Assisting Stakeholders in Class Diagram Interpretation with LLMs: a Work in Progress

Chiara Mannari[1,2], Tommaso Turchi[1], Manlio Bacco[2], Alessio Malizia[1,3]
[1]University of Pisa, Pisa, Italy
[2]Institute of Information Science and Technologies (ISTI), National Research Council (CNR), Pisa, Italy
[3]Molde University College, Molde, Norway
chiara.mannari@isti.cnr.it, tommaso.turchi@unipi.it,
manlio.bacco@isti.cnr.it, alessio.malizia@unipi.it

***Abstract*—Diagrams can be valuable tools in requirements engineering to establish a shared understanding between software engineers and stakeholders. However, interacting with these visual representations can be challenging for some stakeholders who prefer textual descriptions and may need support to interpret notation elements and understand the diagram structure and meaning. To address this need, we explore the use of Large Language Models to effectively assist stakeholders interacting with diagrams by providing automatic textual explanations and contextual guidance. Specifically, we aim to design and evaluate with stakeholders an interactive layer (integrated into an end-user-oriented modelling tool) that provides automatic diagram explanations in natural language. As a first step toward our research objective, this paper investigates the capability of GPT4 to generate appropriate textual descriptions from domain models. We use a test data set consisting of UML class diagrams in various formats, belonging to the domain of digital agriculture, and develop a set of prompts to generate the interactive explanatory layer. We conduct a technical evaluation of the output, focusing on correctness, completeness, and understandability. The results provide valuable insights to inform future design and research, while also revealing potential challenges in real-world applications.**

I. Introduction

Diagrams play an important role in requirements engineering (RE) by providing structured and unambiguous representations of system requirements, processes, goals and interactions, to support subsequent development [1]. The adoption of modelling notations such as UML [2], iStar [3], [4], KAOS [5], URN [6], BPMN [7], promotes consistency and interoperability, fosters a shared understanding among developers, enables tool-supported analysis, transformation, and automation [8].

The overarching goal of our ongoing research is to investigate to what extent MoDRE notations can be applied to address the challenge of co-designing digital solutions in socio-technical systems. We frame our research within the context of digital agriculture, which is characterized by the integration of digital technologies into traditional practices, resulting in complex socio-technical transformations [9]. This domain involves a diverse range of stakeholders with different backgrounds and skills and requires multidisciplinary expertise to effectively address social, economic, environmental, and technological challenges [10]. The adoption of digital solutions in agriculture necessitates not only technical innovation but also careful consideration of the impact within real-world rural contexts [11], underscoring the need for collaborative and context-sensitive RE approaches to innovation [12].

This paper builds on previous findings from our empirical research in the realm of RE and end-user development (EUD) [13]. In [14] we introduced a RE method for *socio-technical process modelling* in digital agriculture. The method applies a set of MoDRE notations, namely iStar, UML, and BPMN to represent the process transformation after the introduction of digital technologies, aiming to support the collaborative impact assessment of process reengineering. Following the design science approach [15], we validated the proposed method with domain experts through a pilot study. Currently, the method is being evaluated through action research in 20 real agricultural contexts within the EU-funded project Maximising the co-benefits of agricultural digitalisation through conducive digital ecosystems (CODECS) [16]. In [17], we proposed a formalisation procedure that supports agronomists in applying the method in real-world contexts, and we report our experience. In [18] we introduced ModeLLer, a prototype of an EUD web tool relying on a block-based visual editor that simultaneously operates block-to-model transformation in multiple semi-formal notations while the end-user is assembling blocks following instructions prompted on the block. To evaluate the tool, we carried out a user study asking 30 participants with no previous expertise in formal notations to model a scenario of digital agriculture [19].

