June 2025

TEACHING GRAPHICAL MODELING WITH INTELLIGENT TUTORING SYSTEMS – A REVIEW

Johannes Kunz
University of Applied Sciences Mittelhessen, johannes.kunz@mni.thm.de

Markus Siepermann
Technische Hochschule Mittelhessen, markus.siepermann@mni.thm.de

TEACHING GRAPHICAL MODELING WITH INTELLIGENT TUTORING SYSTEMS – A REVIEW

Completed Research Paper

Johannes Kunz, University of Applied Sciences Mittelhessen, Gießen, Germany, johannes.kunz@mni.thm.de

Markus Siepermann, University of Applied Sciences Mittelhessen, Gießen, Germany, markus.siepermann@mni.thm.de

Abstract

Despite the fundamental importance of graphical models across numerous disciplines, educators face significant challenges in teaching advanced concepts in an understandable manner. This paper provides a comprehensive review of 129 studies that have investigated automated assessment systems in graphical modeling education. The review is organized around seven research questions that address the critical needs and challenges of intelligent tutoring systems in this area. The analysis reveals persistent shortcomings – such as limited adaptability, fragmented methodologies, and a surprising absence of Large Language Model (LLM) applications – despite their growing importance. Drawing upon effective approaches identified in the reviewed literature, we offer a set of recommendations that could inform the design of a more dynamic, AI-driven tutoring environment. By incorporating modern technologies such as GPT-4, these recommendations aim to provide adaptive guidance, real-time support, and advanced semantic analysis, thereby overcoming current limitations.

Keywords: Literature Review, Intelligent Tutoring Systems, Automated Assessment, Graphical Modeling, Artificial Intelligence in Education.

1 Introduction

The ability to visualize complex relationships and make them understandable to a wider audience has made graphical models an essential tool in almost all scientific and technical fields (Sedlmair et al., 2012; Jordan, 2004). Graphical modeling provides a universal method for visualizing sophisticated concepts, thereby enhancing understanding, communication, and analysis among stakeholders (McInerny et al., 2014; Eppler & Burkhard, 2007). Despite their varied applications, all graphical models share common foundational principles: a notation that defines visual representations through symbols, semantics that assign meaning and interpretation to these symbols, and syntax that specifies the rules for their combination via a defined grammar (Karagiannis & Kühn, 2002). A modeling language then formalizes these principles, enabling a consistent approach to the design and interpretation of graphical models. Building on these foundational principles, a modeling language simplifies interdisciplinary exchange by making complex concepts easily accessible to non-experts (Maathuis et al., 2018) and encourages collaboration and the sharing of diverse perspectives. It is therefore crucial to promote graphical modeling skills, such as creating and interpreting models, at an early stage and to provide students with basic skills. Despite its importance, modeling often leads to challenges in teaching due to its complexity, especially as students struggle with the steep learning curve and usability of complex modeling tools (Liebel et al., 2016). One of the key difficulties in teaching modeling is determining how to assess students' work consistently, given the variety of possible solutions and the subjective nature of some modeling decisions. This has led to the need for clear, objective standards to guide both instruction and evaluation.

Lindland et al. (1994) addressed this by identifying three essential quality dimensions for assessing models: *syntactic correctness*, *semantic accuracy*, and *pragmatic clarity*. These dimensions ensure that models are formally valid, meaningful within their domains, and understandable to their audiences. In an educational context, for example, this could mean assessing whether a model created by a student is correctly structured, contains all the required content, and is easily interpretable by others, such as tutors or fellow students. Thaler et al. (2016) further proposed additional aspects of the three quality dimensions by allowing the customization of rules to fit specific contexts and dividing semantic quality into content completeness, which ensures that all relevant information is captured in the model, and semantic correctness, which acknowledges the validity of different modeling approaches as long as they meet domain-specific requirements. Additionally, they offered a more detailed exploration of pragmatic quality, developing metrics to assess model complexity and readability, focusing on the cognitive effort required to read and interpret the models effectively. From another perspective, Cherfi et al. (2002) proposed a framework centered on three interconnected dimensions: specification, usability, and implementation. Specification ensures accurate representation of reality with clarity and correctness – aligning closely with the syntactic and semantic aspects described by Lindland et al. (1994). Usability, which assesses ease of understanding and interaction, parallel to the notion of pragmatic clarity. Implementation, meanwhile, introduces an additional perspective by focusing on feasibility and maintainability. Their approach emphasizes the intricate interplay, where improvements in one dimension may inherently compromise another. Despite the rigorous criteria established for model evaluation, students often face challenges in applying these principles correctly. The most commonly reported mistakes students make during modeling include incorrect use of syntax and notation, and inaccurate labelling (Chren et al., 2019). Other prominent issues are the difficulty in distinguishing between relevant and irrelevant information within the task description, determining the appropriate level of detail for a diagram and deciding when a diagram is complete (Reuter et al., 2020). The unstructured arrangement of elements often leads to a lack of clarity and overview, further complicating the modeling process. Students often face significant challenges in mastering abstraction, defining the responsibilities of design elements, and accurately applying notation as well as writing reports and understanding the detailed requirements of their tasks (Stikkolorum et al., 2018). This might lead to the assumption that students should receive more practical training in navigating the complexities of syntax, notation, and semantics, but the actual challenge lies in providing effective learning methods, resources and support to help them master these aspects of graphical modeling. This complexity is often understood only through extensive experience and the analysis of practical case studies. Syntax and notation offer the structural and visual framework of a model, defining how elements are represented and organized. However, it is the semantics that impart overall coherence and meaning to the models. Even when syntax and notation are correctly applied, a misunderstanding of semantics can lead to models that are structurally sound but conceptually flawed – much like a mathematically correct equation that does not solve the intended problem.

Exercises with real-world scenarios are fundamental in helping learners bridge the gap between theoretical knowledge and practical application. They provide an opportunity not only to reinforce memorized concepts but also to engage with them on a deeper level. Moreover, exercises allow for iterative refinement, enabling students to continuously improve their work through repeated practice and reflection. However, for this iterative process to be effective, learners require feedback that guides their development and helps them identify areas for improvement. Feedback plays a pivotal role in enhancing learning outcomes (Hattie & Timperley, 2007). Particularly when timely and constructive, feedback is one of the most effective strategies for improving students' understanding and performance. It serves as a guide, helping students reflect on their errors, refine their approaches, and correct misconceptions, ultimately fostering deeper cognitive engagement with the subject matter. Furthermore, immediate feedback plays a critical role in learning by quickly correcting errors and preventing misconceptions, especially in complex tasks like graphical modeling (Ajogbeje, 2023; Canals et al., 2025). Without timely input, students risk persisting with incorrect approaches, as delayed feedback is less effective in addressing misunderstandings (Chou & Zou, 2020). It not only corrects mechanical errors but also clarifies conceptual issues, helping learners avoid incorrect self-assessment and premature disengage-

ment. Despite clear benefits, the process of providing individualized feedback is complex. Effective feedback in modeling must address not only syntactic correctness of the underlying semantics, but more importantly the values and attitudes of the modeler (Soyka et al., 2022). This challenge becomes particularly pronounced in large classes or when students produce diverse and equally valid modeling solutions. As an example, Jayal and Shepperd (2009) demonstrate that even within the scope of synonym variation, a single label can appear in an average of 45 different forms across 160 student submissions, despite applying text preprocessing techniques such as stemming and stop word removal. In such cases, tutors must carefully review each model, identify errors, recognize the modeler's intentions, and provide specific advice that addresses both the structural and conceptual aspects of the model. Providing this level of detailed, individualized feedback is time-consuming, and as class sizes increase, it becomes increasingly challenging for tutors to maintain the same level of personal attention and provide feedback in a timely manner. This often results in trade-offs between providing prompt feedback and ensuring that feedback meets high quality standards.

