RESEARCH TOPICS 2026-01-22

AUTOMATIC ANALYSIS AND GRADING OF UTML UML DIAGRAMS

Douwe Osinga
d.r.osinga@student.utwente.nl

Supervisor
dr. ir. Vadim Zaytsev
v.zaytsev@utwente.nl

Supervisor
dr. Nacir Bouali
n.bouali@utwente.nl

Faculty of Electrical Engineering,
Mathematics, and Computer Science

UNIVERSITY OF TWENTE.

ABSTRACT

During computer science studies, students are often required to submit UML diagrams. The grading of these diagrams is mainly done by humans, resulting in a costly, lengthy, and error-prone process. In this paper, we investigate the theoretical feasability of automatically grading UML diagrams, focusing on the UTML variant developed at the University of Twente. We find that TODO results and propose *Seshat*, an algorithmic autograder that combines principles from graph isomorphism with structural, semantic, and syntactic matching TODO mention this in paper. In the final thesis, we compare the most suitable autograder from our related works to human grading.

1. INTRODUCTION

UML diagrams play a significant role in computer science, as they allow for communicating software designs in a standardised format. During technical studies, students are often required to make UML diagrams for graded assignments or exams.

However, the grading of these diagrams can often be a costly and lengthy process, involving multiple paid members of staff [1][1]. Additionally, this process is prone to grading inconsistencies [1], as humans are inherently unreliable, according to M. Meadows *et al.* [2], who pose two possible solutions: either "report the level of reliability associated with marks/grades, or find alternatives to [grading]." We propose a third alternative: finding alternatives to the grading *process*. Letting the grading process based on a (human) rubric be performed by software[2] instead of a human reduces human inconsistencies and the time it takes to grade.

The automatisation of grading diagrams provides an grading marking method that could both reduce the cost and time required for institutions and reduce the inherently present inconsistencies in human grading[2] [3] [4]. This could result in similar or superior, performance compared to human grading in terms of **accuracy** and **process transparency**, while improving **consistency**.

With *accuracy*, we mean the percentage of points assigned to a submission that are prescribed by the rubric for a particular excercise. With *consistency*, we mean both the extent to which similar grades are given to similar submissions, and the difference between consecutive runs (i.e. determinism). With *process transparency*, we mean the extent to which the reasoning for a particular grade is explained. These properties are desirable in the grading process, as it means that students are graded in a way that reflects their performance. For transparency, it would also be desirable to be able to link Intended Learning Objectives (ILOs) to the autograders, as this would help relate the grading to the objectives of the module [3].

For this research, we focus on the automatic grading of *UTML* UML diagrams, a recent, in-house developed diagram format of the University of Twente [5] [6]. However, as UTML is just a representation format and tool for creating UML diagrams, we aim to generalise these results to provide advice on the automatic grading of UML diagrams as a whole.

1.1. Background

Take a step back, show autograding from its origins, show different types of autograding. What is auto (auto**matic** / auto**mated**), state different formats (XMI, ...) and how these formats affect the view on the design process as well (IDE integration with XMI into Eclipse etc.), state different types of diagrams (UML, BPMN, ER/DB, MetaEdit / MetaCase, Eclipse Modelling Framework tools, ...)

1.2. Research Questions

In order to examin the feasibility of automatically grading UTML UML diagrams, we provide a main research question (**MRQ**):

To what extent can UML diagrams be graded automatically while maintaining or improving the accuracy, consistency, and transparency of human grading?

[1] From personal experience.

[2] Given that the process is deterministic

We aim to answer the main research question with the following sub-research questions:

RQ1: What existing work can be found for automatically analysing and/or grading UML diagrams?

- **RQ1a**: What correction models are employed by existing works?
- **RQ1b**: To what extent can Intended Learning Objectives be translated into different types of autograder correction models?

RQ2: To what extent are existing solutions suitable for use in autograding UTML diagrams with regards to (1) accuracy, (2) consistency, (3) transparency, (4) availability of source code, (5) extent of linking ILOs to grading instructions, (6) ease of integration into the grading process, and (7) UTML support?