In line with previous research [20], the results we achieved thus far confirm that, under proper guidance, non-technical stakeholders can actively contribute to various phases of a RE process — including elicitation and modelling — and that their involvement can add value to the overall process. However, challenges may arise when stakeholders interact with diagrams, as diagrammatic notations can be difficult to understand, and some stakeholders may have a preference for textual descriptions. Furthermore, interpreting notation elements and grasping the overall structure and meaning of diagrams often requires significant time and effort [21].

In this study, we take initial steps toward exploring the extent to which generative large language models (LLMs) can reduce the cognitive effort required to interact with diagrammatic representations. We aim to design a web-based interactive explanatory layer, powered by LLM-generated text, that complements the diagram by offering natural language descriptions and contextual support.

Prior to conducting a user study, we aim to perform a preliminary technical evaluation of the quality of the output produced by LLMs when transforming diagrams into natural language. In this paper, we present our work in progress and we focus on one of the notations included in our socio-technical process modelling method — UML class diagrams — and we execute two prompts in ChatGPT. The first prompt generates a textual description of the model tailored to non-technical users. The second prompt creates a table containing the classes extracted from the diagram, a contextual description of the class role and interaction, a classification of the class, size and coordinates indicating the position of the class within the diagram. Results from the prompts should provide data to build an interactive layer to overlay the diagram with contextual tooltips with class explanation. We execute the prompts both in GPT-4o and GPT-4.1, and we add some variations by testing the prompts with different formats, specifically raster image, vector, and code. We evaluate the output produced by assessing completeness, correctness, understandability, and terminological alignment.

Results show that the output produced is of generally good quality, although it lacks accuracy in describing more granular structures, and there are no substantial differences between the GPT models tested. The complete documentation of this study is available in the supplementary material [22].

II. Background and Related Work

Diagrams are widely used in RE for representing several aspects of system requirements, entities involved, their relationships, components, and interactions. These visual representations relying on MoDRE notations are typically developed by RE experts after requirements elicitation from domain experts. Stakeholders can be subsequently involved in the diagram validation phase, leading to the refinement of the representation to ensure they accurately reflect stakeholders' needs and expectations [23]. This process is essential for achieving shared understanding among developers and stakeholders and minimise the risk of misunderstanding and overlooked requirements before moving on to design and implementation [24].

In the realm of business process modelling (BPM), Leopold et al. [21] addressed the challenge of supporting stakeholders in validating models through automated techniques for model-to-text transformation based on natural language processing. The evaluation was conducted in two phases: a technical evaluation, consisting of a comparative assessment of the structure and semantics of the generated texts against original manual descriptions, and a user study asking participants to reconstruct process models from the generated texts.

The advent of LLMs has provided potential support to BPM, thereby recent research has explored how LLMs, such as GPT, Gemini, LLama, can be leveraged for generating textual descriptions to assist stakeholders with modelling understanding [25], and for streamlining model creation [26]. Kourani et al. [27] proposed ProMOAI, a web tool that leverages LLMs to automatically generate process models from textual descriptions, handling errors and supporting iterative model refinement through user input. The tool interface accepts different input types—textual, model, data—and performs transformation according to input format, as well as, AI provider selected by the user.

Although primarily focused on BPM, these studies suggest that LLMs can serve as interpreters between formal models and human-readable content, a capability that aligns with our focus on interactive class diagram explanations.

A large number of RE studies have recently focused on automating model generation through machine learning techniques and LLMs, particularly targeting the generation of domain models [23], [24], [28], [29]. Ferrari et al. [29] evaluated the use of GPT-3.5 to generate UML sequence diagrams from industrial requirements documents in natural language and evaluated the generated diagrams using five quality criteria: completeness, correctness, adherence to the standard, degree of understandability, and terminological alignment.

Despite the growing body of research in this area, a gap remains in the development of approaches specifically designed to support non-technical stakeholders in diagram interaction. Our work seeks to address this gap by investigating how LLMs can be leveraged to produce textual explanations that support the understanding of diagrams.

