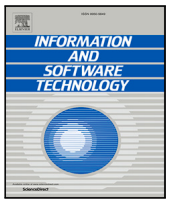
Information and Software Technology 186 (2025) 107806

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Leveraging synthetic trace generation of modeling operations for intelligent modeling assistants using large language models

Vittoriano Muttillo [a,*], Claudio Di Sipio [b], Riccardo Rubei [b], Luca Berardinelli [c]

[a] University of Teramo, Teramo, Italy
[b] University of L'Aquila, L'Aquila, Italy
[c] Johannes Kepler University, Linz, Austria

ARTICLE INFO

Dataset link: https://github.com/hepsycode/MASTER-LLM-IST

Keywords:
Model-driven engineering
Modeling operations
Intelligent modeling assistant
Large language model
Synthetic datasets

ABSTRACT

Context: Due to the proliferation of generative AI models in different software engineering tasks, the research community has started to exploit those models, spanning from requirement specification to code development. Model-Driven Engineering (MDE) is a paradigm that leverages software models as primary artifacts to automate tasks. In this respect, modelers have started to investigate the interplay between traditional MDE practices and Large Language Models (LLMs) to push automation. Although powerful, LLMs exhibit limitations that undermine the quality of generated modeling artifacts, e.g., hallucination or incorrect formatting. Recording modeling operations relies on human-based activities to train modeling assistants, helping modelers in their daily tasks. Nevertheless, those techniques require a huge amount of training data that cannot be available due to several factors, e.g., security or privacy issues.
Objective: In this paper, we propose an extension of a conceptual MDE framework, called MASTER-LLM, that combines different MDE tools and paradigms to support industrial and academic practitioners.
Method: MASTER-LLM comprises a modeling environment that acts as the active context in which a dedicated component records modeling operations. Then, model completion is enabled by the modeling assistant trained on past operations. Different LLMs are used to generate a new dataset of modeling events to speed up recording and data collection.
Results: To evaluate the feasibility of MASTER-LLM in practice, we experiment with two modeling environments, i.e., CAEX and HEPSYCODE, employed in industrial use cases within European projects. We investigate how the examined LLMs can generate realistic modeling operations in different domains.
Conclusion: We show that synthetic traces can be effectively used when the application domain is less complex, while complex scenarios require human-based operations or a mixed approach according to data availability. However, generative AI models must be assessed using proper methodologies to avoid security issues in industrial domains.

1. Introduction

The recent proliferation of generative AI systems has dramatically impacted how software is developed, introducing new opportunities for automation and decision support across the software development lifecycle [1]. Among the various paradigms in software engineering, Model-Driven Engineering (MDE) [2] is a well-established approach that uses high-level models as primary artifacts in the software development process. This abstraction helps manage complexity, improve consistency, and enables tool-based automation.

In this context, the intersection between MDE and Large Language Models (LLMs) has drawn increasing attention from the research community [3], resulting in several proposals supporting different MDE tasks [4]. However, while these approaches show potential, LLMs are not yet capable of replacing human expertise, especially in tasks that require domain-specific knowledge and contextual understanding [5].

Modeling real-world industrial scenarios, in particular, demands a deep understanding of the application domain as well as careful handling of privacy, security, and intellectual property constraints, which often hinder the sharing or reuse of modeling data.

* Corresponding author.
E-mail addresses: vmuttillo@unite.it (V. Muttillo), claudio.disipio@univaq.it (C. Di Sipio), riccardo.rubei@univaq.it (R. Rubei), luca.berardinelli@jku.at (L. Berardinelli).

https://doi.org/10.1016/j.infsof.2025.107806
Received 14 September 2024; Received in revised form 14 May 2025; Accepted 28 May 2025
Available online 16 June 2025

Intelligent Modeling Assistants (IMAs) [6] aim to support modelers by recommending modeling operations or completing partial models. Yet, their effectiveness depends heavily on the availability of training data, specifically, traces of modeling operations. Unfortunately, the scarcity of publicly available training data represents an open challenge.

In such a context, this paper presents an extended version of our previous work, where we propose **MASTER-LLM**, a conceptual framework to support **M**odeling **A**ssistants using **S**ynthetic **T**race g**E**neration of modeling ope**R**ations through **LLM** [7]. In particular, we present a more in-depth exploration of the synthetic trace generation of modeling operations, including new experimental results and enhanced methodologies. In particular, we experiment with six different LLMs using an in-context learning strategy [8] to generate modeling operations given a graphical model editor capable of collecting modeling event traces. We discovered that mixing humans and generated traces is a trade-off solution when training models are unavailable.

In this extension, we consider the Computer Aided Engineering Exchange (CAEX)[1] format and a model-driven editor for CAEX, namely the CAEX Modeling System Environment (CAEX MSE), to further investigate the impact of LLMs in generating modeling operations. In particular, we use the additional MSE to validate the generalizability of the proposed framework, considering an additional domain, i.e., production system automation supported by CAEX. Furthermore, we extend the evaluation conducted in the prior work by analyzing the impact of six different application domains. The results show that synthetic traces lead to better accuracy scores when the CAEX MSE is considered. Contrariwise, manual traces are confirmed to be more effective when less complex domains, e.g., HEPSYCODE MSE, are considered.

The main contributions of the paper are the following:

- An additional MSE, namely, CAEX, and two LLMs, i.e., Claude and Mistral, to assess the generalizability of the MASTER-LLM conceptual framework proposed in Muttillo et al. [7];
- A rigorous evaluation including six application domains and statistical analyses to ensure the robustness of our study;
- A qualitative discussion highlighting critical issues of using generative AI in industrial modeling;
- A replication package to foster further research in this domain.[2]

2. Motivating examples

2.1. MDE in electronic design automation domain

To model high-performance embedded systems, dedicated tools belonging to *Electronic Design Automation* (EDA) [10]. With advancements in EDA, new methods and tools have emerged, enabling higher abstraction models through MDE approaches. HEPSYCODE[3] (HW/SW **CO**-**DE**sign of **HE**terogeneous **P**arallel dedicated **SY**stems) is a prototype EDA methodology and tool designed to reduce the design time of embedded applications. It uses Eclipse MDE technologies to model the behavior of embedded applications with a custom modeling workbench compliant with the HEPSYCODE metamodel. The HEPSYCODE language allows modeling the system as a network of processes communicating through channels.

Fig. 1 represents one real-world example of an embedded application called Digital Camera (DC). The main functionalities of the application, representing a camera that captures photographs in digital memory [11], include acquiring a 64 × 64-pixel image (i.e., *ccdpp* process), performing a zero-bias adjustment (i.e., *cntrl* process), compressing the image (i.e., *codec* process), and transmitting it to an external device (i.e., *uat* process). Data are exchanged through internal channels, while testbench and output feedback use additional external channels and ports. In this scenario, the HEPSYCODE graphical modeling workbench in Fig. 1(a) records the modeler's action events (i.e., the modeling operations), saved in an XES trace file, as shown in Fig. 1(b). Some examples of modeling operations performed by the designer are the following:

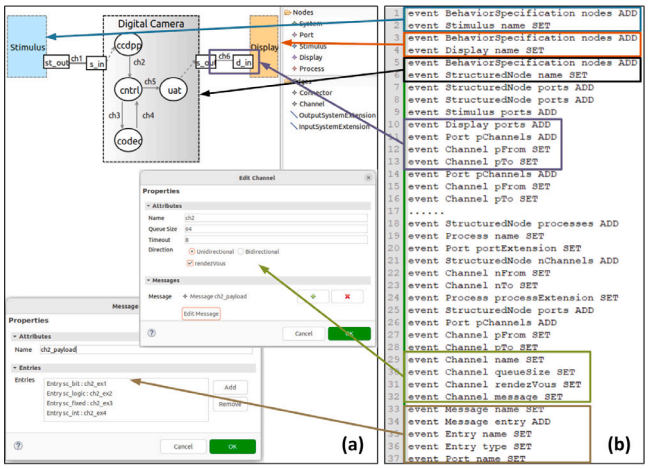


Fig. 1. HEPSYCODE Graphical Modeling Workbench (a) and trace file generated through MER tool (b). The application considered in this scenario is called *Digital Camera* [9].

- A behavioral element for simulating the environment has been added (marked with a light blue square), along with an element for execution feedback (marked with an orange square);
- The main system component (marked with a black square) has been added between the environment simulation component and the feedback component, with ports that connect the system to the feedback through a communication channel (marked with a violet square);
- A message has been added to channel *ch2* (marked with a green square), including its associated parameters and attributes (e.g., direction, queue size, rendezvous), with a payload containing multiple entries and embedded data (marked with a brown square).

It is worth noting that modeling operations are inherently sequential, meaning they follow a specific order during the construction of the model. However, multiple traces can represent different realizations of the same model. In this context, we shift the focus from dependency on a strict sequence of events to ensuring the presence of all required events, or at least the majority of them, across synthetic traces. This perspective prioritizes completeness and coverage over exact ordering and strict dependencies.

2.2. MDE in automotive domain

With increased customer and regulatory emphasis on sustainability, Volvo Construction Equipment (Volvo CE), a partner of the AIDOaRt project, is working on electrifying its construction machines, including battery-electric and fuel-cell technologies. Volvo CE provided an industrial use case whose challenges consisted of fostering the automation of the engineering process of Volvo CE vehicles like the *Dumper System* (DS) shown in Fig. 2(a).

MDE techniques and practices are explicitly introduced, with modeling tools playing a pivotal role in transforming descriptive engineering artifacts produced by office tools into models. The proposed

[1] IEC-62424 CAEX: https://www.iat.rwth-aachen.de.

[2] MASTER-LLM Replication Package: https://github.com/hepsycode/MASTER-LLM-IST.

[3] HEPSYCODE official WebSite: https://hepsycode.github.io/.

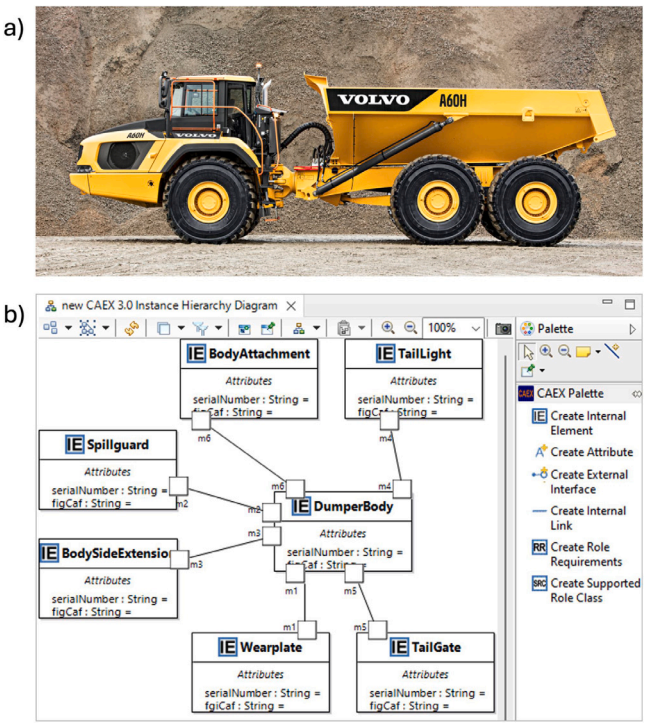


Fig. 2. CAEX Graphical Modeling Workbench. A Volvo dumper (A60H version) (a) and a model excerpt in the CAEX MSE (b).

solution includes the possibility to use AutomationML[4] as modeling language, which in turn relies on CAEX[1], to model vehicles' structural components and their variants [12].

The canvas in Fig. 1 shows the CAEX MSE and its support for the graphical modeling of an *instance hierarchy* and its *internal elements* (IEs) (e.g., `dumper body`). The right-hand panel is a palette containing CAEX modeling elements that are available for editing the model (e.g., some possible modeling operations).

2.3. Challenges and open issues

To assist designers, the modeling environment can collect their modeling operations as traces, as shown in the HEPSYCODE Fig. 1(b). These traces can help IMAs suggest possible modeling operations at any given step, supporting designers as the complexity of the model increases.

Although IMAs can help modelers, several issues need to be carefully handled. Among the others, we elicited the following challenges:

➤ **CH1: Collecting traces is time-consuming:** As shown in this section, collecting modeling operations is time-consuming if the system exploits a traditional event recorder. Moreover, the time needed to produce valuable traces depends strongly on the modeler's level of expertise, given the application domain.

➤ **CH2: Using external training data may leads to inaccurate results :** Alternatively, curated modeling datasets [13] can be used to synthesize traces to feed IMA as done by Di Rocco et al. [14]. Nonetheless, the existing datasets comprise models (and metamodels) specifically created for ML tasks without representing realistic modelers' behavior.

➤ **CH3: Accessing data is often difficult due to privacy issues and industry restrictions.** Industries and research institutions can handle data differently, according to internal or external regulations [15]. This may impact modeling artifacts available to enable automated approaches, resulting in a scarcity of training data. In particular, privacy agreements can negatively affect data disclosing [16], thus preventing researchers from developing accurate automated approaches to support industrial practitioners.

➤ **CH4: Automated tools for capturing modeling operations might be lacking.** A significant challenge in the MDE domain is the lack of automated tools to effectively capture and record modeling operations. This limitation complicates the development of integrated systems, as these modeling operations are essential for tracking and improving the modeling process.

3. Related work

We provide an overview of the state-of-the-art to enlist the closest approaches to our work. To the best of our knowledge, there is no similar approach that employs LLM to generate synthetic traces in the context of an MDE environment. Nevertheless, we describe the state-of-the-art of the works that are related to our internal components.

Modeling Trace Recording. Dehghani et al. [17] presented the Modeling Event Recorder (MER) tool that captures user interaction events through the EMF notification API [18] and saves traces using the IEEE eXtensible Event Stream (XES) format [19]. Our approach leverages the MER component to collect traces. Herrmannsdoerfer et al. [20] discuss a generic operation recorder for model evolution based on an operation metamodel. As MER, it reuses EMF Notifications but neglects compatibility with standards like XES. Brosch et al. [21] exploited the concept of operation recording to perform model versioning. In particular, they relied on the tool Operation Recorder previously introduced by Herrmannsdoerfer et al. [20].

Synthetic Data Generation. In MDE, different approaches have been developed to ease the data scarcity issue [22], exploiting clustering, grammar graphs, and random generators. The clustering approach usually classifies variable values and relationships between components. Then, an instance model is produced from each category to represent that category [23]. To generate the model based on graph grammars, the graph rules are extracted from the metamodel, and then models are generated according to these rules [24]. In the random approach, random procedures are used to generate new models [25]. Furthermore, Soltana et al. [26] present a viable approach for creating system test data using constraint-solving techniques.

The recent advancement of LLMs has motivated the exploration of this technology to generate synthetic datasets [27–29].

The most relevant approach to ours was presented by López et al. [30], where the authors propose the Text2VQL framework, specifically designed to transform natural text to queries expressed in the VIATRA Query Language (VQL). To fine-tune two open-source LLMs, i.e., DeepSeekCoder and Code Llama, they leverage ChatGPT to generate a synthetic dataset comprising pairs of queries and their corresponding natural language descriptions.

