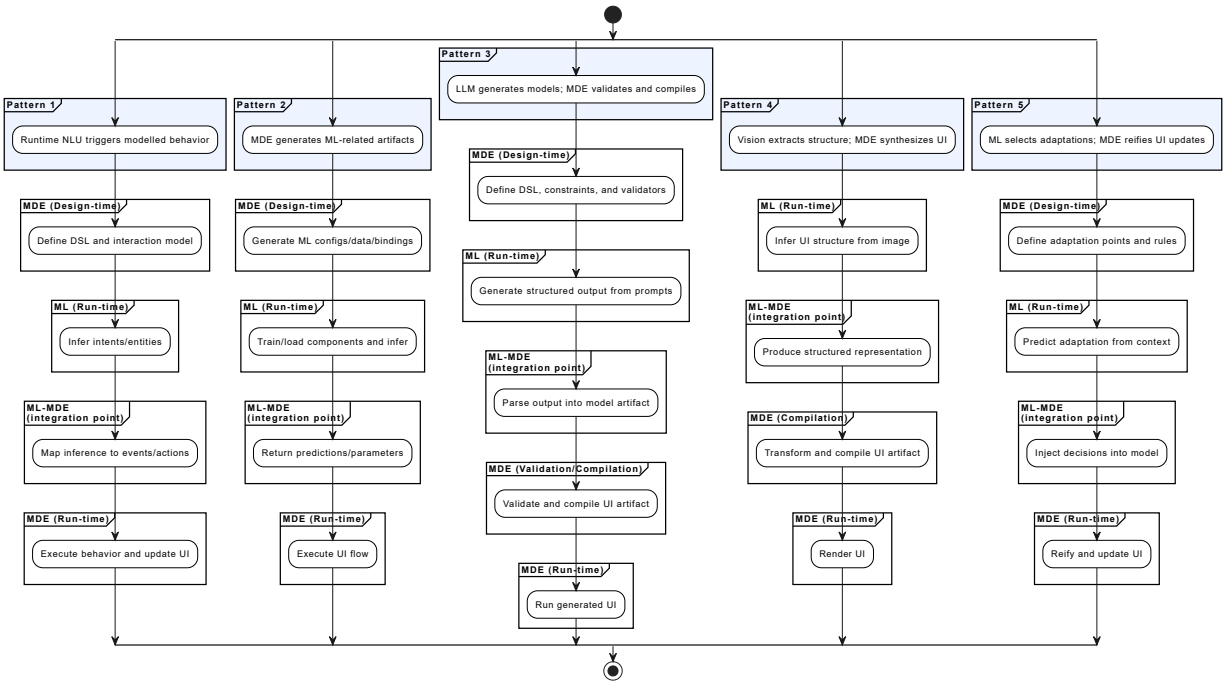
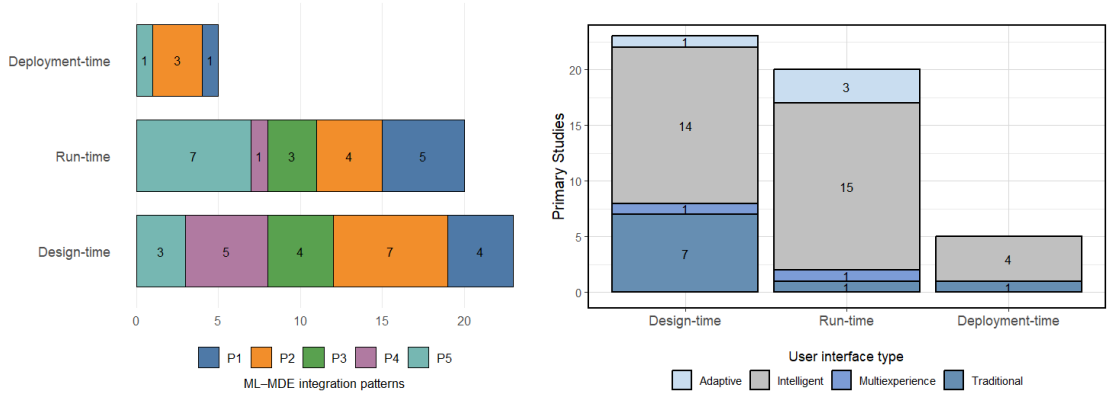


Figure 6. ML–MDE integration workflows (P1–P5).