RQ3: To what extent can a suitable autograder be constructed from previous work to be able to grade UTML UML diagrams?

RQ4: To what extent does the autograder compare to human grading in the context of grading first-year UML exam questions?

RQ1 is answered in Section 2, giving us an overview of existing solutions and their grading methodologies. **RQ2** is answered in Section 2 by analysing these works for suitability of grading. Finally, **RQ3** and **RQ4** are to be answered in the final thesis, where we grade UTML diagrams using an implementation based on related work and compare it to human grading.

2. RELATED WORK

mention that this is an exploratory view into papers. Exact inclusion/exclusion criteria, timeframe etc. will be mentioned in final thesis. In order to answer research questions **RQ1** and **RQ2**, we conduct a small-scale study covering roughly 40 works. These works are collected from sources such as Google Scholar[3] and ResearchGate[4], using terms such as "automatically grading UML diagrams", "autograder diagram", "UML diagram assessment", "machine learning diagrams", "diagram evaluation assessment AI", and similar for autograder-based related works.

2.1. Autograders

Automatic grading of diagrams seems to be a relatively new field, having started somewhere in the early 2000s [7] [8]. Multiple methods and types of diagrams are researched, including purely algorithmic methods for UML class- and use case diagrams, database Entity-Relation Diagrams, and Generative AI (GenAI)-based methods.

2.1.1. Frameworks / Theoretical

N. Smith *et al.* [7] provide a five-step framework for assessing "possibly ill-formed or inaccurate diagrams" that include (1) segmentation, (2) assimilation, (3) identification, (4) aggregation, and (5) interpretation. While the first two steps are aimed at translating images or other "raster-based input" into diagrammatic primitives, the latter stages provide a foundation to grade diagrams used by other papers [9].

N. H. Ali *et al.* [10] propose a UML class diagram assessment system using Rose Petal files, but does not mention enough specifics about algorithms to warrant further investigation.

F. Batmaz [11] takes a broader look at the process of grading, identifying and developing techniques to reduce repetitive actions, focusing on database Entity Relation diagrams. The paper suggests a semi-automatic grading system which identifies identical segments between a submission and the solution. Assuming multiple submission revisions are available, it suggests to "not only [use] the reference text but also the intermediate diagrams" for identifying semantic matches [11, p.40].

V. Vachharajani *et al.* [12] propose a UML use case assessment architecture. It provides a useful catalogue about edge cases related to (use case) diagram assessment, such as the chance of misspellings, synonyms, abbreviations, directionality of relationships, etc.

[3] https://scholar.google.com
[4] https://www.researchgate.net

W. Bian *et al.* [13] establish a metamodel to map submissions to example solutions and present a metamodel to grade submissions. It suggests using syntactic matching, semantic matching, and structural matching, with the goal to optimally match parts of a student submission with those of a teacher, considering spelling mistakes, synonyms and related words, and neighbours / inheritance, respectively.

In conclusion, most autograder strategies recommend structural matching (to identify similar segments of graphs), often in combination with syntactic matching that accounts for misspellings and semantic matching to account for synonyms. Unfortunately, the strategies do not account for integrating ILOs into the grading process explicitly.

2.1.2. Algorithmic

W. Bian *et al.* [4] expand their previous work [13] (see Section 2.1.1) with a case study. Their main findings are that multiple teacher solutions result in more accurate grades with an average accuracy of more than 95% [4, p.10], that grading configurations change per exam if you want similar grades to the teacher, and that their autograding "has shown to be more consistent and able to ensure fairness in the grading process" [4, p.11]. Additionally, their visual feedback system seems to be a nice addition for easily seeing where marks were awarded / taken away (see Figure 1).

