Beyond Single-run Correctness: Nondeterminism-aware Evaluation of LLM-based Model Transformations

Riccardo Rubei
IDD
Mälardalen University
Västerås, Sweden
riccardo.rubei@mdu.se

Alessio Bucaioni
IDD
Mälardalen University
Västerås, Sweden
alessio.bucaioni@mdu.se

Amleto Di Salle
Computer Science Area
Gran Sasso Science Istitute
L'Aquila, Italy
amleto.disalle@gssi.it

Abstract

Model transformation is a core model-driven engineering (MDE) operation in which reproducibility is expected: under fixed metamodels, source model, and transformation rules, a deterministic engine should produce a stable target model. Large Language Models (LLMs) are increasingly explored for MDE tasks, but evaluations often focus on whether an acceptable artifact can be produced once. For deterministic and quasi-deterministic MDE workflows, this single-run correctness view is necessary but insufficient.

This paper introduces *nondeterminism-aware evaluation* for LLM-based MDE. Using model-to-model transformation as a stress case, we compare LLM-generated target models against deterministic ATL references and across repeated executions. Our exploratory evaluation covers five ATL Zoo transformation scenarios, four prompt configurations, three LLMs, and ten executions per transformation -scenario–configuration–LLM combination. We analyze reference deviation, inter-run variation, and morphological differences.

The results show that transformation explicitness does not guarantee convergence. Even when the complete ATL transformation is provided, 13 out of 15 transformation-scenario–LLM combinations show non-zero mean and median deviation from the ATL reference, and 10 out of 15 exhibit non-zero inter-run variation. We further observe stable but not ATL-equivalent behavior and cases where a single exact match coexists with non-zero median deviation. These findings suggest that LLM-based MDE evaluation should treat correctness, reproducibility, and variation meaning as distinct dimensions.

ACM Reference Format:
Riccardo Rubei, Alessio Bucaioni, and Amleto Di Salle. 2026. Beyond Single-run Correctness: Nondeterminism-aware Evaluation of LLM-based Model Transformations. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3822455.3838776

1 Introduction

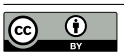
Model transformation is a core operation in model-driven engineering (MDE) [26]. Given the same source model, source metamodel, target metamodel, and transformation rules, a transformation engine is expected to produce a stable target model. This expectation underpins automation, traceability, conformance checking, code generation, model analysis, and other downstream activities that rely on models as structured artifacts [23].

Large Language Models (LLMs) are increasingly shaping software engineering research and practice [20]. Their adoption is often framed *tactically*, in terms of task-level capabilities such as generating artifacts or assisting engineers [20]. Following Babur *et al.* [4], we argue that the *strategic* question is broader: how should MDE processes, tools, and evaluation practices change when generative AI becomes part of engineering workflows?

This paper addresses that question for MDE workflows that traditionally rely on deterministic behavior. Existing LLM-for-MDE evaluations mainly ask whether an LLM can produce a correct or acceptable artifact with respect to a reference, constraint, oracle, or ground truth [10–12, 21, 24]. This single-run correctness view is necessary, but insufficient. In deterministic or quasi-deterministic MDE workflows, practical trust also depends on reproducibility, conformance, semantic stability, and interpretable variation across repeated executions. Model-to-model transformation makes this limitation explicit. An LLM may match a deterministic reference once but vary in later executions, or it may be reproducible while remaining consistently different from the reference. In both cases, single-run correctness can overestimate the suitability of the LLM-based operation. Correctness in one execution is therefore not enough evidence of trust in LLM-based MDE.

This paper introduces *nondeterminism-aware evaluation* for LLM-based MDE. We do not aim to eliminate LLM nondeterminism, enforce deterministic outputs, or rank LLMs as transformation engines. Instead, we use model-to-model transformation as a precise stress case for studying how nondeterminism manifests, how it can be measured, and how its effects can be interpreted in MDE terms. The evaluation question shifts from whether an LLM can perform a modeling task once to what evidence is needed before an LLM-based model operation can be trusted in deterministic workflows. Using ATL transformations as deterministic references, our exploratory evaluation covers five ATL Zoo transformation scenarios [1], four prompt configurations with increasing transformation explicitness, three LLMs belonging to different families, and ten repeated executions per transformation-scenario–configuration–LLM combination. We analyze generated target models along three complementary dimensions: *reference deviation* from the ATL output, *inter-run variation* across repeated executions, and the *morphology*

[1] https://eclipse.dev/atl/atltransformations/



This work is licensed under a Creative Commons Attribution 4.0 International License.
MODELS 2026, Málaga, Spain
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2809-9/26/10
https://doi.org/10.1145/3822455.3838776

of differences, i.e., whether variation appears through namespace, attribute, node, structural, or other model-relevant changes.

The results show that transformation explicitness does not guarantee convergence. Even in the most explicit prompt configuration, where the complete ATL transformation is provided, 13 out of 15 transformation-scenario–LLM combinations show non-zero mean and median deviation from the ATL reference, while 10 out of 15 exhibit non-zero inter-run variation. We also observe stable but not ATL-equivalent behavior and cases where a single exact match coexists with non-zero median deviation across repeated executions. Detailed differences further show that variation is not a single scalar phenomenon: the same numerical distance can correspond to changes with different implications for syntax, analyzability, conformance, traceability, and semantic preservation.

