Terpsichora: a tool to generate synthetic MP-Declare process models

Wesley da Silva Santos[1], Juliana Rezende Coutinho[2], Fernanda Baião[2], Georges Miranda Spyrides[1], and Hélio Côrtes Vieira Lopes[1]

[1] Department of Informatics, PUC-Rio, Rio de Janeiro, Brazil
{wsantos,gspyrides,lopes}@inf.puc-rio.br
[2] Department of Industrial Engineering, PUC-Rio, Rio de Janeiro, Brazil
{julianarezendecoutinho@aluno.,fbaiao@}puc-rio.br

Abstract. Process models play a fundamental role in the Business Process Management lifecycle and are crucial for assessing the robustness of proposed algorithms and conducting benchmarks among different tools. However, public models are limited as they expose strategic knowledge. While some researchers developed public repositories of imperative models, there remains a lack of diverse, publicly available multi-perspective declarative models. Our work aims to bridge this gap by providing a tool for generating synthetic MP-Declare process models. We leverage Large Language Models to generate these models, ensuring coverage of diverse aspects and enhancing the resource pool for the BPM community.

Key words: declarative process management, process models, mp-declare, synthetically generated data, large language models

1 Introduction

Business Process Management (BPM) is a systematic approach to improve an organization's business processes [1]. It involves modeling, automation, execution, control, measurement, and optimization of business processes. A key component of BPM is the use of process models, which serve as blueprints for how processes should be executed. These models are essential for understanding, analyzing, and enhancing the efficiency of a company's operations.

Process models are generally categorized into imperative or declarative models [2]. Imperative models specify the exact sequence of tasks to be performed, whereas declarative models define constraints that specify what should and should not happen during the process execution. Declarative models, such as those using the Declare language, are particularly suited for environments where flexibility and adaptability are crucial. They allow for more dynamic and flexible process execution by focusing on the rules and constraints governing the process rather than the exact flow of activities.

Declare comprises constraints and templates to define the permissible behavior within a process, allowing for greater adaptability and exception handling [3]. Multi-perspective process modeling incorporates various dimensions of process

execution, including control-flow, data, and resources. This holistic approach provides a more comprehensive understanding of business processes, which is essential for accurate analysis and optimization [4]. MP-Declare is the multi-perspective version of Declare [5] which allows the specification of relationships between many entities that are involved in business processes, such as actors, control-flow, rules, and attributes.

Despite their advantages, the availability of public multi-perspective declarative process models is limited. This scarcity is primarily because process models can reveal strategic knowledge about how organizations operate, making companies reluctant to share them publicly [6]. While some repositories of imperative models have been developed ([7], [8] and [9]), there is still a significant gap in the availability of diverse multi-perspective declarative models, as most of the available models were extracted from mining techniques and are in repositories of declarative process mining tools, such as RuM Toolkit and Declare4Py [10].

On the other hand, recent advancements in AI, particularly in generative AI (GAI), have shown promise in generating complex data structures, including process models [6]. These models can learn from vast amounts of data and generate new instances that are both diverse and representative of real-world scenarios, unlike traditional approaches [11]. The use of synthetic data and models is gaining traction in BPM research, with an emphasis on many artifacts, such as log generations ([12], [13] and [14]). Synthetic data generation enhanced with GAI allows the creation of diverse and representative datasets without compromising proprietary business information ([15] and [16]), thereby enabling robust testing, development, and benchmarking.

Our work addresses this gap by providing `Terpsichora`[1], a tool capable of generating synthetic MP-Declare models. By leveraging GAI, specifically large language models (LLM), we systematically generated a wide range of MP-Declare models that cover diverse aspects of business processes and domains. This approach not only enriches the resources available to the BPM community but also demonstrates the potential of using advanced AI techniques in process model generation [17]. Our work builds on these foundations by integrating large language models to generate MP-Declare models, thus addressing the need for diverse, public multi-perspective declarative process models.