In today's educational landscape, digital learning platforms are increasingly being used to address this challenge. While traditional platforms help manage learning materials and simplify administrative tasks, they lack the ability to dynamically adapt to individual learners' needs. Intelligent Learning Systems (ILS) bridge this gap by creating adaptive, data-driven learning environments that personalize instructions based on student progress, engagement, and performance patterns. These systems leverage artificial intelligence to adjust learning paths, recommend resources, and foster a more interactive learning experience. Intelligent Tutoring Systems (ITS) take personalization a step further by mimicking the role of a human tutor. Such systems offer semi- or fully automated assessment capabilities and provide real-time feedback that enables students to correct mistakes instantly and supports independent, self-directed learning (Theelen & van Breukelen, 2022). Unlike general ILS, that focus on adaptive learning environments, ITS specializes in providing individualized guidance, diagnosing misconceptions, and tailoring interventions based on student-specific needs. By automating certain aspects of feedback, such as grading, identifying syntax issues, or offering targeted suggestions for improvement, ITS reduce the time-consuming demands on tutors and make it more feasible to provide both timely and high-quality feedback. This ensures that students receive the personalized instruction they need for individual learning even in larger, more diverse classrooms.

Numerous systems have been developed to address the challenges of teaching graphical modeling and providing automated feedback. Several systematic literature reviews offer insights into specific subareas: like tools to teach UML in software engineering education (Huber & Hagel, 2022); user interface design in modeling tools (Ternes et al., 2021); structural and semantic encoding of conceptual models for machine learning applications (Ali et al., 2023); or the integration of didactic principles to enhance learning outcomes (Ullrich et al., 2023). While these reviews contribute valuable perspectives, they collectively lack a holistic view on graphical modeling in education. To provide a consistent comparison of methodologies and technical solutions, this review introduces three core elements grounded in existing ITS literature (Nwana, 1990; Badaracco & Martínez, 2011). The first element, *Learner-System Interaction*, focuses on the modalities through which learners engage with the system. The design of environments has a profound influence on learner engagement and cognitive processing (Badaracco & Martínez, 2011). The structure of these environments not only impacts on the expressiveness and flexibility available to learners but also shapes what the system is able to observe and interpret. Well-designed input interfaces enhance pedagogical responsiveness by enabling meaningful, context-aware interaction (Nwana, 1990). From this line of reasoning, we derive the following research questions:

RQ1.1: How are modeling languages specified and how do these approaches affect system flexibility and applicability?

RQ1.2: How do task design and user interaction methods impact learner engagement and the development of modeling skills?

The second element, *Analysis and Diagnosis*, evaluates how well learner input aligns with expected structural and conceptual standards through syntactic validation and semantic interpretation. Literature

highlights the challenge of interpreting incomplete or atypical input, especially when terminology or reasoning deviates from expert norms (Badaracco & Martínez, 2011). To address this, systems need diagnostic mechanisms that identify both incorrect solutions and underlying misconceptions. Intelligent instructional systems must dynamically analyze learner performance in relation to expert knowledge representations – going beyond simple correctness to uncover the learner's underlying understanding (Nwana, 1990). These challenges lead to the following research questions:

RQ2.1: What methods are commonly employed for syntax checking to ensure models adhere to structural rules?

RQ2.2: How do the systems interpret and handle model semantics, particularly regarding variations in language and terminology used by learners?

RQ2.3: What are the main techniques used for semantic analysis to evaluate the correctness and completeness of learners' models?

The third element, *Instructional Strategy Application*, concerns how systems translate diagnostic insights into meaningful pedagogical responses. A diagnosis, no matter how accurate, is only valuable for learning if it helps to shape the next teaching steps. Accordingly, this step investigates how systems determine appropriate forms of feedback, scaffolding, and task progression. Effective instructional systems must be sensitive to learners' evolving knowledge states, responding not only with corrective feedback but with interventions tailored to cognitive development and individual learning trajectories (Nwana, 1990; Badaracco & Martínez, 2011). Particularly in competency-based systems, instructional strategies must align with the learner's demonstrated level of understanding and guide them toward increasingly complex or professional-level tasks. To capture these dynamics, we investigate the following research questions:

RQ3.1: What types of feedback mechanisms are implemented, and how do they contribute to learners' understanding and improvement in graphical modeling?

RQ3.2: Which teaching strategies are used to teach graphical modeling, and how do they match the progression of student knowledge?

2 Research Design

A systematic approach was employed to conduct a comprehensive literature review on methods and procedures in the automated assessment of graphical modeling. Following the guidelines proposed by vom Brocke et al. (2009), we aimed at providing a state-of-the-art overview of current research in this field. We conducted an extensive literature search, utilizing selected electronic databases as well as forward, backward, and author searches, as recommended by Webster and Watson (2002). The objective of this review is to systematically identify and analyze publications addressing the automated evaluation of graphical models across multiple disciplines. To ensure broad coverage, we did not limit our focus to standard modeling languages in computer science but extended it to include other subject domains. We limited our search to papers published in peer-reviewed journals and conference proceedings that adhere to a blind review process, ensuring the inclusion of rigorously reviewed, high-quality research. A systematic search was conducted using several databases: Google Scholar, ACM Digital Library, IEEE Xplore, Web of Science Core Collection, AIS eLibrary, and ScienceDirect. The search encompassed all publications available up to September 2024 and was restricted to English-language results. Based on initial exploratory searches, which resulted in 24 relevant publications, we developed a generic search string with neutral terms avoiding references to specific modeling languages (e.g. not using UML, BPMN or ERD) in order to capture a wide range of relevant papers: (("graphical" OR "graph-based" OR "visual" OR "conceptual") AND ("model*" OR "diagram*" OR "draw*")) AND (("automatic" OR "automated") AND ("assessment" OR "grading" OR "marking" OR "feedback" OR "comparison" OR "checking" OR "testing" OR "simulation" OR "refactor*")) OR (("e-learning" OR "automated" OR "tutoring" OR "teaching" OR "testing" OR "checking") AND ("system" OR "platform" OR "software")). After removing duplicates, the initial screening – based on titles, keywords, and abstracts – resulted in 258 papers. We then examined these in greater depth focusing on

whether they addressed the automated assessment of graphical models. We then examined these in greater depth, focusing on whether they addressed the automated assessment of graphical models. During this phase, we categorized each publication according to its relevance to the research questions, identifying 101 papers that offered at least a description of the assessment process. By performing additional manual backward and forward searches and author searches we identified 28 additional relevant papers not included in the initial search. As a result, the final sample comprised 129 papers, all of which were subsequently analyzed in relation to the research questions, thereby offering a thorough overview of existing approaches to automated graphical model assessment. In the following, to avoid overrepresentation, systems with a large number of publications have been grouped together, with reference to the most recent or most comprehensive publication. Very frequently used methods are exemplified by selected papers that best illustrate and represent the approach.