2 Background and Related Work

This section positions our work with respect to determinism in model transformation and nondeterminism in LLM-based generation.

Model transformation and determinism. A model-to-model transformation converts a source model into a target model according to source and target metamodels and transformation rules [9]. In MDE, such transformations are expected to be reproducible: given the same inputs and rules, a transformation engine should produce a stable target model for downstream analysis, code generation, conformance checking, traceability, and synchronization. Determinism has therefore been studied through graph transformation theory, execution semantics, and language-level constraints [9]. In graph-transformation accounts, it is commonly related to *termination* and *confluence* [2, 5, 13, 19, 22, 25], while other work shows how nondeterminism may arise from rule scheduling, location determination, under-specified execution semantics, or parallel updates to shared model graphs [6, 15, 18, 25, 28]. Seminal work further links determinism to consistency checking and repeated tool execution in bidirectional transformations [14, 27], as well as to validation through termination and confluence [22]. This paper does not study nondeterminism in ATL engines; instead, it uses ATL transformations as deterministic reference points for evaluating LLM-generated target models.

LLM nondeterminism and evaluation. LLM outputs are not generally reproducible in the same sense as deterministic model transformations. Even with fixed prompts, generated artifacts may vary because of sampling, numerical precision, batching, or provider-side inference optimizations. Recent work has investigated sources and mitigation of LLM nondeterminism, including scheduling-based approaches [16], numerical-precision effects [29], quantization strategies [30], and repeated-execution studies across prompting strategies and tasks [3]. These studies show that LLM nondeterminism is an active concern, but they do not address how it should be evaluated in MDE workflows where reproducibility is part of the expected behavior of model operations. Existing LLM-for-MDE evaluations mainly ask whether an LLM can generate a correct or acceptable artifact for a given task [12]. We complement this view by studying repeated executions of model-to-model transformation and by reporting reference deviation, inter-run variation, executability, and the modeling meaning of observed differences as separate evaluation dimensions.

3 Experimental procedure

We designed an exploratory evaluation to make LLM nondeterminism observable in model-to-model transformation. The goal is not to benchmark LLMs as replacements for transformation engines, nor to enforce deterministic outputs. Instead, we use model transformation as a controlled MDE setting in which reproducibility is expected: under fixed source and target metamodels, source model, and transformation rules, a deterministic engine should produce a stable target model. Figure 1 summarizes the procedure. We first

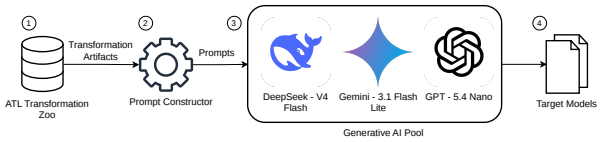


Figure 1: Visualization of the experimental workflow.

select ATL transformations from the ATL Zoo ①, then construct prompts according to four configurations with increasing transformation explicitness ②. Each configuration is executed with three cost-efficient LLMs available at the time of the study (i.e., April–June 2026) ③, and the generated target models are collected ④. We call each transformation-scenario–configuration–LLM tuple an *evaluation cell*. Each cell was executed ten times, yielding 60 evaluation cells and 600 requested generations. Of these, 595 returned artifacts were available for exception analysis and subsequent comparison.

Transformation dataset. We use the ATL Zoo as the source of transformations, metamodels, source models, and ATL rules, as it was widely employed as a benchmarking dataset in several studies [7, 8, 17]. The five selected transformation scenarios vary in provenance and structural complexity, from small didactic cases to larger transformations involving multiple rules, bindings, and helpers. This diversity supports an exploratory analysis of whether variation appears only in underspecified prompts or also when richer transformation artifacts are available. Table 1 reports the structural characteristics of the selected transformations.

Prompt configuration. The prompt configuration determines which transformation artifacts are provided to the LLM. We use four configurations to observe whether increasing transformation explicitness reduces reference deviation and inter-run variation. In C_1, the LLM receives the source model in XMI format and a concise textual description of the target metamodel. This is the most implicit configuration: the LLM must infer both the transformation intent and the target structure. Listing 1 shows the *Persons* target-metamodel description used for *Families2Persons*. These descriptions were generated with LLM support and manually checked by the authors. In C_2, the LLM receives the source metamodel, source model, and target metamodel, but no transformation rules. In C_3, the same artifacts are complemented with a concise natural-language description extracted from the original ATL rules; Listing 2 shows an example.

Listing 1: Excerpt of the metamodels descriptions

Table 1: Structural metrics of the considered transformations.

Transformation Name	Bindings	Bindings Obj Res	Input Patterns	Ouput Patterns	Matched Rules	Rules With Using	Helpers	Attribute Helpers	Attribute Helper Ctx	Operational Helpers	LoC
Book2Publication	3	0	1	1	1	0	3	0	0	3	50
Families2Persons	2	0	2	2	2	0	2	1	0	1	49
IEEE1471_2_MoDAF	54	18	14	14	13	0	1	1	1	0	229
Make2Ant	16	4	5	6	5	0	0	0	0	0	88
Table2SpreadsheetMLSimplified	12	6	3	9	3	2	1	0	0	1	117

```

- **Target Model Description:**
{model_description}
In this system, everyone is a Person, and every
    Person must have a full name. However, a
    person cannot just be a generic human; they
    must be explicitly classified as either a Male
     or a Female.
```

In C_4, the LLM receives the complete transformation context: source metamodel, source model, target metamodel, and ATL transformation file. Listing 3 shows the most explicit prompt configuration, C_4. The prompts were minimally adapted across configurations to reflect the available inputs. For example, in C_3 the prompt states that the transformation rules are expressed in natural language, whereas in C_4 it asks the LLM to analyze and execute the provided ATL rules.

