Using ChatGPT to Refine Draft Conceptual Schemata in Supply-Driven Design of Multidimensional Cubes

Stefano Rizzi[1]

[1]DISI - University of Bologna, Viale Risorgimento, 2, Bologna, 40136 Italy

Abstract

Refinement is a critical step in supply-driven conceptual design of multidimensional cubes because it can hardly be automated. In fact, it relies on the end-users' requirements on the one hand, and on the semantics of measures, dimensions, and attributes on the other. As a consequence, it is normally carried out manually by designers in close collaboration with end-users. The goal of this work is to check whether LLMs can act as facilitators for the refinement task, so as to let it be carried out entirely —or mostly— by end-users. The Dimensional Fact Model is the target formalism for our study; as a representative LLM, we use ChatGPT's model GPT-4o. To achieve our goal, we formulate two research questions aimed at understanding the basic competences of ChatGPT in refinement and investigating if they can be improved via prompt engineering. The results of our experiments show that, indeed, a careful prompt engineering can significantly improve the accuracy of refinement, and that the residual errors can quickly be fixed via one additional prompt. However, we conclude that, at present, some involvement of designers in refinement is still necessary to ensure the validity of the refined schemata.

Keywords

Conceptual design, Multidimensional model, Large Language Models, ChatGPT, Refinement, Supply-driven design

1. Introduction

Conceptual design is a key step in the development of data warehouse (DW) systems and multidimensional databases, since it determines their information content and, ultimately, the set of queries they can answer. The goal is to create an implementation-independent representation of one or more *cubes* structured according to the multidimensional model, i.e., described in terms of measures, dimensions, and attribute hierarchies. A lot of research has been done over the last couple of decades on conceptual design of cubes, mainly distinguishing between *supply-driven approaches*, where the conceptual schema is determined starting from the schema of a source databases, and *demand-driven approaches*, where it is created based on the end-users' requirements.

An advantage of supply-driven design over demand-driven design is that a draft conceptual schema can be obtained from the source schema in automatic fashion, by applying an algorithm that essentially chases the functional dependencies coded in the source schema and uses them to arrange hierarchies [1]. Although this significantly speeds up design, the draft schema must then be refined in the light of the end-users' requirements. Refinement mainly implies the following activities [2]:

- Removing attributes that are deemed not interesting for analyses.
- Finding *descriptive attributes*, i.e., attributes that should not be used for aggregation while being useful for analyses (e.g., the name of a customer).
- Discretizing attributes with dense domains to make them usable for aggregation (e.g., the weight of a product).
- Finding *optional attributes*, i.e., attributes that are undefined for some instances of the hierarchy (e.g., the State attribute in a geographical hierarchy that also includes non-US nations).
- Labeling measures based on whether the SUM operator can be used or not to aggregate them (e.g., the exchange rate of dollars to euros, which cannot be summed).

Unfortunately, these activities can hardly be automated by an algorithm because they rely on the end-users' requirements on the one hand, and on the semantics of measures, dimensions, and attributes as expressed by their names on the other. Then, they must be carried out manually by designers in collaboration with end-users.

This is a typical situation in software engineering where *Large Language Models* (LLMs) may come to the rescue. LLMs have proven to be a great tool for mimicking human linguistic abilities because of their capacity to learn from large corpora, which has had a disruptive effect in a number of fields, and more specifically in software engineering [3, 4, 5]. In particular, the experiments on using LLMs for conceptual design [6, 7] showed that they can help designers with this task by producing draft solutions in a timely manner —although some human intervention is still necessary to guarantee the accuracy of the outcomes.

The goal of this work is to check whether LLMs can act as facilitators for the refinement of conceptual schemata of multidimensional cubes, so as to relieve designers from their role or even, if possible, let refinement be carried out entirely by end-users. The Dimensional Fact Model (DFM [1]) is the target formalism for our study; as a typical LLM, we use ChatGPT's model GPT-4o [8], which has gained popularity for its smooth user interface and natural language generating capabilities [9]. To achieve our goal, we formulate two research questions aimed at (i) understanding the basic competences of ChatGPT in the refinement of a draft DFM schema and (ii) investigating if the latter can be improved via prompt engineering. An extended version of this work is available in [10].