2 Related Work

In the field of BPM, synthetic data generation has become increasingly important for research and algorithm evaluation. This trend spans various aspects of BPM life-cycle, from event logs and process models to improvements and redesign suggestions. Loreti et al. [18] introduced an innovative method for generating synthetic logs with both positive and negative business process traces using abductive reasoning, addressing limitations in existing log generation techniques. Li et al. proposed a new paradigm for automating business process model discovery

[1] Available at: `https://github.com/santos-wesley/Terpsichora`

from natural language documents, creating the MaD dataset [19] to support this approach. Their work highlights the potential of large-scale datasets in training NLP models for BPM applications. Similarly, Yan et al. developed a technique for generating synthetic collections of business process models that mimic real-world properties, providing researchers with realistic datasets for evaluating process management techniques [20].

Leveraging advanced machine learning techniques, Van Dun et al. introduced ProcessGAN [21], a novel approach using generative adversarial networks (GANs) to support business process improvement. Their work demonstrates the potential of AI in enhancing creative aspects of BPM, such as generating improvement ideas. These studies collectively showcase the growing integration of AI and machine learning techniques in BPM, from automating model discovery and improvement suggestions to generating synthetic data for research and evaluation purposes. They highlight the potential for more efficient, accurate, and innovative approaches to process modeling, analysis, and improvement in the field of Business Process Management.

Over the years, research in synthetic data generation for BPM has advanced significantly, evolving from log generation to tackling specific challenges such as process improvement and redesign. However, despite these advancements, many approaches still rely on classical synthetic data generation techniques, which carry inherent limitations rooted in the characteristics of the trained data and the paradigms of the generation methods [22]. These limitations include difficulties in accurately modeling and reproducing complex, multidimensional relationships between variables, challenges in ensuring evaluation and consistency for complex pipelines, and the risk of overfitting, where models trained on synthetic data may become too tailored to the specifics of the generation process and thus fail to generalize effectively to real-world data. Additionally, accurately representing rare but important events or outliers in synthetic data is particularly challenging. Furthermore, adapting these models to new domains often necessitates extensive retraining, requiring handcrafted linguistic features and rules that are labor-intensive to create and maintain.

3 Data Model, LLM Configuration and Prompt Engineering Techniques

To interplay with large language models, we created a metamodel (Figure 1) representing the structural definition of an MP-Declare model ([4] and [5]). This metamodel specifies the constructs involved in MP-Declare in the form of classes and their relationships and was converted to a data model using Pydantic [2], a data validation library for Python which we used to validate if the data complies with our metamodel. Following, we explain the implementation of each construct in the metamodel, detailing the goal of each class, its rules, and attributes.

[2] Available at: `https://docs.pydantic.dev/latest/`

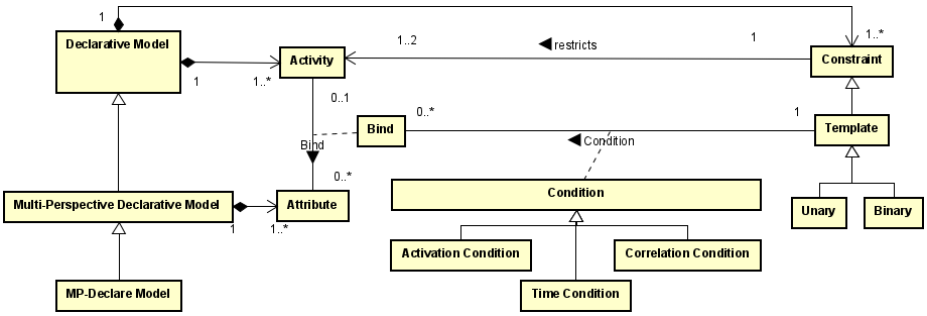

Fig. 1. The UML metamodel for the generation of MP-Declare process models.

The `Activity` class represents an activity within a process model. It is characterized by its `name` (a string that follows the format "<Action Verb><Object>", suggested by [23] as a good pattern for naming an activity), and `description` (which provides a detailed explanation of the activity's purpose). The `Attribute` class defines attributes of a process, characterized by its `type` (which indicates whether the attribute is an integer, float, or enumeration), `name` (specifying the attribute's name) and `description` (providing a detailed explanation of the attribute); optionally, `min_value` and `max_value` fields define the range for integer or float attributes, while `enumeration_values` is a list of possible values for enumeration attributes.