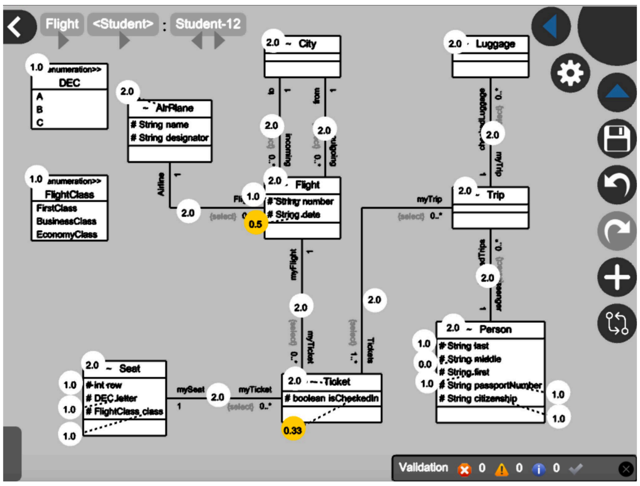

Figure 1: Visual feedback module from W. Bian *et al.* [4, Fig.9]

M. Hosseinibaghdadabadi *et al.* [14] also implements the framework by W. Bian *et al.* [13] by comparing UML use case diagrams to one or multiple example solutions, preferring the maximum grade. It uses a graph similarity strategy which matches nodes based on structural matching, along with syntactic and semantic word matching. Syntactic matching with Levenshtein distance, semantic matching with WordNet similarity score (uses HSO, WUP, LIN metrics). It achieves a very high correlation with human grades, with a similarity percentage of 93.31% [14, p.114].

O. Anas *et al.* [15] compares UML class diagram submissions to an example solution. It uses graph similarity scores based on structural matching along with syntactic and semantic matching. Syntactic matching is done with substring matching, semantic matching is done with neighbour similarity ("the comparison of the neighboring classes" [15, p.1585]), relationship name, type, multiplicity, and inheritance. It achieves a respectable correlation with human grading, with more than 80% is perfectly similar, over 90% >0.85 correlated, and no correlations lower than 0.7.

Multiple papers mention the use of XMI [16] [17], the object notation standard by OMG [18], or Rose Petal files [19], the standard of IBM Rational Rose [20], but fail to mention specifics about matching algorithms or results.

H. AlRawashdeh *et al.* [21] provides an interesting alternative way of grading submissions: by means of combining many UML diagram validators, model checkers, and even LTL properties given by instructors. However, a clear purpose, scope, and results are lacking from the paper.

M. Striewe *et al.* [22] continues H. AlRawashdeh *et al.* [21]'s property checking trend by focusing on graph queries for evaluation, providing a Domain-Specific Language that looks relatively similar to SQL. While it looks promising, the fact that teachers would have to learn a query language and transform their existing rubrics/ example solutions into this format could be a real hurdle, especially given the high similarity to existing grading of graph-isomorphism-based solutions [4] [14] [15]. Additionally, the paper does not provide approximate matching that would account for misspelling or synonyms.

S. Foss *et al.* provide multiple papers on AutoER, a database diagram generator and evaluator that provides direct interaction with a description text [23] [24] [25]. Unfortunately, concrete comparisons to manual grading or source code could not be found.

P. Thomas also provides a selection of paperse on the automatic grading of database diagrams [8] [26] [27] [9] [28]. These papers provide a grading strategy that accounts in its basis for *imprecise* diagrams (diagrams containing misspellings, duplicate entities, etc.), basing their analysing on comparing ever increasing subsets of the graph ((Minimal) Meaningful Units) based on the work of N. Smith *et al.* [7]. By 2009, P. Thomas *et al.* manage to achieve a correlation to human grading of 92%, along with statistically proving that the autograder grades more consistently than human grading. The graphed grading distribution can be viewed in Figure 2.

In 2011, P. Thomas *et al.* provide an online platform for both students and teachers to ease the process of automatic grading further, also used by N. Smith *et al.* [29], which further mathematically specify P. Thomas *et al.*'s work. Unfortunately, we were not able to retrace the source code of this grader.

