2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC) | 979-8-3503-7696-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/COMPSAC61105.2024.00017

LLM-based Class Diagram Derivation from User Stories with Chain-of-Thought Promptings

Yishu Li[1], Jacky Keung[1], Xiaoxue Ma[1], Chun Yong Chong[2], Jingyu Zhang[1], and Yihan Liao[1]
[1]Department of Computer Science, City University of Hong Kong, Hong Kong, China,
{yishuli5-c, xiaoxuema3-c, jzhang2297-c, yihanliao4-c}@my.cityu.edu.hk
{jacky.keung}@cityu.edu.hk
[2]School of Information Technology, Monash University Malaysia, Malaysia,
{chong.chunyong}@monash.edu

***Abstract*—In agile requirements engineering, user stories are the primary means of capturing project requirements. However, deriving conceptual models, such as class diagrams, from user stories requires significant manual effort. This paper explores the potential of leveraging Large Language Models (LLMs) and a tailored Chain-of-Thought (CoT) prompting technique to automate this task. We conducted a comprehensive preliminary study to investigate different prompting techniques applied to the task. The study involved comparing LLM-based approaches with guided and unguided human extraction to evaluate the effectiveness of the proposed LLM-based techniques. Our findings demonstrate that LLM-based approaches, particularly when combined with well-crafted few-shot prompts, outperform guided human extraction in identifying classes. However, we also identified areas of suboptimal performance through qualitative analysis. The proposed CoT prompting technique offers a promising pathway to automate the derivation of class diagrams in agile projects, reducing the reliance on manual effort. Our study contributes valuable insights and directions for future research in this field.**

***Index Terms*—Requirements engineering, user story, large language models, chain of thought prompting**

I. Introduction

Requirements engineering (RE) serves as the foundation for the Software Engineering (SE) process, enabling the effective development of software systems. In agile RE practices, user stories have emerged as a highly popular technique for capturing requirements from a user's perspective [1]. These user stories play a central role as artifacts that guide the subsequent stages of software development [2]. Notably, user stories can undergo refinement to transform them into more detailed and specific specifications [3]. One effective approach for this refinement process is the derivation of conceptual models, such as Unified Modeling Language (UML) diagrams [4].

Bragilovski et al. [3] proposed guidelines aimed at enhancing flexibility and reducing cognitive effort in the task of translating user stories into UML class diagrams. However, despite these advancements, human involvement continues to be necessary. Large Language Models (LLMs) have demonstrated exceptional proficiency in Natural Language Processing (NLP) tasks, making them highly relevant in the field of RE, where effective handling of natural language is critical. In particular, ChatGPT has gained significant traction in RE [5], being employed for tasks such as identifying inconsistencies in natural language requirements [6].

Based on the aforementioned considerations, the main objective of this study is to assess the effectiveness of LLMs in the process of deriving class diagrams within the agile RE context. The design of appropriate prompts plays a crucial role in maximizing the effectiveness of LLMs [7]. One state-of-the-art prompting technique that has shown promising results is Chain-of-Thought (CoT) prompting, as introduced in [8].

We aim to enhance the reliability and comprehensibility of the LLM-based approach for deriving class diagrams from user stories. To reduce the need for human intervention in the process, we have developed a three-phase CoT prompting incorporated with the LLM approach for automating the derivation of class diagrams. This proposed method involves three key phases in its reasoning steps: class identification, attribute identification, and relationship identification. User stories sourced from [3] are utilized as input for the derivation process, and the resultant outputs are formatted to demonstrate identified classes, attributes, and relationships. These generated outputs serve as a reference for the subsequent construction of the class diagram. Following the derivation of class diagrams, we conducted a comprehensive analysis of the generated output. This analysis covered both quantitative and qualitative aspects to evaluate the effectiveness and accuracy of the LLM-based approach.

Overall, our findings support the potential of the LLM-based approach to automate certain aspects of class diagram derivation, particularly in class and relationship identification. However, more attention and investigation are needed to overcome the limitations and further enhance the accuracy and reliability of the LLM-based approach. These insights contribute to the ongoing exploration of leveraging LLMs in agile RE. Our contributions are summarized as follows:

- We investigate prompting techniques for the LLM-based derivation of class diagrams from user stories.
- We introduce a novel prompting technique tailored explicitly for the derivation of class diagrams.
- We present comprehensive case studies evaluating the efficacy of different prompting techniques, which yield valuable insights that merit further exploration in the context of agile requirement analysis.