The `Bind` class represents the linkage between activities and their attributes in a model. Hence, each bind is defined by an `Activity` (which represents the `Activity` involved in the binding), `attributes` (a list of `Attribute` objects bound to the activity), and a `description` (providing a description of the bind's purpose). The `Constraint` class models the constraints governing process execution, characterized by its `type` (specifying if the constraint is unary or binary), `description` (a detailed explanation of the constraint), `template` (which specifies the constraint's template), `activation` (representing the activation activity, required for both unary and binary constraints), its `target` (representing the target activity, required for binary constraints); in particular, the `activation_condition`, `correlation_condition`, and `time_condition` are optional fields for specifying specific conditions that govern the constraint's execution. Finally, `cardinality` defines the cardinality for certain unary templates. The `MPDeclareModel` is the main class and encapsulates the entire MP-Declare process model.

In addition to the structural metamodel illustrated in Figure 1, our implementation of the proposed repository comprises validators to ensure appropriate types for each type of construct of the model, using logical and pattern methods. The `convert_to_string` method provides a string representation of the process model, formatting activities, binds, attributes, and constraints into a coherent textual format suitable for parsing to `Declare4Py`.

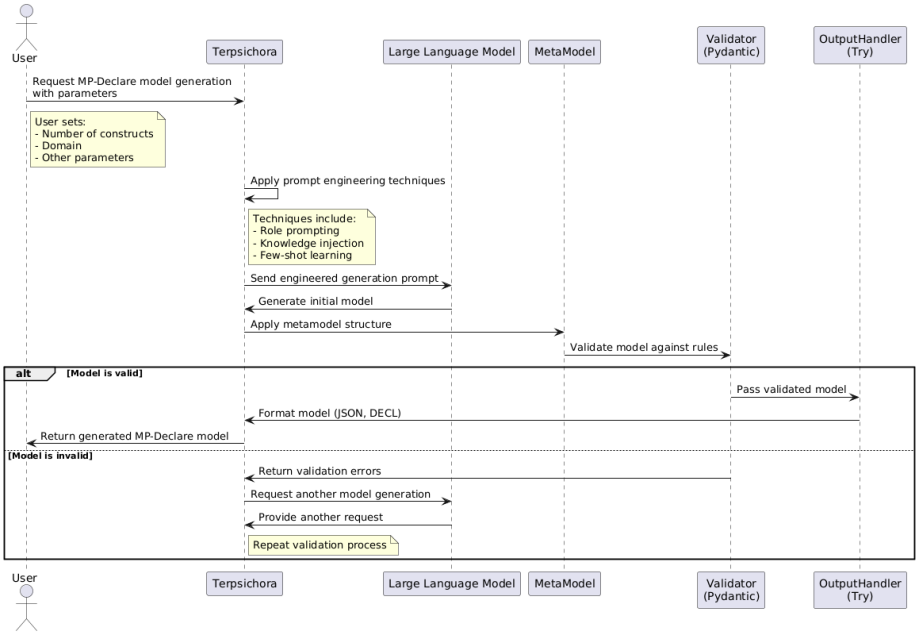

Fig. 2. The generation sequence diagram.

We used `gpt-4o-2024-05-13`, the flagship model of OpenAI, and ChatGPT [3] to generate 500 random domains to feed the generation prompt with a diverse set of domains to generate the models. Alongside with that, we modified the temperature parameter, which is used in the sampling process when generating text to control the randomness or entropy of the text [24]. A higher temperature value increases the randomness, while a lower value makes the model's output more deterministic. We used lower values of temperature (0.2) to make our outputs more reproducible and replicable [25].

We leveraged some prompt engineering (PE) techniques to understand the MP-Declare nuances and generate process models, without the extensive computational and data resources requirements of fine-tuning ([24] and [26]).We used role-prompting [27], which goal is to assign a specific role to the LLM. In our case, we guided the LLM to generate data as an expert in process modeling and analysis and domain expertise using natural language instruction. Knowledge injection [28] involves providing the LLM with specific information that may not have been explored during its training.