III. Technical Evaluation

Adopting the approach by Leopold et al. [21], we conducted a technical evaluation of the output generated by LLMs as a preliminary step before engaging participants in a user study.

A. UML class diagrams

We selected UML class diagrams, which are among the notations incorporated in our *socio-technical process modelling* method. These representations are used to depict the structure of the process to-be, including all involved actors, digital systems, components, and natural resources, as well as the relationships among these entities. As we plan to create an AI-generated web-based layer to integrate into our EUD tool, we selected test data that reflects real-world contexts in which stakeholders may utilise a variety of diagramming tools. In this experiment, we assess diagrams created with: (1) StarUML and (2) draw.io, both recommended in the formalisation procedure [17], and (3) ModeLLer [18], which incorporates the PlantUML library and generates diagrams in the PlantUML style. We added a first variant by considering multiple formats supported by the tools: PNG, SVG, XMI, PlantUML and drawio.

Table I summarises the three real-world cases extracted from the CODECS project, detailing UML structure and format specifications. Diagram 1 represents an IoT system for the remote monitoring of vineyard conditions; diagram 2 depicts a scanner system for real-time soil analysis; diagram 3 refers to a smart irrigation system for remote control of irrigation.

TABLE I: Summary of the three real-world cases extracted from the CODECS project and their format specifications.

Case		Description	UML	Software	Format
1	IoT vineyard monitoring system	An IoT monitoring system based on sensors installed on-field that measures several parameters in the vineyard to optimize organic treatments.	No: nodes #9; arches #12 Types: class, aggregation, direct association	StarUML	png
					svg
					xmi
2	Soil scanner	A soil scanner with sensors and AI-based software that measures soil parameters to optimise fertilisation.	No: nodes #12; arches #20 Types: class, aggregation, direct association	draw.io	png
					svg
					drawio
3	Smart irrigation	An AI-based system to monitor field and weather conditions and manage irrigation	No: nodes #8; arches #9 Types: class, direct association	ModeLLer	png
					PlantUML
					xmi

B. Prompt engineering

For executing prompts, we selected GPT, a prominent LLM that has recently introduced vision capabilities (gpt-image-1), enabling image input understanding. We ran our prompts using the web user interface and adhered to the prompt engineering guidelines provided in the OpenAI Cookbook[1]. In addition, we included a comparative variant by evaluating both GPT-4o — the first model to natively implement gpt-image-1 — and GPT-4.1, which is declared to feature improved instruction following and context examples.

We designed two prompts: the first assesses GPT in describing diagrams; the second explores GPT capability to extract structural and spatial information adding contextual descriptions. The two prompts are as follows:

Prompt 1 "Write a summary that explains the uploaded diagram to a non-technical audience."

Prompt 2 "First, analyse the uploaded UML class diagram. Then, extract all classes and create a table with a row for each class and a column named NAME with the class name. Then, create a column DESCRIPTION containing a 20-30 word description of the class that explains to a non-technical audience the role of the class within the system and operations performed in association with other connected classes. Then, create a column TYPE containing the type of entity represented, either digital, actor/organisation, natural resource or other type of resource. Then, create additional columns: WIDTH, containing the class width; HEIGHT, containing the class height; X, containing the x-position of the class; Y, containing the y-position of the class. Convert all values into relative values. Return the complete table".

In prompt 1, we did not impose length restrictions, as diagrams may vary in size; however we specified the target, consisting of users not having knowledge of formal notations.

Prompt 2 is oriented towards the creation of an interactive layer with tooltips, with content returned in the form of a table. The prompt is articulated in 4 steps requiring respectively to (1) extract classes, (2) write short descriptions based on the class role and neighboring relationships, (3) classify classes into human, natural or technological entities, and (4) extract class size and coordinates within the diagram. Following GPT guidelines, we structured and ran prompt 2 as a chain-of-thought in GPT-4.1, whereas in GPT-4o, we decomposed the prompt into separate, sequential instructions.