2836-3795/24/$31.00 ©2024 IEEE
DOI 10.1109/COMPSAC61105.2024.00017

The rest of the paper is organized as follows. Section II discusses the background and related work. Section III introduces the proposed CoT framework, and Section IV demonstrates the experiment design settings. Section V showcases the preliminary quantitative results. Furthermore, we provide a comprehensive discussion on Section VI. Finally, Section VII concludes our study.

II. Background and Related Work

A. Derivation of Class Diagram

User stories serve as a prevalent notation for expressing requirements in agile projects [1]. Each user story delineates fundamental elements of a requirement: *who* it is for, *what* is expected from the system, and optionally, *why* it is important [9]. We adopted the most popular and widely adopted Connextra template [10]: "As a ⟨role⟩, I want to ⟨goal⟩, [so that ⟨reason⟩]."

There is a scarcity of methods for deriving conceptual models from user stories [3], where most of the work relies on human effort [11, 12]. Lucassen et al. [13] proposed an automated approach, leveraging the Visual Narrator tool, for extracting structural conceptual models. However, its implementation poses challenges due to the extensive array of linguistic patterns inherent in natural language. Based on the previous work, Bragilovski et al. [3] designed example-based guidelines to derive conceptual models from user stories. The results indicate that these guidelines enhance the ability to derive conceptual models in cases of medium complexity. Notably, these approaches mainly rely on human effort.

B. Chain of Thought Prompting

The emergency of LLMs brings a chance to transition from manual efforts to an automated workflow through well-designed prompt input. Establishing accurate class diagrams serves as a foundational step in the continuous development process. The CoT prompting technique [8] is a relatively recent advancement in prompting methods. CoT prompting involves instructing LLMs to generate intermediate natural language reasoning steps, or a CoT, before producing the final output. This technique has demonstrated state-of-the-art performance in natural language generation tasks and has prompted further research in related areas, such as self-consistency prompting [14] and least-to-most prompting [15]. It is important to note that these prompting techniques have not been specifically tailored or extensively explored for the task of deriving class diagrams from user stories.

III. Methodology

In this section, we introduce key components of the CoT prompting methodology, which focuses on class identification, attribute identification, and relationship identification.

A. Steps of Class Diagram Derivation

In accordance with the guidelines presented in [3], we characterize the output of the LLM-based class diagram as triplet format, consisting of three components: *class*, *attribute*, and *relationship*. Each component represents a description that corresponds to the respective element in a finalized class diagram. In a typical human-driven process for class diagram derivation, three identification phases are involved [3]. The first is identifying classes from the given requirements. Subsequently, based on the identified classes, associated attributes are extracted. Lastly, the relationships between classes are defined to culminate in a comprehensive class diagram.

B. CoT Prompting

Based on practical observations and aligned with the human extraction guidelines proposed in [3], we introduce a tailored CoT prompting technique for deriving class diagrams from user stories. This prompting technique encompasses three components: context, CoT examples, and instruction message.

1) Context: The context component of the CoT prompting provides the necessary information and background for the LLMs to understand the task at hand, which adheres to the recommended prompting practices outlined by OpenAI, incorporating contextual information within the prompt to guide the LLM's behavior and the expected output format.

Noteworthy, the class diagram produced by LLM is at the conceptual level rather than the implementation level. Typically, the conceptual class diagram poses greater challenges in derivation, particularly when the development team lacks familiarity with the problem domain. Hence, the focus of our study is on providing a high-level description of the models, which serves as a compass for subsequent stages of code generation or more exhaustive detailed design endeavors.

2) CoT Examples: The CoT prompting consists of intermediate reasoning steps that provide examples that demonstrate the thought process involved in deriving class diagrams. These examples serve as guidance for the LLMs in generating coherent CoT and reasoning. Within the prompting framework, the CoT steps are seamlessly integrated into illustrative prompt examples. Each example is composed of three segments.

- ⟨input⟩ corresponds to the requirement input formatted as a user story. It provides the initial context for the LLMs to understand the specific task.
- ⟨CoT⟩ aligns with the three steps involved in class derivation. It encompasses the distinct reasoning steps and instructions for each of these identification steps.
- ⟨output⟩ represents the anticipated output from the LLM. It encapsulates the outcomes of the three components and provides an example of the desired descriptions.

a) Class Identification: The initial phase involves the identification of classes. In the illustrative CoT examples, special attention is given to the designation of a ⟨role⟩ as the actor in the user story. The instructional CoT directs the LLM to assess whether the actor engages in operations on the system. If the actor is involved in system operations, it is designated as a class. The process primarily entails discerning the primary entity from the user stories. This is typically done by identifying nouns explicitly mentioned within the narratives. These nouns represent the predominant entities within the domain and are often indicative of potential classes.