3 Results

3.1 RQ1.1: Modeling Language Specification and Integration

The specification and integration of modeling languages are critical factors influencing both system flexibility and the effectiveness of automated assessment. The way modeling languages are defined – whether hard-coded into the system or configurable by users – determines a tool's adaptability to different modeling languages and educational contexts. We identified 38 papers where the specification of the modeling language was discernible – either explicitly described or inferred from implementation details. These papers can be divided into three main categories: use of standard software, custom software using programming libraries, and custom software where the use of libraries is either absent or unspecified. Three tools employ **standard software** with hard-coded modeling languages, offering limited flexibility for adaptation. Hasker and Rowe (2011) and Ali et al. (2007) utilize commercial tools like IBM Rational Rose, which provide predefined modeling languages strictly adhering to the UML standard. The notation and syntax are embedded within the program code, and the data formats store only the elements defined by the software, without support for custom configurations or additional modeling languages. For further processing, XML exports are used, which the assessment software must interpret – introducing dependencies and necessitating adjustments to evaluation methods when changes occur. Hasker and Rowe (2011) also mention the use of Microsoft Visio, which offers various notation elements but does not enforce specific semantics. While this allows some flexibility in diagram creation, it lacks the ability to define custom modeling languages or enforce syntactic and semantic rules within the tool. Schramm et al. (2012) reference ArgoUML, limited to predefined UML standards with fixed notation and syntax, offering no support for custom modeling language definitions. Five papers present tools that leverage existing **software libraries** to provide modeling functionalities. Reischmann and Kuchen (2018) developed a custom tool using the JavaScript library JointJS. Notation and syntax are predefined by programming code, limiting accessibility and adaptability to other modeling languages within their tool. Sánchez-Ferreres et al. (2020) and Garaccione et al. (2024) employ BPMN-js, a JavaScript library implementing BPMN 2.0 with all notation and syntax-rules predefined. Although BPMN-js allows adding custom rules and elements by defining them as additional JavaScript code, this approach necessitates programming expertise, and the core modeling language remains BPMN, limiting the tool's applicability to other modeling languages. The majority of tools – 30 papers – are **custom-developed software** solutions with one or two modeling language hard-coded into the system, offering limited or no flexibility for adaptation to other languages. Tools such as Collect-UML (Baghaei et al., 2007) and TouchCORE (Bian et al., 2020; Boubekeur et al., 2020) are narrowly focused, supporting only UML class diagrams. Other papers used similar approaches, focusing on database models, UML diagrams, deterministic finite automaton or circuit boards. In contrast to the predominantly rigid systems, four papers have designed tools with flexibility in mind, allowing for the definition and integration of new modeling languages without modifying the program code. Tselonis et al. (2005) describe the ABC-System, which includes a customizable palette and drawing area for diagrams. The system treats the diagram type independently of the infrastructure, allowing examiners to customize the palette based on the diagram expected in the student's answer.

Correia et al. (2018) introduce a tool based on a Diagrammatic Language Configuration designed to be extensible and capable of incorporating new kinds of diagrams. Their system uses a configuration file following the Diagrammatic Language Definition Language (DL2), specifying features and feedback used in syntax validation. Users can define nodes, edges, and language constraints through configuration files, enabling the addition of new modeling languages on-the-fly without recompiling or redeploying the system. By separating the modeling language definition from the program code, this method enhances flexibility and allows non-programmers to define modeling languages, potentially through a user interface that facilitates configuration.

3.2 RQ1.2: Task Design and User Interaction

3.2.1 Presentation and Learning Cases of Tasks

Task presentation is mostly uniform, typically involving static case-based texts that simulate real-world scenarios. Learners are given the full task upfront and are expected to model solutions from scratch based on the provided requirements. Only one system deviates from the standard task format: Schildgen (2020) introduces a gamified approach in which tasks are revealed step by step, allowing learners to incrementally adapt their models. This reduces the cognitive load by breaking the task into manageable parts. While most systems rely on static task descriptions, there is variation in how tasks are presented within the learning environment. Some present tasks externally, requiring learners to switch between different interfaces – an approach that reflects real-world conditions but may limit immediate guidance and feedback (Ali et al., 2007; Schramm et al., 2012). Others embed tasks directly within the modeling tool, offering a more seamless and supportive learning experience (Foss et al., 2022; Sánchez-Ferreres et al., 2020). In certain cases, task descriptions are further enhanced through formatting and highlighting, helping learners focus on key information and strengthening the connection between task and modeling activity (Baghaei et al., 2005; Foss et al., 2022).

3.2.2 User Interaction

Concerning user interactions, we identified three levels of autonomy: low autonomy, where users rely on structured support; moderate autonomy, where guidance and independence are balanced; and high autonomy, where users operate within minimal restrictions. *Low autonomy* environments are characterized by structured inputs that guide users through predefined pathways, limiting independent model creation. Batmaz and Hinde (2007) and Schildgen (2020) developed frameworks that guide learners through input forms and buttons, ensuring interaction with predefined components while limiting independent modeling. Similarly, Foss et al. (2022) restrict user input to structured interactions aligned with the given task description, reducing the need for learners to independently organize their models. *Moderate autonomy* environments offer a structured yet flexible approach to model creation, balancing guidance with user control. While learners are not completely free to build models from scratch, they are not restricted to rigid input forms. Instead, they work within guided frameworks that enable meaningful interaction while ensuring correctness. Hall and Gordon (1998) and Ivanchikj et al. (2020) proposed text-based input methods for model generation. These systems enable learners to input textual descriptions, which are subsequently translated into graphical models in real-time. Although learners do not directly manipulate graphical elements, they remain actively involved in defining model content through text. The Collect-UML system of Baghaei et al. (2005) takes the opposite approach by giving learners the freedom to select modeling elements and position them within a canvas. However, this flexibility is deliberately restricted. Rather than defining classes, attributes and relationships individually, users extract relevant components directly from the provided text, highlighting key words and phrases. As this highly restricts the modeling process, Ternes et al. (2020) focus on embedding natural language processing (NLP) techniques into their modeling environment, offering real-time suggestions for naming model elements. Their approach not only promotes more consistent semantic labelling but also reduces the cognitive load on students by automating the selection of contextually relevant terms. Beyond these approaches, most systems emphasize modeling within a structured syntax and notation environment, providing learners with a modeling canvas and a set of standardized

elements tailored to the chosen modeling language (e.g. Correia et al., 2018; Vachharajani et al., 2012; Ternes et al., 2020). These systems allow flexible element arrangement within syntactic constraints, fostering the development of modeling skills within formal structural guidelines. *High autonomy* environments provide learners with greater flexibility in model creation, allowing for independent decision-making while minimizing structured guidance. Smith et al. (2013) developed a system where the syntax checking can be toggled on or off by tutors, allowing them to tailor the level of syntactic enforcement based on the learners' needs. However, for novice learners, this approach may present challenges due to the knowledge required to navigate the system syntax and element selection effectively.