[1]http://github.com/openai/openai-cookbook (last accessed: 5 June 2025)

To clarify our objectives, Fig. 1 illustrates an example of the intended output for the interactive layer. While the diagram itself is human-generated, the accompanying explanatory content is produced using the designed prompts.

C. Evaluation criteria

We were inspired by previous work from Ferrari et al. [29] and adopted the four quality criteria that are also relevant to our context. We adapted the statements such that they are applicable to diagram-to-text transformation:

Completeness: the text covers the content of all the (main) entities with a sufficient degree of detail to explain the content of the model to potential stakeholders.

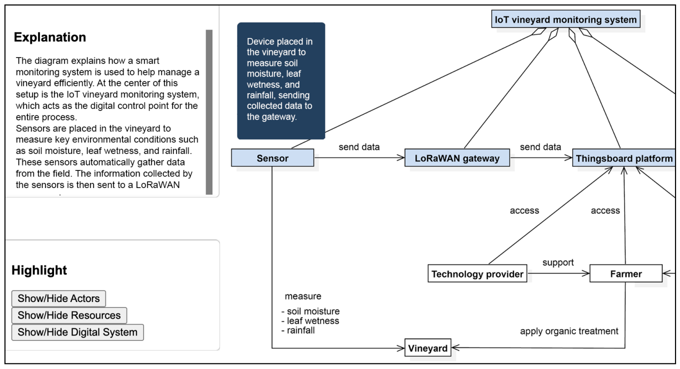
Correctness: the text describes a system structure that is coherent and consistent with the diagram.

Degree of understandability: the text is sufficiently clear, given the complexity of the diagram, and does not contain redundancies.

Terminological alignment: the terminology used in the generated text aligns with the one used in the diagram.

We introduced a fifth criterion which is used to assess coordinates in *Prompt 2*:

Acceptability the extent to which the detected class positions correspond to their true placement in the source model.



Explanation

The diagram explains how a smart monitoring system is used to help manage a vineyard efficiently. At the center of this setup is the IoT vineyard monitoring system, which acts as the digital control point for the entire process.
Sensors are placed in the vineyard to measure key environmental conditions such as soil moisture, leaf wetness, and rainfall. These sensors automatically gather data from the field. The information collected by the sensors is then sent to a LoRaWAN

Device placed in the vineyard to measure soil moisture, leaf wetness, and rainfall, sending collected data to the gateway.

Fig. 1: Example of LLM-generated interactive layer.

D. Evaluation approach

We assessed largely the same criteria across both prompts, introducing minor variations in the evaluation approach.

1) Prompt 1: to evaluate the summaries generated by *Prompt 1*, we assessed each criterion individually. Both prompts were independently evaluated by two raters using a 5-point Likert scale ranging from "1 – Not fulfilled at all" to "5 – Completely fulfilled." The ratings were averaged to determine inter-rater agreement. Additionally, a text field ("*Notes*") was included to document any observations or comments regarding the evaluations.

2) Prompt 2: to evaluate the table obtained from *Prompt 2*, we performed independent evaluations for each column in four steps:

a) Class extraction: this step focuses on completeness, considering the following metrics: true positives (nodes correctly detected), false negatives (nodes missed), and false positives (nodes incorrectly identified). As this evaluation is objective, it is conducted once by a single evaluator.

b) Tooltip description: this addresses three quality criteria: completeness, correctness and degree of understandability, using the same 5-point Likert scale applied in *Prompt 1*, except for completeness, which is assessed objectively as a single accuracy measure based on the ratio of edges mentioned to incident edges. Terminological alignment is not included, as the explanatory and concise nature of tooltips naturally results in terminology differing from the diagram.

c) Classification: this evaluation step is assessed once using a boolean value to indicate correct or incorrect classification, along with a note field for additional observations.

d) Positioning: this evaluation step is addressed through the criterion of acceptability and is evaluated through the 5-point Likert scale applied above, along with the "*Notes*" field.

