JOT
JOURNAL OF
OBJECT TECHNOLOGY

Journal of Object Technology | **RESEARCH ARTICLE**

Leveraging LLMs for Grammar Adaptation: A Study on Metamodel-Grammar Co-Evolution

Weixing Zhang[*], Bowen Jiang[*], Rahul Sharma[*], Regina Hebig[†], and Daniel Strüber[‡,§]
[*]Karlsruhe Institute of Techonology, Germany
[†]Universität Rostock, Germany
[‡]Chalmers University of Technology and University of Gothenburg, Sweden
[§]Radboud University, The Netherlands

ABSTRACT In model-driven engineering, metamodel evolution leads to the need to adapt corresponding grammars to maintain consistency, which typically requires tedious manual work. Existing rule-based methods can achieve partial automation but have limitations when handling complex grammar scenarios. This paper proposes a Large Language Model-based approach that automatically applies adaptations to new grammars after evolution by learning grammar adaptations from previous versions. We evaluated this approach on six real-world Xtext domain-specific languages, using four DSLs as a training set to develop prompting strategies, two DSLs as a test set for validation, and conducting a longitudinal case study on QVTo. The evaluation used three Large Language Models (Claude Sonnet 4.5, ChatGPT 5.1, Gemini 3) and measured grammar adaptation quality from three dimensions: grammar rule-level adaptation consistency, output similarity, and metamodel conformance. Results show that on the test set, all three LLMs achieved 100% adaptation consistency and output similarity, while the rule-based approach achieved only 84.21% on DOT and 62.50% on Xcore. In the QVTo longitudinal study, the LLM-based approach successfully reused learned adaptations across all three evolution steps without manual grammar editing, while the rule-based approach required manual adjustments in two of three transitions. However, on large-scale grammars (EAST-ADL, 297 rules), LLMs' adaptation consistency was far below 90%. This study demonstrates the advantages of LLM-based approaches in handling complex grammar scenarios, while revealing their limitations in large-scale grammar adaptation.

KEYWORDS Large language models, Evolution, Domain-specific language, Metamodel, Grammar.

1. Introduction

In model-driven engineering (MDE), evolution is a long-standing core challenge (Paige et al. 2016). When a metamodel changes, various software artifacts that depend on it—including model instances, model transformation rules, constraint definitions, etc.—need to be adapted accordingly to maintain consistency (Meyers & Vangheluwe 2011; Hebig et al. 2016). This multi-artifact co-evolution problem has been widely studied, with existing work mainly focusing on co-evolution between models and metamodels (Kessentini et al. 2019; Cicchetti et al. 2008b,a), as well as co-evolution between model transformations and metamodels (García et al. 2012; Kessentini et al. 2018). However, in the development and maintenance of domain-specific languages (DSLs), there exists another type of co-evolution scenario: when the abstract syntax (usually represented as a metamodel) evolves, the grammar definition that specifies the concrete syntax also needs to evolve accordingly (Ráth et al. 2010; Tolvanen et al. 2025). This metamodel-driven approach is particularly common in scenarios where the metamodel is the central development artifact in a larger ecosystem, e.g., *blended modeling* (Ciccozzi et al. 2019), where several concrete syntaxes for the same underlying metamodel exist and evolve in parallel. When DSL development adopts

JOT reference format:
Weixing Zhang, Bowen Jiang, Rahul Sharma, Regina Hebig, and Daniel Strüber. *Leveraging LLMs for Grammar Adaptation: A Study on Metamodel-Grammar Co-Evolution*. Journal of Object Technology. Vol. 25, No. 3, 2026.
http://dx.doi.org/10.5381/jot.2026.25.3.a13

the metamodel-driven approach, it is necessary to support co-evolution between metamodels and grammar definitions.

One widely used language framework, Xtext (Eclipse Foundation 2025; Erdweg et al. 2013), supports this metamodel-centric DSL development. In Xtext, the metamodel defines the abstract syntax of the language, while the grammar defines the concrete syntax, and the two adhere to each other. When the metamodel exists first, we can generate grammar from it (Bettini 2016). The generated grammar usually has to be manually adapted, since it may include verbose formatting, e.g, redundant parantheses and keywords, leading to the modification of G_1 to derive G_1' in Figure 1(a)) (Bettini 2016). However, when the metamodel evolves (from M_1 to M_2), language engineers need to adapt G_1' according to the changes in the metamodel (as shown in G_1' to G_2' in Figure 1(a)). This adaptation is particularly laborious and error-prone, since the language engineers manually have to create new rules for newly added elements, based on their understanding of the metamodel changes, an involved task especially in case of complex changes. Xtext provides another alternative scenario (i.e., scenario (b) in Figure 1), where, after the metamodel evolves, language engineers directly generate grammar G_2 from the evolved metamodel M_2 and perform the adaptation on G_2 that occurred in G_1 to G_1'. However, this introduces repetitive work and is equally prone to errors.

To address these challenges in scenario (b), previous research proposed the GrammarTransformer tool (Zhang et al. 2024), which contains 60 predefined grammar adaptation rules that enable automated grammar adaptation through configuration. This approach reduces the complexity of manual modification to some extent; however, when adaptation requirements vary significantly across grammar rules, the configuration process can still become cumbersome. Subsequent work further attempted to automatically extract adaptation configurations by comparing the generated grammar with the target grammar, but its adaptation accuracy still needs improvement (Zhang, Hebig, Strüber, & Steghöfer 2023).

To propose an improved solution to the above challenges, this study explores using Large Language Models (LLMs) to automate the grammar adaptation process. LLMs have demonstrated strong capabilities in code understanding (Wang et al. 2023) and transformation tasks (Cummins et al. 2024), suggesting potential for learning adaptations from examples. A recent systematic mapping of LLM applications in MDE confirms broad interest in this intersection, while also revealing that tasks such as DSL Engineering remain marginal, with only three of 86 surveyed studies addressing them (Zhang, Jiang, Fu, Cheng, et al. 2026). Unlike rule-based methods, our approach does not rely on predefined adaptation rules, but instead investigates whether LLMs can automatically adapt new grammars by learning from historical adaptations. Specifically, we provide the LLM with the grammar generated from the original metamodel (G_1) and its manually adapted version (G_1'), as well as the new grammar generated from the evolved metamodel (G_2), to evaluate whether the LLM can identify the adaptations from G_1 to G_1' and apply the same adaptations to G_2 to generate G_2'. The generated G_2' needs to both maintain consistency with the evolved metamodel and reproduce the manual adaptations from the previous version.

To evaluate the LLM-based approaches, we propose the following research questions:

RQ1: To what extent can LLMs learn and apply grammar adaptations from a prior-version grammar pair (G1, G'1)?

RQ2: To what extent can LLM-based approaches reuse learned adaptations across consecutive evolution steps with minimal human intervention?

RQ3: What are the strengths and limitations of LLM-based approaches compared to rule-based methods for grammar adaptation?

We systematically evaluated the proposed approach on six real-world DSLs from prior work (Zhang, Hebig, Strüber, & Steghöfer 2023), using a training set (EAST-ADL (EAST-ADL Association 2022), BibTeX (Paperpile 2022), Xenia (Misha Rodchenkov 2019), SML (Greenyer 2018)) to develop prompting strategies and validating on a test set (DOT (Eclipse Foundation AISBL 2020), Xcore (Eclipse Foundation 2018)). Additionally, we conducted a longitudinal case study on QVTo spanning four official versions. We used three representative current LLMs (Claude Sonnet 4.5 (Anthropic 2025), ChatGPT 5.1 (OpenAI 2025), Gemini 3 (Studio 2025)) in the evaluation. Results show that on the test set DSLs, all three LLMs outperformed the rule-based approach in adaptation consistency and output similarity. In the QVTo longitudinal study, the LLM-based approach successfully reused learned adaptations across all three evolution steps without manual grammar editing, while the rule-based approach required configuration adjustments in two of the three transitions. However, on large-scale grammars such as EAST-ADL with 297 rules, LLMs encountered challenges, exhibiting systematic omission of identified adaptation operations. This study demonstrates that LLM-based approaches can reduce language engineers' workload, particularly in handling complex grammar scenarios that are difficult for rule-based methods to address (such as syntactic predicates, predicated assignments, and order-insensitive attribute combinations), while also revealing the limitations of current LLMs in large-scale grammar adaptation tasks, pointing out directions that require further exploration in the field of metamodel-grammar co-evolution.

2. Background

2.1. Metamodel-Driven Development and Grammar adaptation

Xtext is a widely-used language development framework (Erdweg et al. 2013) that supports both grammar-driven and metamodel-driven development workflows (Bettini 2016; Paige et al. 2014). In the grammar-driven development workflow, language engineers primarily define the concrete syntax by editing the grammar, from which the corresponding metamodel is automatically generated. In contrast, in the metamodel-driven development workflow, language engineers first create a metamodel to represent domain knowledge and capture the abstract syntax of the language. From this metamodel, the grammar can be automatically generated.