Listing 2: Excerpt of the ATL transformation rules descriptions

```

- **ATL Rules:**
{textual_atl}
assess family name, understand is female, from
    Member to Male, from Member to Female
```

Model generation. For each evaluation cell, we queried DeepSeek-V4 Flash, GPT 5.4 Nano, and Gemini-3.1 Flash Lite ten times using the same input artifacts and prompt configuration. Repeated executions allow us to observe whether equivalent transformation conditions lead to stable target models or to variation across generated outputs. Table 2 summarizes the evaluation settings.

Analysis. We analyze the generated target models along three complementary dimensions: reference deviation, inter-run variation, and difference morphology. *Reference deviation* compares each generated target model with the deterministic ATL reference output for the same transformation scenario. It captures how far an LLM-generated model is from the expected ATL-based result. *Inter-run variation* compares generated models within the same evaluation cell. With ten executions per cell, this yields $\frac{10(10-1)}{2} = 45$ non-repeated pairwise comparisons per cell. It captures whether the LLM produces stable outputs under equivalent conditions. We compute differences with XMLDiff 3.0 [2]. The resulting counts provide an XMI-level approximation of model variation, not a proof of semantic difference. To avoid inflating the main distance measure with ordering effects, we exclude moved nodes from the main count because XML ordering changes do not necessarily correspond to model-level changes. Moved-node counts are retained separately in the replication package. For each reference and inter-run comparison, we report summary statistics including mean, median, standard deviation, minimum, and maximum difference counts. We also record exact matches, stable but not ATL-equivalent cases, and cases where a single exact match coexists with non-zero median deviation across repeated executions. To interpret what the numerical differences mean for MDE, we classify the observed XMLDiff operations into difference categories: namespace changes, attribute-level changes, node-level or structural changes, moved nodes, and text/comment changes. This procedure represents our *morphological analysis.*

Listing 3: Prompt employed for LLMs inference

```

### CONTEXT
You will be provided with four essential
    components:
1. **Source Metamodel (.ecore):...
2. **Source Model (.xmi):...
3. **Target Metamodel (.ecore):...
4. **ATL Transformation Rules (.atl):...

### INPUT DATA
- **Source Metamodel:**
{mm_in}

- **Source Model (XMI):**
{m_in}

- **Target Metamodel:**
{mm_out}

- **ATL Rules:**
{atl}

### TASK & CONSTRAINTS
- Strictly analyze the ATL rules and apply
    them to the elements of the Source Model.
- Ensure the resulting output adheres to the
    constraints and definitions of the Target
    Metamodel.
- **REQUIRED OUTPUT:** Provide ONLY the
    resulting XML/XMI code for the target
    model.
- **STRICT PROHIBITION:** Do not include any
    explanations, introductory text, or
    closing remarks.

### OUTPUT
[Generate the resulting XMI file here]]
```

We also record parsing, comparison, and conformance-checking failures. When minor namespace inconsistencies prevent otherwise

[2] https://xmldiff.readthedocs.io/en/stable/api.html

analyzable models from being compared, we apply minimal transparent normalization, such as replacing `xmlns="http://Persons"` with `xmlns="Persons"`. Normalized models are reported separately. This classification is needed because the same numerical distance may correspond to different modeling consequences, ranging from serialization-level variation to changes affecting conformance, traceability, or semantic preservation.

Table 2: Summary of the experimental settings.

LLMs	DeepSeek V4 Flash, GPT 5.4 Nano Gemini-3.1 Flash Lite
Transformations	Book2Publication, Families2Persons, IEEE1471_2_MoDAF, Make2Ant, Table2SpreadsheetMLSimplified (T_1,T_2,T_3,T_4,T_5)
Configurations	4 (C_1, C_2, C_3, C_4)
Runs	10
Elements Analysed	Models Differences, Exceptions
Metrics	Mean Diff, Median Diff, Standard Deviation, Skewness
Temperature	0 for every run.

4 Results

This section reports the results in terms of: deviation from deterministic ATL outputs, inter-run variation across repeated executions, and morphology of observed differences. Detailed per-run data, generated models, scripts, and supplementary analyses are available in the replication package (Section 8).