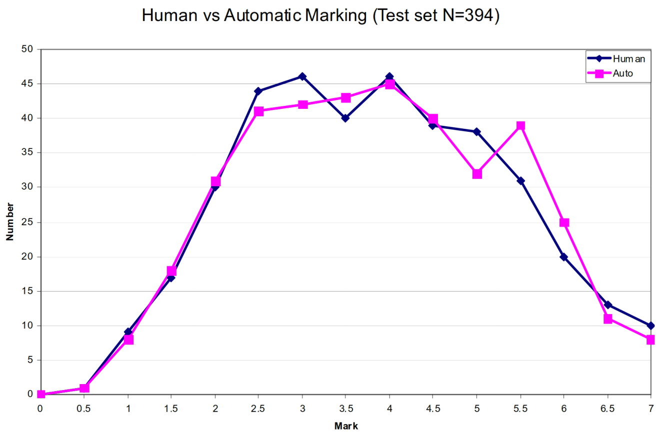

Figure 2: P. Thomas *et al.* [9, Fig. 3]: Human vs. automatic grading in database ER diagrams.

In conclusion, most existing implementations of autotgraders use some form of graph isomorphism algorithm with a combination of structural, semantic, and syntactic matching, also suggested by the majority of frameworks. Some solutions attempt to autograde using property or formula checking, but fail to mention detailed enough methodology or results to warrant further investigation. No autograders provide methods on integrating ILOs into the grading process.

2.1.3. Machine Learning / Generative AI / Large Language Model-driven

There has also been work on using Generative AI / Large Language Models (LLMs) to automatically grade solutions [30] [31] [32] [33].

D. R. Stikkolorum *et al.* [30] is one of the first papers that was found that attempts Machine Learning-based autograding, using several machine learning algorithms to compare it to expert grades. Unfortunately, the grading reaches only a maximum accuracy of 42.76% using a 10-point scale. Exact methods and algorithms are not mentioned.

C. Wang *et al.* [31] evaluate the feasibility of LLM-based grading with the model ChatGPT-4o, specifically for entire student reports, containing multiple types of UML diagrams. They feed pictures of student-submitted UML diagrams directly into the model along with an explanatory prompt that aims to trigger a Chain-of-Thought process (which helps LLMs "tackle complex arithmetic, commonsense, and symbolic reasoning tasks" [34]), and runs the model one time per student, with a temperature of 0.1. It finds that score differences range from −0.25 to +3.75 points, with significantly lower average scores given by the LLM compared to humans. Additionally, there are many occurrences of incorrect grading (wrong identifications, overstrictness, misunderstandings) [31, p.18], which means that, while the authors claim that their solution "demonstrates particular proficiency in the automated evaluation of UML use case diagrams", they do note occurrences of hallucination: "In the evaluation based on UC4, GPT deducts points for missing relationships between specified actors and use cases, but theses relationships existed in the UML use case" [31, p.13]. Furthermore, the paper does not express a strong correlation between LLM grading and human grading, at least compared to papers utilising graph matching algorithms [9] [14], nor does it recognise the inherent bias of LLMs [35] or their inherent non-determinism (even with a zeroed temperature) [36] [37], which make it a sub-optimal solution for consistent, fair grading.

N. Bouali *et al.* [32] uses various LLMs (Llama, GPT-o1 mini, Claude) to grade, translating the models into text instead of giving the LLM images directly such as C. Wang *et al.* [31]. While they achieve a Pearson correlation to human grading of 0.76 with both ChatGPT and Claude, they run into the same inconsistency issues as C. Wang *et al.*: “while the models would provide a final score as requested in the prompt’s response format, this score often did not match the actual sum of points awarded in their criterion-by-criterion assessment”, and “”One ChargingPort is associated with One Vehicle“ was matched with “One ChargingPort is associated with One ChargingStation” with a similarity of 0.92, despite describing different domain relationships” [32, p.164].