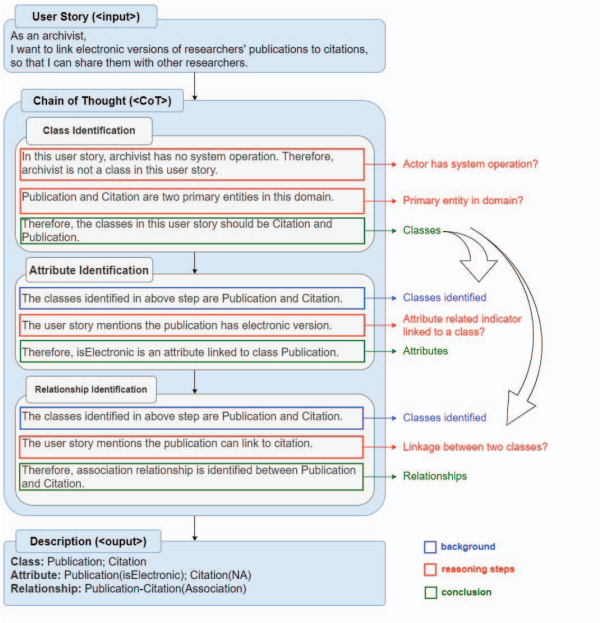

Fig. 1: An example of three-step CoT prompting

b) Attribute Identification: The succeeding phase involves the identification of attributes associated with the entities. In the process of attribute identification, the attribute-related indicator is typically articulated as a noun containing less intricate information. Another prevalent scenario involves a noun signifying a status, denoting that the primary entity possesses such a status. In such instances, the noun indicative of the status should be recognized as the attribute.

c) Relationship Identification: The final phase involves the identification of relationships among entities, with certain verbs playing a contributory role in defining these relationships. Consistently, we initiate the process by introducing background messages. Building upon the identified classes, the prompting message guides the LLM in discerning messages indicating the connection between the two classes. In the examples of CoT prompting, two frequently utilized relationships, namely *association* and *generalization*, are presented.

3) Instruction: Following the prescribed steps, the prompt incorporates instructions directing the LLM to generate an output that enumerates three essential components of the class diagram: *class*, *attribute*, and *relationship*.

C. Implementation

Our proposed CoT prompting is a prompting technique designed for the derivation of class diagrams from user stories, and it is not contingent on specific LLM. In this study, we employ *gpt-3.5-turbo* as the default LLM.

IV. Experiment

A. Research Questions

To gain a comprehensive understanding of LLM-based techniques, we have formulated the following research questions.

RQ1. How are different prompting techniques performed in deriving class diagrams from user stories?
RQ2. What effective of different settings in CoT prompting can impact the performance of driving class diagrams?
RQ3. What is the performance of CoT prompting in distinct cases?

B. Experiment Design

1) Dataset: Following case studies conducted in [3, 16], we select the two projects from user stories collected in [17] to evaluate our proposed framework. Project Data Hub (DH) consists of 20 user stories delineating the functionalities of a web interface designed for dataset operations. Meanwhile, Project Planning Poker (PP) encompasses 22 user stories outlining the requirements for the planningpoker.com website.

2) Comparison Baselines: To answer RQ1 in assessing the effectiveness of our approach, we select three mainstream prompting techniques as baselines.

- **Zero-shot Prompting (ZSP)** directly feeds context and instruction to the LLM without examples.
- **Few-shot Prompting (FSP)** introduce some examples with ⟨input⟩ and ⟨output⟩ pairs fed into the LLM.
- **CoT Prompting (CoT_P)** is a variant of few-shot prompting [8]. It exemplified the reasoning steps ⟨CoT⟩ together with ⟨output⟩ as presented in Figure 1.

3) Settings: To answer RQ2, we designed comprehensive experiments to evaluate the effectiveness of different settings introduced to CoT prompting.

a) Prompting interactions: In the provided prompting examples, the CoT incorporates distinct cognitive steps. To investigate the variances arising from presenting the prompt in a single step versus segregating the examples step-by-step, we formulated two modes of prompting interaction as inputs. In the one-step prompting approach, each example encapsulates all three CoT steps before generating the output. Conversely, in the multiple-step prompting approach, each example encompasses three iterations of the interaction.

b) Number of examples: The CoT technique, a variant of few-shot prompting [8], is susceptible to the influence of the number of examples provided [18]. Based on such an assumption, we formulated distinct example templates as inputs, considering token constraints. The initial template encompasses 5 examples distributed across 3/2/2 scenarios for class, attribute, and relationship identification, respectively. Subsequently, the second template comprises 6 examples (with 3/3/2 scenarios), introducing an additional instance with an extra scenario specifically focused on attribute identification.