Overall, we did not find evidence of using LLMs for trace generation to train IMAs apart from Muttillo et al. [7].

Existing IMAs in MDE context. Several IMAs have been developed to assist modelers in their daily tasks. Above all, model completion is the most supported task, leveraging on NLP techniques [31] and similarity-based algorithms [32] to recommend missing modeling elements given an incomplete model. Adhikari et al. [33] introduce SimVMA, a virtual assistant that analyses an incomplete model and, by inspecting similar projects, can suggest pertinent model completion elements. Iovino et al. [34] extended the tool named PARMOREL to provide personalized and automatic repair suggestions for models leveraging reinforcement learning. Di Rocco et al. [14] present NEMO, an IMA that forecasts the next modeling operations leveraging the LSTM network by relying on a curated dataset of BPMN models. However, we cannot reuse NEMO in a direct comparison since it is tailored for BPMN models. Weyssow et al. [35] proposed a learning-based approach that leverages the RoBERTa pre-trained model to suggest relevant metamodel elements. The metamodels are encoded as structured

[4] AutomationML: https://www.automationml.org/.

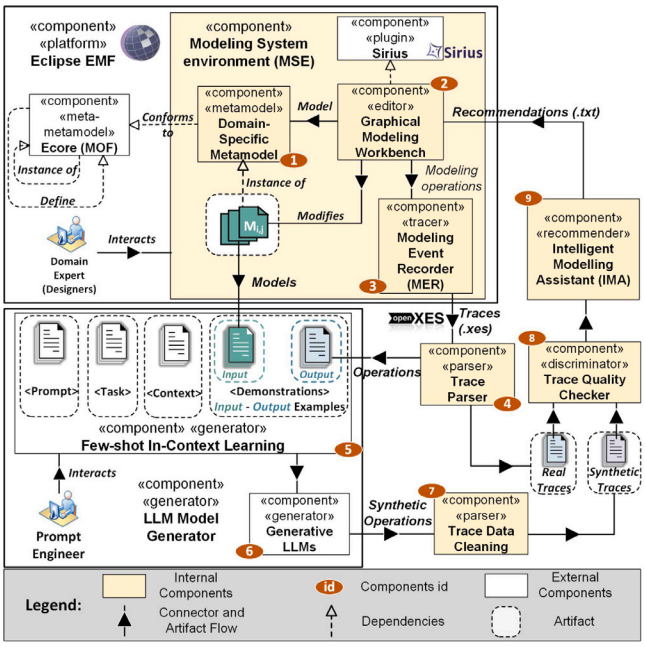


Fig. 3. Overview of the MASTER-LLM framework.

trees to train the underlying model and obtain a textual sequential representation. Subsequently, a test set is generated using a sampling strategy relying on masking where a portion of the input text is randomly modified [36], allowing the model to learn to predict the masked text by relying on the contextual remaining words. The employed model is then used to predict missing elements and provide the modeler with insightful domain concepts. Chaaben et al. [37] leverages the GPT-3 model to complete the model under construction using the few-shot technique. In particular, the model elements are embedded in the prompt as structured text. The approach supports both sequence and class diagrams, even though there is still room for improvement.

Kulkarni et al. [38] exploits GPT-4 to specify digital twins models. The prompt is iteratively refined using the Goal-Measure-Lever (GML) metamodel, in which the modeler can define the goals and sub-goals of the final system.

Finally, Apvrille and Sultan [39] combine ChatGPT and the online TTool framework to complete structural and behavioral SysML models. Given the partial SysML and the domain knowledge encoded as a JSON request, the TTool framework extracts the GPT's response and delivers it to the modelers. The evaluation shows that the proposed framework slightly outperforms students in the modeling tasks, even though the results are worse when complex specifications are considered.

While we acknowledge that LLMs can be used as IMAs, our framework focuses on their usage in generating training data for traditional approaches, e.g., MORGAN.

4. Proposed approach

Fig. 3 depicts the MASTER-LLM, a conceptual framework to support **M**odeling **A**ssistants using **S**ynthetic **T**race g**E**neration of modeling ope**R**ations through **LLM**. The goal is to capture events generated by users' modeling operations and generate traces. Such traces are then injected into recommender systems, enabling the generation of personalized suggestions for modeling actions most relevant to each designer.

4.1. Modeling System Environment (MSE)

The MSE is a graphical model editor. MDE leverages models as core software artifacts. The Meta Object Facility (MOF)[5] by the Object Management Group (OMG) organizes modeling artifacts into three metalayers. Building upon MOF, a modeling framework provides the necessary characteristics for working with these modeling artifacts. EMF[6] is a standard de facto implementation of the MOF architecture. Its metamodeling language, Ecore, represents the meta-metamodel layer. Using Ecore, developers define custom metamodels.

Formally, we also define the set of M models conforming to the Ecore-based metamodel as follows:

$$M = \{M_1, M_2, \dots, M_j, \dots, M_M\} \tag{1}$$

4.2. Modeling Event Recorder (MER)

Formally, in this paper, we use *modeling operations* to refer to activities performed by users on the Graphical Modeling Workbench to create model M_j compliant with the metamodel. Modeling operations generate *events* captured by a notification mechanism and suitably saved as *traces* by a Modeling Event Recorder ③. Moreover, $\Gamma(M)$ is the set of traces obtained from modeling operations that realized the models M, such as:

$$\Gamma(M) = \{\tau_1(M_1), \tau_2(M_2), \dots, \tau_j(M_j), \dots, \tau_M(M_M)\} \tag{2}$$

To simplify matters, we will remove the internal model notation $\tau_j(M_j)$ and only keep τ_j as a generic trace. Each trace τ_j can be split into N events (i.e., single designer modeling operation), as follows:

$$\tau_j := \{e_{j,1}, e_{j,2}, \dots, e_{j,k}, \dots, e_{j,N}\} \tag{3}$$

Each trace event has a fixed syntax, determined by the MER component:

$$e_{j,k} := \text{event } \langle\text{class}\rangle\ \langle\text{attribute}\rangle\ \langle\text{eventType}\rangle \tag{4}$$

With the term modeling event recording, we refer to the collection of modeling traces through modeling event notification mechanisms.

4.3. Intelligent Modeling Assistant (IMA)

To define the *recommender* component ⑨, we rely on the IMA definition provided by Mussbacher et al. [6]. In particular, a *data acquisition layer* must be defined to collect the relevant knowledge from external sources. In addition, an IMA operates in a *context* where modelers perform their activities, thus producing contextual information that can be processed by the IMA. The core component is represented by the *assistant*, namely the algorithm used to perform the actual automated activities, e.g., suggesting missing elements, retrieving similar modeling artifacts, or forecasting the next operations. We focus on IMAs that can retrieve relevant modeling operations given a graphical modeling environment. Formally, given the modeler's context (i.e., a model M_j), the knowledge acquired from external sources or the modeling context (i.e., $\Gamma(M)$ traces set), and A the assistant, the IMA is a function $\mathcal{I}$ defined as follows:

$$\mathcal{I}(M, \Gamma(M), A) = \{Op_1, Op_2, \dots, Op_N\} \tag{5}$$

In the scope of the paper, we consider the *past operations* $\Gamma(M)$ as the unique source of knowledge for the IMA. Given the above-mentioned definition of an event, an explanatory list of recommendations Rec for a given model (M_j) is represented below:

$$Rec(M_j) = \{R(e_{j,1}), R(e_{j,2}), \dots, R(e_{j,N})\} \tag{6}$$

where the list $\{R(e_{j,1}), R(e_{j,2}), \dots, R(e_{j,N})\}$ is the recommended modeling operations.

[5] MetaObject Facility Specification: https://www.omg.org/mof/.
[6] Eclipse Modeling Framework: https://www.eclipse.org.

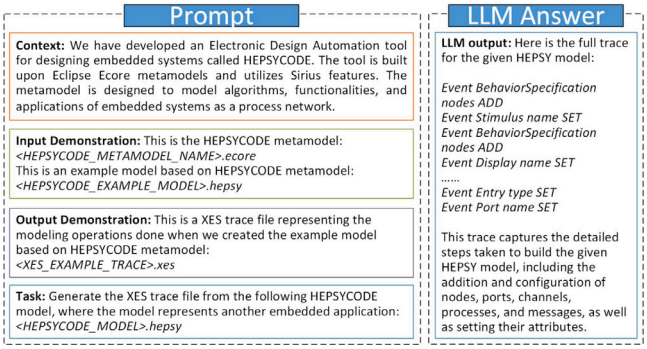


Fig. 4. Prompt schema and LLM answer example for HEPSYCODE.

4.4. Large Language Model (LLM)

A key phase of using LLMs is the design and execution of the most suitable prompt strategy given the goal. *Zero-shots* represents the basic prompt strategy, in which the LLMs are fed with just the query without any example of the expected outputs [40]. The query can be expressed using natural language or embodying specific keywords that refer to the context, e.g., parts of models under development. *Few-shots* requires explanatory outputs in the initial query [41]. In such a way, the task can be executed with weak labeling and minimal supervision from the developers. *Chain-of-Thoughts* is a conversational reasoning task to assess the ability of a model to maintain coherence and context across a series of questions and answers [42]. In such a task, a series of questions is posed, with each of them building upon the context established by the previous question and answer pair.

In this paper, we adopt the *few-shot* prompting strategy ❺ to generate modeling operations ❻ since it is suitable for obtaining the traces using the specified models. Furthermore, we defined $\Gamma^{+}(M)$ as the set of LLM synthetic traces (i.e., emulated human modeling operations) needed to realize the model M_j, such as:

$$\Gamma^{+}(M) = \{\tau_1^{+}(M_1), \tau_2^{+}(M_2), \ldots, \tau_j^{+}(M_j), \ldots, \tau_M^{+}(M_M)\} \tag{7}$$

Following modeling event recorder notation, we removed the internal model notation $\tau_j^{+}(M_j)$ and only kept τ_j^{+} as a generic synthetic trace. Each τ_j^{+} synthetic trace can be split into N synthetic events (i.e., single LLM human emulated modeling operation), as follows:

$$\tau_j^{+} = \{e_{j,1}^{+}, e_{j,2}^{+}, \ldots, e_{j,k}^{+}, \ldots, e_{j,N'}^{+}\} \tag{8}$$

The LLM is trained on demonstrations (i.e., input–output example traces) to emulate the human modeling operation steps using the Few-Shot In-Context Learning prompting approach ❺. Fig. 4 shows an example prompt schema used in this work.

5. Experiment design

This section details the experiment design used in our work. Our approach ensures systematic analysis for reliable and reproducible results.

5.1. Research questions

Based on the research objectives, we aim to answer the following research questions:

➤ **RQ$_1$**: *What is the quality of the synthetic traces?* To answer this question, we assess the quality of the generated traces by applying well-founded correctness and diversity metrics. In addition, we assess the degree of hallucination by defining a novel metric tailored to the context of MDE systems.

➤ **RQ$_2$**: *Is synthetic datasets useful for training a lightweight IMA?* After selecting the best LLM, we trained an existing IMA, i.e., MORGAN [43], with human-generated traces, synthetic ones, and a mix of both. In particular, the goal is to investigate to what extent synthetic traces can be used to replace human ones in real-world scenarios, including industrial use cases.

5.2. Employed tool components

This section presents the tool components employed in the proposed approach.

5.2.1. MSE eclipse-based workbench

The selected MSEs are the *HEPSYCODE Modeling Workbench*[3] (HW/SW *CO-DE*sign of *HE*terogeneous *P*arallel dedicated *SY*stems) and the *CAEX Modeling Workbench* for CAEX[7]. HEPSYCODE and CAEX MSEs have been developed using the Eclipse EMF as the reference language workbench [44] and Sirius to generate its graphical modeling environment as a plugin of the Eclipse platform. The main component "Eclipse («platform»)" in Fig. 3 represents our instantiation of the MOF architecture in which EMF and the Sirius plugin are used to create the Graphical Modeling Workbench ❷. The latter handles models (M) compliant with custom Domain Specific Metamodel ❶. Examples of this Graphical Modeling workbench can be found in Pomante et al. [45] and Cederbladh et al. [12], which further extend the framework with EMF-compliant technologies (e.g., EMF Views [46]).

➤ **CAEX** is a standard for data exchange between engineering tools. CAEX allows the storage of hierarchical object information and provides explicit support for representing objects' attributes and interfaces as well as their relationships, such as inheritance and links among interfaces. CAEX is a native XSD-based data format. An XML Schema is provided by the standard[1]. It is used as the top-level data format for AutomationML[4], a data exchange format used in the manufacturing domain that integrates further standards for representing geometry, kinematics (COLLADA), and logics (PLCOpen XML) of manufacturing systems, which are not considered in this work. The CAEX MSE[7] allows the modeling of hierarchical structure in CAEX based on a CAEX metamodel in EMF[6] (caex.ecore), suitably derived from the standard CAEX XML Schema (caex.xsd).

➤ **HEPSYCODE** is a framework and prototypal tool to improve the design time of embedded applications. HEPSYCODE uses Eclipse MDE technologies, SystemC custom simulator implementation, and AI-augmented algorithms for partitioning activities, all integrated into an automatic framework that drives the designer from the first input specifications to the final solution. The metamodeling language introduced in HEPSYCODE, named *HEPSY* allows modeling the system's behavior as a network of processes communicating through unidirectional synchronous channels.

5.2.2. Modeling event recorder for EMF-based editors

The MER implementation for EMF-based models and editors is presented in Dehghani et al. [17] as subcomponents of a Modeling Process Mining Tool [17] ❸. MER is thus implemented as an Eclipse plugin that interacts with Sirius-based graphical editors for EMF-based models, as the CAEX modeling workbench[7] or HEPSYCODE tool[3], and records users' modeling traces. Traces are encoded in the IEEE Standard for eXtensible Event Stream (XES) [19], which provides an XML schema for log encoding. The MER tool also provides an Ecore-based XES metamodel for encoding traces as EMF-based models.

[7] CAEX MDE Workbench: https://github.com/amlModeling/caex-workbench/tree/variability.

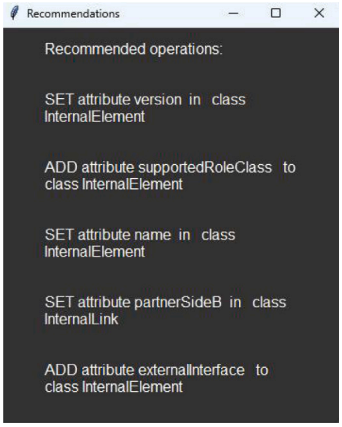


Fig. 5. Excerpt of MORGAN's recommendations using Jinja template.

