

Towards Standardized Evaluation in Automated Domain Modeling: Introducing a Benchmark

Vasiliy Seibert

Institute for Software and Systems Engineering, TU Clausthal
Clausthal-Zellerfeld, Lower Saxony, Germany
vasiliy.seibert@tu-clausthal.de

Abstract

Domain modeling plays an essential role in domain-driven design, capturing essential entities and their relationships within a specific domain. Despite advancements in automated domain modeling, the absence of standardized benchmarks has hindered the comparative assessment of existing approaches. This paper introduces a benchmark designed to address this gap. The benchmark combines the 45-record Golden UML Modelset (Verbruggen et al., 2025) on Zenodo, as distributed by the Text2UML project of Calamo, Meccella, and Snoeck (Calamo et al., 2025), with the 8-record reference archive of Chen et al. (Chen et al., 2023a,b), enabling the evaluation of automated domain modeling approaches across different levels of complexity and scale. Given a natural language description, the task is to generate a corresponding domain model. For each description, a reference domain model is provided as ground truth. A metric is used to compare the generated domain model with the corresponding ground-truth model. To demonstrate the utility of the benchmark, we evaluate multiple automated domain modeling approaches, including heuristic rule-based methods and LLM-driven strategies. In accordance with the FAIR4RS recommendations (Chue Hong et al., 2022), the benchmark is provided as a research artifact to encourage reuse and support future research on automated domain modeling.

CCS Concepts

• **Software and its engineering** → **Software creation and management** → **Software verification and validation**;

Keywords

domain modeling, class diagrams, large language models, benchmark, evaluation, FAIR principles

ACM Reference Format:

Vasiliy Seibert. 2026. Towards Standardized Evaluation in Automated Domain Modeling: Introducing a Benchmark. In . ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

Domain modeling is central to model-driven software engineering [18], and a domain model captures the relevant entities of a domain and their relationships using the syntax of UML class diagrams. The benefit of using UML class-diagram syntax is that the resulting model is precise and well known by software developers, who will use it as a foundation for building the system, and that the model facilitates clear communication in requirements discussions

and reduces misunderstandings between stakeholders.

Assisting or even automating domain modelling has great potential regarding increasing quality and reducing cost. Traditionally, heuristic rule-based methods [1, 2] have been formulated to analyze Requirement Descriptions given in Natural Language Text using tools like the Stanford CoreNLP API and subsequently apply rules to extract entities and meaningful relationships between them. However, these approaches are getting surpassed by modern approaches that rely entirely on Large Language Models (LLMs) [12].

Measuring the performance of automated domain modelling approaches has been neglected in the literature [12]. A benchmark analogous to Stanford’s HELM framework [29], consisting of a clearly defined task, a public dataset, and a deterministic metric, would help the research community by providing a place for healthy competition. The literature has repeatedly called for such a benchmark: Cámara et al. [12] state that the assessment of generative AI in modeling tasks is under-investigated, and Cámara, Burgueño, and Troya [11] argue for robust methodologies to standardize it. On the dataset side, a community-curated domain model corpus [9, 31] has emerged as a common reference set. On the metric side, the Seibert study [24] implements five metrics behind a single interface and evaluates them against human expert ratings [14].

The contribution of this paper is to address the previously described gap by proposing a benchmark. The task is to take a natural-language description of a domain [18] and to generate a domain model in PlantUML class-diagram notation. The benchmark uses a dataset of 53 domain models with their corresponding natural-language descriptions, drawn from two sources [9, 14, 15, 31]. The metric is taken from the Seibert study [24], which implements five metrics from four source papers [13, 27, 30, 34] behind a single interface, each returning three sub-scores in the unit interval, one per element type (class, attribute, association), where each sub-score is the F1 of the element-level match between candidate and reference. The benchmark uses metrik-4, the Triandini [30] metric, on the basis that the Seibert study [24] shows it to be the strongest of the five on per-element agreement with the Chen et al. [14] human expert ratings, winning 2 of 3 element-by-statistic cells.

To encourage reuse, the benchmark is released as a research artifact in accordance with the FAIR4RS recommendations [16, 33] <https://doi.org/10.5281/zenodo.21166386>.