N. Bouali *et al.* identify the problem with grading with LLMs perfectly, stating that “This discrepancy can be attributed to the autoregressive nature of LLMs, where they generate responses token by token” [32, p.164]. Because these models are in their very essence based on predicting tokens [38], there is no formal guarantee that results are internally consistent and thus grades are produced with accuracy. The fact that LLMs produce grades that correlate with human grading does not mean that this grading is done in a fair, consistent, or reliable manner. While N. Bouali *et al.* try to reduce the non-determinism of LLMs by setting the temperature to zero, this does remove non-determinism necesssarily, nor does it correct training biases, as mentioned before.

R. Ramachandran *et al.* [33], unlike the previous papers, use a human-in-the-loop design in combination with both purely algorithmic steps, using LLMs only for similarity matching. Using structural matching algorithms similar to papers presented in Section 2.1.2, it achieves a Mean Average Error of only 0.611, aligning very closely to human grading (see Figure 3). Unfortunately, the sample size was a self-procured test set of only ten images, which negatively impacts the significance of these results, not to mention that the nondeterminism introduced by the LLMs will impact the consistency of grading, although it is unclear how much.

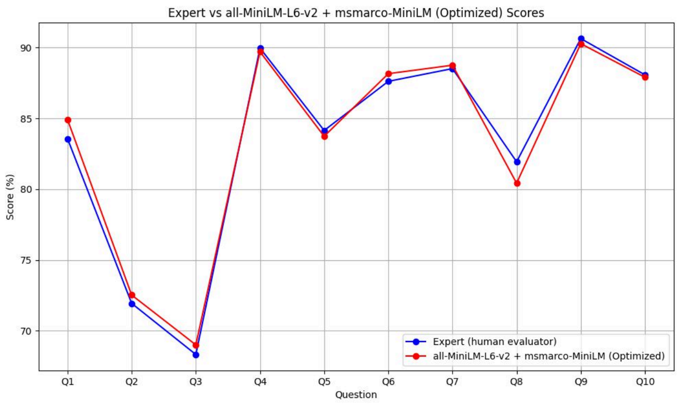

Figure 3: R. Ramachandran *et al.* [33, p.13]: Comparison of expert scores and CodeLLama scores using a combination of all-MiniLM-L6-v2 and msmarc-MiniLM as word similarity models.

In conclusion, while GenAI-based grading has been attempted in recent years, purely GenAI solutions produce lacking similarity to human grading compared to graph isomorphism-based solutions as well as introducing fundamental non-deterministic behaviour / hallucinations. This makes these types of solutions inferior to graph isomorphism solutions for full automatic grading. However, when used particularly for semantic and/or syntactic matching, it may provide similar performance to algorithmic solutions (although it still gives way to nondeterminstic grading and should be carefully evaluated.

2.2. Conclusion

In the explored related work, existing frameworks primarily recommend structural matching in combination with syntactic and semantic matching to be able to match solutions containing spelling mistakes and the use of synonyms. Existing implementations mostly use the methods recommended by the frameworks, with the best results stemming from determinstic, graph isomorphism algorithms, albeit at the cost of the teacher having to produce one or more sample solutions. Purely GenAI methods require less effort from teachers, since they do not need to produce sample solution(s), but produce noticeably subpar results to graph matching algorithms. Using hybrid methods, with GenAI for semantic/syntactic matching and graph isomorphism for structural matching, seems to produce similar results to ‘pure’ graph matching algorithms, but seemingly does not provide major advantages over algorithmic solutions and can additionally introduce nondeterminism in otherwise determinstic solutions, which reduces consistency.

3. TOOLS AND TECHNIQUES

Given existing works, the best approach seems to be to use graph isomorphism algorithms akin to those of W. Bian *et al.* [4] and P. Thomas *et al.* [9], adopting these solutions to UTML UML diagrams. Using a visual representation such as Figure 1 could prove to be a nice addition, so architectural support for visualisations will be taken into account, which can be implemented, should there be enough time.

Since existing solutions that feature these techniques have not published their source code (see Section 5.1), we will develop our own autograder, named *Seshat*[5].