3.3 RQ2.1: Syntax Checking

Syntax checking ensures that the elements of a model are correctly composed in accordance with the formal rules of the modeling language. It focuses on the proper arrangement and interconnection of components, addressing their underlying meanings or semantics. Ensuring syntactic correctness is essential to prevent structural errors that could impede further analysis. Nevertheless, many systems do not implement explicit syntax validation. Only 24 papers reported performing syntax checks, with 13 offering no details on their implementation. This omission is due to reliance on external modeling tools that inherently enforce syntactic correctness or an assumption that the models created by learners are already syntactically correct. For example, Ali et al. (2007), Waugh et al. (2007), and Py et al. (2010) depend on external tools for syntax validation, operating under the assumption that the input models are correct. Other papers explicitly implement syntax checking within their systems. Batmaz and Hinde (2007) and Baghaei et al. (2005) perform syntax checking using manually defined constraints to ensure adherence to the modeling language's syntax rules. By defining specific constraints, these systems evaluate the learner's model against formal syntax requirements, providing immediate feedback on any violations. Furthermore, some papers employ formal methods for syntax validation. Siepermann et al. (2008) defines the syntax of diagrams using temporal logic, enabling precise and formal verification of syntactic rules within the modeling environment. Although syntax checking is considered a solved problem, efficiently addressed through established methods like model checking or simple rule definitions, it remains highly relevant, especially in the context of more flexible systems where the syntax of new modeling languages needs to be defined. In such systems, syntax rules must be specified without hard coding them, necessitating adaptable syntax checking mechanisms that can accommodate the definition of new modeling languages and their associated syntax rules.

3.4 RQ2.2: Semantic Interpretation

While syntax checking ensures structural correctness, semantic analysis focuses on the meaning conveyed by model elements. In low-autonomy systems, this is less of a challenge, as learners work within tightly controlled environments that enforce standardized terminology. However, as autonomy increases, systems need to deal with the variability in terminology by identifying and interpreting underlying concepts, handling synonyms and unfamiliar vocabulary, and correcting spelling errors. Such robust semantic interpretation is essential to support accurate model evaluation, constructive feedback and deeper analysis. A wide range of approaches have been proposed to address these challenges, ranging from basic lexical standardization to more advanced NLP. Ali et al. (2007) illustrate a basic strategy by enforcing language-specific naming conventions in UML diagrams (e.g. using nouns for attribute names), thereby minimizing semantic inconsistencies caused by misnaming. Jayal and Shepperd (2009) extend this principle with text preprocessing methods – such as punctuation removal, case normalization, stop word filtering, and stemming – that further reduce labelling inconsistencies introduced by misspellings, abbreviations, or different morphological forms. Vachharajani et al. (2012) propose a label matching engine that combines spelling correction, stemming, phonetic similarity (Soundex), and both general and domain-specific synonym databases to robustly match labels even when diagrams contain spelling errors or technical domain vocabulary. Similarly, Jayawardena et al. (2018) integrate WordNet, a comprehensive lexical database, and the Natural Language Toolkit (NLTK) to compute text similarity scores – handling synonyms, misspellings, and semantic equivalents through tokenization and part-of-speech tagging. This ensures that nodes expressing the same

meaning are recognized as matches. Thomas et al. (2007) underscore the importance of context in synonym detection, hyponymy, and supertype – subtype relationships within ER diagrams. By applying weights and thresholds to diagram features, they capture subtle semantic distinctions – such as "staff" versus "employee" – and refine matching precision.

3.5 RQ2.3: Semantic Analysis

Semantic analysis evaluates whether a model conveys the intended concepts and relationships. It addresses the challenge of interpreting the meaning behind models by accounting for variations in modeling approaches, terminologies, and potential errors. A key difficulty is the inherent diversity of valid modeling solutions, which hinders one-to-one comparisons. We identified 87 papers describing semantic evaluation methods, including 70 white box and 17 black box approaches. *White box* approaches analyze a model's structure by comparing it with reference solutions or predefined rules, requiring either a complete reference or rule set. We identified three main approaches in the field: rule-based (20), model-based (42) and machine learning techniques (8). *Rule-based* methods rely on predefined rules or constraints to assess the semantic correctness of a model. This approach evaluates a student's model by checking for violations of accepted modeling practices. Baghaei et al. (2007) and Batmaz and Hinde (2007) defined specific constraints for tasks, with their systems flagging any deviations in the student's model. Similarly, Siepermann et al. (2008) applied temporal logic and model-checking. There, students' models are converted into a Kripke structure and evaluated for compliance with formalized requirements. Rule-based methods are time-consuming to set up, as they rely on many manually created configurations. However, they provide precise feedback by identifying the exact errors in the student's model. *Model-based* evaluation compares the student's model directly with one or more reference solutions. Fauzan et al. (2021) use similarity measurements to calculate semantic similarity by comparing properties and relationships, with components weighted according to importance. Coelho and Murphy (2007) implemented graph-matching techniques to locate structural patterns, checking for design features or anomalies. Smith et al. (2013) further refined this approach by analyzing diagrams as graphs of nodes and arcs, matching minimal meaningful units (MMUs) within the student's model to the reference model. This method accommodates errors, omissions, and extraneous items. Foss et al. (2022) employed a stepwise comparison method for ER diagrams, supporting flexible cardinality constraints for tasks with multiple valid answers by systematically matching entities, attributes, and relationships. Additional papers explored Graph Edit Distance (GED) and metric comparison methods to quantify similarity. GED-based methods measure the transformations needed to make one graph resemble another by accounting for modifications, including vertex and edge additions, deletions, and substitutions. Arifin and Siahaan (2020) used inexact graph matching with GED to evaluate their diagrams, providing a quantitative assessment of differences from the reference model. Metric comparison methods, such as those employed by Tselonis et al. (2005), apply heuristic approaches to calculate similarity scores based on attributes like component type, degree, label, and adjacency. These scores are weighted and averaged, allowing flexibility in evaluating student models while maintaining overall similarity to the reference model. *Machine learning* approaches for model evaluation leverage data-driven methods to improve the assessment process, particularly in cases where traditional rule-based or reference model approaches are impractical or time and computing power intensive. Krusche (2022) developed a semi-automated assessment system using supervised machine learning. Initially, all submissions are manually reviewed, as the system lacks training data. It learns from these assessments by clustering similar model elements based on type, name, and context (e.g., using Levenshtein distance). Feedback for one element is then reused for similar elements within a cluster. Human oversight remains essential to ensure contextual accuracy and detect missing elements. Lino and Rocha (2018) developed a machine learning-based system to evaluate ER diagrams by translating them into SQL schemas, extracting metrics, and combining them with diagram-specific features. Models such as neural networks, genetic algorithms, and SVMs were used to predict grades, with neural networks achieving the highest accuracy. Boubekeur et al. (2020) proposed a lightweight approach for the automated assessment of UML class diagrams by combining a simple heuristic with machine learning techniques. Their method achieved up to 89% accuracy in binary quality classifica-