2 Related Work

Prior approaches to automated domain modeling each evaluate on their own dataset, and the field's adoption of a common reference set is recent. Heuristic rule-based methods [1, 3–6, 17, 21, 23, 26, 32] each report a private, case-specific, or industrial evaluation, with no public artefact deposited at a stable identifier. Chen et al. [14] compare different LLMs and prompting strategies, bring their own dataset, and grade candidate models through human expert comparisons despite automated metrics being available, though they do deposit their data on Zenodo [15]. Parts of the Chen data are incorporated in a community-curated domain model corpus [31], published at a stable identifier with peer-reviewed provenance. The Calamo, Mecella, and Snoeck corpus [9, 10] is a cleaned encoding of the Verbruggen 45 published alongside a generation pipeline, and is, to our knowledge, the first reference corpus in this space to be published with the pipeline that consumes it.

Prior approaches to grading a candidate domain model against a reference are diverse in approach, and no metric has become a community standard. In the education application area, Modi, Taher, and Mahmud [20] develop a Java-based tool that produces an automatic grade from student and instructor solution diagrams, Boubekeur, Mussbacher, and McIntosh [8] combine heuristics with machine learning to predict approximate letter grades, and Bouali et al. [7] compare LLM-generated scores against teaching assistants on 92 student submissions. Singh, Boubekeur, and Mussbacher [27] propose a rule-based Mistake Detection System covering 83 of 97 identified mistake types, and Fauzan, Siahaan, Rochimah, and Triandini [19, 30] decompose assessment into a structural and a semantic component, reporting agreement with expert judgements. For model reuse and repository mining, Čech [13] defines a class-model distance via graph edit distance, Yuan, Yan, and Ma [34] decompose structural similarity into intra- and inter-structure components over a UML Class Graph, and Song et al. [28] adopt a graph neural network on use case graphs with TF-IDF cosine similarity. None of these approaches is cross-compared with a metric from another family, and each evaluates on its own dataset. The Seibert study [24] implements five metrics from four source papers [13, 27, 30, 34] behind a single interface and compares them against human expert ratings [14] on a common corpus. This benchmark reuses one of the five implementations, namely metrik-4, the Triandini [30] metric, as the fixed metric.

3 Approach

3.1 Benchmark Approach

The benchmark decomposes into three axes: a data axis, a task axis, and a metric axis. The data axis holds 53 records, each a pair of a natural-language text and a reference PlantUML class diagram drawn from two community-curated corpora. The task axis runs each candidate strategy against the natural-language text to produce a generated PlantUML class diagram. The metric axis is a deterministic comparator that accepts the reference and generated diagrams and returns a 3-tuple of per-element similarity scores, one each for classes, attributes, and associations.

	DS-1 (45)	DS-2 (8)	Total (53)
NLT words (mean)	316	423	—
Classes (mean / total)	9.4 / 425	14.4 / 115	540
Attributes (mean / total)	13.0 / 585	10.3 / 82	667
Enums (mean / total)	2.4 / 109	9.4 / 75	184
Relationships (mean / total)	11.9 / 537	21.6 / 173	710
Association (--)	268	87	355
Aggregation (o--)	6	0	6
Inheritance (-->)	102	34	136
Composition (*--)	116	35	151
Directed assoc. (-->)	32	1	33
Dependency (..)	13	14	27
Association-class	0	2	2
Card (% with multiplicity)	68	76	70

Table 1: Per-dataset overview.

3.2 Data

The benchmark runs on 53 records drawn from two PlantUML class-diagram corpora: a 45-record community-curated corpus [9, 31] and an 8-record reference archive [14, 15]. The eight reference diagrams were derived from the ground truth of the Chen et al. study; while that study produced many more models, those are generated by LLMs, whereas these eight were crafted by human experts in the field of education. Each record is a pair of a natural-language text and a reference PlantUML class diagram that the text describes. Together, the 53 records contribute 540 classes, 667 attributes, 184 enums, and 710 relationships. The two corpora differ in scale and in per-record composition: the 45-record corpus averages 9.4 classes and 13.0 attributes per record, while the 8-record archive averages 14.4 classes and 10.3 attributes, reflecting the denser modelling style of its source domain. Table 1 breaks down the 53 records per dataset and decomposes the 710 relationships into seven relationship types included in the dataset, ranging from 355 plain associations to 2 association-class links. Across both corpora, 70% of relationship endpoints carry a multiplicity, indicating that the majority of relationships specify cardinality information that a candidate model must recover.