5.2.3. MORGAN

We consider the MORGAN tool [43] as the IMA component in our framework. In particular, we modify MORGAN's architecture by introducing a Trace parser ④ to extract relevant information from XES traces obtained by the MER component ③ [17]. We obtain a textual-based representation used by the graph encoder to produce a list of trace graphs by extracting different features for each event, i.e., the type of event and the affected artifacts. Each graph is constructed using a set of Natural Language Preprocessing (NLP) techniques, i.e., stemming and dash-removal. Afterward, MORGAN exploits the Weisfeiler-Lehman algorithm [47] to provide the ranked list of similar operations given an initial XES trace, provided by the Grakel Python library.[8] Listing 1 shows a fragment of MORGAN's recommendations expressed in the MER format for the HEPSYCODE modeling environment. In addition, we encapsulated the recommendations in a simple graphical GUI using the Jinja library[9] to increase usability, as shown in Fig. 5. While a complete integration with the existing modeling environments is out of the scope of this paper, we will see this as possible future work.

Listing 1: Excerpt of MORGAN's textual recommendations.

```
Port pChannels ADD,Message entry ADD, Stimulus
    ports ADD

Channel rendezVous SET, Message entry ADD,
    Channel pTo SET, Entry name SET, Channel
    queueSize SET, Channel name SET, Message
    name SET, Channel nFrom SET, Message entry
    ADD, Entry type SET.
```

5.2.4. Large language model

LLM synthetic dataset generation effectively addresses the challenges of reducing the time and effort to collect traces (CH1), representing realistic designer behavior (CH2), and overcoming security concerns, privacy regulations, and industry restrictions [16]. By leveraging the advanced capabilities of LLMs, our proposed approach provides a robust solution for creating high-quality, relevant, and secure datasets, thereby enhancing the development and performance of ML models.

We experiment with six pre-trained decoder-only transformers-based LLMs as the *generator* component ⑥ in Fig. 3:

1. Llama 3 [48], developed by Meta;
2. Gemini 1.5 Flash,[10] developed by Google DeepMind;
3. Mistral Large 24.07,[11] developed by Mistral AI;
4. Claude 3.5 Sonnet,[12] developed by Anthropic;
5. GPT-3.5.[13] freely available and developed by OpenAI;
6. GPT-4,[14] the professional version of GPT also developed by OpenAI.

Table 1
High-level architecture overview of selected language models.

Model	Layers	Params	Context Windows	Output Tokens
Llama 3	80	70B	8K	4K
Gemini 1.5 Flash	N/A	120B[a]	2M	8K
Mistral Large	N/A	123B	32K	4K
Claude 3.5 Sonnet	N/A	175B[a]	200K	8K
GPT-3.5	up to 96[a]	175B[a]	16K	4K
GPT-4	N/A	1760B[a]	8K	8K

[a] The number of parameters and layers is not officially disclosed but was leaked online [50].

Table 1 presents a high-level architectural overview of the selected LLMs. GPTs, Claude, Mistral, and Gemini are proprietary, meaning their underlying details and weights are not shared publicly. Llama 3 is an open-source model, offering transparency and flexibility with its publicly available architecture and weights. We selected these LLMs because they represent the best proprietary and open-source technologies according to LMSYS LeaderBoard.[15] In addition, we experimented with LLMs proposed by different providers, thus increasing the generalizability of the approach [49]. In the scope of the paper, we used a free web-based interface for all the selected LLMs. Therefore, we cannot configure the token size for each of them. Meanwhile, we experimented with this limitation in some input and output phase cases. We asked the LLM to continue the generation phase in the same chat to cope with this, thus preserving the active context.

5.3. Subjects selection and datasets

The population we consider includes both real and synthetic models used in two domains: *Electronic Design Automation* with HEPSYCODE and *Production System Automation* with CAEX/AutomationML. From this population, we selected a sample of 15 models for HEPSYCODE and 20 models for CAEX, which are included in the D1 dataset. Synthetic datasets generated through LLM in-context learning are collected in the D2 dataset. From the industrial domain, we gathered five real Use Case (UC) models for HEPSYCODE (i.e., from Thalés France, Thales Alenia Space España, Integrasys, and TEKNE companies) and one real UC model for CAEX (i.e., from Volvo CE company), which are included in the D3 dataset. All the datasets are presented below.

➤ **D1 - Real trace dataset:** The trace dataset comprises real traces generated using HEPSYCODE and CAEX modeling workbenches.

The HEPSYCODE models are embedded systems application models taken from the literature, with a total amount of 2379 XES events, as described below:

- **Synthetic**: Four HEPSYCODE synthetic custom applications [51] [478 events], called M_1^H, M_2^H, M_3^H, and M_4^H;
- **Academic**: Sequential and Parallel version of Electronic Digital Camera [9,51] (Embedded System Design) [282 events], called M_5^H and M_6^H; three versions of an academic synthetic application that uses the Finite Impulse Response (FIR) filter [9] [785 events], called M_7^H, M_8^H, and M_9^H;

[8] GraKeL: https://ysig.github.io/GraKeL/0.1a8/.
[9] Jinja Engine: https://jinja.palletsprojects.com/en/stable/.
[10] Google, Google Gemini: https://gemini.google.com/app.
[11] Le Chat: https://mistral.ai/news/le-chat-mistral/.
[12] Claude: https://www.anthropic.com/claude.
[13] OpenAI ChatGPT: https://openai.com/chatgpt.
[14] OpenAI GPT-4: https://openai.com/index/gpt-4.
[15] LMSYS LeaderBoard: https://chat.lmsys.org/.

- **Computer vision and Signal Processing**: Two versions of the JPEG Encoder [52] [413 events], called M_{10}^H and M_{11}^H; Sobel, Susan, and Roberts Filters [51–53] [291 events], called M_{12}^H, M_{13}^H, and M_{14}^H; Relative Spectral Transform (RASTA) filter to support Perceptual Linear Prediction (PLP) preprocessing for speech signals analysis [52] [130 events], called M_{15}^H;

The CAEX models are 20 synthetic models (e.g., $M_1^C, M_2^C, \ldots, M_{20}^C$) that partially cover meta-classes and attributes included in the CAEX metamodel (e.g., we do not consider System Unit Classes), thereby providing a good dataset for generating recommendations to designers for modeling operations in the automotive domain (e.g., the dataset created encompasses the classes and attributes utilized for developing the considered Volvo CE industrial case study model, as discussed below), with a total amount of 807 XES events.

➤ **D2 - LLM synthetic trace datasets:** To evaluate the potential of using LLMs in an MDE context to support designers, we created datasets of the synthetic traces. For the creation of the datasets, we used the same prompt, as shown in Fig. 4, on all the considered LLMs through the online query form, as presented in Section 5.2. In particular, we used one example model and trace in the "Input Demonstration" prompt activity and all the real models from the D1 dataset in the "Task" prompt activity to generate the synthetic traces.

➤ **D3 - Industrial dataset:** HEPSYCODE and CAEX have also been used and validated through several industrial UC studies from European Projects, covering 4 domains (i.e., Avionics, Smart Cities, Space, and Automotive) as follows:

- H2020-ECSEL-2016-1-RIA **AQUAS**[16]
  - **UC$_1$** "Air Traffic Management System" [54] from *Integrasys* in Spain (domain: Smart Cities, 161 events);
  - **UC$_2$** "Space Multicore Architecture with standard Space Satellite tasks" [54] from *Thales Alenia Space España* in Spain (domain: Space, 125 events);
  - **UC$_3$** "Space Multicore Architecture with Onboard Security Encryption Tasks" [54] from *Thales Alenia Space España* in Spain (domain: Space, 200 events);
- H2020-ECSEL-2016-1-RIA **MegaM@aRt2**[17]
  - **UC$_4$** "Flight Management System" [55] from *Thalés* in France (domain: Avionics, 388 events);
- H2020 ECSEL 2020-2-RIA **AIDOaRT**[18]
  - **UC$_5$** "Agile process and Electric/Electronic Architecture of a vehicle for professional applications" [56] from *TEKNE* in Italy (domain: Automotive, 205 events)
  - **UC$_6$** "Data modeling to support product development cost and efficiency" [12] from *Volvo CE* in Sweden (domain: Automotive, 120 events)

The total amount of XES events for the HEPSYCODE industrial models is 1079 for **UC$_1$**, **UC$_2$**, **UC$_3$**, **UC$_4$**, and **UC$_5$**, spanning over three different EU projects. Furthermore, the CAEX industrial model has 120 total XES events from the Volvo CE AIDOaRt **UC$_6$**[18].

Noteworthy, the CAEX models created for the Volvo CE case study use rather small subsets of CAEX metamodel, i.e., `InstanceHierarchy`, `InternalElement`, `Port` metaclasses, to create hierarchical structures of components and their variants (typically referred to as 150%-models [57]), without modeling interconnections (e.g., using `InternalLink`) or other advanced modeling concepts such as roles and libraries[1]. As a result, the D1 dataset reflects the modeling problem complexity, involving only nesting without links, thus creating hierarchical nodes without arcs.

[16] AQUAS EU Project: https://aquas-project.eu/.
[17] MegaM@aRt2 EU Project: https://megamart2-ecsel.eu/.
[18] AIDOaRT EU Project: https://sites.mdu.se/aidoart.

5.4. Evaluation metrics for LLM synthetic trace data

Synthetic data generated from LLMs inherently faces several data limitations that must be acknowledged and addressed. As an inherent characteristic, LLMs may inadvertently propagate inaccuracies or biases present in their pre-training data, leading to outputs that may not always align with factual or unbiased information. Moreover, synthetic data generated by LLMs can sometimes not only be inaccurate but also completely fictitious or disconnected from reality, a phenomenon often referred to as "hallucination".

The quality of the synthetic traces can be assessed in terms of *correctness*, *diversity*, and *hallucination* metrics.

5.4.1. Trace correctness metric

The *Correctness* metric measures whether the data instance is related to the given label. Existing approaches for measuring correctness can be divided into two categories: automatic evaluation and human evaluation. Human evaluation has been conducted by prompt engineers to self-tune the *Few-shot In-Context Learning* component.

Automatic evaluation has been implemented to check the correctness of event syntax using the following metric:

$$C(\tau_j^+) = \frac{\sum_{k=1}^{N'} c(e_{j,k}^+)}{|\tau_j^+|}, \quad \text{where} \quad c(e_{j,k}^+) = \begin{cases} 1 \text{ if } e_{j,k}^+ \text{ has correct syntax} \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

This metric can be evaluated on the full τ_j^+ synthetic trace, while it is possible to cluster the metrics w.r.t syntax features (i.e., MER metamodel classes and attributes).

It is worth noting that we have not enforced strict dependencies on the order of synthetic events because the same model can be built in different ways using various sequences of actions. However, we have ensured that key semantic dependencies are respected in the synthetic modeling operations. This approach allows for flexibility in the traces, representing different valid ways of constructing a model while verifying essential structural and semantic requirements. For example, communication links between classes are only added when at least two classes exist, and structural elements that include or depend on other elements are verified to ensure their dependencies are satisfied. Additionally, we ensured that SET events follow the corresponding ADD events to maintain semantic consistency. To achieve this, we implemented a Python script to validate these semantic relationships.

5.4.2. Trace diversity metric

Diversity measures the difference between a chunk of text and another in the generated instances. In this work, we evaluate differences between τ_j^+ synthetic traces generated by LLMs and real τ_j traces generated by designers using the MER component. The considered metrics are the following:

➤ **Edit-based** similarities, also known as distance-based, measure the minimum number of single-character operations (e.g., insertions, deletions, or substitutions) required to transform one string into another.

Levenshtein: The Levenshtein distance $dist(\tau_j, \tau_j^+)$ between τ_j and τ_j^+ is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one trace into the other, defined as follows:

$$\text{LEV}(\tau_j, \tau_j^+) = 1.0 - \frac{dist(\tau_j, \tau_j^+)}{max(|\tau_j|, |\tau_j^+|)} \tag{10}$$

Longest Common substrings (LCS): The maximum-length common events subsequence LCS(i,k) of τ_j and τ_j^+, considering only characters insertion and deletion, where i and k represent the prefix length

of trace string $\tau_j[i] \in \tau_j$ and $\tau_j^+[k] \in \tau_j^+$, respectively, is given by:

$$LCS(i,k) = \begin{cases} 0 & \text{if } i = 0 \vee k = 0 \\ LCS(i-1,k-1)+1 & \text{if } i,k > 0 \wedge \tau_j[i] = \tau_j^+[k] \\ 0 & \text{if } i,k > 0 \wedge \tau_j[i] \neq \tau_j^+[k] \end{cases} \quad (11)$$

Jaro–Winkler: The Jaro Similarity is calculated using the following formula:

$$\text{JARO}(\tau_j, \tau_j^+) = \begin{cases} 0 & \text{if m} = 0 \\ \frac{1}{3}\left(\frac{m}{|\tau_j|} + \frac{m}{|\tau_j^+|} + \frac{m-t}{m}\right) & \text{Otherwise} \end{cases} \quad (12)$$

where m is the number of matching characters between τ_j and τ_j^+ and t is half the number of transpositions.

➤ Among the **token-based** similarity function, we consider:

Jaccard: measure the size of the intersection divided by the size of the union of the strings, as follows:

$$\text{JACCARD}(\tau_j, \tau_j^+) = \frac{|\tau_j \cap \tau_j^+|}{|\tau_j| + |\tau_j^+| - |\tau_j \cap \tau_j^+|} \quad (13)$$

Overlap: measures the similarity between two strings, calculated as the size of their intersection divided by the size of the smaller string:

$$\text{OVERLAP}(\tau_j, \tau_j^+) = \frac{|\tau_j \cap \tau_j^+|}{\min(|\tau_j|, |\tau_j^+|)} \quad (14)$$

Sorensen-Dice: evaluate twice the number of elements common to both traces divided by the sum of the number of elements in each trace, as follows:

$$\text{DICE}(\tau_j, \tau_j^+) = \frac{2|\tau_j \cap \tau_j^+|}{|\tau_j| + |\tau_j^+|} \quad (15)$$

Q-Gram: count the number of occurrences of different q-grams in the two traces. Given a trace τ_j and let $v \in \Psi^q$ a q-gram, the total number of occurrences of v in τ_j, denoted by $G(\tau_j[v])$, is obtained by sliding a window of length q over the trace tokens. Given two traces τ_j and τ_j^+, the Q-gram similarity is described as follows:

$$\text{Q-GRAM}(\tau_j, \tau_j^+) = 1 - \frac{\sum_{v \in \Psi^q} |G(\tau_j)[v] - G(\tau_j^+)[v]|}{\sum_{v \in \Psi^q} max(G(\tau_j)[v], G(\tau_j^+)[v])} \quad (16)$$

The more q-grams two traces have in common, the more closely related they are.

Cosine: similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them, as follows:

$$\text{COSINE}(\tau_j, \tau_j^+) = cos(\theta) = \frac{\tau_j \cdot \tau_j^+}{\|\tau_j\| \cdot \|\tau_j^+\|} \quad (17)$$

where $\tau_j \cdot \tau_j^+$ is the dot product between the vector τ_j and τ_j^+, and $\|\tau_j\|$ represents the Euclidean norm of the vector τ_j. The resulting measure of similarity spans from −1, signifying complete opposition, to 1, indicating absolute identity. A value of 0 signifies orthogonality or decorrelation, while values in between denote varying degrees of similarity or dissimilarity. For text matching, the attribute vectors τ_j and τ_j^+ are the term frequency vectors of the traces.