Reference deviation from ATL outputs. Figure 2 reports the XMLDiff-based deviation between generated target models and deterministic ATL reference outputs. Deviations occur across transformations, prompt configurations, and LLMs. The smallest deviations appear for *Families2Persons* and *Book2Publication*, while structurally richer transformations generally lead to larger deviations. Across configurations, C_1 and C_2 generally produce higher deviation counts than the more explicit configurations. C_4, where the complete ATL transformation is provided, reduces deviations in several cases but does not eliminate them: 13 out of 15 transformation-scenario–LLM combinations still show non-zero mean and median deviation from the ATL reference. Table 3 reports parsing, comparison, and conformance-related exceptions. These exceptions are important for interpreting the distance results, because reference deviation and inter-run variation can only be computed for comparable outputs. Exception rates vary across LLMs and configurations: Gemini has the highest exception rate, with 96 exceptions over 195 runs (49.8%), while GPT and DeepSeek have rates of 28.0% and 24.0%, respectively. Across configurations, C_4 has the lowest exception rate (11.3%), whereas C_1, C_2, and C_3 have rates of 40.7%, 35.2%, and 47.3%.

Table 3: Details of encountered exceptions.

Criterion	Runs	Exceptions	Nsuri Exc	Rate%
GPT	200	56	31	28.0
DeepSeek	200	48	23	24.0
Gemini	195	96	40	49.8
T_1	119	31	25	26.1
T_2	120	28	24	23.3
T_3	116	46	9	39.7
T_4	120	51	22	34.2
T_5	120	54	14	45.0
C_1	150	61	38	40.7
C_2	145	51	14	35.2
C_3	150	71	36	47.3
C_4	150	17	6	11.3

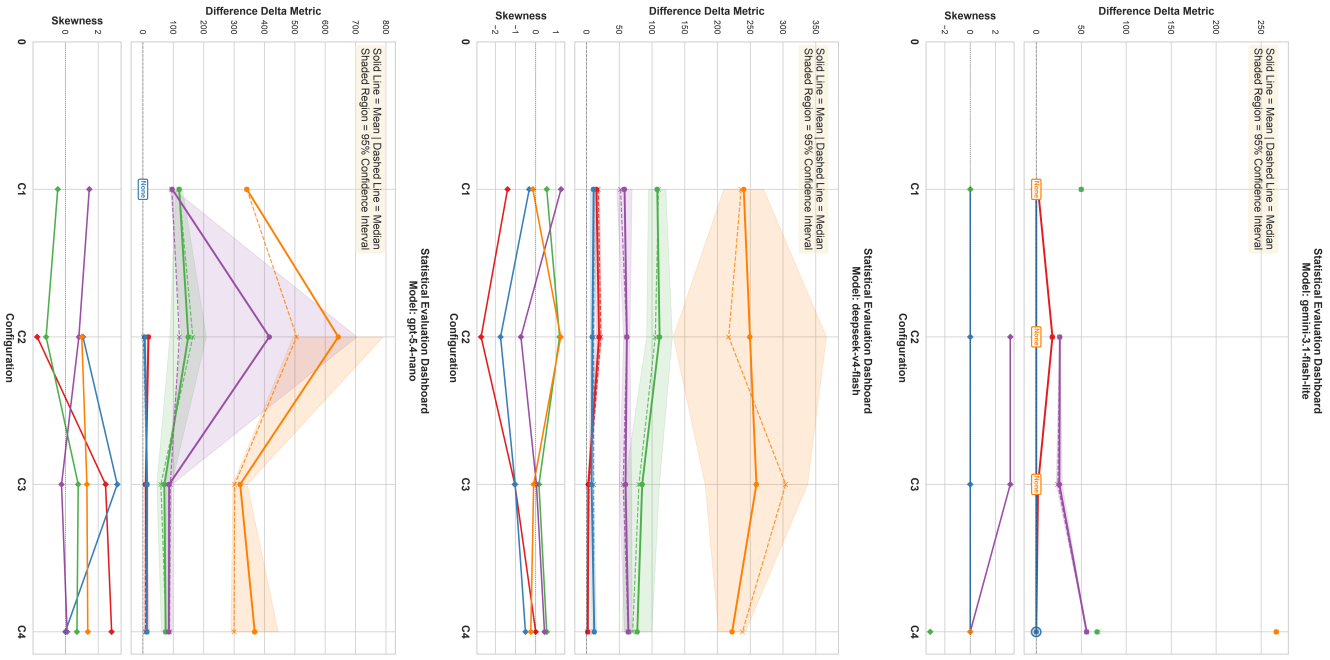
Inter-run variation across repeated executions. Figure 3 reports pairwise differences among generated models produced within the same transformation scenario, prompt configuration, and LLM. Inter-run differences are generally lower than deviations from the ATL reference, but variation remains present across configurations. In C_4, 10 out of 15 transformation-scenario–LLM combinations show non-zero mean inter-run variation, and 9 out of 15 show non-zero median inter-run variation. The remaining five combinations produce identical outputs across repeated executions. These results show that repeated executions can produce stable outputs, variable outputs, and occasional large deviations under otherwise equivalent transformation conditions. Low inter-run variation should therefore not be interpreted in isolation: it must be considered together with reference deviation and with the exception rates reported in Table 3, since non-comparable outputs are excluded from distance computations.

Difference morphology. Table 4 shows XMLDiff operations grouped into five categories: attribute-level changes, node-level or structural changes, moved nodes, namespace changes, and text/comment changes. The table reports aggregate distributions for both comparison types: generated models against ATL references, and generated models against each other within the same evaluation cell. For both comparison types, most differences concern attributes and node-level or structural elements. Attribute-level changes account for 55.8% of ATL-reference differences and 61.6% of inter-run differences, while node-level or structural changes account for 34.6% and 29.1%, respectively. At the operation level, *InsertAttrib* is the most frequent difference, with 19,315 occurrences in the ATL-reference comparison and 38,370 in the inter-run comparison. This confirms that variation is not only a matter of serialization or namespace handling; many differences affect model content that may be relevant for conformance, traceability, and semantic preservation.