TODO architecture, frameworks, languages

4. PLANNING

TODO: Graduation planning. Phases, goals per phase

[5]Named after the Egyptian daughter of *Thoth*, the name of D. Osinga [3]'s autograder.

BIBLIOGRAPHY

[1] F. Ahmed, N. Bouali, and M. Gerhold, "Teaching Assistants as Assessors: An Experience Based Narrative," 2024. [Online]. Available: https://research.utwente.nl/files/457355611/126242.pdf

[2] M. Meadows and L. Billington, "A Review Of THe Literature On Marking Reliability." [Online]. Available: https://assets.publishing.service.gov.uk/media/5a820a57e5274a2e87dc0d5a/0505_Meadows_and_Billington_CERP_RP.pdf

[3] D. Osinga, "Combining Dynamic and Static Analysis for the Automation of Grading Programming Exams," Bachelor thesis, 2024. [Online]. Available: https://github.com/osingaatje/ut-bachelor-thesis

[4] W. Bian, O. Alam, and J. Kienzle, "Is automated grading of models effective?: assessing automated grading of class diagrams," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, in MODELS '20. ACM, Oct. 2020, pp. 365–376. doi: 10.1145/3365438.3410944.

[5] D. Huistra, "UTML - internal GitLab repository." [Online]. Available: https://gitlab.utwente.nl/ewi/eduapps/UTML/

[6] "UTML - old student repository." [Online]. Available: https://github.com/andrewjh9/UTML

[7] N. Smith, P. Thomas, and K. Waugh, "Interpreting imprecise diagrams," in *Diagrammatic Representation and Inference*, in International Conference on Theory and Application of Diagrams. Mar. 2004, pp. 239–241. doi: 10.1007/978-3-540-25931-2_24.

[8] P. Thomas, "Grading Diagrams Automatically," 2004. [Online]. Available: https://oro.open.ac.uk/90155/1/2004_01.pdf

[9] P. Thomas, N. Smith, and K. Waugh, "Automatically Assessing Diagrams," in *Proceedings of the IADIS International Conference on e-Learning*, 2009. [Online]. Available: https://www.researchgate.net/profile/Pete-Thomas/publication/42799920_Automatically_assessing_diagrams/links/0fcfd5060076dd8ba2000000/Automatically-assessing-diagrams.pdf

[10] N. H. Ali, Z. Shukur, and S. Idris, "Assessment System For UML Class DiagramUsing Notations Extraction," 2007. [Online]. Available: https://www.researchgate.net/profile/Zarina-Shukur/publication/253243639_Assessment_System_For_UML_Class_Diagram_Using_Notations_Extraction/links/55487af30cf2b0cf7acec2e4/Assessment-System-For-UML-Class-Diagram-Using-Notations-Extraction.pdf

[11] F. Batmaz, "Semi-Automatic Assessment ofStudents' Graph-Based Diagrams," 2010. [Online]. Available: https://www.academia.edu/download/66135135/70_22270_EM_26aug_20feb_L.pdf

[12] V. Vachharajani and J. Pareek, "A Proposed Architecture for Automated Assessment of Use Case Diagrams," in *International Journal of Computer Applications (0975 – 8887)*, 2014. [Online]. Available: https://www.academia.edu/download/67672696/pxc3900193.pdf

[13] W. Bian, O. Alam, and J. Kienzle, "Automated Grading of Class Diagrams," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, IEEE, Sept. 2019, pp. 700–709. doi: 10.1109/models-c.2019.00106.