IV. Execution and Results

We started our prompts with the instruction "Execute this prompt in isolation," requesting to perform actions without considering previous messages. Both prompts were evaluated independently by the first and second author, and the results from both evaluations were averaged.

A. Diagram summary

Prompt 1 was repeated on 18 instances, considering all cases and format variants (see Table I) and GPT models 4o and 4.1. The average word count for all 18 summaries is 220 words, ranging from 176 to 260. Notably, GPT-4.1 outputs are longer (239 words) compared to GPT-4o (202 words).

The average output quality is *high*. Both GPT models achieve scores between 4 and 5 for all dimensions. Overall, there is no significant difference between cases, formats or models. Only the correctness dimension shows slightly more variability, with GPT-4.1 generally yielding constantly better results.

The notes added by the evaluators suggest that several outputs included *extra content not present in the original data*. Furthermore, the text frequently exhibited *subjective commentary* and *interpretive statements*, for example, "more sustainable practices". There were a few *instances of hallucination* (for example, adding unmentioned operations of calibrations), as well as some omissions.

B. Tooltips table

Prompt 2 was executed on 10 instances, covering all cases and both models, with a focus on input formats of type *image*. Thus, only PNG and SVG formats were considered, while XMI, drawio and PlantUML were excluded, as these do not contain the spatial information required for the evaluation. The execution returned 10 tables containing one row for each detected class, and columns with class name, description, classification, size and localisation data (coordinates).

1) Class extraction: Across all cases, GPT-4.1 achieves perfect scores for *precision* (calculated as the number of nodes correctly detected over the number of nodes correctly detected plus the number of nodes wrongly detected) and *recall* (the number of nodes correctly detected over the nodes correctly detected plus the number of nodes missing). In contrast, GPT-4o shows some variability, with lower scores in case 1 for both variants (0.57 and 0.75 for precision; 0.44 and 1.00 recall). For the remaining cases, both GPT models reach perfect scores.

2) Tooltip description: Results for this evaluation step are presented case by case. In *Case 1 - IoT on-field*, GPT-4.1 outperforms GPT-4o across all metrics and formats. For PNG, GPT-4.1 shows higher completeness (0.71 vs. 0.54), correctness (4.22 vs. 2.11), and understandability (4.50 vs. 2.22). In SVG format, Completeness is slightly higher for GPT-4.1 (0.54 vs. 0.46), with correctness similar for both (3.67), while understandability remains higher for GPT-4.1 (4.17 vs. 4.22). In *Case 2 - Smart Irrigation*, completeness differs significantly, with GPT-4o achieving a much higher score (0.91) compared to GPT-4.1 (0.41). However, GPT-4.1 outperforms GPT-4o in correctness (4.38 vs. 2.69) and understandability (4.69 vs. 2.69). In *Case 3 - Soil Scanner*, GPT-4.1 consistently delivers higher correctness and understandability scores (around 4) for both PNG and SVG, although completeness remains low (0.30–0.35) and similar to GPT-4o. In contrast, GPT-4o achieves lower correctness and understandability (2.25–2.38). Notes from the evaluators are similar to those observed in Prompt 1, highlighting some recurring issues and tendencies. Among issues, it is reported that *aggregation relationships are frequently not recognised or properly explained* or *important piece of information are sometimes missing*. Among the tendencies, it is reported *content additions*, such as background, contextual, or qualitative information, and the inclusion of positive or promotional language about technology benefits. Occasional hallucinations are present, such as actions not depicted in the diagram.