As mentioned in Section 1, the generated grammar contains

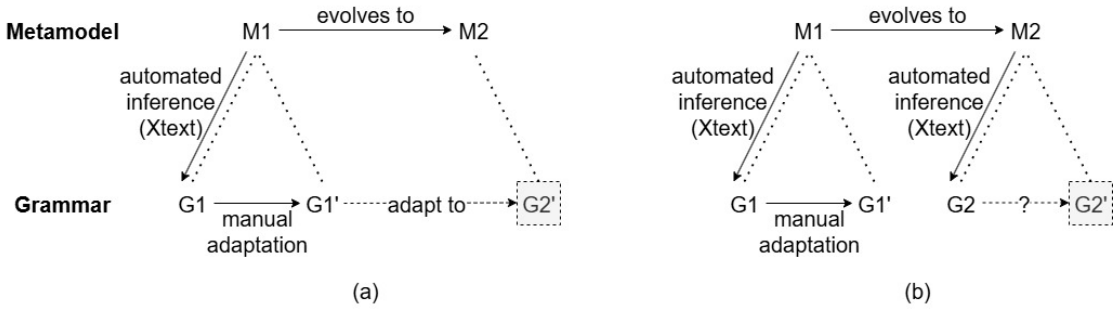

Figure 1 Challenges in metamodel-grammar co-evolution: (a) Direct evolution scenario where language engineers manually propagate metamodel changes from M_1 to M_2 into the grammar evolution from G_1' to G_2' after metamodel evolution; (b) Regeneration scenario where G_2 is generated from M_2 and the adaptations from G_1 to G_1' need to be reapplied to obtain G_2'.

multiple grammar rules and attributes that correspond one-to-one with the metaclasses and properties in the metamodel, sharing the same names (Bettini 2016). The grammar rules of generated grammar always begin with the same name as the metaclass and often contain a keyword with the same name as the metaclass, while attributes are enclosed in curly braces (Bettini 2016). Similarly, attribute keywords and names are the same as the corresponding properties defined in the metamodel.

```
Mission returns Mission:
  'Mission'
  '{'
    'shortName' shortName=Identifier
    ('category' category=Identifier)?
    ('uuid' uuid=String0)?
    ('name' name=String0)?
    ('ownedComment' '{' ownedComment+=Comment ( "," ownedComment+=Comment)* '}' )?
  '}';
```

Listing 1 A grammar rule `Mission` before adaptation.

```
Mission returns Mission:
    'Mission'
         shortName=Identifier
    ('{'
        ('category' category=Identifier ';')?
        ('uuid' uuid=UUID ';')?
        ('name' name=Identifier ';')?
        ( ownedComment+=Comment ( ownedComment+=Comment)* )?
    '}')?;
```

Listing 2 The grammar rule `Mission` after adaptation.

Grammar adaptation refers to the structured textual modification of a grammar to adjust, refine, or improve its concrete syntactic form. In the literature, this process is also known as *grammar transformation* (Zhang et al. 2024) or *grammar optimization* (Zhang, Hebig, Strüber, & Steghöfer 2023). Grammar adaptation operates at multiple syntactic levels, including entire grammar rules, attributes, keywords, symbols (e.g., braces or parentheses), multiplicities, optionality, and other concrete elements.—and consists of adding, removing, or changing these textual elements to obtain a desired concrete syntax style or behavior (Zhang et al. 2024). For example, Listing 1 displays a grammar rule `Mission` containing an attribute `ownedComment`, whose keyword is identical to the attribute name. Through adaptation, the keyword in the attribute is removed (see Listing 2).

While grammar adaptation primarily concerns concrete-syntax refinements, it may intentionally alter the abstract syntax tree structure derived from the grammar when the adaptation involves modifying assignments, types, multiplicities, or rule structure. In Xtext, such structural elements—rather than keywords or other concrete tokens—determine the resulting EMF-based abstract syntax tree (Bettini 2016).

2.2. Rule-based Grammar Adaptation in the Context of Language Evolution

The ***rule-based*** grammar adaptation approach automatically transforms grammars generated from metamodels into target grammars that conform to specific styles and requirements through predefined transformation rules and configuration mechanisms. The core of this approach consists of two tools: *GrammarTransformer* (Zhang et al. 2024) and *ConfigGenerator* (Zhang, Hebig, Strüber, & Steghöfer 2023).

GrammarTransformer contains 60 predefined grammar transformation rules that operate on different elements of the grammar, including keywords, braces, optionality, multiplicity, grammar rule calls, etc. The transformation rules are divided into three types of operations: add, remove, and change. Each rule is associated with a well-defined scope that can be applied at different levels of granularity, such as the entire grammar, a specific grammar rule, or a specific attribute. Applying *GrammarTransformer* requires language engineers to manually configure the application of transformation rules. Taking the transformation from Listing 1 to Listing 2 as an example, the following rules need to be configured: (1) `removeBraces` removes all curly braces in the *Mission* rule; (2) `removeKeyword` removes all keywords such as *'Mission'*, *'ownedComment'*, etc.; (3) `removeOptionality` removes the optionality markers of attributes; (4) `removeComma` removes the comma in the attribute *ownedComment*; and etc.

ConfigGenerator aims to automate the configuration of grammar transformation rules. It automatically extracts the required transformation rule configurations by comparing the generated grammar G_1 with the target grammar G_1'. It automatically extracts transformation rule configurations by comparing G_1 with G_1', which can then be reused for G_2. The extracted configurations can be persisted and reused for the newly generated grammar G_2 in subsequent evolution steps, thereby automati-

Figure 2 LLM-based grammar adaptation process

cally generating G_2' (as shown in Figure 1(b)). This approach achieved fully automated adaptation for languages such as EAST-ADL, BibTeX, and Xenia, but some grammar rules in DOT, Xcore, and SML still could not be fully automatically adapted.

3. Approach

3.1. Overview

Our approach addresses the regeneration scenario illustrated in Figure 1(b). It operates on four grammars: (i) G_1, the grammar generated from the original metamodel; (ii) G_1', the grammar adapted from G_1; (iii) G_2, the grammar generated from the evolved metamodel; and (iv) G_2', the target adapted grammar. The approach takes G_1, G_1', and G_2 as inputs and outputs G_2'.

The approach proceeds in two steps. First, the LLM compares and analyzes G_1 and G_1' to identify and learn the adaptations applied from G_1 to G_1'. Then, the LLM reuses these learned adaptations on G_2 to obtain G_2'. The expected G_2' is intended to continue conforming to the evolved metamodel. We treat the internal processing of the LLM as a black box and evaluate only whether the resulting G_2' conforms to the evolved metamodel and whether the adaptations from G_2 to G_2' are consistent with or similar to those from G_1 to G_1'.

3.2. Experimental Setup and Prompting Strategy

We evaluated our approach using three state-of-the-art LLMs: Claude Sonnet 4.5, Gemini 3, and ChatGPT 5.1, and conducted the experiments on six DSLs adopted from prior work (Zhang, Hebig, Strüber, & Steghöfer 2023): EAST-ADL, BibTeX, Xenia, DOT, Xcore, and SML. Each included a grammar generated from its metamodel and a target grammar, representing diverse language characteristics, e.g., sizes, complexities, and domains. To develop a generalizable prompting strategy while mitigating the risk of overfitting, we split these into a *training set* (EAST-ADL, BibTeX, Xenia, SML) and a *test set* (DOT, Xcore). The training set was designed to cover diverse scenarios: small-scale (BibTeX with 43 grammar rules, Xenia with 15 rules), medium-scale (SML with 96 rules), and large-scale (EAST-ADL with 297 rules).

For prompt design, we followed a manual, expert-in-the-loop development protocol in which prompts were designed and validated on the training set, then frozen before final evaluation. The decision to accept or revise a prompt was guided by evaluating the model's output quality based on the metamodel conformance and the adaptation consistency using metrics defined in Section 3.3. This follows the human-guided refinement process described by Shah (Shah 2024) and rooted in "prompt programming" principles (i.e., stepwise manual prompt improvement) introduced by Reynolds et al. (Reynolds & McDonell 2021).

A single reference LLM, Claude Sonnet 4.5, was selected for prompt development due to its demonstrated reliability in code-related tasks and strong performance in prior benchmarks (Martinez 2025). The core design remained consistent: the LLM received three grammar files (G_1, G_1', G_2), compared G_1 and G_1' to identify adaptations, and applied similar adaptations to G_2 to produce G_2'. We developed prompts starting from the small-to-medium scale DSLs (BibTeX, Xenia, SML), then validated their applicability on the large-scale case (EAST-ADL). The prompts focused on clearly communicating the two-stage task structure—first identifying adaptations from G_1 to G_1', then applying them to G_2— while relying on the prior-version grammar pair to guide the adaptation, rather than encoding task-specific heuristics in the prompt itself. For this exploratory phase, we used the web-based interface of Claude Sonnet 4.5. For the cross-model evaluation, all three LLMs were evaluated manually through their respective web-based interfaces, with no additional automation. We focused on this setup rather than few-shot prompting, as example-based prompting can steer outputs toward the specific examples provided; by keeping the prompt decoupled from any particular DSL examples, we aimed to strengthen external validity and reduce overfitting to specific language characteristics. Further details on the experiment execution procedure are available in our replication package (Zhang 2025).