In our case, we injected knowledge from MP-Declare's constraints templates in the pipeline, explaining them in natural language through an assistant prompt and converted the model from Pydantic to infuse the model with adequate expected generation, through the function call feature of OpenAI [29]. Function calling enables the integration of LLMs with external tools and systems. This

[3] Available at: https://chatgpt.com/

capability has wide-ranging applications, including enhancing AI assistants with expanded functionalities and facilitating seamless interactions between applications and these models.

A set of few-shot learning was prepared to instruct the LLM of expected input and outputs. Few-shot learning in LLMs refers to the ability of these models to perform new tasks with only a small number of examples or demonstrations [30]. In our case, we used few-shots to enhance model's capability of generating data with quality to our proposed metamodel implementation.

4 Synthetic Model Generation and Quality Assessment

A pipeline was developed in a Jupyter Notebook[4], comprising all the steps for generating MP-Declare models guided by our proposed metamodel, the required packages, PE configurations, and post-processing techniques that were created. Figure 2 depicts a sequence diagram of the generation process, showing the interactions between the user, tool, and metamodel, pydantic and Output Handler.

We generated 500 process models in two batches[5], varying quantity of activities, constraint types, conditions, business domains, and complexity through prompts[6]. The models were used to assess the tool and were exported and saved to `decl` and `JSON` formats. The `JSON` file contains descriptions of the process constructs. We used the guidelines of Sandve et al. [31] to make our experiment reproducible, using version control system, log of generation of process models, error handling in tool and open access scripts.

Fig. 3. A model discovered from log of synthetically generated process by Terpsichora.

With regard to quality assessment, the discussion about quality of declarative process models is almost absent in the literature. de Oca et al. [32] systematically reviewed the literature on business process modeling quality, investigating quality issues, relevant frameworks, and gaps. They showed there is no generally accepted framework for process model quality assessment and this is still a broader issue in the BPM field, even in the imperative paradigm. Some aspects of the Seven Process Modeling Guidelines (7PMG), proposed by Mendling et al. [33], were used to evaluate the generated models. The 7PMG proposes guidelines to assess quality of process constructs, such as elements usage, routing paths, model structuring, and so on. Although 7PMG are tailored to imperative

[4] Available at: `https://github.com/santos-wesley/Terpsichora/blob/main/Terpsichora_The_Pipeline.ipynb`

[5] Available at: `https://github.com/santos-wesley/Terpsichora/tree/main/Models`

[6] Generation prompts can be found in the batch subfolders in each model directory.

Table 1. Overall Statistics (Batch 1 vs Batch 2)

Metric	Min		Mean		Median		Max		Std	
	Batch 1	Batch 2	Batch 1	Batch 2	Batch 1	Batch 2	Batch 1	Batch 2	Batch 1	Batch 2
Activities	8	10	9.904	14.888	10	15	10	19	0.356	0.807
Attributes	5	5	9.296	9.768	10	10	11	11	1.329	0.821
Binds	3	3	8.38	10.18	9	9.5	10	16	1.858	3.220
Attributes per Bind	1	1	1.863	1.711	2	2	5	4	0.659	0.661
Constraints	7	11	9.948	14.96	10	15	11	19	0.324	0.611
Unary Constraints	1	2	2.964	4.060	3	4	5	7	0.755	1.058
Binary Constraints	4	8	6.984	10.90	7	11	9	17	0.794	1.167
Integer Attributes	1	1	3.084	3.204	3	3	6	7	0.961	0.886
Float Attributes	0	0	1.512	1.616	1	2	3	4	0.633	0.718
Enumeration Attributes	2	2	4.7	4.948	5	5	8	8	1.220	1.036
Cardinality	1	1	1.817	4.042	1	1	50	1000	3.347	44.07
Activation Conditions	0	0	0.196	0.204	0	0	3	4	0.511	0.553
Correlation Conditions	0	0	0.016	0.016	0	0	1	1	0.125	0.125
Time Conditions	0	0	2.524	3.104	2	3	6	6	0.964	0.945
Templates per Model	3	3	7.492	9.516	8	9	10	15	1.473	2.248

Table 2. Template Usage Statistics (Batch 1 vs Batch 2)