LabTracker natural-language text [14, 15]:

The LabTracker software helps (i) doctors manage the requisition of tests and examinations for patients and (ii) patients book appointments for tests and examinations at a lab. For the remainder of this description, tests and examinations are used interchangeably. For a requisition, a doctor must provide their numeric practitioner number and signature for verification as well as their full name, their address, and their phone number. The signature is a digital signature, i.e., an image of the actual signature of the doctor. Furthermore, the doctor indicates the date from which the requisition is valid. The requisition must also show the patient's information including their alpha-numeric health number, first name and last name, date of birth, address, and phone number. A doctor cannot prescribe a test for themselves but can prescribe tests to someone else who is a doctor. Several tests can be combined on one requisition but only if they belong

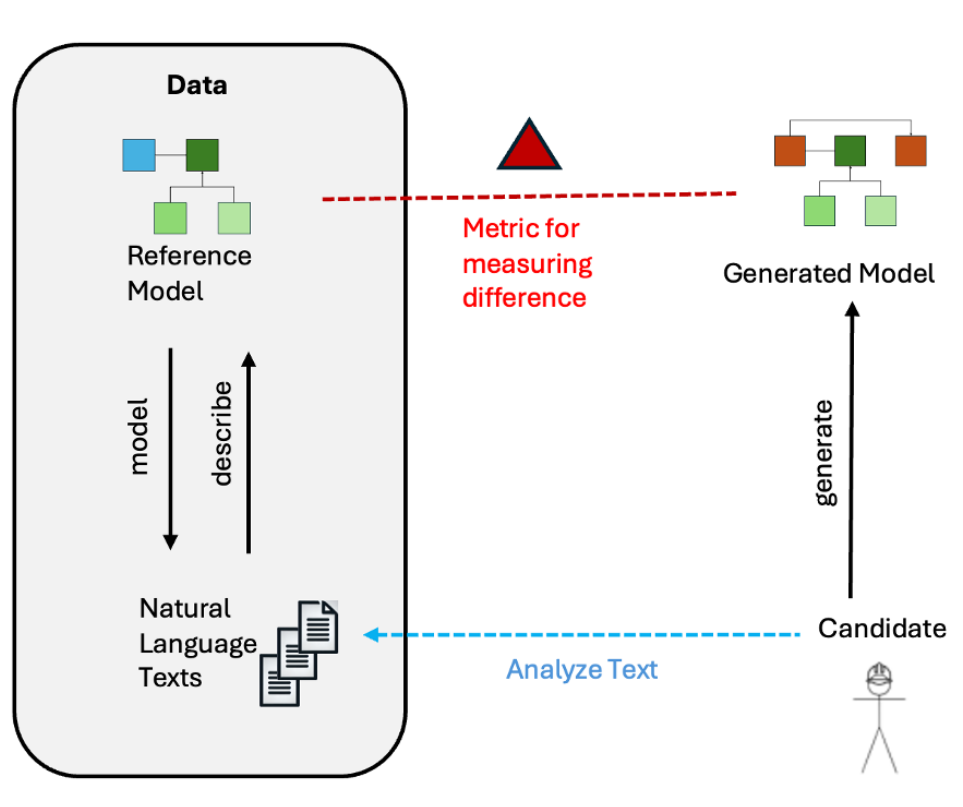


Figure 1: Overview of the benchmark: a (natural-language text, reference PlantUML class diagram) pair is fed to a candidate strategy, which produces a candidate PlantUML diagram that a metric compares against the reference.

to the same group of tests. For example, only blood tests can be combined on one requisition or only ultrasound examinations can be combined. It is not possible to have a blood test and an ultrasound examination on the same requisition. For each test, its duration is defined by the lab network, so that it is possible to schedule appointments accordingly. The duration of a test is the same at each lab. For some kinds of tests, it does not matter how many tests are performed. They take as long as a single test. For example, several blood tests can be performed on a blood sample, i.e., it takes as long to draw the blood sample for a single blood test as it does for several blood tests. A doctor may also indicate that the tests on a requisition are to be repeated for a specified number of times and interval. The interval is either weekly, monthly, every half year, or yearly. All tests on a requisition are following the same repetition pattern. The doctor and the patient can view the results of each test (either negative or positive) as well as the accompanying report. A patient is required to make an appointment for some tests while others are walk-in only. For example, x-ray examinations require an appointment, but blood tests are walk-in only (i.e., it is not possible to make an appointment for a blood test). On the other hand,