Importantly, each DSL was processed in an independent LLM conversation session. Within each session, the LLM received only the three grammar files (G_1, G_1', G_2) of that specific DSL; no information from other DSLs was carried over between sessions. The "training set" designation therefore refers to the DSLs used by the researchers to develop and validate the prompting strategy, not to any form of cross-DSL learning by the LLM. The training-test split follows standard evaluation practice to mitigate the risk of overfitting: prompts developed on the training DSLs might only work well on those specific languages, and the test set verifies that the finalized prompts generalize to unseen cases. The finalized prompts were evaluated across all three LLMs to assess cross-model robustness (Zhuo et al. (Zhuo et al. 2024)).

On the test set, we applied the finalized prompts without modification. Each generated grammar G_2' was evaluated along two dimensions: metamodel conformance (i.e., whether G_2' conformed to the evolved metamodel) and adaptation consistency (i.e., whether the adaptations from G_1 to G_1' were performed on G_2). If syntactic or structural issues arose, we provided up to three targeted follow-up prompts (e.g., "Add semicolons after attributes") to assess whether targeted refinement could resolve the problems.

Beyond the cross-sectional evaluation on test set DSLs, we conducted a longitudinal case study on QVTo to provide a more rigorous assessment of the LLM-based approach for supporting language evolution over multiple versions. QVTo offers a particularly relevant opportunity as it represents a real-world DSL with four official versions (1.0 through 1.3) documented by OMG (The Object Management Group 2016). This enables us to evaluate whether adaptations that LLM-based approaches

learned from one version can be successfully reused across subsequent evolution transitions—a critical aspect of practical language evolution support. Specifically, we applied the finalized prompting strategy to each evolution transition (1.0 to 1.1, 1.1 to 1.2, 1.2 to 1.3) and compared the results against rule-based baselines from prior work, using Claude Sonnet 4.5 as the reference model.

3.3. Evaluation Metrics for Grammar Adaptation Quality

To evaluate whether G_2' reproduces the adaptations from G_1 to G_1', we define the following evaluation metrics.

Grammar Rule-level Adaptation Consistency (RAC) RAC quantifies the proportion of grammar rules in G_2' that successfully reproduce the adaptations observed in G_1', calculated as RAC=$N_{\text{correct}}/N_{\text{total}}$. N_{correct} denotes the count of grammar rules that were correctly adapted according to the previously learned adaptations, and N_{total} represents the total count of grammar rules that required adaptation.

Grammar Rule-level Output Similarity To quantify how closely adapted grammars match the target grammar, we adopt the evaluation framework from prior work (Zhang, Hebig, Strüber, & Steghöfer 2023), i.e., for each DSL, we perform a rule-by-rule comparison between the adapted grammar and the target grammar, reporting the following measures: *Same*: Number of grammar rules that are identical between the adapted grammar and the target grammar; *Diff*: Number of grammar rules that differ between the two grammars; *Percent*: Percentage of identical grammar rules (Same / Total × 100%). For the LLM-based approach, we present both initial results (generalized prompts only) and refined results (after targeted follow-ups when needed). For the rule-based approach, we present the original results from (Zhang, Hebig, Strüber, & Steghöfer 2023).

Metamodel Conformance In addition to adaptation consistency, we verify whether each adapted grammar G_2' conforms to the metamodel M_2. This validation is necessary for the LLM-based approach, as the black-box nature of LLMs may produce grammars that are syntactically valid but semantically incompatible with the metamodel. We validate metamodel conformance by: 1) loading G_2' in the Eclipse Modeling Framework with Xtext; 2) checking for metamodel conformance violations; and 3) recording any violations such as undefined primitive types, missing or renamed attributes, etc. We report conformance results as binary (YES/NO) for each DSL, with detailed violation descriptions for non-conforming cases.

4. Evaluation

This section presents the experimental evaluation of our approach, covering prompt finalization, results on test set DSLs, longitudinal evolution support, and a comparative analysis of LLM-based and rule-based methods.

4.1. Prompt Finalization and Experimental Environment

The decision to finalize a prompt was strictly governed by meeting a set of objective quality thresholds on the training set. For a prompt to be accepted and frozen for final evaluation, the LLM output on the training set had to satisfy three criteria: the adapted target grammar G_2' must conform to the metamodel M_2 (i.e., the grammar can be fully resolved), and both the RAC value and the grammar rule-level output similarity (Percent) value exceed 90%. Metamodel conformance of the generated adapted grammars was validated employing the Eclipse Modeling Tools without enforcing strict version requirements. The authors used the March 2024 release, Java 17, and Xtext 2.36.0. All experimental data are available in our replication package at (Zhang 2025).

4.2. Grammar Adaptation Results (RQ1)

On BibTeX, Xenia, and SML, general-purpose two-step prompts met all three quality thresholds without requiring refinement. We then applied the same prompts to the large-scale case (i.e., EAST-ADL with 297 grammar rules) and conducted five iterations with progressively more specific instructions; however, RAC remained far below 90% (see the analysis in Section 4.4.2). Based on these results, we took the general-purpose two-step prompts that had met all quality thresholds on the three small-to-medium scale DSLs as the finalized prompts. Prompt 1, used when providing G_1 and G_1', and instructs the LLM to identify and extract adaptation rules by comparing the two grammars. Prompt 2, used when providing G_2, instructs the LLM to apply the previously learned adaptations to produce the adapted grammar G_2'. For completeness and transparency, we also applied the finalized prompts to EAST-ADL as a large-scale case, following the same evaluation procedure used for the test set DSLs (i.e., allowing up to three follow-up prompts). The results reported in Tables 1-3 for EAST-ADL are drawn from this evaluation, and reflect the scalability limitations of current LLMs, rather than prompt design, that prevented achieving acceptable quality at this scale of grammar (see Section 4.4.2).

> **Prompt 1:** The attachment contains two Xtext grammars for the same language: the grammar generated from the metamodel and the target grammar. Please identify the adaptations required to transform the generated grammar into the target grammar.

> **Prompt 2:** Now, I'm sending you the grammar generated from the evolved metamodel. Please adapt it using the adaptations you learned previously and output the adapted grammar to me.

To address **RQ1**, Tables 1-3 present evaluation results across all six DSLs in terms of RAC, output similarity, and metamodel conformance. We focus our analysis primarily on the test set DSLs (DOT and Xcore), as these results—obtained by applying finalized prompts to unseen languages—demonstrate generalization capability. Training set results are included in Tables 1-3 for completeness but are not the focus of analysis, as prompt development inherently used these DSLs.

Table 1 presents a comparison of grammar rule-level adaptation correctness between three LLM-based methods and the rule-based method on the test set DSLs (DOT and Xcore). In DOT, 19 grammar rules need to be adapted from the generated

Table 1 Grammar Rule-Level Adaptation Correctness: LLM-based vs. Rule-based Approaches on Test Set DSLs

DSL	required adaptations[1]	Claude-Sonnet-4.5-Based Approach		ChatGPT-5.1-based Approach		Gemini-3-based Approach		Rule-based Approach	
		correct adaptations[2]	RAC	correct adaptations	RAC	correct adaptations	RAC	correct adaptations	RAC
EAST-ADL	234	37	15.81%	1	0.43%	139	59.4%	234	100%
Xenia	15	15	100%	15	100%	15	100%	43	100%
SML	96	96	100%	96	100%	96	100%	94	97.91%
BibTeX	43	43	100%	43	100%	43	100%	43	100%
DOT	19	19	100%	19	100%	19	100%	16	84.21%
Xcore	32	32	100%	32	100%	32	100%	20	62.50%

[1] The metric "required adaptations" refers to the count of grammar rules that need to be adapted from the generated grammar to the target grammar.
[2] The metric "correct adaptations" refers to the count of grammar rules that the approach correctly adapted.