Template	Count		Percentage of Total Constraints		Models Using Template	
	Batch 1	Batch 2	Batch 1	Batch 2	Batch 1	Batch 2
Precedence	498	773	20.02412545	20.6684492	238	242
Response	378	591	15.19903498	15.80213904	219	234
Chain Response	314	426	12.6256534	11.39037433	194	216
Init	251	252	10.0924809	6.737967914	250	247
End	231	232	9.288299156	6.203208556	230	231
Chain Succession	212	305	8.524326498	8.155080214	162	188
Existence	148	254	5.950944914	6.79144385	141	196
Alternate Response	101	159	4.061117813	4.251336898	94	127
Absence	62	172	2.49296341	4.598930481	62	163
Exactly	49	105	1.970245275	2.807486631	47	97
Succession	45	88	1.809408926	2.352941176	41	78
Chain Precedence	37	63	1.487736228	1.684491979	37	57
Responded Existence	29	54	1.16606353	1.443850267	28	49
Not Co-Existence	28	53	1.125854443	1.417112299	28	53
Co-Existence	24	46	0.965018094	1.229946524	23	42
Alternate Succession	22	46	0.88459992	1.229946524	22	42
Alternate Precedence	17	20	0.683554483	0.534759358	16	19
Choice	10	20	0.402090873	0.534759358	10	20
Not Precedence	9	8	0.361881785	0.213903743	9	8
Not Succession	8	26	0.321672698	0.695187166	8	23
Not Responded Existence	6	1	0.241254524	0.026737968	6	1
Exclusive Choice	4	19	0.160836349	0.50802139	4	19
Not Chain Succession	2	13	0.080418175	0.347593583	2	13
Not Response	1	4	0.040209087	0.106951872	1	4
Not Chain Response	1	7	0.040209087	0.187165775	1	7
Not Chain Precedence	0	3	0	0.080213904	0	3

Table 3. Complexity Metrics (Batch 1 vs Batch 2)

Metric	Min		Mean		Median		Max		Std	
	Batch 1	**Batch 2**	**Batch 1**	**Batch 2**	**Batch 1**	**Batch 2**	**Batch 1**	**Batch 2**	**Batch 1**	**Batch 2**
Size	17	22	19.852	29.848	20	30	21	34	0.513	1.072
Density	0.5	0.667	0.822	0.888	0.833	0.875	1	1.333	0.081	0.081
Separability	0.05	0.029	0.160	0.156	0.150	0.167	0.353	0.308	0.048	0.047
Constraint Variability	0	0	0.768	0.789	0.802	0.826	1	1	0.211	0.172

languages and typically assume control-flow, we used two guidelines to assess our models quality: **G3: Use one start and one end event** and **G6: Use verb-object activity labels**.

We enforced the activities' names to have more than one word and use the verb-object pattern through PE and assessed a sample of the models. Every activity name has between 2 and 5 words, complying with our instruction and guideline G6. Most of the models comply with G3, but some generated models (batch 1: 72 and 77; batch 2: 100, 165, 211, 82, 86 and 87) have not more than two init or end constraints, and their usage relates to real-world possibilities. Table 1 and Table 2 present statistics about the model generation process of the two batches. For the two batches, we observed that the tool were capable of following our instructions regarding the quantity of elements of the model and the variety of constraints. The data of the two batches reveals that the tool progressively enhanced its model generation capabilities, as evidenced by the increase in complexity and diversity of the generated models. This progression demonstrates the tool's robustness in handling a wider range of process scenarios. The adherence to guidelines instructed with PE, coupled with the generation of models that closely resemble real-world processes.

Additionaly to these statistics, we used metrics of declarative models already employed and validated in literature proposed by Abbad-Andaloussi et al.[34]. Table 3 show these metrics, which explore the models in therms of size, density, separability and constraint variability. A comparative analysis of Batch 1 and Batch 2 reveals significant differences in model complexity and structure. Batch 2 models are notably larger, with a mean size of 29.848 compared to 19.852 in Batch 1, and exhibit greater size variability (std dev 1.072 vs 0.513). This substantial size increase suggests that Batch 2 represents more complex processes, and thus reflect that the strategy we employed in PE worked.