2. Related work

An experiment to use LLMs for creating specifications from requirements documents in the realm of smart devices is described in [7]. The authors contend that the fundamental skill of conceptual design is still lacking, but they acknowledge that LLMs are very useful in later phases of the devel-

DOLAP 2025: 27th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2025, March 25, 2025, Barcelona, Spain

stefano.rizzi@unibo.it (S. Rizzi)

0000-0002-4617-217X (S. Rizzi)

opment process, like creating class diagrams and generating source code. Additional experiments with ChatGPT for conceptual modeling are discussed in [6]. The authors note that ChatGPT can rapidly produce an initial draft diagram from a natural language description; nevertheless, considerable modeling expertise is still needed to improve and verify the outcomes. The authors of [9] describe an experiment they conducted using ChatGPT and come to the conclusion that while adding LLMs to human-driven conceptual design does not dramatically affect outcomes, it does greatly reduce the time required to complete the design by requiring fewer design steps. In [11], many conceptual schemata produced by an LLM are contrasted with a baseline of crowdsourced solutions. On average, it is shown that crowdsourced ideas are more innovative, whereas LLM-generated solutions are more practical. In [12], the benefits of utilizing LLMs to improve morphological analysis in conceptual design are examined. The tests demonstrate how LLMs give designers access to interdisciplinary knowledge; for optimal outcomes, LLMs and designers should work closely together and use smart prompt engineering. With relation to use case and domain modeling, [13] examines how users engage with LLMs during conceptual modeling. The primary conclusions speak to the necessity of particular prompt templates to assist users.

As to multidimensional conceptual design, the main types of methods in the literature are *supply-driven* (or data-driven), *demand-driven* (or requirement-driven), *mixed*, and *query-driven*. Supply-driven methods begin by designing conceptual schemata from the schemata of the data sources (such as relational schemata); end-user requirements influence design by enabling the designer to choose which data are important for making decisions and by figuring out how to structure them using the multidimensional model [14]. Demand-driven techniques begin with identifying end-users' business requirements, and only then do they look into how to map these requirements onto the available data sources [15]. Mixed techniques integrate requirements-driven and data-driven methods; here, both end-user requirements and data source schemata are used simultaneously [16]. The set of OLAP queries that end-users are willing to formulate is the starting point for the creation of a multidimensional schema in query-driven approaches. These queries can be specified using SQL statements [17], MDX expressions [18], pivot tables [19], or query trees [20]. Multidimensional modeling techniques are reviewed in [21], and their cost-benefit analysis is provided in [22].

3. The investigation process

As stated in the Introduction, our goal in this work is to assess the performance of ChatGPT in the refinement of a draft DFM schema obtained by supply-driven design starting from a source relational schema. We take as a reference an advanced form of the DFM including, besides the basic constructs of fact, measure, dimension, and attribute, the advanced constructs of descriptive attributes, optional attributes, and additivity. In this form, a DFM schema is a graph whose root is the *fact* (represented as a box with the fact name —e.g., SALES— followed by a list of *measures* —e.g., Amount), whose other nodes are *attributes* —e.g., Product— represented as circles and connected by arcs representing many-to-one *roll-up relationships*, i.e., functional dependencies (FDs, for instance, Product → Category). Descriptive attributes are represented without a circle; optional attributes are dashed; a non-additive measures is represented by adding its aggregation operator to its name (e.g., ExchangeRate (AVG)).

3.1. Research questions

We formulate the following research questions:

RQ.1: Is ChatGPT capable of refining a draft DFM schema by (i) making attribute names more intuitive for end-users, (ii) showing additivity, (iii) finding descriptive attributes or discretizing them, (iv) finding optional attributes, (v) completing time hierarchies, and (vi) removing uninteresting attributes?

RQ.2: Can the performance of ChatGPT in refining a draft DFM schema be improved via prompt engineering?

3.2. Experiment design

Our experiment relies on five cornerstones, described in the following subsections.