some tests only require a sample to be dropped off (e.g., a urine or stool sample). To make an appointment for a requisition, a patient selects the desired lab based on the lab's address and business hours. For requisitions with repeated tests, a patient is only allowed to make one appointment at a time. The confirmation for an appointment also shows a confirmation number, the date as well as start/end times, and the name of the lab as well as its registration number. It is possible to change or cancel an appointment at any time but doing so within 24 hours of the appointment incurs a change/cancellation fee. Each lab determines its own fee and business hours. All labs are open every day of the year and offer all tests. The business hours of a lab do not change from one week to the next. Each day a lab is open from the day's start time to its end time, i.e., there are no breaks.

LabTracker reference PlantUML (truncated) [14, 15]:

```
@startuml
enum Interval {
    WEEKLY
    MONTHLY
    HALF_YEARLY
    YEARLY
}
```

```

class Doctor {
  +signature: string
}
class Patient {
  +dateOfBirth: string
}
class Requisition {
  +effectiveDate: string
  +repetitionInterval: Interval
}
(...)
Doctor "1" -- "*" Requisition
Requisition -- "*" Test
Patient "1" -- "0..*" Requisition
Appointment "0..*" -- "1" Lab
Test "1" -- "0..1" Appointment
@enduml

```

3.3 Metric

This benchmark makes use of the metric implementations provided by the Seibert study [24], which packages five metrics from four source papers [13, 27, 30, 34] behind a single interface that accepts two PlantUML strings and returns a 3-tuple of per-element similarity scores, one each for classes, attributes, and associations. The study identifies two risks in reusing published metrics: a wrong implementation of the metric, and a wrong translation of the metric's native output format into the per-element F1 scale of the human expert ratings. To address the first risk, each metric is restated as a design-by-contract specification with pre- and post-conditions, decomposed into sub-functions, and handed unchanged to two independent LLMs through the same harness; if both implementations produce identical output on all 39 ground-truth comparisons from Chen et al. [14], the specification is assumed to be well-defined and the implementations are assumed to be correct. To address the second risk, the specification is extended to translate each metric's native output into the per-element F1 scale, and the translation is tested for partial-order preservation against the metric's native output, so that the ordering of the metric results is consistent with the ordering of the original output across the 39 comparisons. The five implementations are then compared against human expert ratings using a consistency table of four statistics — per-element MAD, residual standard deviation, absolute bias, and Pearson r (linear correlation between metric score and human F1) — computed across the 39 comparisons for each of the three element types. No single metric dominated across all cells: the per-element winners split across M-1, M-3, M-4, and M-5, with M-3 achieving the highest correlation on attributes ($r = 0.65$) and M-4 achieving the best correlation on classes ($r = 0.42$) and relationships ($r = 0.42$). A qualitative analysis complemented the statistical comparison, examining where each metric agreed with or diverged from the human judgments and identifying systematic biases such as M-4's constant overestimation on relationships. The benchmark chooses metrik-4, the Triandini [30] metric, because it wins 2 of the 3 element-by-r cells and also achieves the best attribute MAD (0.14), making it the most balanced metric on the consistency table. The implementation is reused from the upstream domain-model-metrics package

(v1.0.0) without modification, so the per-element scores are reproducible from the version pin.

Seibert study metric interface:

```

Metric M
  compute(ReferencePlantUML,
          GeneratedPlantUML)
  -> (class_score      : [0, 1],
      attribute_score  : [0, 1],
      association_score : [0, 1])

```

3.4 Candidates

The benchmark reuses three open-source candidate strategies: one deterministic rule-based pipeline and two LLM-driven zero-shot strategies. The rule_based candidate is a Python-over-spaCy re-implementation of the NLP-based pipeline of Abdelnabi, Maatuk, Abdelaziz, and Elakeili [1], which parses the natural-language text with spaCy's dependency parser and applies heuristic rules for extracting classes, attributes, and relationships, emitting a PlantUML class diagram with no LLM call. The two zero-shot candidates were derived from the literature: one from Chen et al. [14, 15] and one from Calamo, Mecella, and Snoeck [9, 10]. The Chen et al. prompt asks the LLM to extract the class diagram as an intermediate domain-specific language (Enumerations, Class, Relationships); the Calamo et al. prompt asks the LLM to emit the PlantUML class diagram directly, guided by a 5-step directive covering classes, attributes, relations and inheritance, relation-to-class assignment, and cardinalities. Each LLM-driven candidate is run with two LLMs (GLM-5.1 and Kimi K2) and repeated three times to account for the non-determinism of cloud-served LLMs. To ensure that the LLM output adheres to the PlantUML syntax accepted by the parser, the Chen et al. candidate makes a second LLM call prompted with a grammar encoding the parser's strict grammar, which translates the intermediate DSL into canonical PlantUML.