tion and correctly predicted letter grades within one grade level in 86% of cases. However, most rule- and model-based approaches in the reviewed literature do not report formal metrics for accuracy or reliability, making it difficult to evaluate their effectiveness. While most machine learning approaches aim to automate the evaluation of student models, Sánchez-Ferreres et al. (2020) use machine learning to address a core limitation of rule- and model-based methods: their difficulty in handling the diversity of valid solutions. By applying NLP, their system annotates key elements of textual task descriptions, enabling the semi-automated generation of flexible reference models. This hybrid approach blends the structured accuracy of rule- and model-based evaluation with the adaptability of data-driven techniques, providing a scalable solution that better reflects diverse responses. *Black box* approaches focus on evaluating the observable behavior of a system without examining its internal structure, making them particularly suitable for educational applications where the focus is on understanding system functionality. In the domain of electrical circuits, several papers have developed simulation-based systems that enable students to explore circuit behavior dynamically, fostering an understanding of the impact of various parameters and configurations. Pănoiu et al. (2010), Burch (2002), and White and Frederiksen (1990) developed tools to manipulate circuit parameters and observe the effects on system behavior, thus supporting learning without requiring detailed insight into circuit internals. Pănoiu et al. (2010) provides a tool that targets power electronics, allowing students to adjust parameters of rectifiers, inverters, and other components to observe the impact on current and voltage waveforms, with a particular emphasis on control variables like rectifier control angles. Burch (2002) enabling students to construct circuits with logic gates and test configurations, including the use of black-boxed circuits within more complex designs. This feature allows students to work with modular designs and reinforce the principles of hierarchical circuit construction. White and Frederiksen (1990) use a tool that emphasizes conceptual understanding by offering causal feedback on circuit configurations, enabling students to troubleshoot and experiment to deepen their understanding of electrical principles. Ogata and Kayama (2019) introduced the SML4I tool, which supports state machine diagrams in UML. This tool allows students to simulate state machine models by inputting event sequences, with the tool displaying transitions and current states. The simplicity of setup and use makes it ideal for educational settings, as students can immediately interact with their models and see state transitions in action. While these systems support intuitive exploration, they often lack mechanisms for structured feedback. Their black box nature limits their ability to diagnose misconceptions or provide targeted guidance based on learners' modeling decisions.

3.6 RQ3.1: Feedback Generation and Communication

Feedback helps learners identify errors and understand misconceptions. It appears in various forms – from simple numerical scores to detailed textual explanations and visual cues. Among the reviewed papers, 61 implemented feedback mechanisms, spanning from summative approaches that provide overall scores or grades to formative strategies that guide learners through targeted, progress-oriented support. Some papers also incorporated gamification elements to enhance motivation. We classified the feedback mechanisms into three main categories: numerical (30), textual (33), and visual (13), with several papers combining multiple types for better support. *Numerical feedback* provides quantitative assessments such as grades or scores. It is a quick performance indication for learners but often lacks detailed information about specific errors or areas for improvement. Several papers employed numerical feedback as their primary mechanism. Siepermann et al. (2008) and Lino and Rocha (2018) provided grades based on modeling submissions. Bian et al. (2020) used points to quantify the correctness of models, while Fauzan et al. (2021) utilized scales to represent assessment outcomes. An extension of this approach is presented by Foss et al. (2022), who implement differentiated scoring metrics for individual model components, providing a point distribution that highlights specific strengths and weaknesses more effectively than aggregate scores. A unique numerical feedback mechanism is introduced by Siepermann (2016), who replaces traditional grades with a virtual currency in a gamified approach. The amount awarded varies according to task difficulty and model quality, with optional time-based penalties for a longer completion time. *Textual feedback* provides written comments, often highlighting errors or suggesting improvements. Unlike numerical feedback, it addresses

issues within the learner's model and is typically generated from predefined text elements. It can contain global statements about the diagram or specific comments with placeholders for missing or unnecessary element names. Ali et al. (2007) and Soler et al. (2010) used predefined text modules for each error, combining static and dynamic feedback that inserts specific details (e.g., incorrect attribute or class names) for personalized guidance. Hoggarth and Lockyer (1998) provided two types of textual feedback: specific error documentation, such as missing flows in the model, and highlighting differences compared to the reference solution. These are presented as lists of comments to help learners understand discrepancies between their submissions and expected outcomes. *Visual feedback* involves graphical indications within the modeling environment to highlight incorrect elements, display icons, or change visual attributes of model components. It is often combined with numerical or textual feedback to enhance effectiveness. Siepermann et al. (2013) developed an interface combining numerical, textual, and visual feedback. Errors are marked directly on the model with hints, and overall performance is indicated through a total score. Garacci et al. (2024) applied visual feedback by coloring diagram elements: green for correct parts, purple for wrong order, yellow for near matches, and altered borders for misplaced or misnamed elements.

3.7 RQ3.2: Instruction Models and Adaptive Improvement

The way feedback interacts with learning processes directly influences student progression and engagement. We identified 54 papers that describe or propose instructional strategies for teaching graphical modeling. As summative assessments offer limited learning value, we focus on approaches that actively foster knowledge and skill development. Several papers reference structured learning models, such as Bloom's Taxonomy (Krusche, 2022), which structures learning into hierarchical cognitive stages, requiring students to first understand, then analyze, and finally create solutions. Similarly, the four-stage learning model guides learners through a structured progression: beginning with exposition (theoretical input), followed by elicitation (practical exercises using modeling tools), verification (feedback-based refinement), and concluding with autonomous application (independent modeling tasks) (Hoggarth & Lockyer, 1998). While these methods establish a solid foundation, they do not actively propose adjustments to students' learning needs. To address this limitation, several systems implement adaptive feedback mechanisms with iterative learning loops, where feedback is dynamically adapted to student performance. Baghaei et al. (2007) propose an error-tracking system that reduces redundant feedback on already mastered concepts, while still offering detailed guidance on new or persistent mistakes. Correia et al. (2018) implement progressive feedback adaptation, where initial mistakes trigger detailed explanations, but repeated errors receive gradually refined hints, encouraging independent problem-solving. Thomas et al. (2009) and Garaccione et al. (2024) introduce progressive task complexity, where exercises become increasingly challenging as students demonstrate proficiency. This ensures a scaffolded learning experience that prevents students from stagnating at an easy level while avoiding excessive difficulty jumps. In addition, some systems which consider personalized learning paths based on historical performance data. Suraweera and Mitrovic (2002) use constraint-based student models to track mastery of specific modeling rules, ensuring feedback is targeted toward unresolved learning gaps. Similarly, Sanchez-Ferreres et al. (2020) and Siepermann et al. (2013) emphasize progress tracking through model versioning, allowing students to compare past solutions, fostering self-reflection and deeper conceptual understanding. To further enhance engagement, some papers propose gamification-based learning approaches. Garaccione et al. (2024) and Siepermann (2016) integrate experience points systems, level-based progression, and adaptive hints, motivating students through game-like reward structures. As these approaches fit well with modern digital learning environments, intrinsic motivation is a critical factor in keeping students in the feedback loop.

4 Discussion

4.1 Critical comments

Teaching graphical modeling with automated assessment is a well-established research area, with numerous studies presenting a variety of approaches. However, most of the systems remain rigid in their

implementation of modeling languages, as they focus on teaching one language. Only one approach successfully decouples the modeling logic from the core system, allowing seamless extensions to incorporate new modeling languages (Correia et al., 2018). In terms of user interaction, most systems employ static task descriptions without adapting to the learner's current proficiency level. Only one approach utilizes gamification to reduce cognitive load by structuring tasks into smaller, incremental steps (Schildgen, 2020). Regarding the modeling process, we identified three levels of autonomy, ranging from highly guided environments with strict guidance to more autonomous systems that promote advanced modeling skills. However, no paper proposes a fully adaptive system that dynamically adjusts the level of interaction based on the learner's progress. As the degree of autonomy also influences the analysis and diagnosis mechanisms, systems with less autonomy often bypass complex evaluation challenges, while systems with more flexibility must interpret and analyze both syntax and semantics to provide meaningful evaluations. Within semantic analysis, white box testing approaches are predominant due to their effectiveness in generating reliable feedback. While rule- and model-based evaluations dominate this space, machine learning techniques – despite their increasing relevance in other domains – remain significantly underrepresented, indicating a gap between current research and state-of-the-art advancements. Most papers adopt a traditional instructional model for feedback and instruction, assuming a linear process in which students receive feedback, revise their work, and submit it for further review. Based on these mechanisms most systems remain largely static, offering numerical, textual, or visual feedback without fostering real-time interaction between students and the system. However, some approaches attempt to address this limitation by integrating adaptive mechanisms that dynamically adjust feedback and personalize learning paths. However, one major limitation remains: Unlike generative language models such as ChatGPT, existing systems are not able to facilitate a direct, interactive feedback exchange, which prevents students from asking specific follow-up questions to deepen their understanding. Given the importance of this capability – both for enhancing learning outcomes and reducing the likelihood of students turning to external AI tools for direct solutions – we propose several directions for future development in this area.