Table 2 Grammar Rule-level Similarity Comparison between LLM-based/Rule-based Adapted Grammars and Target Grammars on Test Set DSLs

DSL	Claude-Sonnet-4.5-based Approach			ChatGPT-5.1-based Approach			Gemini-3-based Approach			Rule-based Approach		
	Same[1]	Diff	Percent[2]	Same	Diff	Percent	Same	Diff	Percent	Same	Diff	Percent
EAST-ADL	100	197	33.67%	1	296	0.34%	202	95	68.01%	297	0	100%
Xenia	15	0	100%	15	0	100%	15	0	100%	15	0	100%
SML	75	0	100%	75	0	100%	75	0	100%	51	24	68.0%
BibTeX	43	0	100%	43	0	100%	43	0	100%	43	0	100%
DOT	24	0	100%	24	0	100%	24	0	100%	21	3	87.50%
Xcore	40	0	100%	40	0	100%	40	0	100%	28	12	70.00%

[1] The metric "Same" refers to the count of grammar rules in the adapted grammar that are identical to those in the target grammar. The metric "Diff" is the same principle.
[2] The metric "Percent" refers to the proportion of the count of identical grammar rules in the adapted grammar and the target grammar to the total count of grammar rules in the target grammar.

grammar to the target grammar, while in Xcore, 32 grammar rules require adaptation. From the RAC metric perspective, all three LLM-based methods achieved 100% adaptation correctness on both test set DSLs. In contrast, the rule-based method achieved an RAC value of 84.21% on DOT (16 out of 19 rules correctly adapted) and only 62.50% on Xcore (20 out of 32 rules correctly adapted). These results indicate that on the test set DSLs, the LLM-based methods demonstrate superior performance in identifying and applying adaptations learned from the training set compared to the rule-based method. On the training set, LLM-based approaches achieved high RAC (100%) on BibTeX, Xenia, and SML, while the rule-based approach achieved 100% on EAST-ADL, Xenia, and BibTeX but 97.91% on SML. EAST-ADL posed challenges for LLM-based approaches (see Section 4.4.2 for in-depth analysis).

Table 2 further presents the similarity comparison results between the adapted grammars and the target grammars. The results show that all three LLM-based methods achieved 100% similarity on both test set DSLs, meaning that the grammars adapted by the LLM-based methods are identical to the target grammars in all grammar rules. This result perfectly echoes the RAC values in Table 1. In contrast, the rule-based method achieved 87.50% similarity on DOT, with 21 identical rules and 3 differing rules compared to the target grammar, and only 70.00% on Xcore (28 identical rules and 12 differing rules). These differences primarily stem from the rule-based method's inability to handle certain complex adaptations, such as the predicated assignment structures in DOT and the order-insensitive boolean attribute combinations in Xcore (see Section 4.4 for details). On the training set, LLM-based approaches achieved 100% similarity on BibTeX, Xenia, and SML, while still, faced challenges in EAST-ADL. The rule-based approach achieved 100% on EAST-ADL, Xenia, and BibTeX but only 68.0% on SML.

Table 3 presents the metamodel conformance validation results of the adapted grammars. On the test set, all three LLM-based methods passed metamodel conformance validation on both DOT and Xcore, while the rule-based method failed validation on Xcore. On the training set, LLM-based methods passed validation on BibTeX, Xenia, and SML, but encountered challenges on EAST-ADL; the rule-based method failed validation on SML, though it succeeded on EAST-ADL and BibTeX. These results indicate that on the test set DSLs, the adapted grammars generated by the LLM-based methods are not only consistent with the target grammars at the textual level, but also satisfy all constraints of the metamodel at the semantic level, demonstrating practical usability. However, the failure of LLM-based methods on EAST-ADL suggests potential limitations when handling large-scale grammars (see Section 4.4.2 for in-depth analysis). To verify the stability of the results, we re-

Table 3 Metamodel Conformance Validation of Adapted Grammars on Test Set DSLs

DSL	Claude-based	ChatGPT-based	Gemini-based	Rule-based
EAST-ADL	NO	NO	NO	YES
Xenia	YES	YES	YES	YES
SML	YES	YES	YES	NO
BibTeX	YES	YES	YES	YES
DOT	YES	YES	YES	YES
Xcore	YES	YES	YES	NO

peated the experiments three times for each LLM-based method on the test set DSLs, and all runs yielded consistent results.

Answer to RQ1: On the test set of small-to-medium scale DSLs, all three LLMs achieved 100% RAC and output similarity, demonstrating that LLMs can learn and apply grammar adaptations from historical versions; however, on large grammars (EAST-ADL, 297 rules), adaptation consistency was far below 90%, revealing the limitations of current LLMs in scaling.

4.3. Language Evolution Support: A Longitudinal Study on QVTo (RQ2)

To address **RQ2**, Table 4 presents the comparative results of LLM-based and rule-based approaches across QVTo's evolution sequence. This study was conducted using Claude Sonnet 4.5 (the reference model for prompt development), as its focus is cross-version reusability rather than cross-model robustness—the latter being evaluated in Section 4.2. Both approaches achieved 100% RAC across all three evolution steps, demonstrating their capability to successfully maintain grammar adaptations during language evolution driven by metamodel changes. The key distinction lies in required human effort: the LLM-based approach required no manual grammar editing across all evolution steps, whereas the rule-based approach necessitated configuration adjustments in two out of three transitions (two changes for V1.0→V1.1 and one change for V1.2→V1.3). This suggests that for DSLs of moderate scale like QVTo, LLM-based adaptation can potentially achieve high reusability across evolution steps once a suitable prompting strategy is established. All adapted grammars passed metamodel conformance validation across all three evolution steps.

Answer to RQ2: Across three evolution steps in QVTo, the LLM-based approach successfully reused learned adaptations across all consecutive evolution steps without manual grammar editing, demonstrating its capability to reuse adaptation strategies across multiple evolution steps.

4.4. Comparative Analysis: Strengths and Limitations (RQ3)

To address **RQ3**, we perform a comparative analysis to identify in which adaptation scenarios LLM-based approaches outperform rule-based methods, and in which scenarios they encounter challenges.

4.4.1. Advantages of the LLM-based Approach: Adapting Complex Grammar Scenarios

As shown in Sections 4.2-4.3, the LLM-based approach achieved 100% RAC and output similarity on both test set DSLs while passing metamodel conformance validation, outperforming the rule-based approach (84.21% on DOT, 62.50% on Xcore, conformance failure on Xcore). In QVTo, it achieved fully automated adaptation across all three evolution steps, while the rule-based approach required manual adjustments in two.

In-depth analysis of the failure cases of the rule-based approach on the test set reveals its fundamental limitation: it relies on predefined rules for matching based on surface features and cannot understand the contextual intent and semantic relationships of grammar structures. In DOT, the rule-based approach fails in three types of scenarios: when encountering the syntactic predicate `=>`, it cannot recognize its role in backtracking parsing — in the Port rule shown in Listing 3, `=>` instructs the parser to perform backtracking among multiple choice branches, but the rule-based approach cannot understand how to adapt this special construct; when facing multiple optionality (such as the nested inner and outer optionality in `('subgraph' (name=ID)?)?`), it cannot coordinate the hierarchical relationships; when handling multi-symbol "or" combinations, it cannot determine the overall optionality.

```
Port returns Port:
    {Port}
    ':'
    (=> compass_pt=COMPASS_PT |
        name=ID |
        name=ID ":" compass_pt=COMPASS_PT);
```

Listing 3 Port rule in DOT grammar containing syntactic predicate.

In Xcore, the rule-based approach similarly fails in four types of situations: it cannot recognize the structural intent of mutually exclusive alternatives, cannot handle attribute movement across same-named keywords, and although it can identify individual operations in a combination, it cannot determine the correct execution order—as shown in Listing 4, the bounds attribute requires multiple operations such as removing curly braces, renaming keywords, and changing delimiters; while the rule-based approach can identify these independent operations, it cannot determine their correct execution order, leading to adaptation failure. Additionally, when cross-reference types are missing, the rule-based approach is also disrupted and unable to adapt—in Listing 5, the type attribute in the generated grammar is missing one cross-reference type, which prevents the rule-based approach's adaptation.

```
// Generated grammar
XTypeParameter returns XTypeParameter:
  {XTypeParameter}
  'XTypeParameter'
  name=EString
  '{'
    ('annotations' '{' annotations+=XAnnotation (
    "," annotations+=XAnnotation)* '}' )?
```

Table 4 Evolution Step Analysis on QVTo (LLM-based: Claude Sonnet 4.5)

QVTo Evolution Context				Claude-based Approach		Rule-based Approach	
Versions	Grammar Rules	Evolution Step	Metamodel Changes	RAC	Follow-ups	RAC	#cORA
V1.0	1026	-	-	-	-	-	-
V1.1	992	V1.0 → V1.1	29 differences	100%	0	100%	2
V1.2	992	V1.1 → V1.2	0 differences (QVTo part)[a]	100%	0	100%	0
V1.3	991	V1.2 → V1.3	1 difference	100%	0	100%	1

[a] Version 1.2 introduced three metamodel changes, but all were in the excluded OCL part.

```
    ('bounds' '{' bounds+=XGenericType ( "," bounds
    +=XGenericType)* '}' )?
  '}';

// Target grammar
XTypeParameter returns XTypeParameter:
    {XTypeParameter}
          (annotations+=XAnnotation)*
    name=ID
        ('extends'  bounds+=XGenericType ( "&"
    bounds+=XGenericType)*  )?
    ;
```

Listing 4 Example of combination operation ordering failure in Xcore.

```
// Generated grammar - cross-reference type missing
XGenericType returns XGenericType:
    {XGenericType}
    'XGenericType'
    '{'
        ('type' type=[|EString])?
        ...
    '}';

// Target grammar - complete cross-reference type
XGenericType returns XGenericType:
    {XGenericType}
    type=[genmodel::GenBase|XQualifiedName]
    ...;
```

Listing 5 Example of adaptation failure due to missing cross-reference type in Xcore.