Chen et al. [14, 15] — Stage 1, system prompt:

Generate the lists of model classes and associations from a given description.

Chen et al. [14, 15] — Stage 1, task prompt:

Create a class diagram for the following description by giving the enumerations, classes, and relationships using format:

```

Enumerations:
enumerationName(literals)
(there might be no or multiple enumerations)

```

```

Class:
className(attributeType attributeName)
(there might be multiple attributes)
(there might be multiple classes)

```

```

Relationships
mul1 class1 associate mul2 class2
(class1 and2 are classes above. mul1 and mul2
are one of the following options
[0..*, 1, 0..1, 1..*])
(there might be multiple associations)

```

```
class1 inherit class2
(class1 and class2 are classes above)
(there might be multiple inheritance)

mul1 class1 contain mul2 class2
(class1 and2 are classes above. mul1 and mul2
are one of the following options
[0..*, 1, 0..1, 1..*])
(there might be multiple composition)
```

Chen et al. [14, 15] — Stage 2, translate prompt:

You are a PlantUML translator. You will be given a draft class diagram extracted from a natural-language specification. The draft may be in a free-form text format (the upstream zenodo S1 format) OR in a malformed PlantUML form. Your job is to produce a single, clean PlantUML class diagram in the canonical grammar described below. (...)

Calamo, Mecella, and Snoeck [9, 10] — prompt:

You will be asked by the user to create a plant UML model from specification text. Do so in the most clear way possible, avoid class properties and assign multiplicity.

Do include attributes for classes. For example the class Book would be:

```
class Book{ String Title, String Author,
Date PublicationDate }
```

Use only bi-directional arc for relations and no description. For example a relation between the class Book and the class Page, if the Book can have from one to many pages and the pages could have exactly one book, would be:

```
Book "1..1" -- "1..*" Page
```

Adapt the cardinality to each case. Where no specific cardinality is specified, use the default "0..*". If necessary, feel free to use inheritance. The plantuml has to be the class diagram. In generating the diagram perform this steps in order

1. Extract class from text
2. Extract attributes for each class
3. Extract relations form text and look for the inheritance
4. Assign the relation to the corresponding class
5. Add cardinality to the relations

Put everything in this order: first all classes and then all relations. In our example would be:

```
@startuml
```

```
class Book{ String Title, String Author,
Date PublicationDate }
class Page{ String Content}
```

```
Book "1..1" -- "1..*" Page
```

```
@enduml
```

Output plantuml without futher text or explanation.

4 Results

The benchmark results are summarised in Tables 2 and 3, which report the cross-run stability of the three candidate strategies on the two community-curated corpora introduced in Section 3.1: DS-1, the 45-record community-curated corpus of Verbruggen et al. [31] as re-encoded by Calamo, Mecella, and Snoeck [9], and DS-2, the 8-record reference archive of Chen et al. [14, 15], whose reference diagrams were crafted by human experts in the field of education. Together the two corpora contribute 53 records, comprising 540 classes, 667 attributes, 184 enumerations, and 710 relationships. Each cell in the tables corresponds to one (candidate, LLM) combination, repeated over three independent runs to capture the non-determinism of cloud-served LLMs. The three candidates are the rule-based pipeline of Abdelnabi, Maatuk, Abdelaziz, and Elakeili [1], the two-stage zero-shot strategy of Chen et al. [14, 15], and the single-stage zero-shot strategy of Calamo, Mecella, and Snoeck [9, 10]. The two LLMs are GLM-5.1 [35] and Kimi K2 [22], invoked through Ollama with think mode disabled, at an extract temperature of 0.7 and a translate temperature of 0.0 for the Chen et al. candidate's second-stage grammar prompt. Scoring is performed with metrik-4 from the Seibert study [24], pinned to v1.0.0 of the domain-model-metrics package [25], which returns one F1 score per element type (class, attribute, relationship), each in the unit interval. For each cell, the tables report the 3-run median and cross-run standard deviation of the per-run mean score, along with the median percentage of records that failed validation across the three runs. The rule-based candidate is deterministic and involves no LLM call, so it appears with no LLM and zero cross-run variance on every element.