5 Implications for Nondeterminism-Aware MDE Evaluation

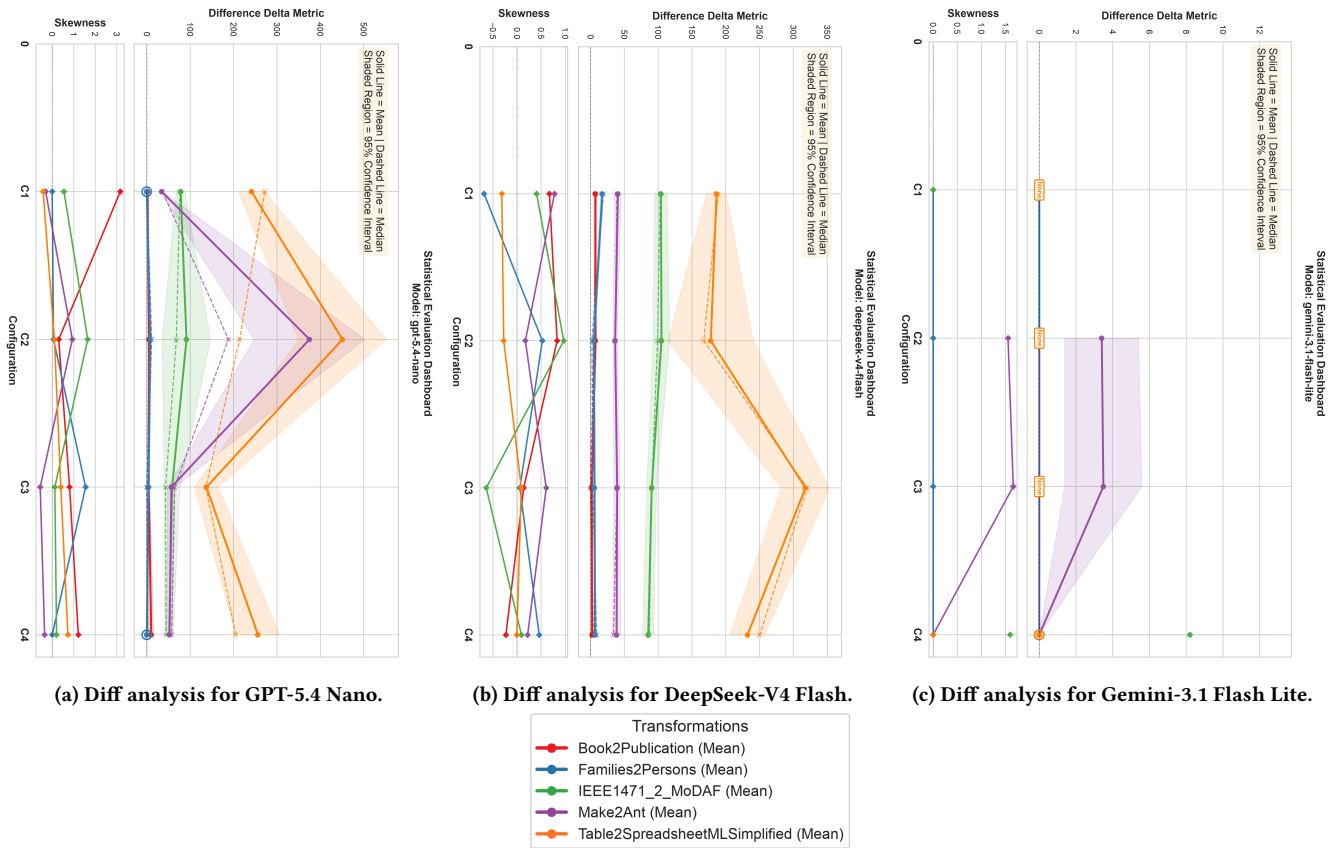
The results in Section 4 suggest that LLM-based model operations should be evaluated as repeated operations, not only as isolated generated artifacts. For deterministic and quasi-deterministic MDE workflows, nondeterminism-aware evaluation should make five dimensions explicit.

(1) *Executability is the entry condition.* Reference deviation and inter-run variation can be measured only for generated models that are parseable, comparable, and analyzable. As Table 3 shows, this condition cannot be assumed. Evaluations should therefore report exceptions, normalization steps, and comparable outputs separately from distance-based metrics.
(2) *Correctness and reproducibility must be separated.* Figures 2 and 3 capture different properties: agreement with the deterministic ATL output and stability across repeated executions. Exact or near-exact outputs can coexist with deviations in



(a) Diff analysis for GPT-5.4 Nano. (b) Diff analysis for DeepSeek-V4 Flash. (c) Diff analysis for Gemini-3.1 Flash Lite.

Figure 2: Comparison of Diff between generated elements and the original.



(a) Diff analysis for GPT-5.4 Nano. (b) Diff analysis for DeepSeek-V4 Flash. (c) Diff analysis for Gemini-3.1 Flash Lite.

Figure 3: Comparison of Diff among the generated elements.