These failures reveal a common problem: correct adaptation requires not only identifying "what grammar elements exist" but also understanding "what intent these elements express in this context". For example, the symmetric choice structure in Xcore shown in Listing 6 is not a regular mutually exclusive choice, but rather expresses order insensitivity; multiple optionality in DOT is not simple stacking, but rather hierarchical structural relationships. The 60 predefined rules of the rule-based approach are based on surface features of grammar elements and cannot capture these deep design intents.

```
    ...
    (unordered?='unordered' unique?='unique'?|
    unique?='unique' unordered?='unordered'?)?
    ...
```

Listing 6 Example of order-insensitive boolean attribute combination.

In contrast, the LLM-based approach learns context-relevant handling strategies from examples by analyzing the complete adaptation process from G_1 to G_1'. LLMs do not need to explicitly understand the semantic definitions of syntactic predicates or multiple optionality, but rather observe "when a certain structure appears in historical adaptations, how it was handled", thereby implicitly learning strategies such as preserving special constructs, coordinating hierarchical relationships, and maintaining structural intent. This ability to learn from complete adaptation examples not only enables LLMs to handle complex scenarios that the predefined rules of the rule-based approach cannot cover, but also demonstrates better generalization capability—as the QVTo longitudinal study shows, once an LLM learns adaptations from one evolution step, it can automatically reuse these adaptations in subsequent evolutions without manual grammar editing.

Furthermore, the LLM-based approach is more reliable in ensuring metamodel conformance. Although the rule-based approach achieved 70% textual similarity on Xcore, the generated grammar does not conform to the metamodel. This is because transformations based on surface patterns may break the mapping relationship between grammar and metamodel. In contrast, by learning the complete adaptation process $G_1 \rightarrow G_1'$ that conforms to the metamodel, LLMs implicitly learn the constraints for maintaining conformance, resulting in grammars that are both textually correct and semantically compliant with the metamodel.

4.4.2. Limitations of the LLM-based Approach: Challenges with Large-Scale Grammars

Although the LLM-based approach performed well on the test set DSLs, it encountered difficulties on the EAST-ADL language in the training set. The RAC values of the three LLM-based approaches on EAST-ADL were all far below 90%, and acceptable adaptation quality could not be achieved even after multiple iterations of prompt optimization (as we mentioned in Section 4.1, the results presented in Tables 1-3 for EAST-ADL represent the fifth iteration of our prompt refinement). EAST-ADL is the largest among all evaluated DSLs, containing 291 metaclasses, 297 grammar rules, and approximately 3000 lines of grammar text. We found that grammar scale may be a key factor affecting the performance of the LLM-based approach: on large-scale grammars like EAST-ADL, LLMs omit a large number of identified adaptation operations during the application phase.

Table 5 Adaptation-type-level analysis on EAST-ADL: correctness of each adaptation type across LLMs.

		Claude Sonnet 4.5		Gemini 3		ChatGPT 5.1	
Adaptation Type[a]	**Occ.**[b]	**Cor.**[c]	**Inc.**[d]	**Cor.**	**Inc.**	**Cor.**	**Inc.**
Brace/optionality removal	194	78	116	193	1	2	192
Keyword removal	194	190	4	191	3	3	191
shortName promotion	188	188	0	134	54	2	186
Separator modification	225	47	178	180	45	5	220
Type system adaptation	202	151	51	201	1	5	197

[a] Adaptation types are derived from the grammar transformation rules identified in (Zhang et al. 2024).
[b] Occ.: number of grammar rules requiring this adaptation type.
[c] Cor.: number of grammar rules where the adaptation was correctly applied.
[d] Inc.: number of grammar rules where the adaptation was not correctly applied.

In the third iteration, Claude successfully identified approximately 1500 semantic adaptations, including moving the shortName attribute from inside curly braces to outside, adding semicolons after attributes with multiplicity "0-1", removing comma separators from collection attributes, and other key adaptations. This suggests that, for Claude, the primary bottleneck was not in the identification stage—where the LLM analyzes G_1 and G_1' to recognize adaptation operations—but rather in the application stage, where identified adaptations must be consistently executed across all grammar rules. Indeed, when Claude was asked to apply these identified adaptations to the evolved grammar G_2, the generated adapted grammar systematically omitted a large number of identified adaptations, especially the addition of semicolons. The three grammar files of EAST-ADL (G_1, G_1', G_2) produced a total of approximately 12,000 tokens, which is far below Claude Sonnet 4.5's context window limit of 200,000 tokens, but when adaptations need to be applied to G_2 containing hundreds of grammar rules, LLMs seem to have difficulty maintaining consistency in detail conventions across the entire grammar scope.

It is worth noting that this limitation is unrelated to grammar complexity—EAST-ADL's adaptations are relatively systematic (e.g., applying the same shortName promotion and curly brace optionalization to all entity rules), and should theoretically be easier to learn than the complex adaptations involving special constructs in DOT and Xcore. The real challenge lies in scale: the six evaluated DSLs exhibit an inverse relationship between grammar scale and adaptation complexity: DOT and Xcore are small-to-medium in scale but involve complex, context-dependent adaptations (syntactic predicates, predicated assignments, order-insensitive attribute combinations), whereas EAST-ADL is large in scale but requires repetitive, uniform adaptations across rules (Table 5). This distinction helps explain why LLMs succeeded on the former but struggled with the latter—the challenge is scale-dependent rather than complexity-dependent.

Gemini 3's failure on EAST-ADL exhibited different problem characteristics: in addition to the same systematic omission of adaptations, Gemini also performed erroneous destructive operations, including deleting the shortName attribute that should not be deleted, modifying attribute names without reason causing attributes to be unparseable, and adding semicolons where semicolons should not be added. These errors resulted in the generated grammar not only having an RAC value below 90%, but also failing metamodel conformance validation. Multiple targeted follow-up prompts (such as explicitly requesting "do not delete attributes that should not be deleted") also failed to resolve these issues.

ChatGPT 5.1 encountered additional challenges on EAST-ADL: its input was truncated to approximately 1,200 of 3,000 lines, preventing it from receiving the complete grammar. This means that in the first step of identifying required adaptations, ChatGPT had already missed most of the grammar content, making accurate adaptation impossible.

To quantify these observations, we grouped the adaptation operations required to transform the generated grammar into the target grammar of EAST-ADL into five adaptation types based on their operation characteristics, and analyzed correctness by adaptation type (Table 5). Results show that Claude's difficulties lie in consistently applying operations: keyword removal achieved high correctness (190 out of 194 occurrences), but separator modification succeeded in only 47 out of 225 cases. Gemini achieved high correctness on most types but struggled with shortName promotion (134 out of 188) and separator modification (180 out of 225), with some failures involving deletion of attributes that should have been preserved, causing metamodel conformance failures. ChatGPT's performance was limited by input truncation as discussed earlier. These results confirm that the bottleneck lies in the application stage rather than in identification: Claude and Gemini were able to describe the required adaptation types in their intermediate outputs, yet failed to execute them completely across all grammar rules; ChatGPT, by contrast, could not complete the identification stage itself due to input truncation. Despite refinement attempts through follow-up instructions, the systematic omission persisted, suggesting that this limitation reflects the capability boundaries of the LLMs evaluated in this study, rather than a prompt engineering issue that could be resolved through further refinement.

The rule-based approach achieved 100% adaptation correctness (RAC) on EAST-ADL. This is because once adaptation rules are determined, the rule-based approach can systematically apply the rules to all grammar rules in a programmatic manner, unaffected by grammar scale. This contrast indicates that when handling large-scale grammar adaptation tasks with high repetition of identical operations, the reliability and scalability of the rule-based approach are superior to the current LLM-based approach.

Answer to RQ3: LLM-based approaches demonstrate advantages in handling complex grammar scenarios that are difficult for rule-based methods to address, such as syntactic predicates, predicated assignments, and order-insensitive attribute combinations, successfully learning and applying adaptations in these cases. However, LLM-based approaches encountered challenges on large-scale grammars, whereas rule-based approaches demonstrate better reliability and scalability in adapting large languages.

5. Discussion

5.1. Practical Guidelines

Based on our evaluation, we provide the following guidelines for selecting adaptation approaches:

For grammars with fewer than 100 rules, LLM-based approaches are recommended when adaptations involve syntactic predicates, predicated assignments, or order-insensitive attribute combinations—constructs that require extensive manual configuration in rule-based methods. For grammars exceeding 200 rules, rule-based approaches are more reliable, as our EAST-ADL evaluation (297 rules, Table 5) demonstrates LLMs' systematic omission issues at this scale.

The choice between approaches should consider both grammar scale and adaptation characteristics. Systematic and repetitive adaptations favor rule-based methods for their deterministic execution, while context-dependent adaptations that vary across rules benefit from LLM-based approaches' reduced configuration effort. For grammars between 100-200 rules, practitioners should evaluate whether adaptation complexity outweighs the risks of systematic omission, potentially testing both approaches on representative subsets.