4.2 Recommendations

Many papers propose advanced approaches, none provides a comprehensive solution. By synthesizing their strengths, we propose an adaptive learning environment for graphical modeling, based on the three core elements introduced earlier. *Learner System Interaction.* Building on existing approaches to reduce cognitive load in the initial learning phase, we propose a system that incrementally presents both task complexity and descriptions. To further enhance adaptability, the system offers varying levels of autonomy based on individual proficiency. Beginners benefit from restricted modeling palettes and structured guidance, helping them develop foundational skills within clear boundaries. In contrast, advanced learners gain access to open-ended modeling environments that foster creativity and problem-solving. This scaffolded learning approach, supported by extensive research, enables deeper comprehension and minimizes conceptual errors, ensuring a more effective and personalized learning process. As part of this, we suggest a language-independent system supporting diverse modeling notations. While this adds flexibility, it also introduces challenges: abstracting interaction across different syntactic and semantic rules may increase complexity and risk oversimplification. Ensuring usability and pedagogical clarity requires careful interface and rule design. Still, studies like Correia et al. (2018) show that such implementations are feasible when logic and interface are cleanly separated. *Analysis and Diagnosis.* Building on the innovative approach of Sanchez-Ferreres et al. (2020), which aligns task descriptions with process models, current developments in NLP could unify existing methodologies. By identifying syntactic structures, semantic roles, and dependencies within student submissions, they enable a comprehensive framework for automated model validation. However, the presented approach is primarily effective for structured process models, where descriptions closely follow the reading structure. Other modeling paradigms that rely on implicit representations do not map directly to textual descriptions, making automated validation more complex. These challenges become even more pronounced in language-independent systems, where diagnostic mechanisms must accommodate diverse syntactic rules and semantic conventions. Designing such flexible yet precise valida-

tion frameworks remain a key technical hurdle. Nevertheless, advancements in large language models (LLMs), such as GPT-o1, offer new possibilities for bridging this gap. With enhanced reasoning and contextual understanding, LLMs can infer implicit structures, generate missing links, and provide adaptive suggestions, improving automated validation across various domains. Ma et al. (2023) demonstrated that while traditional NLP struggled with generating diverse and high-quality conceptual designs, LLMs like GPT produced feasible, human-like solutions from textual descriptions. By leveraging few-shot learning, these models not only enhanced adaptability but also generated outputs more aligned with human reasoning, showing their potential to unify structured and unstructured modeling approaches. *Instructional Strategy Application.* The effectiveness of an ITS extends beyond its ability to analyze and diagnose student submissions; it also depends on how effectively it integrates feedback into an adaptive instructional strategy. As suggested by several reviewed studies, we propose leveraging adaptive feedback loops, where beginners receive detailed, explanatory guidance, while advanced learners are provided with concise hints that encourage independent problem-solving. However, despite efforts to simulate human tutoring, existing systems still struggle to achieve truly interactive and meaningful communication. The reviewed papers often rely on a rigid cycle of submission, correction, and resubmission, lacking the dialogic depth necessary for fostering deeper engagement and reflection. Recent advancements in LLMs, such as Chat-GPT, offer new opportunities to enhance feedback communication in ITS by enabling interactive dialogue, clarifying misconceptions, and guiding reasoning. Unlike static feedback, LLMs can generate context-aware explanations and ask follow-up questions, mimicking human tutoring. However, studies indicate that unrestricted use of LLMs can lead to shallow knowledge retention and hinder problem-solving skills (Adeshola & Adepoju, 2023). This underscores the importance of distinguishing between AI-generated answers – which often replace the learning process – and formative feedback aimed at guiding students through reflection on their own modeling attempts. As Bransford et al. (2000) emphasize, meaningful learning is demonstrated not merely through correctness, but through transfer – the ability to apply acquired knowledge to novel and unfamiliar contexts. Recent papers emphasize the role of prompt engineering in structuring AI-generated feedback to encourage reflection, provide adaptive hints, and maintain pedagogical intent, ensuring that learners actively engage with the material rather than passively receiving solutions (e.g. Mzwri & Turcsányi-Szabo, 2025). Nevertheless, the risk of LLMs generating incorrect information remains, potentially reinforcing misconceptions if students rely on faulty outputs. To mitigate this, an educator-in-the-loop approach becomes essential, ensuring that human oversight guides and corrects AI-generated content. As Holmes et al. (2021) emphasize, artificial intelligence in education must not only be ethical by intention, but also by design. They argue that pedagogical choices encoded in AI systems require human oversight to account for unintended consequences and to ensure ethical learning experiences. This includes critical decisions about the accuracy of AI-generated feedback, fairness, and the role of technology in complementing rather than replacing teachers. Additionally, the impact of reduced human interaction in AI-driven learning environments is still underexplored, raising concerns about motivation, emotional support, and cognitive development. Despite advancements in LLMs, their implementation requires significant ongoing costs and substantial technical and financial investment, particularly for personalized solutions. In addition, institutional resistance can hinder adoption – often driven by concerns over data privacy, lack of trust in AI-based systems, and limited staff training (Holmes et al., 2021; Nwana, 1990). Addressing this requires transparent system design, educator involvement in development, and professional training to build acceptance and ensure effective integration.

4.3 Future Research

Future research should investigate how language models can enhance automated assessment in graphical modeling through *modular AI architectures* that separate assessment from pedagogical interaction. Domain-specific expert modules can handle assessment tasks such as syntax validation and semantic analysis, generating diagnostics that are subsequently interpreted by an LLM acting as a teaching interface. LLMs are ideal for interactive learner support, providing real-time, dialog-based feedback through which learners can ask questions, explore alternatives, and receive tailored, pedagogically

meaningful guidance. They can also support adaptive autonomy control, dynamically adjusting task complexity or scaffolding based on learner progress using rule-based or learning-based meta-agents. In addition, micro-simulations powered by LLMs could help visualize the downstream effects of modeling decisions. However, LLMs are less suitable for evaluation tasks due to limited transparency and nondeterministic behavior, which are critical concerns in educational contexts where fairness and reproducibility are essential. Small Language Models (SLMs) provide an efficient, lightweight alternative for targeted NLP tasks such as entity extraction or label normalization. They can be developed and deployed locally and provide greater control over model behavior, resulting in greater consistency. SLMs can also be fine-tuned to specific modeling languages or educational contexts, either on-demand when a new modeling notation is integrated into the system, or through a more generalized approach in which the model is trained to interpret configuration data such as rule sets and task descriptions.