These edit- and token-based diversity metrics can be used to evaluate how well LLMs can emulate both the designer's modeling approach and patterns, as well as human-based modeling approaches.

5.4.3. Trace hallucination metric

In the scope of the paper, we define hallucination as the number of additional operations, namely *non-realistic events*, generated compared to the human ones by specifying the following metric:

$$H(\tau_j, \tau_j^+)_{\langle \text{event} \rangle} = \frac{\sum_{k=1}^{N'} (\text{Synthetic Events } e_{j,k}^+ \text{ of type } \langle \text{event} \rangle)}{\sum_{k=1}^{N} (\text{Real Events } e_{j,k} \text{ of type } \langle \text{event} \rangle)} \quad (18)$$

This metric can be evaluated on the full τ_j^+ synthetic trace file and τ_j real trace file, as well as for all the considered events, classes, attributes, or specific event types (e.g., ADD or SET). If these metrics are greater than 1, then the LLM produces an incorrect synthetic trace file (i.e., hallucination results, the LLM adds more classes or attributes than those present in the real trace model).

5.5. Evaluation metrics for modeling recommendations

Concerning the produced recommendations, we set up an automatic evaluation of accuracy metrics aiming at mimicking the modeler's behavior. In the scope of this paper, we define the **True Positive (TP)** as the correct recommended operation, **False Positive (FP)** as the wrong operation, and **False Negatives (FN)** as the operations that should be included in the recommendations but actually are not. Given these definitions, we define **Precision (PR)**, **Recall (REC)**, and **F1-score (F1)** as follows:

$$\text{PR} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (19)$$

$$\text{REC} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (20)$$

$$\text{F1} = 2 \times \frac{\text{PR} \times \text{REC}}{\text{PR} + \text{REC}} \quad (21)$$

Concerning the assessed modeling operations, we evaluate the approach using two different parameters, i.e., **context ratio (CR)** and **cutoff (CO)** as done by Muttillo et al. [7]. The first parameter represents the number of operations captured at a certain timestamp, i.e., past operations. In the scope of the paper, we mimic three different stages of models, i.e., early stage, medium, and almost complete, considering three thresholds, i.e., $CR_1 = 0.2$, $CR_2 = 0.5$, and $CR_3 = 0.8$ of the original testing model. Similarly, we vary the number of recommended operations by setting the CO parameters with $CO_1 = 3$, $CO_2 = 5$, and $CO_3 = 10$ operations as thresholds. Thus, we derived nine different configurations, i.e., $C_{1.1}$, $C_{1.2}$, $C_{1.3}$, $C_{2.1}$, $C_{2.2}$, $C_{2.3}$, $C_{3.1}$, $C_{3.2}$, $C_{3.3}$.

For instance, configuration $C_{1.2}$ represents the situation where the system recommends few operations in an early stage of development, i.e., $CO_1 = 3$ and $CR_2 = 0.5$, respectively.

Each configuration has been evaluated using the 5-fold cross-validation since it is a well-founded strategy to evaluate ML-based recommender systems automatically [58]. In particular, we split the operations into train, test, and **Ground Truth (GT)** data by resembling the MORGAN original setting presented by Di Sipio et al. [43]. The train traces are used to feed the underpinning graph kernel engine and are compared with the testing ones. We obtained the GT data by relying on the CR parameter to vary the number of already performed operations. Concretely, augmenting CR reduces the number of operations to be predicted, i.e., the GT operations. We eventually use the test and GT data to compute the accuracy using the metrics presented in Section 5.4. To avoid any bias in the evaluation, we randomize the testing and GT operations, thus assuming that there is no temporal relationship between them. In addition, we analyze the time required to perform (i) the loading of training traces and encoding them in a graph-based format and (ii) the recommendation for all the testing operations.

5.6. Experimental variables

To answer RQ_1, the synthetic datasets used for recommendations are the **independent variable**, with **treatments** or variations for this independent variable depending on the LLMs (i.e., OpenAI, Meta, Mistral, Anthropic, and Google).

For RQ_2, the nine configurations from Section 5 are the **independent variable**. We analyze two aspects: training data size and using synthetic data to replace human-generated operations. To support the first aspect, we run a 5-fold validation on three datasets. Concerning the second aspect, we create a new synthetic dataset, D_2, from human

traces in D_1. Additionally, we mix 50% of traces from D_1 and D_2 to form the D_{m05} dataset. The **treatments** depend on the dataset types (human-generated, LLM-generated, and mixed).

For the domain analysis, the **independent variable** is the dataset type (D_1, D_2, and D_{m05}). In addition, we derive two datasets by mixing different ratios of synthetic and real traces, i.e., D_{m02} and D_{m08}, where the ratios of synthetic traces are 0.2 and 0.8 out of the total number, respectively. The **treatments** or variations depend on the industrial use cases.

5.7. Experimental hypotheses

➤ **RQ$_1$**: The null hypotheses for **RQ$_1$** are related to metrics defined above. The null hypothesis for the correctness metric states that the mean correctness metric value $\mu^{Corr}_{LLM_i}$ of the ith LLM is lower than 0.95. In contrast, the alternative hypothesis states that their mean value is greater than 0.95.

$$\begin{aligned} H_0^{Corr} &: \mu^{Corr}_{LLM_i} < 0.95 \ \forall i = 1, \dots, k \\ H_A^{Corr} &: \mu^{Corr}_{LLM_i} > 0.95 \ \forall i = 1, \dots, k \end{aligned} \tag{22}$$

The null hypothesis checks if the generated synthetic traces are syntactically incorrect. Rejecting it allows us to assess the quality of the synthetic dataset in terms of syntax.

For diversity metrics, we aim to determine if some LLMs perform better than others by comparing their distributions and statistical values. The null hypothesis states that the mean diversity metric values across different LLMs are the same, while the alternative hypothesis tests whether at least two LLMs have different mean diversity metrics.

$$\begin{aligned} H_0^{Div} &: \mu^{Div}_{LLM_1} = \mu^{Div}_{LLM_2} = \dots = \mu^{Div}_{LLM_k} \\ H_A^{Div} &: \forall i \neq j \ \exists \ i, j = 1, \dots, k \mid \mu^{Div}_{LLM_i} \neq \mu^{Div}_{LLM_j} \end{aligned} \tag{23}$$

Finally, we conducted a statistical analysis to evaluate the hallucination metric for all LLMs. The null hypothesis states that the average hallucination value is greater than 1 for each LLM, while the alternative hypothesis asserts the opposite.

$$\begin{aligned} H_0^{Hall} &: \mu^{Hall}_{LLM_i} > 1 \ \forall i = 1, \dots, k \\ H_A^{Hall} &: \mu^{Hall}_{LLM_i} < 1 \ \forall i = 1, \dots, k \end{aligned} \tag{24}$$

This test is designed to determine whether the effect of hallucination is mitigated or not, and whether it can be deemed negligible. By evaluating these hypotheses, we aim to establish if hallucinations in the models can be safely ignored.

➤ **RQ$_2$**: The dataset created through human-based modeling operations (i.e., D_1) is compared to the dataset generated by the LLM (i.e., D_2) and the mixed human-LLM dataset (i.e., D_{m05}), using the mean precision, recall, and F1 values. To represent the combinations considered, $opt \in D \times C$ refers to the Cartesian product of the datasets $D = \{D_1, D_2, D_{m05}\}$ and the configuration set $C = \{C_{1.1}, C_{1.2}, C_{1.3}, C_{2.1}, C_{2.2}, C_{2.3}, C_{3.1}, C_{3.2}, C_{3.3}\}$. Therefore, $H^{LLM,opt}$ measures the impact of the LLM-generated synthetic dataset on IMA performance compared to the human-generated dataset. The null hypothesis for **RQ$_2$** tests whether the mean values of precision, recall, and F1 are equal across all configurations and datasets.

$$\begin{aligned} H_0^{LLM,opt} &: \mu^{opt}_{D_1} = \mu^{opt}_{D_{m05}} = \mu^{opt}_{D_2} \\ H_A^{LLM,opt} &: \mu^{opt}_{D_1} \neq \mu^{opt}_{D_{m05}} \ \text{ OR } \ \mu^{opt}_{D_1} \neq \mu^{opt}_{D_2} \ \text{ OR } \\ & \quad \mu^{opt}_{D_{m05}} \neq \mu^{opt}_{D_2} \end{aligned} \tag{25}$$

Finally, we aim to determine if IMA performs better in certain use cases by comparing distributions and statistical values. The null hypothesis states that the mean IMA metric values for the different use cases are the same.

$$H_0^{IMA} : \mu^{IMA}_{UC_1} = \mu^{IMA}_{D_{UC2}} = \mu^{IMA}_{D_{UC3}} = \mu^{IMA}_{D_{UC4}} = \mu^{IMA}_{UC5} = \mu^{IMA}_{UC6} \tag{26}$$

The alternative hypothesis tests whether at least two UCs have different mean IMA metrics, as follows:

$$H_A^{IMA} : \forall i \neq j \ \exists \ i, j = 1, \dots, 6 \mid \mu^{IMA}_{UC_i} \neq \mu^{IMA}_{UC_j} \tag{27}$$

5.8. Experimental plan

To address the RQs, we designed an experiment plan to evaluate the effectiveness of LLMs in generating synthetic traces for MDE systems and their potential to replace human-generated traces. The experiments are described below.

For RQ_1, we conducted a **single-factor** experiment with **multiple treatments**, using datasets generated by different LLMs. We compared them using correctness, diversity, and hallucination metrics to see if LLM-generated traces differ significantly from human-generated ones. For RQ_2, we conducted a **multi-factor** experiment with **multiple treatments**, training MORGAN with human-generated traces (D_1), LLM-generated traces (D_2), and a mixed dataset (D_{m05}). The goal was to assess whether synthetic traces can replace human-generated ones in real-world scenarios. For the domain analysis, we evaluated the practical use of LLM-generated traces in real industrial settings. We conducted another **multi-factor** experiment with **multiple treatments** across various datasets, including mixed datasets (i.e., D_{m02}, D_{m05}, D_{m08}). The goal was to determine how well LLM-generated traces could be applied in industrial environments.

The experimental steps included: (1) collecting real models and traces, (2) generating synthetic datasets using LLMs, (3) applying correctness, diversity, and hallucination metrics, (4) training and evaluating MORGAN with different data configurations, and (5) validating the approach in real-world industrial use cases.

5.9. Data analysis

First, we use descriptive statistics, shown through boxplots, violin plots, and density plots, to assess metrics for LLM-generated synthetic trace data.

Next, data normality is evaluated using Q-Q plots and the Shapiro–Wilk test (significance level 0.05). If data is normally distributed, we use one-sample and independent t-tests to assess the hypothesis. If not, the non-parametric Wilcoxon-Mann–Whitney Rank-Sum test is used to evaluate potential differences between the samples. The data is independent due to the use of different LLMs, which also applies to the Wilcoxon test.

Additionally, to identify significant differences among metrics, we conducted Fisher's and Welch's One-Way ANOVA tests, which account for equal or varying variances among LLM groups, respectively. The one-way ANOVA is considered a robust test against the normality assumption. However, we choose to check the results from the one-way ANOVA with the nonparametric Kruskal–Wallis H Test, which does not require the assumption of normality. For a detailed analysis of metric group differences, we further utilized the *Games-Howell* test for post-hoc analysis.

5.10. Experiment execution

We ran our experiment, considering all the tools and the evaluation approaches, on a single Ubuntu 22.04.2 LTS (GNU/Linux 6.5.0-44-generic x86 64) with a PC equipped with an Intel® Xeon CPU E3-1225 v5 @ 3.30 GHz, 32 GB system memory, 128 KB LI cache, 1 MB L2 cache, and 8 MB L3 cache. The quality of the synthetic data has been assessed from the perspectives of diversity, correctness, and hallucination, measured using quantitative metrics through the *python-text-distance*[19] library version 0.1.6 and Python version 3.11.4. All the statistical analyses have been performed using Jamovi[20] software version 2.3.28.0.

[19] Pytextdist: https://pypi.org/project/pytextdist/.
[20] Jamovi: https://www.jamovi.org/.

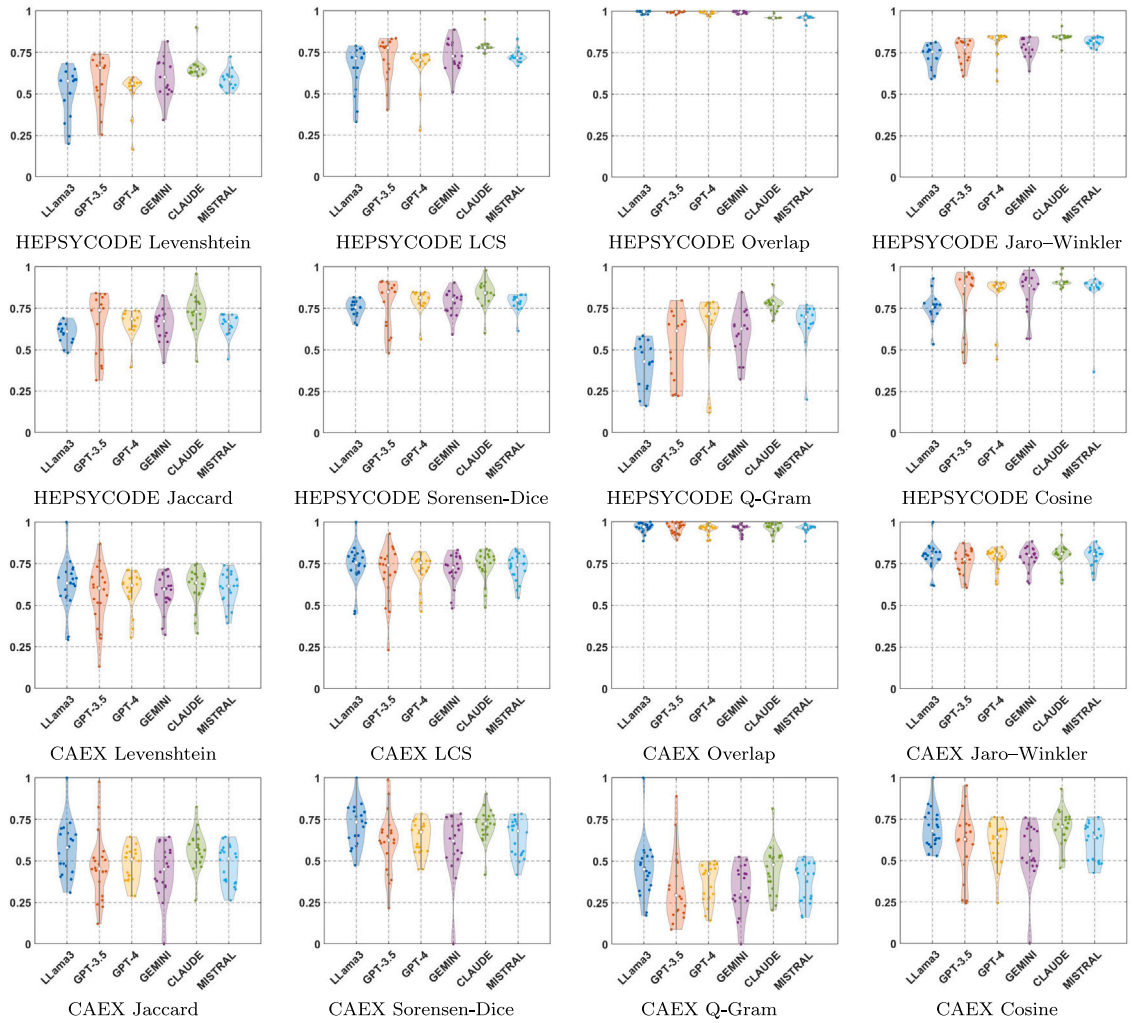


Fig. 6. Synthetic data quality evaluation diversity results for HEPSYCODE and CAEX. The violin plots show the distribution of points with the scatter plot. The small black squares in the center of the boxplots represent the mean.