3.2.1. Base criteria and technology

The criteria we follow for our experiment are listed below:

- **Learning**. For learning we adopt a prompt-based learning method, which is often used as an alternative to fine-tuning [23]. Specifically, for RQ.1 we adopt *0-shot learning* (which operates with no labeled examples); for RQ.2 we adopt *few-shot learning* and provide two training examples [24]. To further improve learning, for RQ.2 we also employ the *chain-of-thought* technique [25], which includes a list of reasoning steps in the examples.
- **Reproducibility**. The lack of reproducibility of the tests is a significant challenge when working with LLMs because of their non-deterministic nature. The level of "creativity" of ChatGPT is ruled by its temperature parameter; in principle, no creativity is required for refining draft schemata, so we set the temperature to 0 for every chat.
- **Domain**. The issue domain is acknowledged to be crucial for LLMs; the more domain knowledge an LLM has, the better the model it generates [26]. Every example we present describes actual domains, some of which are well-known (like purchases) and others are less common (like crossfit workouts).
- **Conversation-awareness**. The answers obtained from ChatGPT may depend heavily on the previous questions asked during a conversation. Thus, as also suggested in [26], we start a new chat for each case.
- **Iteration**. In all our tests, the first answer obtained is considered. However, keeping in mind that the refinement process is inherently iterative, in RQ.2 we tried to improve the first answer by further prompting ChatGPT with suggestions.

As to the technological environment, experiments have been carried out on the ChatGPT-4o model.

3.2.2. Input/output format

A draft DFM schema must be provided in input for each of our research questions, and a refined one must be provided as output. We employ *YAML*[1], a human-readable data

[1] https://yaml.org/spec/history/2001-08-01.html

serialization language that is frequently used for configuration files and in applications where data is being saved or communicated, as a format to express DFM schemata. Since ChatGPT is familiar with YAML, it does not need any further instruction on the syntax of the language; nevertheless, it needs to be taught about the particular tags we added to denote multidimensional concepts (e.g., *measures* to introduce the list of measures).

3.2.3. Prompt templates

Following the suggestions given in [3, 4], the prompts we adopt during our chats are structured according to the following templates:

- *Instruction prompts.* These are used in RQ.1 and RQ.2 to assign ChatGPT a task and explain how to execute it. Their structure includes: (1) ROLE: the specific roles assigned to ChatGPT and to the user to provide a context for the task; (2) FORMAT: how the input and output (i.e., the draft and refined DFM schemata) should be coded; (3) TASK: the task assigned; (4) PROCEDURE (optional): the method suggested to perform the task; (5) EXAMPLE (optional): an example of some test cases, an explanation of the procedure suggested to solve it (according to the chain-of-thought principle), and the expected output.
- *Case prompts.* These are used in RQ.1 and RQ.2 to assign a specific task to ChatGPT. Their structure includes: (1) INPUT: the input of the task (a draft DFM schema coded in YAML); (2) TASK: the task assigned; (3) OUTPUT: the output required (a refined DFM schema coded in YAML).

3.2.4. Test cases

We created a set of five test cases with increasing difficulties, based on some exercises in supply-driven design assigned to the students of a master course in Business Intelligence. Each exercise provided a source relational schema; from this schema, a draft DFM schema was created in supply-driven mode using the FD-chasing algorithm in [1]. The number of dimensions and measures in the test cases ranges from 3 to 5 and from 0 to 5, respectively, while the overall number of attributes (dimensions plus hierarchy levels) ranges from 10 to 34. Two of the test cases include shared hierarchies (i.e., nodes entered by two or more arcs, as often is the case with temporal hierarchies).

3.2.5. Evaluation of the results

Refinement is, to some extent, a subjective process because it largely depends on the end-user requirements. For instance, given a ProductWeight attribute, both making it descriptive and discretizing it into WeightRanges are reasonable refinements. As a consequence, creating a single ground truth for each test case is hardly feasible. So we had to proceed manually, by first identifying a set of feasible refinements for each part of each draft DFM schema, and then counting an error in the solution proposed by ChatGPT for each deviation from this set of feasible refinements.

3.3. Answer to RQ.1: Refinement

To put ChatGPT to the test on refinement, we fed it with our five test cases. In order to enable a more precise evaluation of the abilities of ChatGPT, separate prompts are submitted for each refinement step entailed by RQ.1. Thus, for each test case we adopt a simple instruction prompt that explains the DFM constructs followed by a request to carry out a list of refinement steps; no PROCEDURE and EXAMPLE components are present that suggest ChatGPT how to operate. Then we formulate a sequence of case prompts that (i) specify a draft DFM schema in input, (ii) assign as a task one single refinement step, and (iii) require a refined DFM schema in output. In the following we will separately consider each step and briefly review its result.