References

Adeshola, I., & Adepoju, A. P. (2023). The opportunities and challenges of ChatGPT in education. *Interactive Learning Environments*, *32*(10), 6159–6172.

Ajogbeje, O. J. (2023). Enhancing Classroom Learning Outcomes: The Power of Immediate Feedback Strategy. *International Journal of Disabilities Sports and Health Sciences*, *6*(3), 453–465.

Ali, N. H., Shukur, Z., & Idris, S. (2007). A Design of an Assessment System for UML Class Diagram. *2007 International Conference on Computational Science and Its Applications (ICCSA 2007)*, 539–546.

Ali, S. J., Gavric, A., Proper, H., & Bork, D. (2023). Encoding Conceptual Models for Machine Learning: A Systematic Review. *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 562–570.

Arifin, M. N., & Siahaan, D. (2020). Structural and Semantic Similarity Measurement of UML Use Case Diagram. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, *11*(2), 88-100.

Badaracco, M., & Martínez, L. (2011). An Intelligent Tutoring System Architecture for Competency-Based Learning. In A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, & L. C. Jain (Eds.), *Knowlege-Based and Intelligent Information and Engineering Systems* (pp. 124–133). Springer.

Baghaei, N., Mitrovic, A., & Irwin, W. (2005). A Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML. *ICCE*, *2005*, 11–18.

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of Computer-Supported Collaborative Learning*, *2*(2–3), 159–190.

Batmaz, F., & Hinde, C. J. (2007). A web-based semi-automatic assessment tool for conceptual database diagrams. *Proceedings of the Sixth IASTED International Conference on Web-Based Education*, 427–432.

Bian, W., Alam, O., & Kienzle, J. (2020). Is automated grading of models effective?: Assessing automated grading of class diagrams. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 365–376.

Boubekeur, Y., Mussbacher, G., & McIntosh, S. (2020). Automatic assessment of students' software models using a simple heuristic and machine learning. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 1–10.

Bransford, J., Brophy, S., & Williams, S. (2000). When Computer Technologies Meet the Learning Sciences: Issues and Opportunities. *Journal of Applied Developmental Psychology*, *21*(1), 59–84.

Burch, C. (2002). Logisim: A graphical system for logic circuit design and simulation. *J. Educ. Resour. Comput.*, *2*(1), 5–16.

Canals, L., Granena, G., Yilmaz, Y., & Malicka, A. (2025). The relative effectiveness of immediate and delayed corrective feedback in video-based computer-mediated communication. *Language Teaching Research*, *29*(1), 242–268.

Cherfi, S. S.-S., Akoka, J., & Comyn-Wattiau, I. (2002). Conceptual Modeling Quality – From EER to UML Schemas Evaluation. In S. Spaccapietra, S. T. March, & Y. Kambayashi (Eds.), *Conceptual Modeling – ER 2002,* 2503, 414–428. Springer Berlin Heidelberg.

Chou, C.-Y., & Zou, N.-B. (2020). An analysis of internal and external feedback in self-regulated learning activities mediated by self-regulated learning tools and open learner models. *International Journal of Educational Technology in Higher Education*, *17*(1), 55.

Chren, S., Buhnova, B., Macak, M., Daubner, L., & Rossi, B. (2019). Mistakes in UML Diagrams: Analysis of Student Projects in a Software Engineering Course. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 100–109.

Coelho, W., & Murphy, G. (2007). ClassCompass: A software design mentoring system. *Journal on Educational Resources in Computing*, *7*(1), 2.

Correia, H., Leal, J. P., & Paiva, J. C. (2018). Improving Diagram Assessment in Mooshak. In E. Ras & A. E. Guerrero Roldán (Eds.), *Technology Enhanced Assessment*, 829, 69–82. Springer International Publishing.

Eppler, M., & Burkhard, R. (2007). Visual representations in knowledge management: Framework and cases. *J. Knowledge Management*, *11*, 112–122.

Fauzan, R., Siahaan, D., Rochimah, S., & Triandini, E. (2021). A Different Approach on Automated Use Case Diagram Semantic Assessment. *International Journal of Intelligent Engineering and Systems*, *14*(1), 496–505.

Foss, S., Urazova, T., & Lawrence, R. (2022). Automatic Generation and Marking of UML Database Design Diagrams. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, 626–632.

Garaccione, G., Coppola, R., & Ardito, L. (2024). Gamifying Business Process Modeling Education: A Longitudinal Study. *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, 580–589.

Hall, L., & Gordon, A. (1998). A virtual learning environment for entity relationship modelling. *ACM SIGCSE Bulletin*, *30*(1), 345–349.

Hasker, R., & Rowe, M. (2011). UMLint: Identifying Defects in UML Diagrams. *118th Annual Conference of the American Society for Engineering Education*, 22.1558.1-22.1558.14.

Hattie, J., & Timperley, H. (2007). The Power of Feedback. *Review of Educational Research*, *77*(1), 81–112.

Hoggarth, G., & Lockyer, M. (1998). An automated student diagram assessment system. *ACM SIGCSE Bulletin*, *30*(3), 122–124.

Holmes, W., Porayska-Pomsta, K., Holstein, K., Sutherland, E., Baker, T., Shum, S. B., Santos, O. C., Rodrigo, M. T., Cukurova, M., Bittencourt, I. I., & Koedinger, K. R. (2021). Ethics of AI in Education: Towards a Community-Wide Framework. *International Journal of Artificial Intelligence in Education*, *32*(3), 504–526.

Huber, F., & Hagel, G. (2022). Tool-supported teaching of UML diagrams in software engineering education—A systematic literature review. *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, 1404–1409.

Ivanchikj, A., Serbout, S., & Pautasso, C. (2020). From text to visual BPMN process models: Design and evaluation. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 229–239.

Jayal, A., & Shepperd, M. (2009). The Problem of Labels in E-Assessment of Diagrams. *Journal on Educational Resources in Computing*, *8*(4), 1–13.

Jayawardena, R. R. A. M. P., Thiwanthi, G. A. D., Suriyaarachchi, P. S., Withana, K. I., & Jayawardena, C. (2018). Automated Exam Paper Marking System for Structured Questions and Block Diagrams. *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, 1–5.

Jordan, M. I. (2004). Graphical Models. *Statistical Science*, *19*(1), 140–155.

Karagiannis, D., & Kühn, H. (2002). Metamodelling Platforms. In K. Bauknecht, A. M. Tjoa, & G. Quirchmayr (Eds.), *E-Commerce and Web Technologies*, 2455, 182–182. Springer Berlin Heidelberg.

Krusche, S. (2022). Semi-Automatic Assessment of Modeling Exercises using Supervised Machine Learning. *Proceedings of the 55th Hawaii International Conference on System Sciences*, 871–880.

Liebel, G., Heldal, R., & Steghöfer, J.-P. (2016). Impact of the Use of Industrial Modelling Tools on Modelling Education. *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, 18–27.

Lindland, O. I., Sindre, G., & Solvberg, A. (1994). Understanding quality in conceptual modeling. *IEEE Software*, *11*(2), 42–49.

Lino, A., & Rocha, A. (2018). Automatic evaluation of ERD in e-learning environments. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–5.