6. Results

6.1. RQ_1: *What is the quality of the synthetic traces?*

To answer this research question, we evaluate the quality of the generated synthetic data using the considered LLMs and the evaluation approach presented in Section 5.

6.1.1. *Correctness evaluation*

To assess the correctness of the generated data, we apply two activities, prompt engineering and automatic trace evaluation.

Prompt engineering was conducted by prompt engineers in component ⑤ of Fig. 3. In this activity, prompt engineers selected input and output examples, analyzed the MSE context and task (synthetic trace creation), and refined prompts to ensure high-quality results with correct syntax, sufficient events, and no inaccuracies. The basic prompt structure in Fig. 4 was realized in a short time. Therefore, we proceeded to the human data cleaning activities in component ⑦ (e.g., remove unwanted text and/or characters and format the files).

The automatic validation was performed following Eq. (9) in component ⑧ (i.e., the Quality Checker). The mean values for all the LLM are greater than 0.95, while the 25% percentile and the 75% percentile belong to 1.00, and the confidence interval ranges from 0.984 to 1. To test the experimental hypothesis for correctness, we first check the normality assumptions and then apply the appropriate statistical test. The results for normality testing show that the p-value is lower than 0.001, so we can reject the null hypothesis of the normality of the dataset, and we use the one-sample Wilcoxon-Mann–Whitney Rank-Sum t-test. This test confirms that the correctness $C(\tau_j^+)$ is greater than 95% with p-values < 0.001 for all LLMs and MSEs under consideration. Therefore, we can reject the null hypothesis H_0^{Corr} and we can assess that all the LLMs can accurately emulate, from a syntactic point of view, the generation of reference traces under the considered configuration with a syntactic accuracy greater than 95%.

Finally, the check on semantic dependencies did not detect any violations, confirming that all the LLMs are capable of generating valid sequences of modeling operations.

6.1.2. *Diversity evaluation*

Concerning the diversity, we evaluate differences between τ_j^+ synthetic traces generated by LLMs and real τ_j traces generated by designers using the diversity metrics defined in Section 5.4.2, from Eq. (10) to Eq. (17). Fig. 6 presents the violin plot results with the point distributions related to each considered metric value. The plots show that all the LLMs behave similarly except for GPT-4, Claude, and Mistral, which have metric values closer to the median (i.e., lower variance) for the HEPSYCODE workbench.

To identify significant differences among metrics and test the experimental null hypothesis H_0^{Div}, we conducted Fisher's and Welch's One-Way ANOVA tests along with the Kruskal–Wallis H test, as stated in Section 5.9. This analysis reveals substantial differences in the Levenshtein,

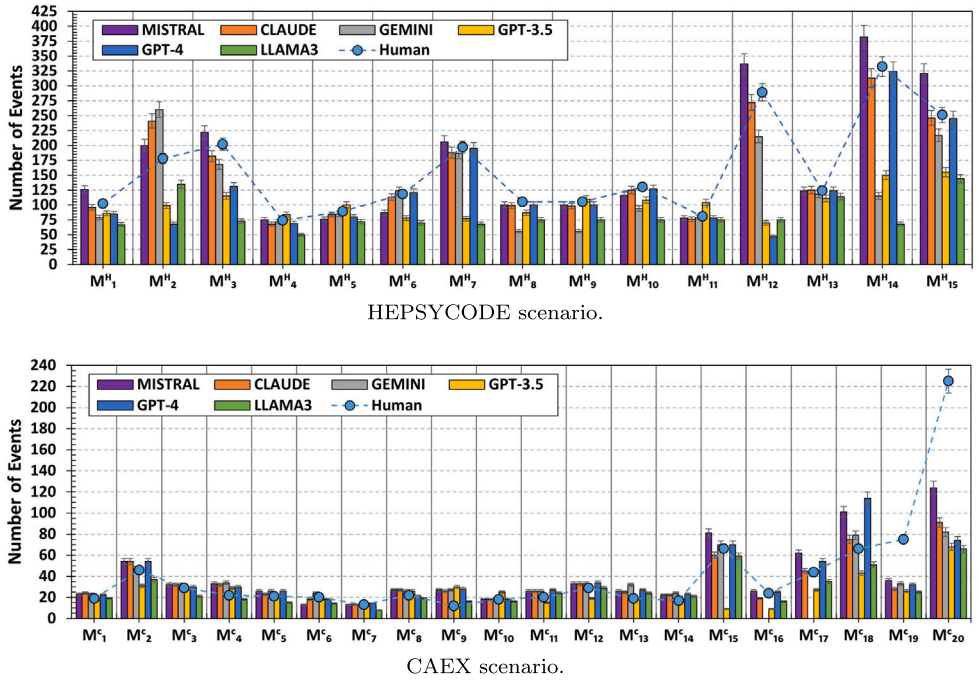


Fig. 7. Barplots comparing the number of human modeling operations and LLM-generated synthetic data events for each model.

LCS, Overlap, Jaro, Cosine, and Q-Gram metrics for HEPSYCODE and Cosine for CAEX across at least one LLM (p-values < 0.05).

For a detailed analysis of metric group differences, we conducted a post-hoc analysis using the Games-Howell test, which provided p-values for each metric group combination. The results indicate significant differences between pairs of LLMs with p-values less than 0.05. In the HEPSYCODE scenario, significant differences were observed between Claude and GPT-4, Claude and Llama 3, as well as Claude and Mistral for the Levenshtein and LCS metrics. For the Overlap metric, significant differences were found between Claude and all the other LLMs, as well as between Mistral and the other LLMs. Regarding the Jaro metric, significant differences were identified between Claude, Gemini, GPT-3.5, and Llama 3, and among Mistral, Llama 3, and GPT-3.5. For the Q-Gram metric, significant differences were observed between Claude, Gemini, GPT-3.5, and Llama 3, as well as between GPT-4 and Llama 3, and Mistral and Llama 3. Finally, the Cosine metric revealed significant differences between Claude and Llama 3. In the CAEX scenario, significant differences were identified in the Cosine[21] metric between Gemini and Llama 3. These findings align with the visual inspection of violin plots in Fig. 6 and suggest that the LLMs differ significantly across several metrics, especially in the EDA domain.

Finally, we want to determine which of the LLMs considered in the previous combinations is the best in terms of the mean metric values. To do this, we applied the non-parametric Wilcoxon-Mann–Whitney Rank-Sum test, as it is appropriate for comparing two independent groups when the assumptions of normality are not met. In the HEPSYCODE scenario, Claude outperforms GPT-4 in the Levenshtein, Q-gram, and Cosine metrics. Claude also outperforms Llama 3 and Mistral Large in the Levenshtein, Jaro, Q-gram, and Cosine metrics. Additionally, Claude outperforms Gemini and GPT-3.5 in the Jaro and Q-gram metrics. GPT-4 outperforms Claude and Mistral in the Overlap metric and outperforms Gemini and GPT-3.5 in the Jaro and Q-gram metrics. Moreover, GPT-4 also outperforms Llama 3 in the Jaro, Q-gram, and Cosine metrics. Mistral outperforms Gemini in the Q-gram metric, while Gemini surpasses Mistral in the Overlap metric. Mistral also outperforms GPT-3.5 in the Jaro and Q-gram metrics and demonstrates better performance compared to Llama 3 in the Jaro, Q-gram, and Cosine metrics. For the CAEX metrics, Gemini 1.5 Flash and GPT-3.5 outperform Llama 3 70B across Q-Gram and Cosine metrics. However, when comparing Claude 3.5 Sonnet, Mistral Large, GPT-3.5, GPT-4, and Gemini 1.5 Flash, there is not enough evidence to conclude that one LLM is definitively better or worse than another in terms of diversity metrics.

This analysis confirms that Claude, Mistral, and GPT-4, the most recent models with the highest number of parameters, outperform the other LLMs in the evaluated similarity measures within the HEPSYCODE EDA domain. Meanwhile, in the CAEX scenario, no significant differences were observed among the LLMs.

6.1.3. Hallucination evaluation

Before evaluating the effects of hallucination, we analyze the number of events generated by LLMs. Fig. 7 shows a barplot comparison of human modeling operations and LLM-generated synthetic events. The data indicates that as model complexity increases, reflected in a higher number of human modeling operations, LLMs tend to deviate more significantly from human patterns, leading to greater discrepancies and potential hallucinations. This divergence is particularly evident in highly complex models such as M^H_{12}, M^H_{14}, M^H_{15}, and M^C_{20}. These results highlight the importance of further research to identify the causes of these inconsistencies and to develop strategies for improving the alignment of LLM outputs with human behavior.

Furthermore, automatic evaluation of non-realistic events has been implemented considering the Hallucination metric defined in Eq. (18) in Section 5.4.3. Table 2 shows the statistics related to the hallucination metrics. The Confidence Interval (CI) of the mean assumes that sample means follow a t-distribution with N-1 degrees of freedom.

Based on Table 2, GPT-4 and Claude 3.5 Sonnet are the only LLMs with better values than the others for HEPSYCODE scenario, with an IQR equal to 0, and with SE, SD, and Variance lower than all other LLMs. Notably, for GPT-4, the hallucination metric stays at 1 up to the 95th percentile and increases only marginally to 1.02 at the 99th percentile, highlighting a significantly reduced impact of hallucination. In the CAEX scenario, GPT-3.5 and GPT-4 are the best-performing LLMs, with a median of 1.00, along with a moderate variance of 0.261

[21] Note that Kruskal–Wallis was higher than 0.05, but we still found some differences in at least two LLMs with the post-hoc analysis.

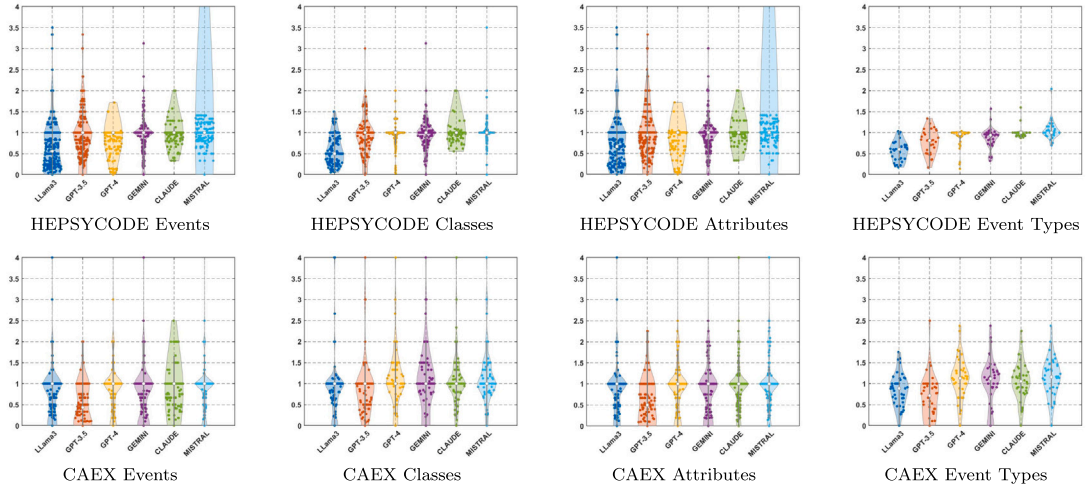


Fig. 8. Synthetic data quality evaluation hallucination results for HEPSYCODE and CAEX. The violin plots show the distribution of points in the scatter plot. The small black squares in the center of the box plots represent the mean.

Table 2
Event hallucination metric results.

HEPSYCODE											
LLM	N	μ	SE	CI-L	CI-U	Median	95th	99th	SD	Variance	IQR
Gemini	364	0.881	0.02400	0.834	0.929	1.000	1.64	2.00	0.458	0.2096	0.337
GPT-3.5	364	0.900	0.03422	0.823	0.967	1.000	2.00	3.00	0.653	0.4262	0.500
GPT-4	364	0.890	0.01391	0.863	0.918	1.000	1.00	1.02	0.265	0.0704	0.000
Llama 3	364	0.652	0.03062	0.592	0.713	0.558	1.50	2.69	0.584	0.3413	0.822
Claude	364	0.982	0.00783	0.966	0.997	1.000	1.00	1.42	0.149	0.0223	0.000
Mistral	364	1.017	0.03726	0.944	1.090	1.000	1.33	1.41	0.711	0.5052	0.000
CAEX											
LLM	N	μ	SE	CI-L	CI-U	Median	95th	99th	SD	Variance	IQR
Claude	214	0.868	0.0519	0.765	0.970	1.000	2.00	2.44	0.759	0.576	0.500
GPT-3.5	214	0.502	0.0349	0.433	0.570	0.437	1.12	2.00	0.511	0.261	1.000
GPT-4	214	0.809	0.0375	0.735	0.883	1.000	1.78	2.00	0.548	0.301	0.656
Llama 3	214	0.760	0.0462	0.669	0.851	1.000	1.78	2.87	0.676	0.457	0.741
Mistral	214	0.885	0.0601	0.766	1.003	1.000	2.00	4.94	0.880	0.774	0.550
Gemini	214	0.777	0.0538	0.671	0.883	1.000	2.00	3.81	0.787	0.619	1.000

μ: Mean; SE: Standard Error; SD: Standard Deviation; IQR: Interquartile range; CI-L: 95% Confidence Interval Lower Bound; CI-U: 95% Confidence Interval Upper Bound; 95th: 95% Percentile; 99th: 99% Percentile;

and 0.301, respectively. Furthermore, GPT-3.5 and GPT-4 exhibit the lowest values at both the 95th and 99th percentiles, suggesting that they are the best LLMs to mitigate the effects of hallucination in the CAEX scenario.

The results for normality testing show that the p-value is lower than 0.001 for all the LLM under consideration, so we can reject the null hypothesis of the normality of the dataset, and we use the one-sample Wilcoxon-Mann–Whitney Rank-Sum t-test to address H_0^{Hall}. Applying the Wilcoxon *One Sample T-Test* for all the LLMs and the HEPSYCODE and CAEX scenarios, we get a p-value < 0.001, allowing us to reject H_0^{Hall}. This result confirms that the mean number of synthetic modeling operations generated by the LLMs is statistically significantly less than 1. This finding suggests that, on average, the effects of hallucination are mitigated, demonstrating a closer alignment between LLM outputs and human behavior.