3) Classification: The overall accuracy of classification is high across all cases, with an average of 91% (9 errors out of 100 instances). GPT-4.1 achieved perfect accuracy (0% error rate), while GPT-4o committed all 9 errors (18% error rate). Analysing by case, GPT-4o error rates are higher for *Case 1 - IoT on-field* (27.8%) and *Case 3 - smart irrigation* (25%), while for *Case 2 - soil scanner* are lower (8.3%). The errors made by GPT-4o are mainly due to empty values — stemming from unextracted classes, which were counted as errors. However, two instances within Case 3 — weather station and moisture sensor, were classified as *natural* resources, instead of *digital*. This may indicate a more accurate understanding of domain-specific concepts by the newer model.

4) Positioning: Results of the Acceptability of detected class positioning indicate that GPT-4.1 generally achieves higher scores, consistently reaching 4 or above in most instances. In contrast, GPT-4o shows greater variability and generally lower scores, with some values at or near 1. In several cases, all classes were extracted and positioned close to their actual locations; however, overlaps or minor misalignments, especially in y-positioning for crowded areas — were observed, while x-positioning was generally more accurate. Notably, there are exceptions — such as in Case 2, format svg —where GPT-4o outperforms GPT-4.1. This output demonstrated precise mapping with boxes aligned with class centres and correct sizing.

V. Discussion

Results from prompt executions indicate GPT capability of producing quality content, with neither model consistently outperforming the other across all criteria. GPT-4.1 tends to generate more verbose text and interpretations, while offering greater accuracy in class detection, classification and positioning. Instead, GPT-4o occasionally achieves higher completeness but manifests more variability and reduced quality in other dimensions. In line with previous research [25], [29], we observe limitations in contextual understanding and in the ability to capture granular information, such as aggregations and labels. Additionally, some instances of hallucinations were observed, although these could be related to the tendency to add contextual information and interpretations.

VI. Conclusion, Limitations and Future Work

In this study, we conducted an initial investigation into the extent to which LLMs may support stakeholders in interacting with diagrams. We acknowledge several limitations, both in terms of the small number of cases evaluated, the number of UML elements contained, and LLMs assessed. Despite these limitations, the results confirm the feasibility of using GPT models in our specific real-world context and conducting a study with users. On the one hand, the results obtained help inform the design of specific features in future applications, such as adding warnings informing that the interactive layer is AI-generated, or enabling users to choose model configurations, as already offered by [27]. On the other hand, results outline a roadmap for future work, which could involve experimenting with additional models, testing variable prompt strategies, and incorporating further evaluation criteria such as text complexity.

References

[1] S. Assar, "Model driven requirements engineering: Mapping the field and beyond," in *2014 IEEE MoDRE*, 2014.

[2] "Unified modeling language (UML) 2.5.1 core specification," https://www.omg.org/spec/UML, 2017, accessed: 05-31-2025.

[3] J. Horkoff, F. B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos, and P. Giorgini, "Goal-oriented requirements engineering: An extended systematic mapping study," *Requir. Eng.*, vol. 24, no. 2, p. 133–160, jun 2019.

[4] E. Yu, P. Giorgini, N. Maiden, J. Mylopoulos, and S. Fickas, *Social Modeling for Requirements Engineering*. The MIT Press, 2011.

[5] E. Gonçalves, L. Monte, S. Souza, M. de Oliveira, and J. Araujo, "A systematic literature review of kaos extensions," in *REFSQ*, 2025.

[6] D. Amyot, "Amyot, d.: Introduction to the user requirements notation: Learning by example." *Computer Networks*, 2003.

[7] "Business process model and notation (BPMN) v2.0," https://www.omg.org/spec/BPMN/2.0/, 2010, accessed: 05-31-2025.

[8] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven software engineering in practice*. Morgan & Claypool Publishers, 2017.

[9] A. Ferrari, M. Bacco, K. Gaber, A. Jedlitschka, S. Hess, J. Kaipainen, P. Koltsida, E. Toli, and G. Brunori, "Drivers, Barriers and Impacts of Digitalisation in Rural Areas from the Viewpoint of Experts," *IST*, 2022.

[10] M. Ryan, G. Isakhanyan, and B. Tekinerdogan, "An interdisciplinary approach to artificial intelligence in agriculture," *NJAS*, vol. 25, 01 2023.