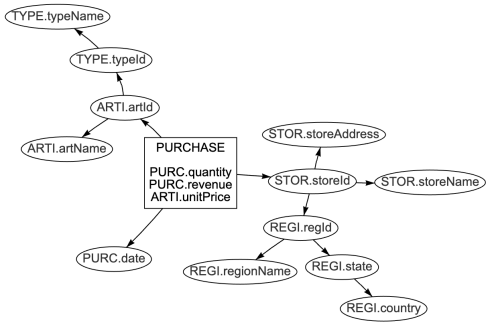

Figure 1: Draft schema for test case C2 (only the first four letters of relation names are shown)

(i) **Make names intuitive**. The attribute and measure names in the draft DFM schema have the form RELATION_NAME.attributeName, being RELATION_NAME a table of the source relational schema used for deriving the draft schema and attributeName one of its attributes (see Figure 1). Making these names more intuitive for end-users is mostly done correctly even if no specific procedure is suggested. In some cases the relation name was simply dropped, in others it was prefixed to the attribute name (e.g., SUPPLIER.name became SupplierName). In case C5, the most complex one, the shared hierarchy was mistaken and the direction of some FDs was inverted.

(ii) **Label measures**. ChatGPT is quite good at dealing with additivity. This is surprising, considering that this task is often not easy even for end-users. The main errors we found were syntactical: the measure was renamed in the YAML code under the "measure" tag but not under the "dependencies" tag, resulting in additional fake nodes.

(iii) **Find descriptive attributes**. ChatGPT performs poorly in this task, with an average of almost four errors per test case. On the one hand, it does not know under which conditions an attribute should be made descriptive or discretized; on the other, it does not use the correct syntax as stated in the FORMAT section of the instruction prompt.

(iv) **Find optional attributes**. The identification of optional attributes is strictly related to the end-user requirements. Thus, for this refinement step the prompt simulates an end-user statement; for instance, "Not all regions have a state". As in the previous case, ChatGPT always fails in the syntax used (although it correctly identifies the optional attribute).

(v) **Complete time hierarchies**. Here, ChatGPT correctly adds Month → Year hierarchies to Date attributes. However, it always fails in recognizing and managing shared hierarchies (in C4 and C5).

(vi) **Remove attributes**. For the last refinement step, an indication from end-users about which attributes are deemed uninteresting for their analyses is required. Thus, like

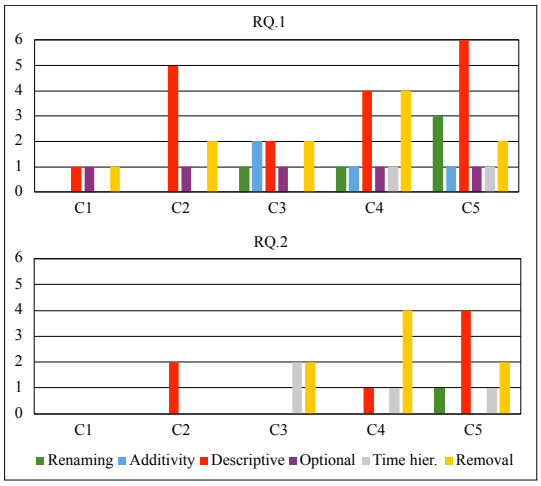

Figure 2: Number of errors in the refinement of draft DFM schemata (top: basic prompts, bottom: improved prompts)

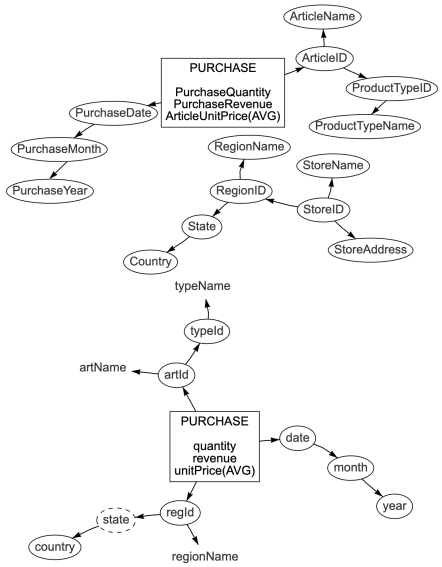

Figure 3: Refined schema for test case C2 with basic prompt (top) and improved prompt (bottom; descriptive attributes are shown with no circle, optional attributes with dashed circles)

for optional attributes, the prompt simulates an end-user statement; for instance, "StoreId is not interesting to me". ChatGPT does not know how to correctly rearrange FDs after removing an attribute, so it makes an average of 2 errors per test case for this refinement step.