Interestingly, despite this size difference, both batches maintain similar density characteristics (means of 0.822 and 0.888), indicating a consistent ratio of constraints to activities. However, Batch 2's higher maximum density (1.333) points to some instances of highly constrained models, potentially reflecting more intricate rule structures in certain cases. The separability metric shows a slight decrease in Batch 2 (mean 0.156 vs 0.160), suggesting that the additional elements in these larger models are integrated into existing components rather than forming discrete sub-processes. This integration trend implies more interconnected and complex process representations in Batch 2. Both batches demonstrate high constraint variability (means of 0.768 and 0.789), indicating

the use of diverse constraint types across models. Notably, Batch 2 shows slightly more consistency in this diversity (std dev 0.172 vs 0.211), suggesting a more uniform application of varied constraints across its models.

Collectively, these metrics reflect increased complexity in Batch 2, characterized by larger, more integrated models with consistently diverse constraint usage. The similarities in density and separability metrics between batches, despite the significant size difference, suggest that the modeling approach scales effectively to larger, more complex processes without fundamentally altering the model structure. This scalability, combined with the trend towards integration rather than fragmentation as processes grow more complex, indicates that the modeling technique is well-suited for representing increasingly sophisticated business processes while maintaining structural integrity and coherence.

We used several models (batch 1: 135, 250; batch 2: 39, 240) to generate logs and rediscover them[7]. We visually assessed them, and they were similar in terms of discovered constraints and control flow. Figure 3 shows a discovered model that appears very imperative. When compared with the synthetically generated model (model 250 from batch 1), it has the same constructs.

We inspected the generated JSON of these models to assess the descriptions and determine if they fit the knowledge we injected into the model. An example of a constraint description generated by Terpsichora for model 250 is: `Chain Succession[Approve Transfer, Execute Transfer] | | |0,2h`, described as "**Execute Transfer must directly follow Approve Transfer within 2 hours.**" This description is compliant with our instructions and aligns with the semantics of the constraint.

5 Conclusions

In this work we presented Terpsichora, a tool for generating MP-Declare models leveraging LLM capabilities of PE and tool calling. To evaluate the proposal, we generated 500 models and assessed them using statistics of output, metrics established in literature and visually assessed a sample of these models qualitatively to attest that they are real-world related.

Our proposal enables a novel generation of a diverse set of MP-Declare models, representing a resource for assessing the robustness of new algorithms, benchmarking tools, and generating logs for simulation purposes, among others. Our pipeline is both reproducible and customizable, allowing users to generate models while overcoming the classic limitations of synthetic data generation, which often requires data acquisition and model training.

Terpsichora's utilization of Large Language Models (LLMs) offers significant advantages in addressing key challenges in synthetic process model generation. By leveraging the complex pattern recognition capabilities of LLMs, Terpsichora can capture and reproduce intricate multidimensional relationships between process elements, as evidenced by the diverse constraint types and their distributions

[7] Available at: `https://github.com/santos-wesley/Terpsichora/tree/main/Logs`

shown in Table 2. This approach not only generates models with sophisticated control-flow patterns and multi-perspective aspects but also enhances adaptability to new domains. Unlike traditional methods that require extensive retraining and domain-specific adjustments, Terpsichora's prompt-based generation allows for quick adaptation to different business contexts simply by modifying the input prompts. This flexibility is particularly valuable for researchers and practitioners working across various industries.

Furthermore, the use of LLMs significantly reduces the need for handcrafted linguistic features and rules, which are often labor-intensive to create and maintain in traditional synthetic data generation methods. By automating the interpretation and application of process semantics with a metamodel and coding it into a datamodel in Pydantic, Terpsichora offers a more efficient and scalable approach to generating diverse, realistic MP-Declare models, potentially accelerating research and development in the context of model generation.

Along with the traditional constructs of MP-Declare, we add a new attribute to generate descriptions for each construct. This resource enables semantic anomaly detection not only for activities, as proposed by [35], but also for constraints. Another potential application is in generating multi-perspective process logs, incorporating additional perspectives inexistent in traditional log generation from structured models.

However, there are limitations. First, the current output context window of the model is limited to 4096 tokens, which restricts the ability to generate larger models and to test the LLM capacity to produce and maintain coherence in extensive, real-world models. Second, to ensure reproducibility, we kept a low temperature. Higher temperatures generally lead to more creative outputs, which could be beneficial for evaluating different generations of the same process.