On DS-1, the 45-record community-curated corpus [9, 31], both LLM-driven candidates outperform the rule-based baseline [1] on every element. The Chen et al. candidate [14, 15] paired with GLM-5.1 [35] achieves the highest scores on class (0.746) and attribute (0.780), while the Calamo et al. candidate [9, 10] paired with GLM-5.1 achieves the highest score on relationship (0.701). The rule-based baseline scores 0.417 on class, 0.447 on attribute, and 0.348 on relationship, so the LLM candidates lead by 0.15–0.35 absolute on every element. Cross-run stability is high: the standard deviation across the three runs is at most 0.025 for every (candidate, LLM) cell, and as low as 0.006 for the Chen et al. candidate with GLM-5.1. The Chen et al. candidate with GLM-5.1 also produces zero validation failures across all three runs, while the Calamo et al. candidate with GLM-5.1 has a 2.2% failure rate. Kimi K2 [22] is competitive but scores lower than GLM-5.1 on both candidates, with failure rates of 8.9% (Chen et al.) and 4.4% (Calamo et al.). The rule-based baseline is deterministic with zero cross-run variance and zero failures, as expected.

On DS-2, the 8-record reference archive of Chen et al. [14, 15], the LLM candidates again outperform the rule-based baseline [1] on every element, but the cross-run standard deviations are 4–10×

Candidate	LLM	Fail %	Class	Attribute	Relationships
Rule-based [1]	—	0.0	0.417 ± 0.000	0.447 ± 0.000	0.348 ± 0.000
Chen et al. [14, 15]	GLM-5.1 [35]	0.0	0.746 ± 0.006	0.780 ± 0.006	0.676 ± 0.007
Chen et al. [14, 15]	Kimi K2 [22]	8.9	0.699 ± 0.013	0.717 ± 0.014	0.641 ± 0.014
Calamo et al. [9, 10]	GLM-5.1 [35]	2.2	0.694 ± 0.024	0.691 ± 0.025	0.701 ± 0.024
Calamo et al. [9, 10]	Kimi K2 [22]	4.4	0.613 ± 0.015	0.603 ± 0.016	0.637 ± 0.015

Table 2: Benchmark results on DS-1 (45 records). Scores are 3-run median ± cross-run std; Fail % is the median percentage of records that failed validation. Best results in bold.

Candidate	LLM	Fail %	Class	Attribute	Relationships
Rule-based [1]	—	0.0	0.312 ± 0.000	0.353 ± 0.000	0.215 ± 0.000
Chen et al. [14, 15]	GLM-5.1 [35]	12.5	0.625 ± 0.013	0.650 ± 0.014	0.576 ± 0.030
Chen et al. [14, 15]	Kimi K2 [22]	25.0	0.596 ± 0.057	0.603 ± 0.053	0.581 ± 0.066
Calamo et al. [9, 10]	GLM-5.1 [35]	0.0	0.655 ± 0.062	0.640 ± 0.064	0.689 ± 0.059
Calamo et al. [9, 10]	Kimi K2 [22]	0.0	0.506 ± 0.080	0.487 ± 0.070	0.548 ± 0.104

Table 3: Benchmark results on DS-2 (8 records). Scores are 3-run median ± cross-run std; Fail % is the median percentage of records that failed validation. Best results in bold.

higher than on DS-1. The Calamo et al. candidate [9, 10] with GLM-5.1 [35] achieves the best class (0.655) and relationship (0.689) scores, while the Chen et al. candidate [14, 15] with GLM-5.1 achieves the best attribute score (0.650). The rule-based baseline scores 0.312 on class, 0.353 on attribute, and 0.215 on relationship, so the LLM leads are comparable to DS-1 in absolute terms (0.15–0.47). However, the cross-run standard deviation reaches 0.104 for the Calamo et al. candidate with Kimi K2 [22] on relationship, compared to 0.015 on DS-1 — a 6.9× inflation. The smaller corpus amplifies single-record noise: with only 8 records, one record flipping its per-record score shifts the cell median by roughly 0.06. Validation failure rates are also higher on DS-2: the Chen et al. candidate with GLM-5.1 has a 12.5% failure rate (versus 0.0% on DS-1), and with Kimi K2 it reaches 25.0% (versus 8.9%). The Calamo et al. candidate has zero validation failures on DS-2 across both LLMs, consistent with its single-stage design that emits PlantUML directly.