(a) ML–MDE coupling stages. **(b) Coupling stage by UI type.**

Figure 7. ML–MDE coupling stages and UI types.

and deployment time (1 case each). Adaptive UIs are observed only at run time (3 cases) and sporadically at design time (1 case). Multi-experience UIs appear only once at design time and once at run time. Figure 7a reports the overall timing distribution.

4. Discussion

4.1. Principal Findings

This study synthesizes how ML and MDE are coupled to design, generate, and adapt user interfaces. The corpus shows rapid growth in recent years. However, proposals remain heterogeneous. Many works are still prototypes. No dominant reference architecture has emerged.

- **The evidence concentrates on IUI.** Most contributions focus on IUI rather than on classical adaptive interfaces. The research community prioritizes interaction-centered "intelligence" mechanisms—such as conversational and generative behavior—over runtime personalization through adaptive UI.
- **ML for interpretation and generation.** Across the corpus, ML is reported mainly to interpret inputs (e.g., intents or requirements) and to generate structured artifacts that drive UI construction. Runtime UI adaptation is reported less frequently and is often described with limited detail compared to specification and generation mechanisms.
- **Neural networks and generative models dominate.** The corpus reports neural-network-based methods and language-based generative models as the most recurrent families, indicating a preference for approaches capable of interpreting natural language, generating structured artifacts, and supporting conversational interaction. Other families appear sporadically, and some studies do not specify the model family used.
- **MDE emphasizes code generation.** MDE is used to formalize the UI and generate executable artifacts through transformations. However, few studies describe syntactic and semantic validation. Without such validation, ML-generated artifacts may fail to conform to the metamodel and, consequently, compromise downstream MDE transformations.
- **The corpus reports multiple integration patterns.** The corpus identifies five integration patterns between ML and MDE. These patterns differ depending on the UI type and the role assigned to MDE. In some cases, MDE governs behavior while ML interprets inputs. In others, ML produces candidate models and MDE enforces constraints and compiles them. In adaptation-oriented cases, ML selects the changes and MDE materializes them. Overall, this indicates that the integration point and the coupling artifact are the primary design decisions.
- **Integration occurs at design time and run time.** IUI dominate across all integration stages—design time, run time, and deployment time—with the strongest emphasis on run time. Moreover, deployment-time coupling is reported almost exclusively for IUI, confirming that this UI type is not confined to a single stage. In contrast, IUA are reported mostly at run time and appear only marginally at design time.

4.2. Limitations and Threats to Validity

We report limitations and threats to validity using the stage-based scheme for SLRs proposed by [Kitchenham 2007]. We group threats into: *(i)* study selection, *(ii)* data extraction and analysis, and *(iii)* cross-cutting process threats.

i) **Threats to the validity of study selection.** Selection bias refers to the risk of excluding relevant studies due to search and screening decisions [Genero Bocco et al. 2023]. We mitigated this risk through a protocol-driven search strategy, seed-study checks, backward snowballing, and a multi-reviewer selection process. To strengthen consistency during screening, the study selection involved double assessment, cross-checking, and systematic discrepancy resolution. However, some relevant studies may still be missed due to database coverage, indexing delays, and term variability. Grey literature was excluded to ensure traceability and comparability, acknowledging the associated publication bias.

ii) **Validity of extracted and analyzed data.** The validity of the extracted and analyzed data depends on corpus size, study heterogeneity, and the accuracy of extraction and classification [Genero Bocco et al. 2023]. We mitigated these threats by using operational definitions aligned with RQ1–RQ8 and an extraction protocol with uniform criteria. Nevertheless, heterogeneity in reporting remains a limitation, particularly in studies that do not specify ML families, learning paradigms, or validation procedures with sufficient detail.

iii) **Cross-cutting threats to the SLR process.** Process threats include protocol deviations, limited comparability across studies, and potential domain-knowledge bias [Genero Bocco et al. 2023]. We minimized protocol deviations by applying the same workflow across all phases. Domain-knowledge bias may still affect interpretation; we addressed this threat through team expertise and expert guidance. Nevertheless, some subjectivity may remain in borderline classifications.