Table 4: Details of the differences of the generated models against ATL references, and against each other.

	Generated models against ATL references													Generated models against each other										
Diff	**Total**	**GPT**	**Gemini**	**DeepSeek**	T_1	T_2	T_3	T_4	T_5	C_1	C_2	C_3	C_4	**Total**	**GPT**	**Gemini**	**DeepSeek**	T_1	T_2	T_3	T_4	T_5	C_1	C_2	C_3	C_4
DeleteAttrib	1826	54.5	25.6	19.9	0.0	0.0	20.6	36.6	42.8	22.3	12.7	22.7	42.3	15697	55.6	0.6	43.9	0.2	1.1	20.1	17.7	60.9	26.0	18.4	26.8	28.8
DeleteNamespace	260	46.9	3.8	49.2	14.2	18.5	13.5	25.8	28.1	13.1	23.1	29.2	34.6	509	4.5	0.0	95.5	19.6	26.1	17.1	12.0	25.1	26.9	21.0	27.3	24.8
DeleteNode	2281	61.9	7.9	30.2	0.7	0.0	19.7	18.8	60.8	35.8	35.0	16.5	12.7	15981	76.5	0.2	23.4	0.4	0.1	7.8	15.6	76.0	11.1	59.5	14.0	15.4
InsertAttrib	19315	65.1	11.2	23.7	1.7	0.4	14.9	24.2	58.9	11.7	38.5	19.6	30.2	38370	71.2	0.4	28.4	0.8	0.9	11.8	31.8	54.7	16.9	45.1	18.4	19.7
InsertComment	17	0.0	0.0	100.0	0.0	0.0	5.9	5.9	88.2	35.3	0.0	35.3	29.4	65	0.0	0.0	100.0	0.0	0.0	6.2	9.2	84.6	47.7	0.0	44.6	7.7
InsertNamespace	328	59.8	0.0	40.2	22.0	25.9	17.4	17.4	17.4	7.6	29.6	29.6	33.2	546	10.8	0.0	89.2	20.7	27.1	15.4	12.3	24.5	26.7	22.5	24.7	26.0
InsertNode	5438	70.1	4.5	25.4	2.1	0.4	16.2	24.0	57.4	20.1	49.1	13.0	17.8	9222	69.8	0.6	29.6	1.3	0.2	12.7	30.3	55.6	13.7	46.5	17.0	22.8
RenameAttrib	131	29.8	15.3	55.0	0.0	6.9	27.5	62.6	3.1	50.4	14.5	10.7	24.4	3485	91.0	0.3	8.7	0.0	2.3	3.9	5.8	88.0	13.1	2.1	37.2	47.6
RenameNode	6738	41.0	17.1	42.0	0.8	6.8	13.2	13.5	65.6	8.7	16.0	27.0	48.4	13608	46.4	1.4	52.3	1.3	4.2	22.5	16.1	55.9	37.4	12.8	22.1	27.7
UpdateAttrib	2028	25.4	17.1	57.5	12.3	0.1	26.4	27.2	34.1	11.9	28.9	21.9	37.2	24637	60.4	0.9	38.8	1.4	0.2	18.3	9.6	70.5	26.3	22.6	23.9	27.2
UpdateTextIn	644	99.2	0.0	0.8	0.5	0.0	0.2	0.5	98.9	0.0	91.5	8.4	0.2	931	97.2	0.0	2.8	1.6	0.0	0.6	1.9	95.8	0.0	94.6	4.9	0.4

other runs, while stable outputs can still deviate from the ATL reference. Single-run correctness, reference deviation, and inter-run variation should therefore not be collapsed into one score.

(3) *Transformation explicitness improves evaluability, not trust.* Providing the full ATL transformation in C_4 reduces exceptions, suggesting improved parsability and comparability. However, C_4 still exhibits reference deviation and inter-run variation. Prompt explicitness improves the conditions for evaluation but does not make LLM behavior equivalent to the execution of deterministic transformations.

(4) *Reliability requires distributions, not only averages.* Mean differences can hide large deviations, exact matches coexisting with non-zero medians, and failures excluded from distance computations. Evaluations should therefore report medians, exact matches, exception rates, and per-cell distributions, not only aggregate averages.

(5) *Variation needs model-aware interpretation.* As Table 4 shows, most differences affect attributes and node-level or structural elements, while namespace, move, and text/comment changes are less frequent. Raw XMLDiff counts make variation observable, but they should be complemented by model-aware, conformance-aware, and semantic-aware interpretation. LLM-based MDE evaluation should therefore report variation profiles, not only success rates or aggregate distances.

Together, these implications define nondeterminism-aware evaluation as a practical basis for assessing LLM-based model operations before they are integrated into deterministic MDE workflows. Generated model operations should be assessed through executability, correctness, reproducibility, distributional reliability, and the modeling meaning of variation before they are integrated into deterministic workflows.