5.2. Threats to Validity

Internal Validity: Several factors in our study may affect the validity of causal inferences. First, we used only Claude Sonnet 4.5 for prompt strategy development, then applied the finalized prompts to ChatGPT 5.1 and Gemini 3 for cross-model validation; while this approach evaluates the cross-model consistency of prompts, using a different reference model for prompt development might yield different prompt strategies and performance characteristics. Second, the decision criteria for prompt finalization (>90% RAC threshold) were chosen empirically; under different training sets or DSL characteristics, different thresholds might lead to different prompt strategies, thereby affecting performance on the test set. Third, the finalized prompts met all quality thresholds on the small-to-medium scale training DSLs without iterative refinement.

Construct Validity: Our evaluation metrics—RAC, Output Similarity, and Metamodel Conformance—while providing objective and quantifiable measurements, may not fully capture the multidimensional nature of "high-quality grammar adaptation". RAC measures the consistency of adaptations, i.e., whether adaptations are correctly reused, but does not evaluate the reasonableness or desirability of the adaptations themselves. For example, if the target grammar (G_1') itself contains design flaws or does not conform to best practices, the LLM's perfect replication of these adaptations would still achieve 100% RAC, yet the quality of the resulting grammar may not be ideal. Furthermore, our metrics also focus on metamodel conformance and grammar rule similarity, but they do not evaluate other quality properties that grammar engineers might care about, e.g., grammar readability, maintainability, consistent naming conventions, or adherence to team coding style guidelines (Dubey 2006). Therefore, while our metrics can effectively validate whether LLMs learn and apply historical adaptations, they may not be sufficient to comprehensively assess the overall quality of adapted grammars in practical software engineering practice.

External Validity: Although we evaluated our approach on six DSLs from different domains, the sample size remains limited, making it difficult to draw universally applicable conclusions. Furthermore, in the evaluation of these six DSLs, the G_1 and G_2 we used are actually the same generated grammar. However, in actual language evolution scenarios, metamodel changes may lead to differences between the generated grammar G_2 and G_1, e.g., adding or deleting grammar rules. Our longitudinal evaluation on QVTo mitigates this threat to some extent, as QVTo's four official versions did undergo real metamodel evolution, with actual differences in the generated grammars between versions. However, the changes between QVTo versions are relatively small as shown in Table 4, which may not capture the full complexity of real evolution scenarios. Second, our approach is specifically designed for Xtext-based DSLs, where grammars can be generated from metamodels and the generated grammars follow Xtext's conventions and patterns. The applicability of this approach to DSLs developed with other frameworks (such as JetBrains MPS (JetBrains 2025)) remains unclear. Third, we intentionally focused on a setup decoupled from specific DSL examples rather than few-shot prompting, as example-based prompting can steer outputs toward the provided examples and limit generalizability; comparison against alternative strategies such as chain-of-thought prompting remains a direction for future work. Furthermore, our approach encountered significant challenges on the large-scale grammar (i.e., the grammar of EAST-ADL), forcing us to exclude EAST-ADL from the prompt finalization process. This indicates that grammar scale may be a critical factor affecting the effectiveness of LLM-based approaches, and the small-to-medium scale DSLs on which our final evaluation is based may not represent the actual challenges of large-scale industrial DSLs. Future research needs to explore strategies specifically designed for large-scale grammars, such as segmented processing, incremental adaptation, etc., to overcome the systematic omission problems of current LLMs when handling hundreds of grammar rules.

5.3. The Role of LLMs in Model-Driven Engineering

This study explores the potential of LLMs in automating grammar adaptation tasks, contributing empirical evidence to the intersection of AI and MBE. This intersection manifests in two complementary directions: using LLMs to support MDE tasks (LLM4MDE), and using MDE techniques to facilitate the adoption of LLMs (MDE4LLM) (Di Rocco et al. 2025).

LLMs as Components of the MBE Toolchain. LLMs as Components of the MBE Toolchain. Unlike traditional rule-based MDE tools, LLMs introduce probabilistic and non-deterministic characteristics (Astekin et al. 2024; Sawadogo et al. 2025; Ouyang et al. 2025), posing challenges for MDE toolchains that traditionally rely on deterministic transformations to ensure model consistency and traceability (Lucas et al. 2009; Bucchiarone et al. 2020). However, the "black box" nature of LLMs means we cannot fully predict or explain how their generated outputs are produced. In our study, this characteristic is partially

mitigated through mandatory metamodel conformance validation—any grammar generated by LLMs must pass validation through the Eclipse Modeling Framework, ensuring its semantic correctness. This pattern of combining probabilistic AI methods with formal verification may represent a viable architecture for future AI-enhanced MDE tools.

Specificity of Prompt Engineering in MDE Contexts. Our prompt development process reveals the unique requirements of prompt engineering in the MDE domain. Unlike general LLM applications, MDE tasks have clear correctness criteria—generated grammars must conform to metamodel specifications. This enables us to adopt more stringent quality thresholds than traditional prompt engineering: prompts are finalized only when LLMs achieve (RAC>90%), and output (similarity>90%), and pass metamodel conformance validation on the training set. Notably, these metrics can be computed directly against pre-existing target grammars without manual annotation, and metamodel conformance validation provides a formal correctness guarantee, ensuring the semantic validity of generated grammars. Furthermore, our training-test separation methodology—using four DSLs for prompt development and two DSLs for testing—reflects scientific rigor aimed at avoiding prompt overfitting to specific cases.

From Tool Development to Methodological Exploration. The value of this study lies not only in demonstrating that LLMs can perform grammar adaptation tasks, but more importantly in systematically characterizing the capability boundaries of this approach. Through evaluation on DSLs of different scales and complexities, we identified the applicable domain of LLM methods (small-to-medium scale, complex adaptations requiring context-dependent understanding) and the limitation domain (large-scale, requiring systematic application of hundreds of operations). It also suggests a concrete hybrid architecture: since the identification stage was not the primary bottleneck for Claude and Gemini on EAST-ADL while the application stage exhibited systematic omissions, one direction would be to use LLMs exclusively for identifying adaptation operations from G_1 and G_1', and then delegate execution to a deterministic rule-based engine such as GrammarTransformer. However, realizing this architecture faces a challenge: LLMs produce informal natural language adaptation descriptions (e.g., “move shortName outside curly braces”) rather than formal, executable rule specifications, requiring either constrained output formats or a translation layer to the 60 predefined transformation rules. Moreover, some adaptations that LLMs successfully handle (such as syntactic predicates and order-insensitive attribute combinations) fall outside the coverage of current predefined rules, meaning that the rule-based execution engine would also need to be extended. In this sense, our study provides systematic empirical analysis of “the role of AI methods in grammar adaptation as a specific MDE task,” complementing existing literature’s exploration of LLMs in other MDE tasks (such as model generation and model completion (Arulmohan et al. 2023; K. Chen et al. 2023)).

Enhancing Metamodel-Driven Development Viability. Prior large-scale study (Zhang & Struber 2024; Zhang, Strüber, & Hebig 2026) revealed that Xtext-based DSL development predominantly follows the grammar-driven scenario, with metamodel-driven approaches being less common. A reason for this preference is the manual workload: In scenario (b), after metamodel evolution, developers regenerate G_2 and reapply all previous adaptations, whereas scenario (a) requires only localized adjustments to G_1' based on metamodel changes. our study demonstrates that for small-to-medium scale DSLs, LLM-based approaches can further automate the learning and reapplication of historical adaptations, reducing this workload and making scenario (b) more attractive relative to scenario (a).

5.4. LLM-based Approach for Grammar Style Customization

Beyond supporting metamodel-grammar co-evolution, our LLM-based approach opens new possibilities for automated grammar style customization. Previous work demonstrated a semi-automated approach for transforming Xtext-generated grammars into specific styles, such as Python-style DSLs (Zhang, Hebig, Steghöfer, & Holtmann 2023), requiring manual keyword refinement decisions and predefined transformation scripts for operations like brace removal and whitespace-awareness introduction.

Many of the adaptation operations that LLMs successfully handled in this study—such as removing braces, refining keywords, and adjusting attribute positions—are similar in nature to those in prior style customization work.

6. Related Work

6.1. Metamodel-Grammar Co-evolution in MDE

The evolution of DSLs is a challenge in MDE. As detailed by Meyers et al.(Meyers & Vangheluwe 2011), the "coupled evolution" problem arises when changes in a metamodel (abstract syntax) necessitate the propagation of updates to related artifacts, such as models, transformations, and grammars (concrete syntax). Traditionally, this process has been handled through manual adaptation or rule-based automation. For instance, in the context of Xtext,changes to the Ecore metamodel often break the mapping to the Xtext grammar.

Zhang et al.(Zhang et al. 2024) addressed this by proposing GrammarTransformer, an approach that utilizes predefined transformation rules to semi-automate the synchronization between evolved metamodels and grammars.