5. Conclusions and Future Work

This SLR synthesizes empirical evidence on how ML and MDE are combined to design, generate, and adapt UI. We reviewed studies published between 2014 and 2025. The final corpus includes 32 primary studies that met the inclusion and quality criteria. The reviewed evidence indicates that ML–MDE integration still lacks a solid engineering foundation. The main issue is not the absence of proposals, but rather the lack of technical specifications. This limitation hinders reproducibility, comparability, and the reliable operation of proposed solutions. Although the review addresses UI development broadly, the evidence shows that Intelligent User Interfaces constitute the dominant application context for ML–MDE integration.

The primary contribution of this review is a shared scheme for describing ML–MDE integration in UI development. The scheme specifies which artefacts are involved, for what purpose, and how they are coupled, allowing proposals to be framed as technically verifiable workflows rather than as high-level solutions. Future work should: *(i)* formalize integration points and coupling artifacts into a unified reference framework for IUI; *(ii)* incorporate validation and error handling as explicit process components; *(iii)* standardize an end-to-end workflow covering design, deployment, and runtime; and *(iv)* define an empirical protocol with implementation-level metrics that evaluates integration quality, not only model performance. These actions can support more verifiable and transferable UI engineering processes in real-world contexts, particularly in IUI development.

Artifact Availability

The replication package is available at: https://doi.org/10.5281/zenodo.19081341. It includes the selected primary studies, the data extraction matrix with the corresponding columns for each aspect evaluated in the research questions, the quality assessment questions and their resulting scores, and the complete version of Figure 6. These materials support the transparency, traceability, and reproducibility of the SLR.

References

Abrahão, S., Insfran, E., Sluÿters, A., and Vanderdonckt, J. (2021). Model-based intelligent user interface adaptation: challenges and future directions. *Software and Systems Modeling*, 20(5):1335 – 1349.

Ahmed, A., Azab, S., Moussa, S. M., and Abdelhamid, Y. (2025). Vit-dtc: vision transformer-based design-to-code framework for code generation from generated ui designs and hand-drawn sketches. *Neural Computing and Applications*, 37(29):24243 – 24264.

Akiki, P. A., Bandara, A. K., and Yu, Y. (2014). Adaptive model-driven user interface development systems. *ACM Comput. Surv.*, 47(1):1 – 33.

Almonte, L., Guerra, E., Cantador, I., and de Lara, J. (2022). Recommender systems in model-driven engineering: A systematic mapping review. *Softw. Syst. Model.*, 21(1):249–280.

Alti, A. and Lakehal, A. (2025). Ai-mdd-ux: Revolutionizing e-commerce user experience with generative ai and model-driven development. *Future Internet*, 17(4):1 – 34.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software engineering for machine learning: a case study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '19, page 291–300. IEEE Press.

Budgen, D., Turner, M., Brereton, P., and Kitchenham, B. (2008). Using mapping studies in software engineering. In *Proceedings of PPIG 2008*, pages 195–204. Lancaster University.

Cardona-Reyes, H., Muñoz-Arteaga, J., Mitre-Ortiz, A., and Villalba-Condori, K. O. (2021). Model-driven approach of virtual interactive environments for enhanced user experience. *Applied Sciences*, 11(6):2804.

Criado, J., Vicente-Chicote, C., Padilla, N., and Iribarne, L. (2010). A model-driven approach to graphical user interface runtime adaptation. In *Proceedings of the 5th Workshop on Models@run.time at the 13th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2010)*, volume 641 of *Models@Run.Time*, pages 49–59. CEUR-WS.