6 Threats to validity

This study provides exploratory evidence on nondeterminism-aware evaluation, not a definitive assessment of LLM suitability for model transformation. The main construct threat concerns the use of XMLDiff. XML-level differences make variation observable, but they do not establish semantic equivalence or semantic difference. Namespace changes or moved nodes may be harmless, while small attribute or reference changes may affect model meaning. We mitigate this by reporting difference morphology separately and excluding moved nodes from the main distance measure. Future work should add model-aware, conformance-aware, and semantic-aware comparison. A related threat is the use of ATL outputs as deterministic references: our results measure deviation from ATL reference outputs, not full semantic correctness. Results may be affected by prompt construction, generated outputs, comparison scripts, or exception handling. We reduce this risk by applying the same procedure across transformations, configurations, and LLMs, and by providing prompts, artifacts, scripts, exceptions, normalization steps, and detailed results in the replication package. Possible data leakage cannot be ruled out because ATL Zoo artifacts may have appeared in public training data. However, the study focuses on repeated-execution behavior, deviation, exceptions, and difference morphology rather than absolute performance or LLM ranking. The evaluation covers five ATL Zoo transformation scenarios, four prompt configurations, three cost-efficient LLMs, and ten executions per evaluation cell. This is sufficient for an exploratory NIER study, but it does not support broad generalization across all MDE tasks, transformation languages, modeling notations, or LLM families. Larger models, fine-tuned models, agentic workflows, or specialized MDE tools may behave differently. Exact replication may be affected by provider-side model updates, unavailable model versions, or hidden decoding settings. We mitigate this by documenting the evaluated models, prompts, generated artifacts, comparison scripts, and detailed results in the replication package.

7 Conclusion and future plans

This paper introduced *nondeterminism-aware evaluation* for LLM-based MDE. The main message is that correctness in one execution is not enough: generated model operations must also be evaluated for executability, reproducibility, reference deviation, and the modeling meaning of variation.

Our future plans will move from observing nondeterminism to engineering workflows that make it measurable and controllable. First, we will define evaluation contracts for LLM-based model operations, specifying evidence such as parseability, metamodel conformance, constraint satisfaction, exception rates, deviation distributions, and inter-run variation. Second, we will develop model-aware differencing and adequacy checks that relate observed changes to metamodel elements, references, containment structures, traceability links, and semantic constraints. Third, we will investigate *agentic model transformation*, in which LLMs generate and revise candidate transformations, while deterministic MDE services provide metamodel access, checking, execution, diagnostics, comparison, and repair. The longer-term goal is to provide the MDE community with reusable evidence standards and service-based infrastructure for LLM-assisted MDE, so that generated artifacts are not treated

as isolated successes but as outputs of inspectable, repeatable, and controllable modeling workflows.

8 Data availability

The replication package, including prompts, input artifacts, generated models, comparison scripts, detailed results, and supplementary analyses, is available at [1].

Acknowledgments

This work is supported by the Swedish Agency for Innovation Systems through the project "iSecure: Developing Predictable and Secure IoT for Autonomous Systems" (2023-01899), by the Key Digital Technologies Joint Undertaking through the project "MATISSE: Model-based engineering of digital twins for early verification and validation of industrial systems" (101140216), and by the Clean Energy Transition Partnership through the project "FLEXI: Human-centered AI and digital twin powered energy system integration for flexibility markets" (101069750).