Ma, K., Grandi, D., McComb, C., & Goucher-Lambert, K. (2023). Conceptual Design Generation Using Large Language Models. *Volume 6: 35th International Conference on Design Theory and Methodology (DTM)*. Proceedings of the ASME 2023 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference., Boston, Massachusetts, USA.

Maathuis, M., Drton, M., Lauritzen, S., & Wainwright, M. (Eds.). (2018). *Handbook of Graphical Models*. CRC Press.

McInerny, G. J., Chen, M., Freeman, R., Gavaghan, D., Meyer, M., Rowland, F., Spiegelhalter, D. J., Stefaner, M., Tessarolo, G., & Hortal, J. (2014). Information visualisation for science and policy: Engaging users and avoiding bias. *Trends in Ecology & Evolution*, *29*(3), 148–157.

Mzwri, K., & Turcsányi-Szabo, M. (2025). The Impact of Prompt Engineering and a Generative AI-Driven Tool on Autonomous Learning: A Case Study. *Education Sciences*, *15*(2), 199.

Nwana, H. S. (1990). Intelligent tutoring systems: An overview. *Artificial Intelligence Review*, *4*(4), 251–277.

Ogata, S., & Kayama, M. (2019). SML4C: Fully Automatic Classification of State Machine Models for Model Inspection in Education. *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 720–729.

Pănoiu, C., Pănoiu, M., Muscalagiu, I., & Iordan, A. (2010). Visual interactive environment for study the power electronics using PSCAD-EMTDC simulation program. *Computer Applications in Engineering Education*, *18*(3), 469–475.

Py, D., Auxepaules, L., & Alonso, M. (2010). *Diagram, a Learning Environment for Initiation to Object-Oriented Modeling with UML Class Diagrams*.

Reischmann, T., & Kuchen, H. (2018). An Interactive Learning Environment for Software Engineering Design Patterns. *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, 1–2.

Reuter, R., Stark, T., Sedelmaier, Y., Landes, D., Mottok, J., & Wolff, C. (2020). Insights in Students' Problems during UML Modeling. *2020 IEEE Global Engineering Education Conference (EDUCON)*, 592–600.

Sanchez-Ferreres, J., Delicado, L., Andaloussi, A. A., Burattin, A., Calderon-Ruiz, G., Weber, B., Carmona, J., & Padro, L. (2020). Supporting the Process of Learning and Teaching Process Models. *IEEE Transactions on Learning Technologies*, *13*(3), 552–566.

Schildgen, J. (2020). MonstER Park—The Entity-Relationship-Diagram Learning Game. *Proc. 39th Int. Conf. Conceptual Modeling*, 150–157.

Schramm, J., Strickroth, S., Le, N.-T., & Pinkwart, N. (2012). Teaching UML Skills to Novice Programmers Using a Sample Solution Based Intelligent Tutoring System. *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, 472–477.

Sedlmair, M., Meyer, M., & Munzner, T. (2012). Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Transactions on Visualization and Computer Graphics*, *18*(12), 2431–2440. IEEE Transactions on Visualization and Computer Graphics.

Siepermann, M. (2016). Database Engineering Game. In R. Bottino, J. Jeuring, & R. C. Veltkamp (Eds.), *Games and Learning Alliance*,10056, 70–79. Springer International Publishing.

Siepermann, M., Lackes, R., & Börgermann, C. (2008). Using Model Checking to Automatically Mark Graphical E-Learning Exercises. *Proceedings of EdMedia + Innovate Learning 2008*, 5302–5307.

Siepermann, M., Siepermann, C., & Lackes, R. (2013). Electronic Exercises for the Metra Potential Method: *Proceedings of the 15th International Conference on Enterprise Information Systems*, 435–442.

Smith, N., Thomas, P., & Waugh, K. (2013). Automatic Grading of Free-Form Diagrams with Label Hypernymy. *2013 Learning and Teaching in Computing and Engineering*, 136–142.

Soler, J., Boada, I., Prados, F., Poch, J., & Fabregat, R. (2010). A web-based e-learning tool for UML class diagrams. *IEEE EDUCON 2010 Conference*, 973–979.

Soyka, C., Schaper, N., Bender, E., Striewe, M., & Ullrich, M. (2022). Toward a Competence Model for Graphical Modeling. *ACM Trans. Comput. Educ.*, *23*(1), 15:1-15:30.

Stikkolorum, D. R., De Oliveira Neto, F. G., & Chaudron, M. R. V. (2018). Evaluating Didactic Approaches used by Teaching Assistants for Software Analysis and Design using UML. *Proceedings of the 3rd European Conference of Software Engineering Education*, 122–131.

Suraweera, P., & Mitrovic, A. (2002). KERMIT: A Constraint-Based Tutor for Database Modeling. In S. A. Cerri, G. Gouardères, & F. Paraguaçu (Eds.), *Intelligent Tutoring Systems*, 2363, 377–387. Springer Berlin Heidelberg.

Ternes, B., Rosenthal, K., & Strecker, S. (2021). User Interface Design Research for Modeling Tools: A Literature Study. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, *16*, 4:1-30.

Ternes, B., Rosenthal, K., Strecker, S., & Bartels, J. (2020). TOOL–A Modeling Observatory & Tool for Studying Individual Modeling Processes. *39th International Conference on Conceptual Modeling*, 178–182.

Thaler, T., Houy, C., Fettke, P., & Loos, P. (2016). Automated Assessment of Process Modeling Exams: Basic Ideas and Prototypical Implementation. *Modellierung 2016 - Workshopband. Workshop Zur Modellierung in Der Hochschullehre (MoHoL-2016), Befindet Sich Modellierung 2016, March 2-2*, *255*, 63–70.

Theelen, H., & Van Breukelen, D. H. J. (2022). The didactic and pedagogical design of e-learning in higher education: A systematic literature review. *Journal of Computer Assisted Learning*, *38*(5), 1286–1303.

Thomas, P. G., Waugh, K., & Smith, N. (2007). Learning and automatically assessing graph-based diagrams. *Beyond Control: Learning Technology for the Social Network Generation. Research Proceedings of the 14th Association for Learning Technology Conference (ALT-C, 4—6 September, Nottingham, UK, 2007)*, 61–74.

Thomas, P., Waugh, K., & Smith, N. (2009). Generalised diagram revision tools with automatic marking. *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, 318–322.

Tselonis, C., Sargeant, J., & McGee Wood, M. (2005). *Diagram matching for human-computer collaborative assessment*.

Ullrich, M., Houy, C., Stottrop, T., Striewe, M., Willems, B., Fettke, P., Loos, P., & Oberweis, A. (2023). Automated Assessment of Conceptual Models in Education. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 18, 2:1-36.

Vachharajani, V., Pareek, J., & Gulabani, S. (2012). Effective Label Matching for Automatic Evaluation of Use—Case Diagrams. *2012 IEEE Fourth International Conference on Technology for Education*, 172–175.

Vom Brocke, J., Simons, A., Niehaves, B., Niehaves, B., Riemer, K., Plattfaut, R., & Cleven, A. (2009). Reconstructing the giant: On the importance of rigour in documenting the literature search process. *ECIS 2009 Proceedings*, 161.

Waugh, K., Thomas, P., & Smith, N. (2007). Teaching and Learning Applications Related to the Automated Interpretation of ERDs. *24th British National Conference on Databases*, 39–47.

Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), 13–23.