Da Silva, M. R., Leite, L. L., and Garcia Costa Dos Santos, G. L. (2022). Algorithms for the development of adaptive web interfaces: A systematic literature review. In *2022 XVII Latin American Conference on Learning Technologies (LACLO)*. IEEE.

Fernandez-Garcia, A. J., Iribarne, L., Corral, A., and Wang, J. Z. (2015). Evolving mashup interfaces using a distributed machine learning and model transformation methodology. In *On the Move to Meaningful Internet Systems: OTM 2015 Workshops*, volume 9416 of *Lecture Notes in Computer Science*, pages 401–410. Springer, Cham.

Gaspar-Figueiredo, D., Fernández-Diego, M., Nuredini, R., Abrahao, S., and Insfran, E. (2024). Reinforcement learning-based framework for the intelligent adaptation of user interfaces. In *Companion Proceedings of the 16th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '24 Companion, page 40–48, New York, NY, USA. Association for Computing Machinery.

Genero Bocco, M., Piattini Velthuis, M. G., Cruz-Lemus, J. A., and Díaz García, Ó. (2023). *Métodos de Investigación en Informática*. AQC Lab.

Kitchenham, B. (2007). Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report Version 2.3, Keele University, UK.

Kitchenham, B., Brereton, P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, 51(1):7–15.

Kitchenham, B. A., Budgen, D., and Brereton, P. (2015). *Evidence-Based Software Engineering and Systematic Reviews*. Chapman and Hall/CRC, 1 edition.

Klotins, E., Unterkalmsteiner, M., and Gorschek, T. (2015). Software engineering knowledge areas in startup companies: A mapping study. In Fernandes, J. M., Machado, R. J., and Wnuk, K., editors, *Software Business*, pages 245–257, Cham. Springer International Publishing.

Kristić, M., Zakarija, I., Škopljanac-Mačina, F., and Car, Ž. (2025). Machine learning for adaptive accessible user interfaces: Overview and applications. *Applied Sciences*, 15(23):12538.

Maybury, M. (1999). Intelligent user interfaces: An introduction. In *Proceedings of the 4th International Conference on Intelligent User Interfaces*, IUI '99, pages 3–4, New York, NY, USA. Association for Computing Machinery.

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica*, 22(3):276 – 282.

Mejias, J. C., Silega, N., Noguera, M., Rogozov, Y. I., and Lapshin, V. S. (2022). Model-driven user interface development: A systematic mapping. In Agredo-Delgado, V., Ruiz, P. H., and Correa-Madrigal, O., editors, *Human-Computer Interaction*, pages 114–129, Cham. Springer International Publishing.

Mezhoudi, N. and Vanderdonckt, J. (2021). Toward a task-driven intelligent gui adaptation by mixed-initiative. *International Journal of Human-Computer Interaction*, 37(5):445 – 458.

Naveed, H., Arora, C., Khalajzadeh, H., Grundy, J., and Haggag, O. (2024). Model driven engineering for machine learning components: A systematic literature review. *Information and Software Technology*, 169:107423.

Omar, K. and Gómez, J. M. (2025). Intelligent user interfaces for metaverse. In *2025 1st International Conference on Computational Intelligence Approaches and Applications (ICCIAA)*. IEEE.

Peters, M. D. J., Godfrey, C., McInerney, P., Munn, Z., Tricco, A. C., and Khalil, H. (2020). Chapter 11: Scoping reviews (2020 version). In Aromataris, E. and Munn, Z., editors, *JBI Manual for Evidence Synthesis*. JBI.

Planas, E., Daniel, G., Brambilla, M., and Cabot, J. (2021). Towards a model-driven approach for multiexperience ai-based user interfaces. *Software and Systems Modeling*, 20(4):997 – 1009.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 1–10, New York, NY, USA. Association for Computing Machinery.