References

[1] Anonym. 2026. Replication Package. https://zenodo.org/records/21107094.
[2] Thorsten Arendt, Enrico Biermann, Stefan Jurack, Christian Krause, and Gabriele Taentzer. 2010. Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations. In *Model Driven Engineering Languages and Systems - 13th International Conference, MODELS 2010, Oslo, Norway, October 3-8, 2010, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 6394)*. Springer, 121–135. doi:10.1007/978-3-642-16145-2_9
[3] Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, et al. 2024. Non-determinism of" deterministic" llm settings. *arXiv preprint arXiv:2408.04667* (2024).
[4] Önder Babur, Sébastien Mosser, and Alfonso Pierantonio. 2026. Selling Shovels in the LLM Gold Rush: Why Software Engineering Research Risks Missing the Real Transformation. *J. Object Technol.* 25, 2 (2026), 1.
[5] Enrico Biermann, Claudia Ermel, and Gabriele Taentzer. 2012. Formal foundation of consistent EMF model transformations by algebraic graph transformation. *Softw. Syst. Model.* 11, 2 (2012), 227–250. doi:10.1007/S10270-011-0199-7
[6] Jesús Sánchez Cuadrado and Jesús M. Perera Aracil. 2014. Scheduling model-to-model transformations with continuations. *Softw. Pract. Exp.* 44, 11 (2014), 1351–1378. doi:10.1002/SPE.2202
[7] Jesús Sánchez Cuadrado, Esther Guerra, and Juan de Lara. 2014. Reverse Engineering of Model Transformations for Reusability. In *Theory and Practice of Model Transformations - 7th International Conference, ICMT@STAF 2014, York, UK, July 21-22, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8568)*, Davide Di Ruscio and Dániel Varró (Eds.). Springer, 186–201. doi:10.1007/978-3-319-08789-4_14
[8] Jesús Sánchez Cuadrado, Esther Guerra, and Juan de Lara. 2018. AnATLyzer: an advanced IDE for ATL model transformations. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 85–88. doi:10.1145/3183440.3183479
[9] Krzysztof Czarnecki and Simon Helsen. 2006. Feature-based survey of model transformation approaches. *IBM Syst. J.* 45, 3 (2006), 621–646. doi:10.1147/SJ.453.0621
[10] Duy Dao, Alessio Bucaioni, and Antonio Cicchetti. 2025. Learning to Transform: Evaluating LLMs on Model Transformation by Example. In *MDE Intelligence 2025*. http://www.es.mdu.se/publications/7236-
[11] Duy Dao, Alessio Bucaioni, and Antonio Cicchetti. 2026. Change-aware round-trip benchmarking of LLMs for reliable and efficient artifact co-evolution. *Journal of Systems and Software* (2026), 113014.
[12] Juri Di Rocco, Davide Di Ruscio, Claudio Di Sipio, Phuong T Nguyen, and Riccardo Rubei. 2025. On the use of large language models in model-driven engineering: J. Di Rocco et al. *Software and Systems Modeling* 24, 3 (2025), 923–948.
[13] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. 2006. *Fundamentals of Algebraic Graph Transformation*. Springer. doi:10.1007/3-540-31188-2
[14] Romina Eramo and Alessio Bucaioni. 2013. Understanding bidirectional transformations with tggs and jtl. *Electronic Communications of the EASST* 57 (2013).
[15] Eclipse Foundation. [n. d.]. Eclipse Viatra. https://eclipse.dev/viatra/. Accessed: 2026-6-26.
[16] Raja Gond, Aditya K Kamath, Ramachandran Ramjee, and Ashish Panwar. 2026. Llm-42: Enabling determinism in llm inference with verified speculation. *arXiv preprint arXiv:2601.17768* (2026).
[17] Raffaela Groner, Peter Bellmann, Stefan Höppner, Patrick Thiam, Friedhelm Schwenker, and Matthias Tichy. 2024. Predicting the Performance of ATL Model Transformations. In *Software Engineering 2024, Fachtagung des GI-Fachbereichs Softwaretechnik, Linz, Austria, February 26 - March 1, 2024 (LNI, Vol. P-343)*, Rick Rabiser, Manuel Wimmer, Iris Groher, Andreas Wortmann, and Bianca Wiesmayr (Eds.). Gesellschaft für Informatik e.V., 33–34. doi:10.18420/SW2024_4
[18] Object Management Group. [n. d.]. QVT — MOF Query/View/Transformation. https://www.omg.org/spec/QVT/1.3/About-QVT/. Accessed: 2026-6-26.
[19] Reiko Heckel. 2004. Graph Transformation in a Nutshell. In *Proceedings of the School of SegraVis Research Training Network on Foundations of Visual Modelling Techniques, FoVMT 2004, Dagstuhl, Germany, May 3-7, 2004 (Electronic Notes in Theoretical Computer Science, Vol. 148)*. Elsevier, 187–198. doi:10.1016/J.ENTCS.2005.12.018
[20] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79.
[21] Gabriel Kazai, Ronnie Agyeiwaa Osei, Alessio Bucaioni, and Antonio Cicchetti. 2025. Model Transformations Using LLMs Out-of-the-Box: Can Accidental Complexity Be Reduced?. In *First Workshop on Large Language Models For Generative Software Engineering*. http://www.es.mdu.se/publications/7201-
[22] Jochen Malte Küster. 2006. Definition and validation of model transformations. *Softw. Syst. Model.* 5, 3 (2006), 233–259. doi:10.1007/S10270-006-0018-8
[23] Tom Mens and Pieter Van Gorp. 2006. A taxonomy of model transformation. *Electronic notes in theoretical computer science* 152 (2006), 125–142.
[24] Zahra Moezkarimi, Kevin Eriksson, Albin Alm Johansson, Alessio Bucaioni, and Marjan Sirjani. 2024. Harnessing ChatGPT for Model Transformation in Software Architecture: From UML State Diagrams to Rebeca Models for Formal Verification. In *4th International Workshop of Model-Driven Engineering for Software Architecture*. http://www.ipr.mdu.se/publications/7130-
[25] Olga Runge. [n. d.]. The Attributed Graph Grammar System. https://www.user.tu-berlin.de/o.runge/AGG/WWW/agg.html. Accessed: 2026-6-26.
[26] Shane Sendall and Wojtek Kozaczynski. 2003. Model transformation: The heart and soul of model-driven software development. *IEEE software* 20, 5 (2003), 42–45.
[27] Perdita Stevens. 2007. A landscape of bidirectional model transformations. *International Summer School on Generative and Transformational Techniques in Software Engineering* (2007), 408–424.
[28] Massimo Tisi, Salvador Martínez Perez, and Hassene Choura. 2013. Parallel Execution of ATL Transformation Rules. In *Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 8107)*, Ana Moreira, Bernhard Schätz, Jeff Gray, Antonio Vallecillo, and Peter J. Clarke (Eds.). Springer, 656–672. doi:10.1007/978-3-642-41533-3_40
[29] Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. 2026. Understanding and mitigating numerical sources of nondeterminism in llm inference. *Advances in Neural Information Processing Systems* 38 (2026), 169819–169851.
[30] Zhenting Zhu, Lucas Thai, Shan Yu, Yicheng Liu, Yifan Qiao, Chenxi Wang, Harry Xu, and Junyi Shu. 2026. Demystifying Numerical Instability in LLM Inference: Achieving Reproducible Inference for Mission-Critical Tasks with HEAL. *arXiv preprint arXiv:2606.21023* (2026).