References

- [1] Eman A. Abdelnabi, Amin M. Maatuk, Tawfig M. Abdelaziz, and Samir M. Elakeili. 2020. Generating UML Class Diagram using NLP Techniques and Heuristic Rules. In *2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE, Monastir, Tunisia, 277–282. doi:10.1109/STA50679.2020.9329301
- [2] Sayf Ahmed, Adnan Ahmed, and Nasir U. Eisty. 2022. Automatic Transformation of Natural to Unified Modeling Language: A Systematic Review. In *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, Las Vegas, NV, USA, 112–119. doi:10.1109/SERA54885.2022.00026
- [3] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. 2016. Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS '16)*. Association for Computing Machinery, New York, NY, USA, 250–260. doi:10.1145/2976767.2976771
- [4] Imran Sarwar Bajwa and M. A. Choudhary. 2012. From Natural Language Software Specifications to UML Class Models. In *Proceedings of the 13th International Conference on Enterprise Information Systems (ICEIS 2011), Revised Selected Papers*. Springer, 224–237. doi:10.1007/978-3-642-29958-2_16
- [5] Imran Sarwar Bajwa, Abdus Samad, and Sobia Mumtaz. 2009. Object Oriented Software Modeling Using NLP Based Knowledge Extraction. *European Journal of Scientific Research* 35, 1 (2009), 22–33.
- [6] Wahiba Ben Abdesslem Karaa, Zeineb Ben Azzouz, Ashish Singh, Nilanjan Dey, Amira S. Ashour, and Henri Ben Ghazala. 2016. Automatic Builder of Class Diagram (ABCD): An Application of UML Generation from Functional Requirements. *Software: Practice and Experience* 46, 11 (2016), 1443–1458. doi:10.1002/spe.2392
- [7] Ayoub Bouali et al. 2025. LLM-Based Grading of Software Models: An Exploratory Study. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 423–435.
- [8] Yasmine Boubekeur, Gunter Mussbacher, and Shane McIntosh. 2020. Automatic grading of software modeling assignments. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '20)*. ACM, 267–278. doi:10.1145/3417991.3418019
- [9] Marco Calamo, Massimo Mecella, and Monique Snoeck. 2025. Text2UML: A Tool for the Automated Generation of UML Class Diagrams from Natural Language Text. In *Proceedings of the ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems (MODELS '25)*. Association for Computing Machinery.
- [10] Marco Calamo, Massimo Mecella, and Monique Snoeck. 2026. Text2UML Results with Golden UML Dataset. <https://doi.org/10.5281/zenodo.19599470>.
- [11] Javier Cámara, Loli Burgueño, and Jesús Troya. 2024. Towards Standardized Benchmarks of LLMs in Software Modeling Tasks: A Conceptual Framework. *Software and Systems Modeling* (2024). doi:10.1007/s10270-024-01195-x
- [12] Javier Cámara, Jesús Troya, Loli Burgueño, and Antonio Vallecillo. 2023. On the Assessment of Generative AI in Modeling Tasks: An Experience Report with ChatGPT and UML. *Software and Systems Modeling* 22, 3 (2023), 781–793. doi:10.1007/s10270-022-01069-8
- [13] Istvan Cech et al. 2019. Matching Object-Oriented Models: A Graph-Edit-Distance Approach. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 217–227.
- [14] Kua Chen, Yujing Yang, Boqi Chen, J. A. Hernández López, Gunter Mussbacher, and Dániel Varró. 2023. Automated Domain Modeling with Large Language Models: A Comparative Study. In *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, Västerås, Sweden, 162–172. doi:10.1109/MODELS58315.2023.00037
- [15] Kua Chen, Yujing Yang, Boqi Chen, J. A. Hernández López, Gunter Mussbacher, and Dániel Varró. 2023. Automated Domain Modeling with Large Language Models: A Comparative Study — Research Artefact. <https://doi.org/10.5281/zenodo.8105098>.
- [16] Neil P. Chue Hong et al. 2022. FAIR4RS Principles. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '18 Workshop)*. doi:10.5281/zenodo.20942596
- [17] Manuel Clavel, M. S. E. González, and V. T. da Silva. 2007. The MOVA Tool: A Rewriting-Based UML Modeling, Measuring, and Validation Tool. In *Proceedings of the XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2007)*. 393–394.
- [18] Eric Evans. 2004. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional (2004).
- [19] Reza Fauzan et al. 2021. Automated Assessment of Object-Oriented Models Using Structural and Lexical Similarity. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 112–122.