The number of errors made at each step for each test case is shown in Figure 2 (top). As an example, Figure 3 (top) shows the final DFM schema for test case C2; note that descriptive and optional attributes are not shown as such because the visualizer does not recognize the wrong YAML syntax for them, and that the graph is non-connected because some FDs were dropped. Overall, the performance are not very good but acceptable, with an average number of total refinement errors per test case equal to 9. The errors clearly tend to increase with the complexity of the draft schema; the main problems are due to the YAML syntax and to the presence of shared hierarchies. The most critical refinement steps appear to be the identification of descriptive attributes and the removal of uninteresting attributes.

3.4. Answer to RQ.2: Improved refinement

To answer RQ.2 we incrementally crafted an instruction prompt by first trying to address the main issues emerged in RQ.1, then progressively adding specific sentences to try to fix the residual (or new) errors. The ROLE, FORMAT, and TASK components are exactly like in RQ.1. However, for each refinement step, we added a PROCEDURE component to suggest ChatGPT how to operate and an EXAMPLE component with two examples. The case prompts are exactly the same used for RQ.1.

No errors in additivity and optional attributes are made when the improved prompt is used. A single renaming error is made in C5, due to an unrecognized shared hierarchy. A few errors are made on time hierarchies, again due to shared hierarchies. Overall, the main causes of errors are related to descriptive/discretized attributes (in some cases, a few of them are not identified) and to the removal of uninteresting attributes (sometimes, arcs are not correctly repositioned). Noticeably, all these errors could be fixed in a single iteration via specific prompts, e.g., "Merge drop-off date and pick-up date into a single date node" to fix a shared hierarchy. In some cases, even generic prompts were used successfully to fix errors, e.g., "Some arcs are missing, please try again" to fix the FDs after removal. Figure 3 (bottom) shows the final DFM schema obtained for C2 after correcting two errors in descriptive attributes via an iteration prompt.

The results, in terms of number of errors made at each step, are summarized in Figure 2 (bottom). It appears that prompt engineering can significantly improve the accuracy of refinement, with the average number of total refinement errors per test case decreasing from 9 to 4. The main residual errors are related to the recognition of shared hierarchies and of descriptive/discretized attributes, as well as to the removal of uninteresting attributes. In our tests, all these errors could be fixed via an additional prompt that either explains exactly how to proceed, or simply suggests to try again paying more attention to some specific aspect.

4. Conclusion

In this work we have investigated the capabilities of ChatGPT to cope with a specific task in conceptual design, namely, the refinement of draft DFM schemata obtained by supply-driven conceptual design of multidimensional data cubes —a task that is normally carried out manually by designers and end-users in close collaboration. It turned out that, although ChatGPT tends to mix the conceptual level (DFM) with the logical level (star/snowflake schemata), it can provide some acceptable results on test cases with different degrees of complexity using simple prompts. Noticeably, our tests show that, when prompts are enhanced with detailed instructions and examples, the results produced significantly improve in all cases. Indeed, when using an improved prompt the average number of errors per multidimensional concept across all test cases decreases from 0.5 to 0.2. In practice, the residual errors are still too many to state that no involvement of designers is necessary and that end-users can carry out refinement by directly interacting with an LLM. However, we can conclude that LLMs can significantly support designers in refinement, even considering that all residual errors in our tests could quickly be fixed via a simple additional prompt.

References

[1] M. Golfarelli, S. Rizzi, Data warehouse design: Modern principles and methodologies, McGraw-Hill, 2009.

[2] L. Antonelli, S. Bimonte, S. Rizzi, Multidimensional modeling driven from a domain language, Autom. Softw. Eng. 30 (2023) 6.