Similarly, Kusel et al.(Kusel et al. 2015) provided a systematic classification of co-evolution approaches, highlighting that while operator-based and inference-based methods exist, they often require significant manual configuration or rigorous formal definitions that are difficult to maintain in rapid development cycles. Our work differs from these traditional approaches by replacing rigid transformation rules with the probabilistic reasoning capabilities of LLMs, aiming to learn adaptations implicitly from history rather than requiring explicit rule definition.

6.2. Large Language Models in Software Engineering

The emergence of Large Language Models (LLMs) has significantly impacted numerous domains within Software Engineering (SE). Many recent studies have explored applying LLMs to various SE tasks (Hou et al. 2024). LLMs have demonstrated state-of-the-art performance in code generation, completion, and translation (M. Chen et al. 2021).

In the specific domain of MDE, the integration of LLMs is gaining traction (Di Rocco et al. 2025). Chaaben et al. (Chaaben et al. 2023) pioneered the use of few-shot prompting for model completion, demonstrating that GPT-3 can complete UML models when provided with examples from the ModelSet dataset. Studies have explored using LLMs to generate OCL constraints (Pan et al. 2024), reverse engineer models from code (Pearce et al. 2022), and assist in creating DSL instances (Netz et al. 2024). For example, LLMs can effectively parse natural language requirements into formal model specifications, reducing the barrier to entry for DSL users. However, these applications generally focus on generation (creating new artifacts) rather than evolution (maintaining consistency between existing, changing artifacts), which requires a deeper understanding of the structural dependencies inherent in DSLs.

6.3. LLM-based Artifact Co-evolution

Most relevant to our study is the emerging body of work applying LLMs specifically to the co-evolution and migration of software artifacts. Jiang et al.(Jiang et al. 2025) introduced Codeditor, an LLM-based tool designed to migrate code changes across programming languages (e.g., Java to C#), demonstrating that LLMs can learn edit patterns more effectively than traditional transpilers. Closer to the MDE context, Kebaili et al. (Kebaili et al. 2024) conducted an empirical study using ChatGPT to co-evolve source code in response to metamodel changes. Their results showed that while LLMs can successfully repair most broken code (achieving up to 88.7% correctness in specific scenarios), specifically target generated code repair. In contrast, our work targets the adaptation of the grammar definition itself—a more upstream challenge in the language infrastructure. Furthermore, recent investigations by Zhang et al. (Zhang et al. 2025; Zhang, Jiang, Fu, Koziolek, et al. 2026) have begun to explore the co-evolution of textual DSL instances using LLMs, evaluating how models can migrate user scripts when the underlying grammar changes. Our work complements and extends these efforts by shifting the focus upstream: instead of fixing instances or generated code. By evaluating LLMs' ability to adapt Xtext grammars directly from metamodel changes—demonstrated across real-world DSLs like QVTo.

7. Conclusion

This paper explores the potential of large language models in automating metamodel-grammar co-evolution. We proposed an LLM-based approach that learns grammar adaptations from historical versions and applies them to new grammars after metamodel evolution. Evaluation on six real-world DSLs demonstrates that LLMs outperform the rule-based approach on complex grammar scenarios and successfully reuse adaptations across consecutive evolution steps without manual grammar editing; however, systematic omission of adaptation operations on large-scale grammars reveals the current limitations of LLMs at scale.

Future work includes three directions: improving LLM reliability on large-scale grammars, extending to co-evolution of other language artifacts (such as model transformation rules and OCL constraints with metamodels), and exploring the potential of LLMs in supporting co-evolution of modeling artifacts at different abstraction levels.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1608 - 501798263 and supported by the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF). It was partially funded by Vetenskapsrådet, grant 2021-04881_VR.

References

Anthropic. (2025). *Claude Sonnet 4.5.* Retrieved from https://www.anthropic.com/claude/sonnet (Accessed December, 2025)

Arulmohan, S., Meurs, M.-J., & Mosser, S. (2023). Extracting domain models from textual requirements in the era of large language models. In *2023 acm/ieee international conference on model driven engineering languages and systems companion (models-c)* (pp. 580–587).

Astekin, M., Hort, M., & Moonen, L. (2024). An exploratory study on how non-determinism in large language models affects log parsing. In *Proceedings of the acm/ieee 2nd international workshop on interpretability, robustness, and benchmarking in neural software engineering* (pp. 13–18).

Bettini, L. (2016). *Implementing domain-specific languages with xtext and xtend.* Packt Publishing Ltd.

Bucchiarone, A., Cabot, J., Paige, R. F., & Pierantonio, A. (2020). Grand challenges in model-driven engineering: an analysis of the state of the research. *Software and Systems Modeling*, *19*(1), 5–13.

Chaaben, M. B., Burgueño, L., & Sahraoui, H. (2023). Towards using few-shot prompt learning for automating model completion. In *2023 ieee/acm 45th international conference on software engineering: New ideas and emerging results (icse-nier)* (pp. 7–12).

Chen, K., Yang, Y., Chen, B., López, J. A. H., Mussbacher, G., & Varró, D. (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)* (pp. 162–172).

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., … Zaremba, W. (2021). *Evaluating large language models trained on code.* Retrieved from https://arxiv.org/abs/2107.03374

Cicchetti, A., Di Ruscio, D., Eramo, R., & Pierantonio, A. (2008a). Automating co-evolution in model-driven engineering. In *2008 12th international ieee enterprise distributed object computing conference* (pp. 222–231).

Cicchetti, A., Di Ruscio, D., Eramo, R., & Pierantonio, A. (2008b). Meta-model differences for supporting model co-evolution. In *Proceedings of the 2nd workshop on model-driven software evolution-modse* (Vol. 1).

Ciccozzi, F., Tichy, M., Vangheluwe, H., & Weyns, D. (2019). Blended modelling-what, why and how. In *2019 acm/ieee 22nd international conference on model driven engineering languages and systems companion (models-c)* (pp. 425–430).

Cummins, C., Seeker, V., Armengol-Estapé, J., Markosyan, A. H., Synnaeve, G., & Leather, H. (2024). Don't transform the code, code the transforms: Towards precise code rewriting using llms. *arXiv preprint arXiv:2410.08806*.

Di Rocco, J., Di Ruscio, D., Di Sipio, C., Nguyen, P. T., & Rubei, R. (2025). On the use of large language models in model-driven engineering: J. di rocco et al. *Software and Systems Modeling*, *24*(3), 923–948.

Dubey, A. (2006). Goodness criteria for programming language grammar rules. *ACM SIGPLAN Notices*, *41*(12), 44–53.

EAST-ADL Association. (2022). *EATOP Repository.* Retrieved from https://bitbucket.org/east-adl/east-adl/src/Revison/ (Accessed February, 2023)

Eclipse Foundation. (2018). *Eclipse xcore wiki.* Retrieved from https://git.eclipse.org/c/emf/org.eclipse.emf.git/tree/plugins/org.eclipse.emf.ecore.xcore/src/org/eclipse/emf/ecore/xcore/Xcore.xtext (Accessed February, 2023)

Eclipse Foundation. (2025). *Xtext homepage.* (https://www.eclipse.org/Xtext/. Last accessed Nov 2022)

Eclipse Foundation AISBL. (2020). *Dot xtext grammar.* Retrieved from https://github.com/eclipse/gef/blob/master/org.eclipse.gef.dot/src/org/eclipse/gef/dot/internal/language/Dot.xtext (Accessed February, 2023)

Erdweg, S., Van Der Storm, T., Völter, M., Boersma, M., Bosman, R., Cook, W. R., … others (2013). The state of the art in language workbenches: Conclusions from the language workbench challenge. In *International conference on software language engineering* (pp. 197–217).

García, J., Diaz, O., & Azanza, M. (2012). Model transformation co-evolution: A semi-automatic approach. In *International conference on software language engineering* (pp. 144–163).

Greenyer, J. (2018). *Scenario Modeling Language (SML) Repository.* Retrieved from https://bitbucket.org/jgreenyer/scenariotools-sml/src/master/ (Accessed February, 2023)

Hebig, R., Khelladi, D. E., & Bendraou, R. (2016). Approaches to co-evolution of metamodels and models: A survey. *IEEE Transactions on Software Engineering*, *43*(5), 396–414.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., … Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, *33*(8), 1–79.

JetBrains. (2025). *MPS Meta Programming System.* Retrieved from https://www.jetbrains.com/mps/ (Accessed December, 2025)

Jiang, J., Li, Z., Qin, H., Jiang, M., Luo, X., Wu, X., … Chen, T. (2025, April). Unearthing gas-wasting code smells in smart contracts with large language models. *IEEE Trans. Softw. Eng.*, *51*(4), 879–903. Retrieved from https://doi.org/10.1109/TSE.2024.3491578 doi: 10.1109/TSE.2024.3491578

Kebaili, Z. K., Khelladi, D. E., Acher, M., & Barais, O. (2024, July). An empirical study on leveraging llms for metamodels and code co-evolution. *Journal of Object Technology*, *23*(3), 1-14. Retrieved from http://www.jot.fm/contents/issue_2024_03/article6.html (The 20th European Conference on Modelling Foundations and Applications (ECMFA 2024)) doi: 10.5381/jot.2024.23.3.a6

Kessentini, W., Sahraoui, H., & Wimmer, M. (2018). Automated co-evolution of metamodels and transformation rules: A search-based approach. In *International symposium on search based software engineering* (pp. 229–245).