[14] M. Hosseinibaghdadabadi, O. A. N. Almerge, and J. Kienzle, "Automated Grading of Use Cases," in *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, IEEE, 2023. [Online]. Available: https://ieeexplore.ieee.org/iel7/10343461/10343549/10343598.pdf

[15] O. Anas, T. Mariam, and L. Abdelouahid, "New method for summative evaluation of UML class diagrams based on graph similarities," 2021. [Online]. Available: https://www.academia.edu/download/66135135/70_22270_EM_26aug_20feb_L.pdf

[16] S. Modi, H. A. Taher, and H. Mahmud, "A Tool to Automate Student UML diagram Evaluation," 2021. [Online]. Available: https://www.academia.edu/download/72488756/575.pdf

[17] R. Jebli, J. E. Bouhdidi, and M. Y. Chkouri, "Assessing Students' UML Class Diagrams: a NewAutomated Solution," in *2023 7th IEEE Congress on Information Science and Technology (CiSt)* |, IEEE, 2023. [Online]. Available: https://ieeexplore.ieee.org/iel7/10409867/10409868/10409936.pdf

[18] OMG, "XMI specification." [Online]. Available: https://www.omg.org/spec/XMI

[19] N. H. Ali, Z. Shukur, and S. Idris, "A Design of an Assessment System for UML Class Diagram," in *2007 International Conference on Computational Science and its Applications (ICCSA 2007)*, IEEE, Aug. 2007, pp. 539–546. doi: 10.1109/iccsa.2007.2.

[20] IBM, "Rational Rose." [Online]. Available: https://www.ibm.com/support/pages/ibm-rational-rose-enterprise-7004-fix-pack-4-7000

[21] H. AlRawashdeh, S. Idris, and A. M. Zin, "Using Model Checking Approach for Grading the Semantics of UML Models," 2014. [Online]. Available: https://iieng.org/images/proceedings_pdf/8684E0114567.pdf

[22] M. Striewe and M. Goedicke, "Automated Checks on UML Diagrams," in *ITiCSE'11*, in ITiCSE '11. ACM, June 2011, pp. 38–42. doi: 10.1145/1999747.1999761.

[23] S. Foss, T. Urazova, and R. Lawrence, "Learning UML database design and modeling with AutoER," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, in MODELS '22. ACM, Oct. 2022, pp. 42–45. doi: 10.1145/3550356.3559091.

[24] S. Foss, T. Urazova, and R. Lawrence, "Automatic Generation and Marking of UML Database Design Diagrams," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, in SIGCSE 2022. ACM, Feb. 2022, pp. 626–632. doi: 10.1145/3478431.3499376.

[25] S. Foss, "AutoER: A System for the Automatic Generation and Evaluation of UML Database Design Diagrams," 2022. [Online]. Available: https://open.library.ubc.ca/media/download/pdf/24/1.0421624/4

[26] P. Thomas, N. Smith, and K. Waugh, "An approach to the automatic grading of imprecise diagrams," technical report, 2006. doi: .org/10.21954/ou.ro.00016046.

[27] P. Thomas, N. Smith, and K. Waugh, "Automatically assessing graph-based diagrams," *Learning, Media and Technology*, vol. 33, no. 3, pp. 249–267, 2008, doi: 10.1080/17439880802324251.

[28] P. Thomas, K. Waugh, and N. Smith, "Generalised Diagramming Tools with Automatic Marking," in *Innovation in Teaching and Learning in Information and Computer Sciences*, 2011. doi: 10.11120/ital.2011.10010022.

[29] N. Smith, P. Thomas, and K. Waugh, "Automatic Grading of Free-Form Diagrams with Label Hypernymy," in *2013 Learning and Teaching in Computing and Engineering*, IEEE, Mar. 2013, pp. 136–142. doi: 10.1109/latice.2013.33.

[30] D. R. Stikkolorum, P. van der Putten, C. Sperandio, and M. R. Chaudron, “Towards Automated Grading of UML Class Diagrams with Machine Learning,” 2019. [Online]. Available: https://ceur-ws.org/Vol-2491/paper80.pdf

[31] C. Wang, B. Wang, P. Liang, and J. Liang;, “Assessing UML Diagrams by GPT: Implications for Education,” technical report, 2025. [Online]. Available: https://www.researchgate.net/publication/397720325_Assessing_UML_Diagrams_by_GPT_Implications_for_Education

[32] N. Bouali, M. Gerhold, T. U. Rehman, and F. Ahmed, “Toward Automated UML Diagram Assessment: Comparing LLM-Generated Scores with Teaching Assistants,” 2025. [Online]. Available: https://research.utwente.nl/files/496461589/134819.pdf

[33] R. Ramachandran, P. Vijayan, A. Anilkumar, and V. Gandadharan, “AI Assisted System for Automated Evaluation of Entity-Relationship Diagram and Schema Diagram Using Large Language Models,” technical report, Dec. 2025. doi: 10.3390/bdcc10010002.