[3] W. Ma, S. Liu, W. Wang, Q. Hu, Y. Liu, C. Zhang, L. Nie, Y. Liu, LLMs: Understanding code syntax and semantics for code analysis, CoRR abs/2305.12138 (2023).

[4] J. White, S. Hays, Q. Fu, J. Spencer-Smith, D. C. Schmidt, ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design, CoRR abs/2303.07839 (2023).

[5] H. Fill, J. Cabot, W. Maass, M. van Sinderen, AI-driven software engineering - the role of conceptual modeling, Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 19 (2024).

[6] H. Fill, P. Fettke, J. Köpke, Conceptual modeling and large language models: Impressions from first experiments with ChatGPT, Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 18 (2023) 3.

[7] R. Lutze, K. Waldhör, Generating specifications from requirements documents for smart devices using large language models (LLMs), in: Proc. HCI, Washington, DC, USA, 2024, pp. 94–108.

[8] W. Hariri, Unlocking the potential of ChatGPT: A comprehensive exploration of its applications, advantages, limitations, and future directions in natural language processing, CoRR abs/2304.02017 (2023).

[9] Z. Zhou, J. Li, Z. Zhang, J. Yu, H. Duh, Examining how the large language models impact the conceptual design with human designers: A comparative case study, Int. J. Hum. Comput. Interact. (2024) 1–17.

[10] S. Rizzi, Using ChatGPT to refine draft conceptual schemata in supply-driven design of multidimensional cubes, arXiv 2502.02238v1 (2025).

[11] K. Ma, D. Grandi, C. McComb, K. Goucher-Lambert, Conceptual design generation using large language models, CoRR abs/2306.01779 (2023).

[12] L. Chen, Y. Tsang, Q. Jing, L. Sun, A LLM-augmented morphological analysis approach for conceptual design, in: Proc. DRS, Boston, USA, 2024, pp. 1–19.

[13] S. J. Ali, I. Reinhartz-Berger, D. Bork, How are LLMs used for conceptual modeling? An exploratory study on interaction behavior and user perception, in: Proc. ER, Pittsburgh, USA, 2024, pp. 257–275.

[14] O. Romero, A. Abelló, Data-driven multidimensional design for OLAP, in: Proc. SSDBM, Portland, OR, USA, 2011, pp. 594–595.

[15] P. Jovanovic, O. Romero, A. Simitsis, A. Abelló, D. Mayorova, A requirement-driven approach to the design and evolution of data warehouses, Inf. Syst. 44 (2014) 94–119.

[16] F. D. Tria, E. Lefons, F. Tangorra, Hybrid methodology for data warehouse conceptual design by UML schemas, Inf. Softw. Technol. 54 (2012) 360–379.

[17] O. Romero, A. Abelló, Automatic validation of requirements to support multidimensional design, Data Knowl. Eng. 69 (2010) 917–942.

[18] T. Niemi, J. Nummenmaa, P. Thanisch, Constructing OLAP cubes based on queries, in: Proc. DOLAP, Atlanta, Georgia, USA, 2001, pp. 9–15.

[19] S. Bimonte, L. Antonelli, S. Rizzi, Requirements-driven data warehouse design based on enhanced pivot tables, Req. Eng. 26 (2021) 43–65.

[20] R. Nair, C. Wilson, B. Srinivasan, A conceptual query-driven design framework for data warehouse, Int. Jour. of Computer and Information Engineering 1 (2007) 62–67.

[21] O. Romero, A. Abelló, A survey of multidimensional modeling methodologies, Int. J. Data Warehous. Min. 5 (2009) 1–23.

[22] F. D. Tria, E. Lefons, F. Tangorra, Cost-benefit analysis of data warehouse design methodologies, Inf. Syst. 63 (2017) 47–62.

[23] K. Chen, Y. Yang, B. Chen, J. A. H. López, G. Mussbacher, D. Varró, Automated domain modeling with large language models: A comparative study, in: Proc. MODELS, Västerås, Sweden, 2023, pp. 162–172.

[24] T. B. Brown, et al., Language models are few-shot learners, in: Proc. NeurIPS, 2020.

[25] J. Wei, et al., Chain-of-thought prompting elicits reasoning in large language models, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Proc. NeurIPS, New Orleans, LA, USA, 2022.

[26] 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 (2023) 781–793.