Finally, Fig. 8 presents the synthetic data quality evaluation hallucination results for HEPSYCODE and CAEX, considering single events, classes, attributes, and specific types of modeling operations (i.e., ADD and SET). The boxplots confirm previous findings, showing that GPT-4 and Claude are the LLMs that perform best in mitigating the effects of hallucination at least in the HEPSYCODE scenario.

We also calculate the total evaluation time by summing the time needed to complete each activity, from creating a domain-specific metamodel to generating a final dataset for training IMAs. Moreover, using the PC configuration presented in Section 5.10, the total time for modeling and trace collection in MSE was approximately 150 min, while the time for generating models using the six LLMs and performing validation was approximately 25 min.

6.2. RQ_2: *Is synthetic datasets useful for training a lightweight IMA?*

LLM analysis. To obtain D_2, we use the in-context few-shots learning setting described in Section 4 by relying on GPT-4, as it represents one of the best models according to the conducted evaluation in the previous research question. In addition, we analyze the recommendation capabilities in suggesting operations that affect (i) classes and (ii) attributes, resembling the original MORGAN experiment in Di Sipio et al. [43]. Fig. 9 shows the results obtained by MORGAN in recommending class and attribute operations considering the three datasets and the nine configurations in HEPSYCODE and CAEX scenarios.

To identify significant differences among configurations and test the experimental null hypothesis $H_0^{LLM,opt}$, we conducted Fisher's and Welch's One-Way ANOVA tests along with the Kruskal–Wallis H test, as stated in Section 5.9. This analysis reveals substantial differences in the configuration for HEPSYCODE and CAEX across at least one configuration (p-values < 0.001). These findings align with the visual inspection of box plots in Fig. 9.

For a detailed analysis of configuration differences, we conducted a post-hoc analysis using the Games-Howell test, which provided p-values

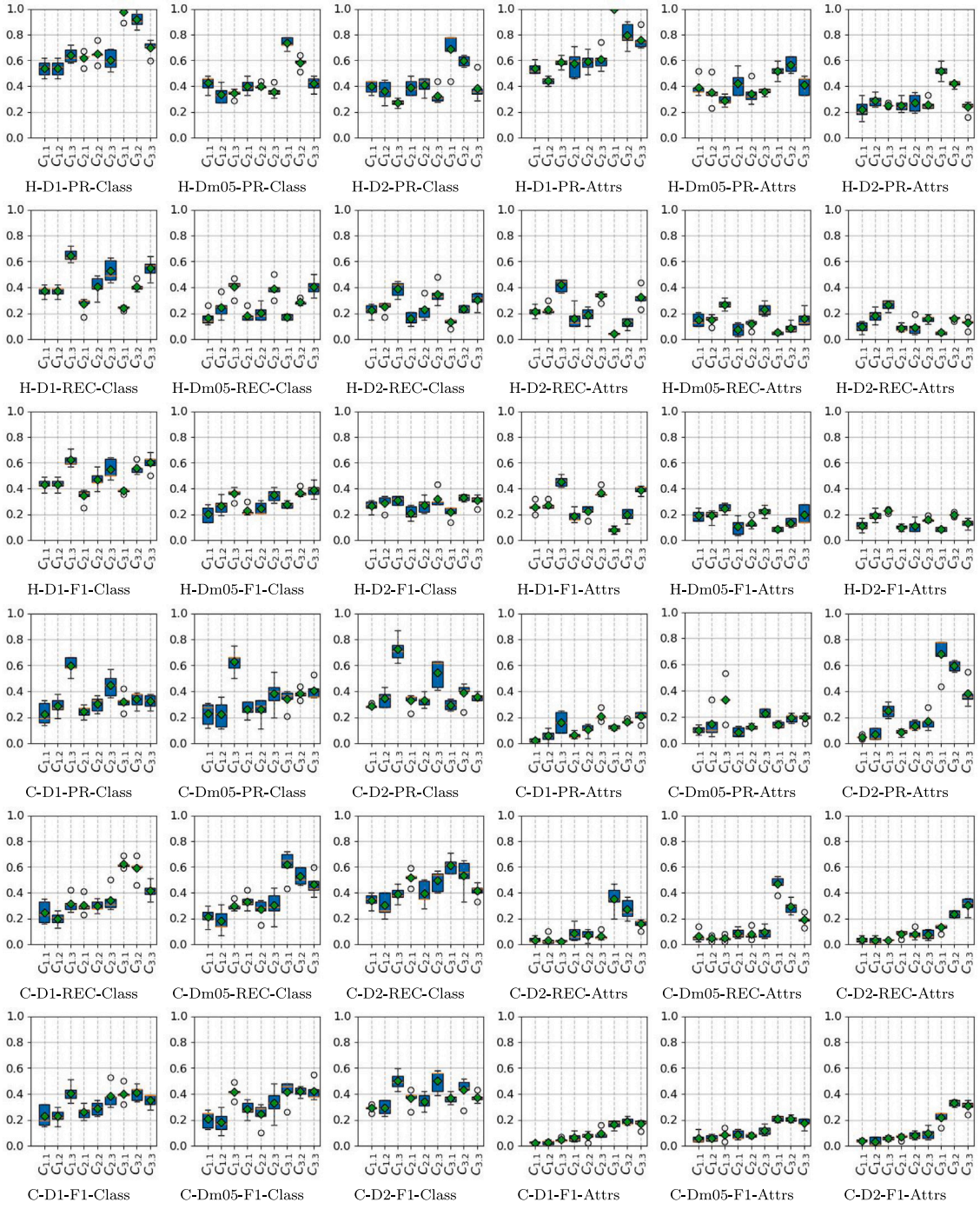


Fig. 9. Recommendations results for HEPSYCODE (H-) and CAEX (C-). The green diamond-shaped marker represents the mean.

for each metric group combination. All results are freely available in the replication package repository[2].

Specifically, in recommending *class* operations, the system performs better when real traces are used across all the configurations considered, as presented in Section 5. In particular, configurations $C_{3.1}$, $C_{3.2}$, and $C_{3.3}$ result in better performance. Moreover, while the value obtained by MORGAN using synthetic traces is slightly less accurate compared to the human dataset for HEPSYCODE, there is an improvement in performance for CAEX when using synthetic data compared to real traces.

Nonetheless, we report that the results obtained using the synthetic datasets are similar to the ones obtained in the original MORGAN paper in which the same algorithm has been tested using a curated modeling dataset, i.e., ModelSet [13]. While we acknowledge that the purpose is different, i.e., it was used to support model completion, the study highlights the importance of curating and preprocessing training data for ML-based IMAs.

Intuitively, powerful LLMs like GPT-4 can be used to generate modeling operations when real traces are not available. This claim is confirmed by analyzing the results obtained for D_{m05} where the results are slightly increased compared to synthetic traces. On the one hand, we report that using only half of the real traces has limited impact. On the other hand, a small portion of synthetic data can contribute to enabling IMAs focused on recommending modeling operations. It is worth noting that we consider MORGAN as the IMA component in this paper. Thus, we anticipate that using other IMAs can lead to better

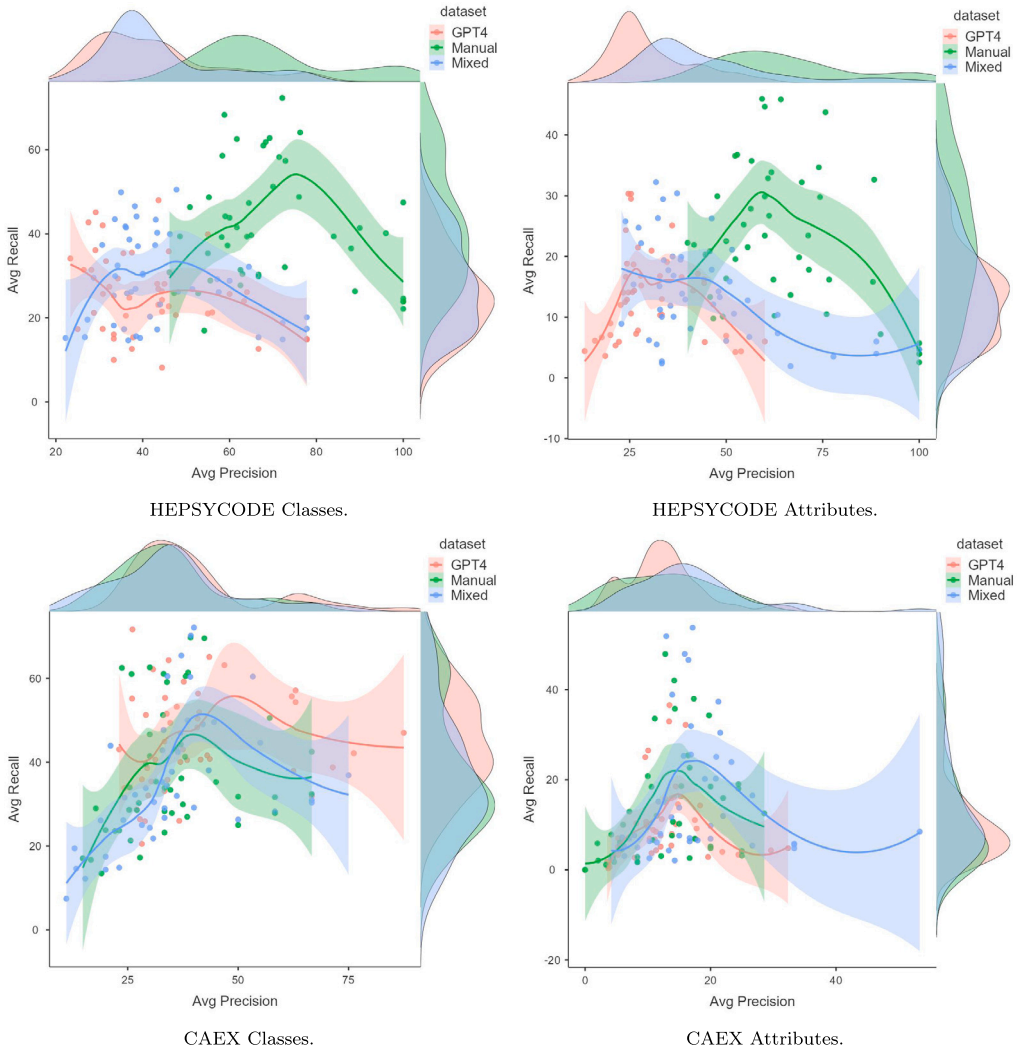


Fig. 10. Total recommendations results scatterplot for the different dataset D1 (Manual), D2 (GPT4) and Dm05 (Mixed).

performance (e.g., NEMO by Di Rocco et al. [14]). Concerning the configurations, we report that increasing both the CR and CO values contributes to increasing the overall performance of MORGAN. This is quite expected since the MORGAN's engine is based on a graph kernel similarity [47] that benefits from more training data.

A similar trend in performance accuracy can be observed for recommending *attribute* operations summarized in Fig. 9. As expected, the accuracy is lower compared to class operations due to the higher variability in defining modeling operations. Concerning the impact of synthetic operations, the results confirm that D_1 offers better performance for HEPSYCODE, even though the delta between the synthetic data is lower than the class recommendations, while synthetic datasets lead to better MORGAN performances in the CAEX scenario. It is worth mentioning that the obtained performance is in line with state-of-the-art IMAs [32] used in modeling completion tasks. Furthermore, we report that the NEMO approach by Di Rocco et al. [14], the most relevant approach to our work, achieves 0.60 in accuracy on a curated dataset. In this respect, our approach employs traces extracted from real-world models.

Fig. 10 shows a scatterplot of total recommendations across different datasets, along with the distributions of Precision and Recall. LOESS regression lines [59] are added to highlight trends, with shaded areas representing standard error variability. In the CAEX scenarios, the addition of synthetic data, especially in combination with real data, improves overall performance, as shown by Precision distributions, Recall distributions, and trend lines. Contrarily, in the HEPSYCODE scenario, performance decreases with synthetic data, though results remain within an acceptable range. This highlights the importance of meta-model complexity and scenario-specific considerations when integrating synthetic data, as its impact can vary depending on the underlying characteristics of the datasets and the specific meta-model.

Finally, we evaluated the time needed to compute the training and testing phase considering only the MORGAN tool, thus excluding the time to generate the traces. Overall, the time to load and encode the training traces is equal to 0.07 s on average for each fold, while 7 s are required to perform the recommendation phase. In addition, we analyze the time required to generate the synthetic traces. By running a prompt using the chatGPT service, we report that the generation of a single trace takes approximately 0.73 s.

Domain analysis. To evaluate the impact of different application domains, we run MORGAN on Dataset D_3 using the previous datasets as training, i.e., D_1, $Dm02$, $Dm05$, $Dm08$, and D_2 considering the novel CAEX MSE.

Concretely, we used D_3 datasets as the *validation set* of our approach, aiming to evaluate how the produced data can be used in different application domains. Fig. 11 depicts the results obtained for HEPSYCODE and CAEX. Notably, we used configuration $C_{3.3}$ to compute the results, as it is one of the configurations that offers the best performance, as shown in the previous research question.

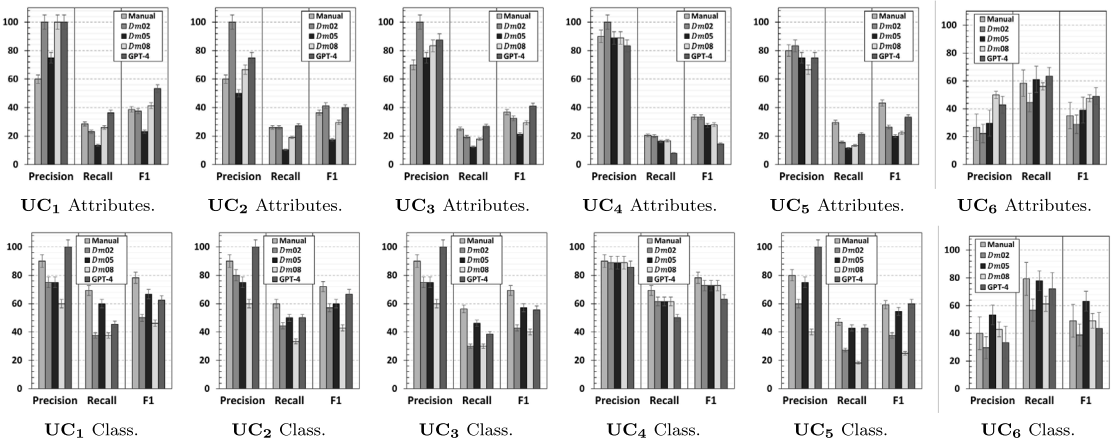


Fig. 11. Validation results for the considered MSEs on dataset D_3 for the different dataset D1 (Manual), D2 (GPT4), Dm02, Dm05, and Dm08 (Mixed).