[11] A. Ferrari, F. Lepore, L. Ortolani, and G. Brunori, "Eliciting the double-edged impact of digitalisation: a case study in rural areas," in *RE 2023*.

[12] J. Doerr, A. Hess, and M. Koch, "Re and society - a perspective on re in times of smart cities and smart rural areas," in *RE'18*. IEEE, 2018.

[13] B. R. Barricelli, F. Cassano, D. Fogli, and A. Piccinno, "End-user development, end-user programming and end-user software engineering: A systematic mapping study," *JSS*, vol. 149, pp. 101–137, 2019.

[14] C. Mannari, M. Bacco, G. O. Spagnolo, A. Malizia, and A. Ferrari, "Towards a method for modelling socio-technical process transformation in digital agriculture," in *REW 2024*.

[15] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*, 01 2014.

[16] "Codecs," https://www.horizoncodecs.eu/, accessed: 05-31-2025.

[17] C. Mannari, M. Sportelli, H. Meesala, O. F. Okoye, F. Lepore, M. Bacco, G. Brunori, A. Malizia, and A. Ferrari, "End-user requirements modelling: An experience report from digital agriculture," in *REFSQ 2025*.

[18] C. Mannari, E. Anichini, M. Bacco, A. Ferrari, T. Turchi, and A. Malizia, "Modeller – enabling end-users to model systems: a case study in digital agriculture," in *AVI 2024*.

[19] C. Mannari, T. Turchi, C. Frosali, M. Bacco, A. Ferrari, and A. Malizia, "Assessing computational thinking skills through artefacts: the case of modeller," in *IS-EUD 2025*, to appear.

[20] U. Abelein and B. Paech, "Understanding the influence of user participation and involvement on system success – a systematic mapping study," *Empirical Softw. Engg.*, vol. 20, no. 1, p. 28–81, Feb. 2015.

[21] H. Leopold, J. Mendling, and A. Polyvyanyy, "Supporting process model validation through natural language generation," *IEEE TSE*, vol. 40, 2014.

[22] C. Mannari, T. Turchi, M. Bacco, and A. Malizia, "Assisting stakeholders in class diagram interpretation with llms: a work in progress - replication package [data set]. zenodo." https://doi.org/10.5281/zenodo.15684118, 2025, accessed: 17-06-2025.

[23] R. Saini, G. Mussbacher, J. L. C. Guo, and J. Kienzle, "Automated, interactive, and traceable domain modelling empowered by artificial intelligence," *Softw. Syst. Model.*, vol. 21, no. 3, 2022.

[24] M. Bragilovski, A. T. Van Can, F. Dalpiaz, and A. Sturm, "Deriving domain models from user stories: Human vs. machines," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, 2024.

[25] N. Klievtsova, J. Mangler, T. Kampik, and S. Rinderle-Ma, "Utilizing process models in the requirements engineering process through model2text transformation," in *RE 2024*.

[26] H. Kourani, A. Berti, D. Schuster, and W. M. P. van der Aalst, "Process modeling with large language models," in *Enterprise, Business-Process and Information Systems Modeling*, H. van der Aa, D. Bork, R. Schmidt, and A. Sturm, Eds. Cham: Springer Nature Switzerland, 2024.

[27] H. Kourani, A. Berti, D. Schuster, and W. M. van der Aalst, "Promoai: Process modeling with generative ai," in *IJCAI*, 2024.

[28] G. Çelikmasat, A. Özgövde, and F. B. Aydemir, "Generating domain models with llms using instruction tuning: A research preview," in *REFSQ*, A. Hess and A. Susi, Eds. Springer Nature Switzerland, 2025.

[29] A. Ferrari, S. Abualhaija, and C. Arora, "Model generation with llms: From requirements to uml sequence diagrams," in *2024 IEEE REW*.