Future work include testing with other models, including the newest OpenAi `gpt-4o-2024-08-06`, which implemented structured outputs techniques to enhance model inference in a given structured schema with complex rules [36] and increased the output tokens to 16384. Also, the usage of other prompt techniques to evaluate model quality at a semantic level, such as LLM-as-a-Judge [37], which enables open-ended questions evaluation using LLMs. Finally, addressing the understandability and cognitive impact of models generated by Terpsichora.

References

1. M. Dumas, L. M. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*, Springer, 2018.
2. W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," *Computer Science-Research and Development*, vol. 23, pp. 99–113, 2009.
3. C. Di Ciccio and M. Montali, "Declarative Process Specifications: Reasoning, Discovery, Monitoring," in *Process Mining Handbook*, W. M. P. van der Aalst and J. Carmona, Eds., Springer International Publishing, Cham, 2022, pp. 108–152, doi: 10.1007/978-3-031-08848-3_4.

4. S. Schönig, C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovery of Multi-perspective Declarative Process Models," in *Service-Oriented Computing*, Springer International Publishing, Cham, 2016, pp. 87–103.
5. A. Burattin, F. M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models," *Expert Systems with Applications*, vol. 65, pp. 194–211, 2016.
6. S. Feuerriegel, J. Hartmann, C. Janiesch, and P. Zschech, "Generative AI," *Business & Information Systems Engineering*, vol. 66, no. 1, pp. 111–126, 2024.
7. F. Corradini, F. Fornari, A. Polini, B. Re, F. Tiezzi, and others, "RePROSitory: a Repository Platform for Sharing Business PROcess modelS," *BPM (PhD/Demos)*, vol. 2420, pp. 149–153, 2019.
8. M. Weske, G. Decker, M. Dumas, L. Rosa, J. Mendling, and H. A. Reijers, "Model collection of the business process management academic initiative," *Zenodo*, doi: 10.5281/zenodo.3758705, 2020.
9. D. Sola, C. Warmuth, B. Schäfer, P. Badakhshan, J. R. Rehse, and T. Kampik, "SAP Signavio Academic Models: a large process model dataset," in *International Conference on Process Mining*, Springer, 2022, pp. 453–465.
10. A. Alman, I. Donadello, F. M. Maggi, and M. Montali, "Declarative Process Mining for Software Processes: The RuM Toolkit and the Declare4Py Python Library," in *International Conference on Product-Focused Software Process Improvement*, Springer, 2023, pp. 13–19.
11. R. Sutton, A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.
12. V. Skydanienko, C. Di Francescomarino, C. Ghidini, and F. M. Maggi, "A Tool for Generating Event Logs from Multi-Perspective Declare Models," *BPM (Dissertation/Demos/Industry)*, vol. 2196, pp. 111–115, 2018.
13. I. Donadello, F. M. Maggi, F. Riva, and M. Singh, "ASP-Based Log Generation with Purposes in Declare4Py," in *ICPM Doctoral Consortium/Demo*, 2023.
14. C. Di Ciccio, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Generating event logs through the simulation of declare models," in *Enterprise and Organizational Modeling and Simulation: 11th International Workshop, EOMAS 2015, Held at CAiSE 2015, Stockholm, 2015, Selected Papers 11*, Springer, 2015, pp. 20–36.
15. J. Recker, M. Indulska, M. Rosemann, and P. Green, "How Good is BPMN Really? Insights from Theory and Practice," in J. Ljungberg and M. Andersson, Eds., *Proceedings 14th European Conference on Information Systems*, Goeteborg, 2006.
16. A. Burattin, "Artificial datasets for multi-perspective Declare analysis," Zenodo, Jul. 2015, doi: 10.5281/zenodo.20030.
17. H. van der Aa, C. Di Ciccio, H. Leopold, and H. A. Reijers, "Extracting declarative process models from natural language," in *Advanced Information Systems Engineering: 31st International Conference, CAiSE 2019, Rome, Italy, June 3–7, 2019, Proceedings 31*, Springer, 2019, pp. 365–382.
18. D. Loreti, F. Chesani, A. Ciampolini, and P. Mello, "Generating synthetic positive and negative business process traces through abduction," *Knowledge and Information Systems*, vol. 62, pp. 813–839, 2020.
19. X. Li, L. Ni, R. Li, J. Liu, and M. Zhang, "MaD: A Dataset for Interview-based BPM in Business Process Management," in *2023 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2023, pp. 1–8.
20. Z. Yan, R. Dijkman, and P. Grefen, "Generating synthetic process model collections with properties of labeled real-life models," in *Asia Pacific Business Process Management: Second Asia Pacific Conference, AP-BPM 2014, Brisbane, QLD, Australia, July 3-4, 2014. Proceedings 2*, Springer, 2014, pp. 74–88.