[34] J. Wei *et al.*, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” 2023. [Online]. Available: https://arxiv.org/abs/2201.11903

[35] R. Ranjan, S. Gupta, and S. N. Singh, “A Comprehensive Survey of Bias in LLMs: Current Landscape and Future Directions,” 2024. doi: 10.48550/arXiv.2409.16430.

[36] M. Brenndoerfer, “Why Temperature=0 Doesn't Guarantee Determinism in LLMs,” 2025, [Online]. Available: https://mbrenndoerfer.com/writing/why-llms-are-not-deterministic

[37] B. Atil *et al.*, “Non-Determinism of “Deterministic” LLM Settings,” 2025. [Online]. Available: https://arxiv.org/pdf/2408.04667

[38] A. F. Ferraris, D. Audrito, L. D. Caro, and C. Poncibò, “The architecture of language: Understanding the mechanics behind LLMs,” *Cambridge Forum on AI: Law and Governance*, vol. 1, pp. 1–19, 2025, doi: 10.1017/cfl.2024.16.

5. APPENDICES

5.1. Autograder suitability table

Author	Diagram(s)	Ac	Co	Tr	OSS	ILO	Int	UTML
W. Bian *et al.* [4]	UML Class	H	H	H	N	N	?	N
M. Hosseinibaghdadabadi *et al.* [14]	UML Use Case	H	H	?	N	N	?	N
O. Anas *et al.* [15]	UML Class	M	H	?	N	N	?	N
S. Modi *et al.* [16]	UML Class	?	H	?	N	N	?	N
R. Jebli *et al.* [17]	UML Class	?	H	?	N	N	?	N
N. H. Ali *et al.* [10]	UML Class	?	H	?	N	N	?	N
H. AlRawashdeh *et al.* [21]	UML State/ Sequence	?	H	?	N	N	?	N
M. Striewe *et al.* [22]	UML Class	?	H	?	N	N	?	N
S. Foss *et al.* [23] [24] [25]	ER	?	H	?	N	N	?	N
P. Thomas *et al.* [8] [26] [9] [9] [28]	ER	H	H	?	M	N	?	N
D. R. Stikkolorum *et al.* [30]	UML Class	L	L	L	L	N	?	N
C. Wang *et al.* [31]	UML	M	L	M	H	N	M	N
N. Bouali *et al.* [32]	UML Class	M	M	M	H	N	M	N
R. Ramachandran *et al.* [33]	ER	H	M	H	L	N	?	N

Table 1: TODO FIX TRANSPARENCY
TODO explain integration ease
TODO explain Transparency
Autograders and their suitability scores.

*Di(*agram type*), Ac(*curacy*), Co(*nsistency*), Tr(*ansparency*), OSS = *availability of source code*, ILO = *ease of linking grading to ILOs*, Int(*egration ease*), UTML *support*.

Scoring is divided into "N" (*No Support*), "L" (*Low*), "M" (*Medium*), "H" (*High*), and "?" (*Unknown*), which gives an indication of suitability w.r.t. that particular criterium. The scoring is done in a comparative way, with the lowest-scoring solution receiving a "L", the highest scoring receiving a "H". A high **consistency** is awarded for determinstic solutions. High **transparency** is awarded for solutions that explain the exact grade that was given in terms of rubrics (medium for full rubrics that might not match (i.e. LLM solutions)). High **integration ease** is given to solutions that features solutions that have guides on building and deploying them (medium for small custom programs that are needed).