Kessentini, W., Sahraoui, H., & Wimmer, M. (2019). Automated metamodel/model co-evolution: A search-based approach. *Information and Software Technology*, *106*, 49–67.

Kusel, A., Etzlstorfer, J., Kapsammer, E., Retschitzegger, W., Schwinger, W., & Schönböck, J. (2015). Consistent co-evolution of models and transformations. In *Proceedings of the 18th international conference on model driven engineering languages and systems* (p. 116–125). IEEE Press.

Lucas, F. J., Molina, F., & Toval, A. (2009). A systematic review of uml model consistency management. *Information and Software technology*, *51*(12), 1631–1645.

Martinez, C. (2025). *Claude Sonnet 4.5: Features, Benchmarks & Pricing (2025).* Retrieved from https://www.leanware.co/insights/claude-sonnet-4-5-overview (Accessed December, 2025)

Meyers, B., & Vangheluwe, H. (2011). A framework for evolution of modelling languages. *Science of Computer Programming*, *76*(12), 1223-1246. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167642311000141 (Special Issue on Software Evolution, Adaptability and Variability) doi: https://doi.org/10.1016/j.scico.2011.01.002

Misha Rodchenkov. (2019). *Xenia xtext.* Retrieved from https://github.com/rodchenk/xenia/blob/master/com.foliage.xenia/src/com/foliage/xenia/Xenia.xtext (Accessed February, 2023)

Netz, L., Michael, J., & Rumpe, B. (2024). From natural language to web applications: Using large language models for model-driven software engineering. In *Modellierung 2024* (pp. 179–195).

OpenAI. (2025). *GPT-5.1: A smarter, more conversational ChatGPT.* Retrieved from https://openai.com/index/gpt-5-1/ (Accessed December, 2025)

Ouyang, S., Zhang, J. M., Harman, M., & Wang, M. (2025). An empirical study of the non-determinism of chatgpt in code generation. *ACM Transactions on Software Engineering and Methodology*, *34*(2), 1–28.

Paige, R. F., Kolovos, D. S., & Polack, F. A. (2014). A tutorial on metamodelling for grammar researchers. *Science of Computer Programming*, *96*, 396–416.

Paige, R. F., Matragkas, N., & Rose, L. M. (2016). Evolving models in model-driven engineering: State-of-the-art and future challenges. *Journal of Systems and Software*, *111*, 272–280.

Pan, F., Zolfaghari, V., Wen, L., Petrovic, N., Lin, J., & Knoll, A.

(2024). Generative ai for ocl constraint generation: Dataset collection and llm fine-tuning. In *2024 ieee international symposium on systems engineering (isse)* (pp. 1–8).
Paperpile. (2022). *A complete guide to the BibTeX format.* Retrieved from https://www.bibtex.com/g/bibtex-format/ (Accessed February, 2023)
Pearce, H., Tan, B., Krishnamurthy, P., Khorrami, F., Karri, R., & Dolan-Gavitt, B. (2022). Pop quiz! can a large language model help with reverse engineering? *arXiv preprint arXiv:2202.01142*.
Ráth, I., Ökrös, A., & Varró, D. (2010). Synchronization of abstract and concrete syntax in domain-specific modeling languages: By mapping models and live transformations. *Software & Systems Modeling*, *9*(4), 453–471.
Reynolds, L., & McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 chi conference on human factors in computing systems* (pp. 1–7).
Sawadogo, S., Sabane, A., Kafando, R., Kabore, A. K., & Bissyande, T. F. (2025). Revisiting the non-determinism of code generation by the gpt-3.5 large language model. In *2025 ieee international conference on software analysis, evolution and reengineering (saner)* (pp. 36–44).
Shah, C. (2024). From prompt engineering to prompt science with human in the loop. *arXiv preprint arXiv:2401.04122*.
Studio, G. A. (2025). *Gemini 3.* Retrieved from https://aistudio.google.com/models/gemini-3 (Accessed December, 2025)
The Object Management Group. (2016). *MOF Query/View/Transformation.* https://www.omg.org/spec/QVT, last accessed December 2025.
Tolvanen, J.-P., Kelly, S., Di Rocco, J., Pierantonio, A., & Tinella, G. (2025). A framework for evaluating tool support for co-evolution of modeling languages, tools and models. *Software and Systems Modeling*, *24*(2), 311–338.
Wang, Y., Le, H., Gotmare, A., Bui, N., Li, J., & Hoi, S. (2023). Codet5+: Open code large language models for code understanding and generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing* (pp. 1069–1088).
Zhang, W. (2025). *'leveraging llms for grammar adaptation: A study on metamodel-grammar co-evolution'.* Retrieved from https://osf.io/3evtx/overview?view_only=92d852dba36341738c0e2f42e9c5457c
Zhang, W., Hebig, R., Steghöfer, J.-P., & Holtmann, J. (2023). Creating python-style domain specific languages: A semi-automated approach and intermediate results. In *Modelsward* (pp. 210–217).
Zhang, W., Hebig, R., & Strüber, D. (2025). Leveraging llms to support co-evolution between definitions and instances of textual dsls. In *Proceedings of the first large language models for software engineering workshop (llm4se), co-located with staf 2025.* Koblenz, Germany.
Zhang, W., Hebig, R., Strüber, D., & Steghöfer, J.-P. (2023). Automated extraction of grammar optimization rule configurations for metamodel-grammar co-evolution. In *Proceedings of the 16th acm sigplan international conference on software language engineering* (pp. 84–96).
Zhang, W., Holtmann, J., Strüber, D., Hebig, R., & Steghöfer, J.-P. (2024). Supporting meta-model-based language evolution and rapid prototyping with automated grammar transformation. *Journal of Systems and Software*, *214*, 112069.
Zhang, W., Jiang, B., Fu, Y., Cheng, H., Hummel, M., Scotti, V., … Koziolek, A. (2026). *Large language models in model-driven engineering: A systematic mapping study.* Karlsruhe Institute of Technology (KIT). Retrieved from https://publikationen.bibliothek.kit.edu/1000193410 doi: 10.5445/IR/1000193410
Zhang, W., Jiang, B., Fu, Y., Koziolek, A., Hebig, R., & Strüber, D. (2026). Leveraging llms to support co-evolution between definitions and instances of textual dsls: A systematic evaluation. *arXiv preprint arXiv:2602.11904*.
Zhang, W., & Struber, D. (2024). Tales from 1002 repositories: Development and evolution of xtext-based dsls on github. In *2024 50th euromicro conference on software engineering and advanced applications (seaa)* (pp. 172–179).
Zhang, W., Strüber, D., & Hebig, R. (2026). Development and evolution of xtext-based dsls on github: an empirical investigation. *Empirical Software Engineering*, *31*(3), 48.
Zhuo, J., Zhang, S., Fang, X., Duan, H., Lin, D., & Chen, K. (2024). Prosa: Assessing and understanding the prompt sensitivity of llms. *arXiv preprint arXiv:2410.12405*.

About the authors

Weixing Zhang is a PostDoc at Karlsruhe Institute of Technology. He received his PhD at the University of Gothenburg, Sweden in 2025. His research interests include SE, Empirical SE, AI4SE. You can contact the author at weixing.zhang@kit.edu or visit https://wilson008.github.io/.

Bowen Jiang is a PhD researcher at Karlsruhe Institute of Technology, Germany. She received her M.Sc. from WASEDA University, Japan in 2024. Her research interests include MDE, software testing, and AI4SE. You can contact the author at bowen.jiang@kit.edu or visit https://mcse.kastel.kit.edu/staff_bowen_jiang.php.

Rahul Sharma is a PostDoc at Karlsruhe Institute of Technology. His research interests include LLMs, ML, Data Science. You can contact the author at rahul.sharma@kit.edu or visit https://dsis.kastel.kit.edu/staff_rahul_sharma.php.

Ragina Hebig is a Professor for Software Engineering at the University of Rostock, Germany. Her research interests include Empirical SE, SE4AI & AI4SE, MDE, Software Process. You can contact the author at regina.hebig@uni-rostock.de or visit https://se.informatik.uni-rostock.de/team/lehrstuhlinhaber/prof-dr-rer-nat-regina-hebig/.

Daniel Strüber is an Asso. Professor at Chalmers | University of Gothenburg, Sweden, and an Ass. Professor at Radboud University in Nijmegen, the Netherlands. His research interests include MDE, AI engineering, software product lines. You can contact the author at danstru@chalmers.se or visit https://www.danielstrueber.de/.