21. C. van Dun, L. Moder, W. Kratsch, and M. Röglinger, "ProcessGAN: Supporting the creation of business process improvement ideas through generative machine learning," *Decision Support Systems*, vol. 165, p. 113880, 2023.
22. S. I. Nikolenko, *Synthetic Data for Deep Learning*, vol. 174, Springer, 2021.
23. H. Leopold, R. H. Eid-Sabbagh, J. Mendling, L. G. Azevedo, and F. A. Baiao, "Detection of naming convention violations in process models for different languages," *Decision Support Systems*, vol. 56, pp. 310–325, 2013.
24. C. Li, F. Luo, C. Tan, M. Wang, S. Huang, S. Li, and J. Bai, "Parameter-efficient sparsity for large language models fine-tuning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. De Raedt, Ed., ijcai.org, 2022, pp. 4223–4229.
25. P. Ivie and D. Thain, "Reproducibility in scientific computing," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–36, 2018.
26. I. Thangarasa, A. Gupta, W. Marshall, T. Li, K. Leong, D. DeCoste, S. Lie, and S. Saxena, "SPDF: sparse pre-training and dense fine-tuning for large language models," in *Uncertainty in Artificial Intelligence, UAI 2023, July 31 - 4 August 2023, Pittsburgh, PA, USA*, R. J. Evans and I. Shpitser, Eds., vol. 216, Proceedings of Machine Learning Research, PMLR, 2023, pp. 2134–2146.
27. B. Xu, A. Yang, J. Lin, Q. Wang, C. Zhou, Y. Zhang, and Z. Mao, "Expert-prompting: Instructing large language models to be distinguished experts," *CoRR*, abs/2305.14688, 2023.
28. A. Martino, M. Iannelli, and C. Truong, "Knowledge injection to counter large language model (LLM) hallucination," in *European Semantic Web Conference*, Springer, 2023, pp. 182–185.
29. OpenAI, "Function Calling Guide," 2024. [Online]. Available: `https://platform.openai.com/docs/guides/function-calling`. Accessed: 2024-07-03.
30. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, 33, pp. 1877–1901, 2020.
31. G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten simple rules for reproducible computational research," *PLoS Computational Biology*, vol. 9, no. 10, p. e1003285, 2013.
32. I. Moreno-Montes de Oca, M. Snoeck, H. A. Reijers, and A. Rodríguez-Morffi, "A systematic literature review of studies on business process modeling quality," *Information and Software Technology*, vol. 58, pp. 187–205, 2015.
33. J. Mendling, H. Reijers, and W. van der Aalst, "Seven process modeling guidelines (7PMG)," *Information and Software Technology*, 52(2), pp. 127–136, 2010.
34. A. Abbad-Andaloussi, A. Burattin, T. Slaats, E. Kindler, and B. Weber, "Complexity in declarative process models: Metrics and multi-modal assessment of cognitive load," *Expert Systems with Applications*, vol. 233, p. 120924, 2023.
35. H. van der Aa, A. Rebmann, and H. Leopold, "Natural language-based detection of semantic execution anomalies in event logs," *Information Systems*, vol. 102, p. 101824, 2021, doi: 10.1016/j.is.2021.101824.
36. OpenAI, "Introducing structured outputs in the API," 2024. [Online]. Available: `https://openai.com/index/introducing-structured-outputs-in-the-api/`. Accessed: 2024-08-11.
37. L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," *Advances in Neural Information Processing Systems*, vol. 36, 2024.