- [20] Hasib Mahmud. 2021. MOSS: A tool to grade object-oriented programming assignments. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 534–540. doi:10.1145/3408877.3432530
- [21] Manoj Mohanan and Philip Samuel. 2018. Natural Language Processing Approach for UML Class Model Generation from Software Requirement Specifications via SBVR. *International Journal on Artificial Intelligence Tools* 27, 6 (2018), 1850027. doi:10.1142/S0218213018500274
- [22] Moonshot AI. 2025. Kimi K2.6 (K2.7-code). <https://kimi.moonshot.cn/>.
- [23] Daniel Popescu, Spencer Rugaber, Nenad Medvidovic, and Daniel M. Berry. 2008. Reducing Ambiguities in Requirements Specifications via Automatically Created Object-Oriented Models. In *Proceedings of the 14th Monterey Workshop on Software Engineering for Self-Adaptive Systems*. Springer, 103–124. doi:10.1007/978-3-540-89378-3_7
- [24] Vasily Seibert. 2025. Comparing Domain-Model Similarity: A Reference Implementation of Five Literature Metrics Behind One Interface, Evaluated Against Human Expert Ratings. <https://doi.org/10.5281/zenodo.20942597>.
- [25] Vasily Seibert. 2025. domain-model-metrics: Python Package of Five Domain-Model Similarity Metrics. <https://github.com/VasilySeibert/domain-model-metrics>.
- [26] Vibha S. Sharma, Suman Sarkar, Kunal Verma, Anitha Panayappan, and Alex Kass. 2009. Extracting High-Level Functional Design from Software Requirements. In *Proceedings of the 16th Asia-Pacific Software Engineering Conference (APSEC '09)*. IEEE, 35–42. doi:10.1109/APSEC.2009.62
- [27] Chanchal K. Singh et al. 2022. Detecting Mistakes in Object-Oriented Models: A Synonym-Aware Approach. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 1–11.
- [28] Wei Song et al. 2024. A Deep Learning Approach to UML Use Case Model Similarity. In *Proceedings of the 27th International Conference on Model Driven Engineering Languages and Systems (MODELS '24)*. ACM.
- [29] Stanford Center for Research on Foundation Models (CRFM). 2024. Holistic Evaluation of Language Models (HELM). <https://crfm.stanford.edu/helm/>.
- [30] Evi Triandini et al. 2021. Automated Class Diagram Assessment using Semantic and Structural Similarities. *International Journal of Intelligent Engineering and Systems* (2021). doi:10.22266/IJIES2021.0430.06
- [31] Charlotte Verbruggen, Lukas Netz, Philipp-Lorenz Glaser, Marion Scholz, Christian Huemer, Marco Calamo, Bernhard Rumpe, Monique Snoeck, and Dominik Bork. 2025. Toward a Community-Curated Golden Dataset of UML Models. In *Proceedings of the ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems (MODELS '25 Educators Symposium)*. Association for Computing Machinery, New York, NY, USA. Dataset: <https://doi.org/10.5281/zenodo.16985873>.
- [32] S. Vinay, S. Aithal, and P. Desai. 2009. An Approach towards Automation of Requirements Analysis. In *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2009)*, Vol. 1.
- [33] Mark D. Wilkinson et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. In *Scientific Data*. doi:10.1038/sdata.2016.18
- [34] Yi Yuan, Yilong Yan, and Yanbo Ma. 2020. Structural Matching of UML Class Diagrams: A Graph-Pattern Approach. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 1–11.
- [35] Zhipu AI. 2025. GLM-5.1/GLM-5.2 Model Card. <https://github.com/THUDM/GLM>.

Received 2026-07-03