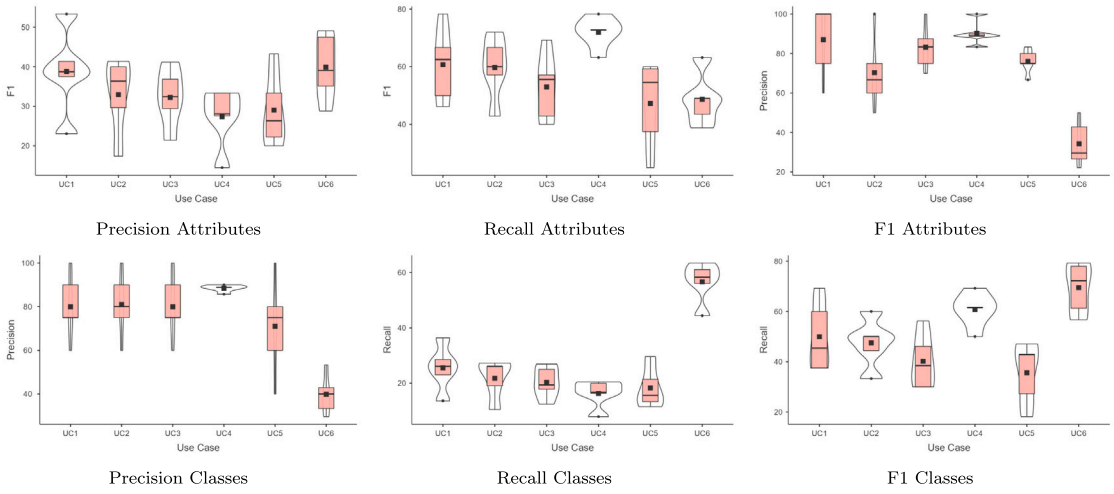


Fig. 12. Synthetic Data IMA Metric Results for CAEX and HEPSYCODE into the industrial domain. The violin plots show the distribution of points with the scatter plot. The small black squares in the center of the boxplots represent the mean.

In contrast to the results of RQ_2, the human-based traces are not the best training set for MORGAN, while the usage of synthetic data improves the whole performance for recommended operations in different industrial contexts. In addition, we see that the operations generated by the GPT-4 model tend to decrease the recall values, while human-generated traces have higher precision values on average in the CAEX and HEPSYCODE scenarios. Therefore, mixing human and synthetic traces can represent an adequate trade-off to reduce the number of false negatives. This is confirmed by analyzing the results obtained with the novel datasets, i.e., *Dm*02 and *Dm*08. Concerning the time needed for the recommendations, it is worth noticing that the training time is reduced from 7 to 2 s on average as we are using MORGAN pre-trained weights to perform the recommendations activity. Therefore, the computation times are similar to those presented in RQ_2 to perform the testing phase.

Concerning the experimental hypotheses, we evaluate differences between use cases using the IMA metrics defined in Section 5.5. Fig. 12 presents the violin plot results with the point distributions related to each considered IMA metric value. The plots show that all the UCs behave similarly except for UC_4 and UC_6, which have metric values closer to the median (i.e., lower variance) for the HEPSYCODE and CAEX workbench.

To identify significant differences among metrics and test the experimental null hypothesis H_0^{IMA}, we conducted Fisher's and Welch's One-Way ANOVA tests along with the Kruskal–Wallis H test, as stated in Section 5.9. This analysis reveals substantial differences in precision and recall for class and attributes (p-value < 0.05 for all the cases), while F1 has substantial differences in class but not in attributes.

For a detailed analysis of metric group differences, we conducted a post-hoc analysis using the Games-Howell test, which provided p-values for each metric group combination. Specifically, in the *attributes* scenario, significant differences were observed in the precision of UC_6 compared to UC_1, UC_3, UC_4, and UC_5, and in the recall of UC_6 compared to UC_1, UC_2, UC_3, UC_4, and UC_5. Therefore, in the *classes* scenario, significant differences were observed in the precision of UC_6 compared to UC_1, UC_2, UC_3, and UC_4, in the recall of UC_6 compared to UC_3 and UC_5, and in the F1 of UC_6 compared to UC_4. There were no significant differences among the F1 in the attributes scenario.

Finally, we want to determine which of the UCs considered is the best in terms of the IMA mean metric values. To do this, we applied the non-parametric Wilcoxon-Mann–Whitney Rank-Sum test, as it is appropriate for comparing two independent groups when the assumptions of normality are not met. In the *attributes* scenario, UC_6 is outperformed by all other UCs in precision. Therefore, UC_1 outperforms UC_4 in recall and F1, while UC_4 outperforms UC_5 in precision. Finally, UC_6 outperforms UC_1, UC_2, UC_3, UC_4 in recall, while UC_6 outperforms UC_4 and UC_5 in F1. In the *classes* scenario, UC_6 is outperformed by all other UCs in precision, and by UC_4 in F1. Therefore, UC_6 outperforms UC_1, UC_2, UC_3, and UC_5 in recall.

7. Discussion

In this section, we present a discussion of the findings from our study, specifically addressing the research questions outlined earlier (i.e., RQ_1, RQ_2, and RQ_3).

7.1. Answer and reflections on RQ_1

The results in Section 6.1 show that the traces generated by the LLMs are very similar to those generated by a human, especially considering GPT-4 and Claude, which have the best statistical results and reduced hallucination effects in the EDA domain. In the CAEX scenario, no differences were observed in the results across the different LLMs, except for Llama 3 70B, which shows the worst performance. Moreover, the hallucination metric is never mitigated in the CAEX scenario, so this perturbation in the dataset must be taken into account. It is worth noting that, although the effects of hallucination are mitigated, the model complexity introduces challenges related to the coverage of the model itself and the correct number of events generated by LLMs. These issues could potentially impact the performance of recommender systems. However, these results demonstrate that LLMs can be used as tools to emulate modeling operations that a human could perform, offering a fairly reliable level of accuracy. This suggests a promising potential for LLMs to support designers in modeling activities in various complex domains.

Summary of RQ_1: Our analysis reveals that GPT-4 and Claude are the best models considering the EDA domain, outperforming open-source and other proprietary models, while Llama 3 70B shows the worst results in all scenarios. However, most LLMs still produce adequate traces and can represent an alternative way to produce modeling operations.

7.2. Answer and reflections on RQ_2

The analysis of our results in Section 6.2 provides significant insights into the use of LLM-generated synthetic datasets for training an IMA. Specifically, the impact of synthetic datasets appears to vary depending on the complexity of the domain and the comprehensiveness of the available real datasets.

For the HEPSYCODE workbench, the real dataset was extensive enough to cover the HEPSYCODE metamodel's classes and attributes. The results suggest that synthetic datasets did not significantly enhance IMA performance. This is demonstrated by the hallucinations, diversity, and correctness metrics, where synthetic traces generated by GPT-4 closely match previous real-world outcomes. Thus, LLM-generated synthetic traces may come in handy where the real dataset is comprehensive and fully represents the domain's complexity. However, synthetic traces can expand the training set more quickly than human-based modeling activities while maintaining the accuracy of other state-of-the-art tools.

In contrast, the CAEX environment presented a more complex domain, characterized by richer and more intricate syntax and semantics, allowing the definition of various application domains. Here, the use of LLM-generated synthetic traces proved beneficial as it addressed gaps not covered by the real dataset, improving IMA performance. This is confirmed by the hallucination metric, which consistently exceeded a value of 2 across all LLMs used. The higher hallucination metric indicates that LLMs were able to produce a greater number of events and modeling operations than those present in the real dataset. Despite the divergence from real-world data, these synthetic traces were syntactically accurate and did not introduce bias into the IMA. Instead, they contributed to enhanced overall performance by covering a wider range of modeling operations.

The domain analysis shows that UC_6 has the lowest precision but the highest recall. Furthermore, when comparing UC_6 to UC_5 (i.e., from the automotive domain), similar outcomes are observed across other UCs, indicating that performance is not influenced by the application domain (e.g., UC_5 and UC_6), but rather by the specific MSE, the underlying metamodel, and the complexity of the domain.

The findings also reveal that, for CAEX, the use of a synthetic dataset reduces precision but enhances tolerance to false negatives. The high precision and low recall observed in HEPSYCODE, contrasted with the low precision and high recall in CAEX, are attributed to the dataset size and domain complexity. In the case of HEPSYCODE, we utilized a large, real-world dataset with several case studies that covered the entire metamodel. The results suggest that while the synthetic dataset improves precision, it does so at the cost of recall. However, this trade-off is acceptable, as the outcomes are consistent with previous real-world studies.

In contrast, for CAEX, the greater complexity of the domain prevented us from fully covering the metamodel. Consequently, the synthetic models generated were unable to capture the full complexity, and the use of synthetic data did not enhance the IMA's precision. This is because the synthetic traces introduced bias into the dataset, making it less suitable for real-world industrial applications. In practice, LLMs generate synthetic traces that are syntactically accurate but often include cases not present in real data, i.e., high hallucination rates, leading to the generation of additional events and modeling operations. As a result, synthetic datasets differ significantly from real ones, and in industrial settings, the introduction of unrealistic traces decreases precision, although it does reduce the likelihood of false negatives by increasing recall.

Concretely, the provided recommendations might be useful in different application domains, thus assisting modelers in different real-world scenarios. In addition, we demonstrate that our approach can be used in different application contexts, thus representing a suitable alternative to cope with **CH3** and **CH4** discussed in Section 2. Concretely, IMAs can be trained with synthetic traces that are not identical but similar to the target ones, thus overcoming privacy issues in an industrial context.

Summary of RQ_2: The examined LLMs are able to generate synthetic traces that are comparable to human ones. Therefore, LLMs can help mitigate training data scarcity for IMAs by generating synthetic datasets, though their impact depends on the domain's complexity and the completeness of the real dataset. These synthetic traces can quickly expand the training set, covering a wider range of operations while preserving accuracy, even if they differ somewhat from real-world data. The conducted evaluation shows that the impact of the application domain on the IMA performances is minimal compared to other factors, such as the metamodel structure and the domain complexity. Across different domains, similar performance trends were observed, indicating that domain-specific factors are less significant. In addition, our approach can be used to support the specification of IMAs when training data are missing, thus overcoming the lack of data due to privacy issues.

8. Limitations

This section discusses the limitations of the proposed framework. A first limitation concerns the possible hallucination of incorrect operations, which can lead the IMA component to process unsuitable data. Although we mitigated this risk by experimenting with six popular LLMs and evaluating the generated data with well-founded metrics, hallucination remains an intrinsic challenge. Additionally, we did not address the removal of operations, which could introduce biases in the computation of the hallucination metric.

Another aspect relates to the evaluation of the IMA component itself. The computed metrics might lead to inaccuracies. To reduce this risk, we designed three different configurations simulating varying levels of completion and validated the outcomes through a 5-fold

cross-validation using established statistical indices. All experiments were randomized and conducted in an isolated environment, allowing repeated executions with identical prompts across different LLMs to ensure accurate comparisons.

The generalizability of our framework is also a concern, as results may vary when considering different modeling tools. We addressed this partially by validating the approach using distinct, well-established modeling components for each architectural block (i.e., the modeling environment, the trace recorder, and the IMA assistant), and by exploiting an additional modeling dataset from several EU projects, thus covering different domain applications, as discussed in RQ_2. Nonetheless, further validations on a broader set of modeling tools and application domains are needed.

Moreover, reconstructing complete models from the generated traces is currently infeasible, as the traces lack essential semantic information necessary to capture the full model structure and behavior, as shown in Section 4.2. Although our diversity metrics assess structural variety and alignment, they do not guarantee end-to-end consistency. Enriching traces with semantic details is a promising direction for future work [17].

Another potential improvement concerns the learning strategy. While our approach relies on prompt-based in-context learning, fine-tuning LLMs on domain-specific, high-quality traces could enhance accuracy and reasoning capabilities. However, this is constrained by the scarcity of publicly available traces, especially in industrial contexts. Future work will explore LLM fine-tuning strategies to overcome this limitation [60].

Finally, we used the Goal-Question-Metric (GQM) approach to prevent inadequate pre-operational explication of constructs. We addressed mono-operation bias by introducing multiple variations of independent variables across RQ_1 and RQ_2, including different LLMs, dataset configurations, and a mix of human-generated and LLM-generated datasets. This approach provided a broader representation and enhanced the generalizability of our results. To mitigate mono-method bias, we applied a range of metrics across experiments, i.e., multiple metrics and multi-factor experiments using mixed datasets.

Overall, when analyzing the results of our experiment, we used the appropriate statistical tests to prevent any errors, biases, or violated assumptions. Additionally, we created a replication package to enable others to reproduce our work and confirm our methods and results. Despite all these efforts, residual limitations remain and call for further empirical replication.

9. Conclusion

This paper proposes MASTER-LLM, a conceptual framework to support automated activities in the context of MDE, leveraging modeling event recorders, intelligent modeling assistants, and large language models. In particular, we used prominent LLMs to generate synthetic traces using an in-context few-shot prompt engineering strategy, aiming at resembling human-style operations. The findings of the study demonstrate that LLMs can be used to generate traces in a specific format, even though the evaluated assistant suffers from degradation of performance when delivering recommendations.

In future work, we plan to investigate how synthetic traces impact qualitative aspects such as internal regulations, privacy concerns, and security issues. Additionally, we aim to generate multiple traces from the same model to evaluate how this affects the overall metrics and performances (e.g., using different prompt formats). Furthermore, we intend to include additional modeling assistants and LLMs that account for long-range temporal dependencies. Last but not least, we will deploy MASTER-LLM by integrating the complete set of components and collecting feedback from modelers.

CRediT authorship contribution statement

Vittoriano Muttillo: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Claudio Di Sipio:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Riccardo Rubei:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Luca Berardinelli:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL-JU) through the project AIDOaRt, grant agreement No. 101007350, and the Key Digital Technologies Joint Undertaking (KDT-JU) through the project MATISSE, grant agreement No. 101140216. We also thank Thalés France, Thales Alenia Space España, Integrasys, Volvo Construction Equipment, and TEKNE companies.

Data availability

Freely available on GitHub: https://github.com/hepsycode/MASTER-LLM-IST.

References

[1] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang, Large language models for software engineering: A systematic literature review, ACM Trans. Softw. Eng. Methodol. 33 (8) (2024) http://dx.doi.org/10.1145/3695988.

[2] A.L. Ramos, J.V. Ferreira, J. Barceló, Model-based systems engineering: An emerging approach for modern systems, IEEE Trans. Syst. Man Cybern. C (Appl. Rev.) 42 (1) (2011) 101–111.

[3] J. Di Rocco, D. Di Ruscio, C. Di Sipio, P.T. Nguyen, R. Rubei, On the use of large language models in model-driven engineering, Softw. Syst. Model. (2025) http://dx.doi.org/10.1007/s10270-025-01263-8.

[4] J. Cámara, J. Troya, L. Burgueño, A. Vallecillo, On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML, Softw. Syst. Model. 22 (3) (2023) 781–793, http://dx.doi.org/10.1007/s10270-023-01105-5.

[5] J. Cámara, L. Burgueño, J. Troya, Towards standardized benchmarks of LLMs in software modeling tasks: a conceptual framework, Softw. Syst. Model. 23 (6) (2024) 1309–1318, http://dx.doi.org/10.1007/s10270-024-01206-9.

[6] G. Mussbacher, B. Combemale, J. Kienzle, S. Abrahão, H. Ali, N. Bencomo, M. Búr, L. Burgueño, G. Engels, P. Jeanjean, J.-M. Jézéquel, T. Kühn, S. Mosser, H. Sahraoui, E. Syriani, D. Varró, M. Weyssow, Opportunities in intelligent modeling assistance, Softw. Syst. Model. 19 (5) (2020) 1045–1053, http://dx.doi.org/10.1007/s10270-020-00814-5.

[7] V. Muttillo, C. Di Sipio, R. Rubei, L. Berardinelli, M. Dehghani, Towards synthetic trace generation of modeling operations using in-context learning approach, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 619–630, http://dx.doi.org/10.1145/3691620.3695058.

[8] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, MyMediaLite: a free recommender system library, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 305–308, http://dx.doi.org/10.1145/2043932.2043989.

[9] V. Muttillo, L. Pomante, M. Santic, G. Valente, Systemc-based Co-Simulation/Analysis for system-level Hardware/Software Co-Design, Comput. Electr. Eng. 110 (2023) 108803, http://dx.doi.org/10.1016/j.compeleceng.2023.108803.

[10] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, X. Ning, Y. Ma, H. Yang, B. Yu, H. Yang, Y. Wang, Machine learning for electronic design automation: A survey, ACM Trans. Des. Autom. Electron. Syst. 26 (5) (2021) 1–46, http://dx.doi.org/10.1145/3451179.

[11] F. Vahid, T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction, first ed., John Wiley & Sons, Inc., USA, 2001.

[12] J. Cederbladh, L. Berardinelli, H. Bruneliere, A. Cicchetti, M. Dehghani, C. Di Sipio, J. Miranda, A. Rahimi, R. Rubei, J. Suryadevara, Towards automating model-based systems engineering in industry - An experience report, in: 2024 IEEE International Systems Conference (SysCon), 2024, pp. 1–8, http://dx.doi.org/10.1109/SysCon61195.2024.10553610.

[13] J.A.H. López, J.L. Cánovas Izquierdo, J.S. Cuadrado, ModelSet: a dataset for machine learning in model-driven engineering, Softw. Syst. Model. 21 (3) (2021) 967–986, http://dx.doi.org/10.1007/s10270-021-00929-3.

[14] J. Di Rocco, C. Di Sipio, P.T. Nguyen, D. Di Ruscio, A. Pierantonio, Finding with NEMO: a recommender system to forecast the next modeling operations, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, MODELS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 154–164, http://dx.doi.org/10.1145/3550355.3552459.

[15] V. Garousi, K. Petersen, B. Ozkan, Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review, Inf. Softw. Technol. 79 (2016) 106–127, http://dx.doi.org/10.1016/j.infsof.2016.07.006.

[16] E.P. Enoiu, A. Čaušević, Enablers and impediments for collaborative research in software testing: an empirical exploration, in: Proceedings of the 2014 International Workshop on Long-Term Industrial Collaboration on Software Engineering, WISE '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 49–54, http://dx.doi.org/10.1145/2647648.2647655.

[17] M. Dehghani, L. Berardinelli, M. Wimmer, Towards modeling process mining for graphical editors, in: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2023, pp. 929–933, http://dx.doi.org/10.1109/MODELS-C59198.2023.00146.

[18] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks, EMF: Eclipse Modeling Framework 2.0, second ed., Addison-Wesley Professional, 2009.

[19] IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams, IEEE Std 1849- 2023 (Revision of IEEE Std 1849-2016), 2023, pp. 1–55, http://dx.doi.org/10.1109/IEEESTD.2023.10267858.

[20] M. Herrmannsdoerfer, M. Koegel, Towards a generic operation recorder for model evolution, in: Proceedings of the 1st International Workshop on Model Comparison in Practice, IWMCP '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 76–81, http://dx.doi.org/10.1145/1826147.1826161.

[21] P. Brosch, G. Kappel, P. Langer, M. Seidl, K. Wieland, M. Wimmer, An introduction to model versioning, in: Formal Methods for Model-Driven Engineering, Springer Berlin Heidelberg, 2012, pp. 336–398, http://dx.doi.org/10.1007/978-3-642-30982-3_10.

[22] P. Pietsch, H.S. Yazdi, U. Kelter, Generating realistic test models for model processing tools, in: 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), 2011, pp. 620–623, http://dx.doi.org/10.1109/ASE.2011.6100140.

[23] C.A.G. Pérez, J. Cabot, Test Data Generation for Model Transformations Combining Partition and Constraint Analysis, 8568, 2014, p. 25, http://dx.doi.org/10.1007/978-3-319-08789-4_3.

[24] S. Sen, B. Baudry, Mutation-based model synthesis in model driven engineering, in: Second Workshop on Mutation Analysis (Mutation 2006 - ISSRE Workshops 2006), IEEE Computer Society, Los Alamitos, CA, USA, 2006, p. 13, http://dx.doi.org/10.1109/MUTATION.2006.12.

[25] A. Mougenot, A. Darrasse, X. Blanc, M. Soria, Uniform random generation of huge metamodel instances, in: R.F. Paige, A. Hartman, A. Rensink (Eds.), Model Driven Architecture - Foundations and Applications, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2009, pp. 130–145, http://dx.doi.org/10.1007/978-3-642-02674-4_10.

[26] G. Soltana, M. Sabetzadeh, L.C. Briand, Practical constraint solving for generating system test data, ACM Trans. Softw. Eng. Methodol. 29 (2) (2020) http://dx.doi.org/10.1145/3381032.

[27] C. Xu, D. Guo, N. Duan, J. McAuley, Baize: An open-source chat model with parameter-efficient tuning on self-chat data, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 6268–6278, http://dx.doi.org/10.18653/v1/2023.emnlp-main.385.

[28] W. Zeng, C. Xu, Y. Zhao, J.-G. Lou, W. Chen, Automatic instruction evolving for large language models, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 6998–7018, http://dx.doi.org/10.18653/v1/2024.emnlp-main.397.

[29] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N.A. Smith, D. Khashabi, H. Hajishirzi, Self-instruct: Aligning language models with self-generated instructions, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 13484–13508, http://dx.doi.org/10.18653/v1/2023.acl-long.754.

[30] J.A.H. López, M. Földiák, D. Varró, Text2VQL: Teaching a model query language to open-source language models with ChatGPT, in: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, MODELS '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 13–24, http://dx.doi.org/10.1145/3640310.3674091.

[31] L. Burgueño, J. Cabot, S. Li, S. Gérard, A generic LSTM neural network architecture to infer heterogeneous model transformations, Softw. Syst. Model. 21 (1) (2021) 139–156, http://dx.doi.org/10.1007/s10270-021-00893-y.

[32] J. Di Rocco, D. Di Ruscio, C. Di Sipio, P.T. Nguyen, A. Pierantonio, MemoRec: a recommender system for assisting modelers in specifying metamodels, Softw. Syst. Model. 22 (1) (2022) 203–223, http://dx.doi.org/10.1007/s10270-022-00994-2.

[33] B. Adhikari, E.J. Rapos, M. Stephan, SimIMA: a virtual simulink intelligent modeling assistant: Simulink intelligent modeling assistance through machine learning and model clones, Softw. Syst. Model. 23 (1) (2023) 29–56, http://dx.doi.org/10.1007/s10270-023-01093-6.

[34] I. Ludovico, A. Barriga, A. Rutle, R. Heldal, Model repair with Quality-Based reinforcement learning, in: R. Paige, A. Vallecillo (Eds.), J. Obj. Technol. 19 (2) (2020) 17:1, http://dx.doi.org/10.5381/jot.2020.19.2.a17.

[35] M. Weyssow, H. Sahraoui, E. Syriani, Recommending metamodel concepts during modeling activities with pre-trained language models, Softw. Syst. Model. 21 (3) (2022) 1071–1089, http://dx.doi.org/10.1007/s10270-022-00975-5.

[36] L. Wang, Y. Lepage, Learning from masked analogies between sentences at multiple levels of formality, Ann. Math. Artif. Intell. (2023) 1–25, http://dx.doi.org/10.1007/s10472-023-09918-2.

[37] M.B. Chaaben, L. Burgueño, H. Sahraoui, Towards using few-shot prompt learning for automating model completion, in: Proceedings of the 45th International Conference on Software Engineering: New Ideas and Emerging Results, in: ICSE-NIER '23, IEEE Press, Melbourne, Australia, 2023, pp. 7–12, http://dx.doi.org/10.1109/ICSE-NIER58687.2023.00008.

[38] V. Kulkarni, S. Reddy, S. Barat, J. Dutta, Toward a symbiotic approach leveraging generative AI for model driven engineering, in: 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2023, pp. 184–193, http://dx.doi.org/10.1109/MODELS58315.2023.00039.

[39] L. Apvrille., B. Sultan., System architects are not alone anymore: Automatic system modeling with AI, in: Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering - MODELSWARD, SciTePress, INSTICC, 2024, pp. 27–38, http://dx.doi.org/10.5220/0012320100003645.

[40] B. Romera-Paredes, P.H.S. Torr, An embarrassingly simple approach to zero-shot learning, in: R.S. Feris, C. Lampert, D. Parikh (Eds.), Visual Attributes, Springer International Publishing, Cham, 2017, pp. 11–30, http://dx.doi.org/10.1007/978-3-319-50077-5_2.

[41] X. Li, S. Yuan, X. Gu, Y. Chen, B. Shen, Few-shot code translation via task-adapted prompt learning, J. Syst. Softw. 212 (2024) 11, http://dx.doi.org/10.1016/j.jss.2024.112002.

[42] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E.H. Chi, Q.V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS), NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2022.

[43] C. Di Sipio, J. Di Rocco, D. Di Ruscio, P.T. Nguyen, MORGAN: a modeling recommender system based on graph kernel, Softw. Syst. Model. (2023) 1–23, http://dx.doi.org/10.1007/s10270-023-01102-8.

[44] A. Iung, J. Carbonell, L. Marchezan, E. Rodrigues, M. Bernardino, F.P. Basso, B. Medeiros, Systematic mapping study on domain-specific language development tools, Empir. Softw. Eng. 25 (5) (2020) 4205–4249, http://dx.doi.org/10.1007/s10664-020-09872-1.

[45] L. Pomante, V. Muttillo, M. Santic, P. Serri, SystemC-based electronic system-level design space exploration environment for dedicated heterogeneous multi-processor systems, Microprocess. Microsyst. 72 (2020) 18, http://dx.doi.org/10.1016/j.micpro.2019.102898.

[46] H. Bruneliere, J.G. Perez, M. Wimmer, J. Cabot, EMF views: A view mechanism for integrating heterogeneous models, in: Conceptual Modeling, Springer International Publishing, 2015, pp. 317–325, http://dx.doi.org/10.1007/978-3-319-25264-3_23.

[47] N. Shervashidze, P. Schweitzer, E.J. van Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeiler-Lehman graph kernels, J. Mach. Learn. Res. 12 (77) (2011) 2539–2561.

[48] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, et al., The Llama 3 Herd of models, 2024, arXiv:2407.21783.

[49] W.-L. Chiang, L. Zheng, Y. Sheng, A.N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M.I. Jordan, J.E. Gonzalez, I. Stoica, Chatbot arena: an open platform for evaluating LLMs by human preference, in: Proceedings of the 41st International Conference on Machine Learning, ICML '24, JMLR.org, 2025.

[50] J. Gerstmayr, P. Manzl, M. Pieber, Multibody models generated from natural language, Multibody Syst. Dyn. 62 (2) (2024) 249–271, http://dx.doi.org/10.1007/s11044-023-09962-0.

[51] V. Muttillo, G. Valente, L. Pomante, Design space exploration for mixed-criticality embedded systems considering hypervisor-based SW partitions, in: 2018 21st Euromicro Conference on Digital System Design, DSD, 2018, pp. 740–744, http://dx.doi.org/10.1109/DSD.2018.00115.

[52] K. Rosvall, I. Sander, A constraint-based design space exploration framework for real-time applications on MPSoCs, in: 2014 Design, Automation & Test in Europe Conference & Exhibition, DATE, 2014, pp. 1–6, http://dx.doi.org/10.7873/DATE.2014.339.

[53] G. Valente, T. Fanni, C. Sau, T.D. Mascio, L. Pomante, F. Palumbo, A composable monitoring system for heterogeneous embedded platforms, ACM Trans. Embed. Comput. Syst. (TECS) 20 (5) (2021) http://dx.doi.org/10.1145/3461647.

[54] L. Pomante, V. Muttillo, B. Křena, T. Vojnar, F. Veljković, P. Magnin, M. Matschnig, B. Fischer, J. Martinez, T. Gruber, The AQUAS ECSEL project aggregated quality assurance for systems: Co-Engineering inside and across the product life cycle, Microprocess. Microsyst. 69 (2019) 54–67, http://dx.doi.org/10.1016/j.micpro.2019.05.013.

[55] V. Muttillo, G. Valente, L. Pomante, H. Posadas, J. Merino, E. Villar, Run-time monitoring and trace analysis methodology for Component-based embedded systems design flow, in: 2020 23rd Euromicro Conference on Digital System Design, DSD, 2020, pp. 117–125, http://dx.doi.org/10.1109/DSD51259.2020.00029.

[56] H. Bruneliere, V. Muttillo, R. Eramo, L. Berardinelli, A. Gómez, A. Bagnato, A. Sadovykh, A. Cicchetti, AIDOaRt: AI-augmented automation for DevOps, a model-based framework for continuous development in Cyber–Physical systems, Microprocess. Microsyst. 94 (2022) 104672, http://dx.doi.org/10.1016/j.micpro.2022.104672.

[57] D. Bilic, E. Brosse, A. Sadovykh, D. Truscan, H. Bruneliere, U. Ryssel, An integrated model-based tool chain for managing variability in complex system design, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2019, pp. 288–293, http://dx.doi.org/10.1109/MODELS-C.2019.00045.

[58] D.I. Ignatov, J. Poelmans, G. Dedene, S. Viaene, A new cross-validation technique to evaluate quality of recommender systems, in: M.K. Kundu, S. Mitra, D. Mazumdar, S.K. Pal (Eds.), Perception and Machine Intelligence, Springer Berlin Heidelberg, 2012, pp. 195–202, http://dx.doi.org/10.1007/978-3-642-27387-2_25.

[59] W.G. Jacoby, Loess:: a nonparametric, graphical tool for depicting relationships between variables, Elect. Stud. 19 (4) (2000) 577–613, http://dx.doi.org/10.1016/S0261-3794(99)00028-1.

[60] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, T.-S. Chua, Data-efficient fine-tuning for LLM-based recommendation, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 365–374, http://dx.doi.org/10.1145/3626772.3657807.