C. Evaluation

1) Metrics: The two most frequent variables, taken from conceptual modeling research [19, 20], used for measuring conceptual models are *validity* and *completeness*.

True Positive (TP) signifies the number of instances present in the solution generated by the LLM that also aligns with the instances in the gold standard. Conversely, False Positive (FP) denotes the count of instances in the LLM-generated solution

that do not correspond to instances within the gold standard. The metric ***validity*** (val) is defined as the ratio of TP to the total instances in the LLM-based solution. In information retrieval terms, validity equates to precision: $val = \frac{|TP|}{|TP|+|FP|}$.

The term False Negative (FN) denotes the count of instances that exist in the gold standard but cannot be identified in LLM-generated output. The metric ***completeness*** (com) is defined as the ratio between TP and the number of instances in the gold standard. In information retrieval terms, completeness equates to recall: $com = \frac{|TP|}{|TP|+|FN|}$.

Given that the identification of attributes and relationships is contingent upon the prior identification of connected class entities, we followed the *adjusted version* of val and com introduced in [3, 16], which calculates them specifically in relation to those attributes and relationships within the gold standard that pertain to identified class entities.

2) Human Evaluation: Our authors and invited practitioners systematically compared outputs from LLM-based approaches with the established gold standard sourced from [3].

V. Results

A. Answer to RQ1

In response to RQ1, we assess the *adjusted* completeness and validity of LLM-based solutions compared to two human extraction outcomes as documented in [3]: 1) **Guided Derivation (GD)**, involving evaluation of class diagrams formulated by students following example-based guidelines; 2) **Unguided Derivation (UD)**, where students developed class diagrams without additional guidance. Comparative results for Project PP and DH are summarized in Table I, with bold items indicating superior performance in prompting techniques and results surpassing human extraction highlighted in grey.

TABLE I: Completeness and Validity

Project	Approaches	Cls		Att		Rel	
		val	com	val	com	val	com
PP	ZSP	0.75	0.67	0.33	0.50	0.20	0.11
	FSP	0.75	0.67	0.40	0.50	0.75	0.38
	CoT_P	**0.78**	**0.78**	**1.00**	0.50	**1.00**	**0.40**
	GD	0.78	0.53	0.6	0.49	0.59	0.57
	UD	0.74	0.57	0.53	0.51	0.56	0.59
DH	ZSP	**0.75**	0.60	0.17	0.17	0.30	0.27
	FSP	0.67	0.67	**0.50**	**0.33**	0.60	0.40
	CoT_P	0.65	**0.73**	**0.50**	**0.33**	**0.75**	**0.50**
	GD	0.67	0.37	0.58	0.31	0.50	0.36
	UD	0.62	0.36	0.44	0.29	0.37	0.33

In terms of class identification, the LLM-based approaches show higher completeness compared to guided and unguided manual derivation in both two projects. However, for attribute identification, the LLM-based methods do not perform as well as human extraction in Project PP. In some cases, few-shot and CoT prompting techniques yield a higher count of completed attributes than guided human extraction in Project DH. Similarly, regarding relationship identification, the prompting LLM method does not generate satisfactory results in Project PP, but the few-shot and CoT prompting techniques improve LLM's efficiency. In Project DH, these techniques achieve better completeness than human extraction.

It is evident that CoT prompting attains the best completeness and validity across all three sub-tasks, with the exception of class identification in Project DH.

B. Answer to RQ2

To address RQ2, we present the results of different settings applied to the prompting techniques, as demonstrated in Table II and III. The bold items indicate settings with better completeness introduced in CoT prompting.

Concerning Project PP, the outcomes derived from one-step and multi-step interactive inputs exhibit similarity. Notably, the one-step input demonstrates superior performance in the completeness of relationship identification. In Project DH, the one-step input yields greater completeness in class and attribute identification, with no discernible difference in relationship identification. Overall, the one-step input proves to enhance completeness in the class diagram derivation.

TABLE II: Effectiveness of Prompting Interaction

Project	Interaction	Cls		Att		Rel	
		val	com	val	com	val	com
PP	One	0.78	0.78	1.00	0.50	1.00	**0.40**
	Multiple	0.71	0.78	1.00	0.50	1.00	0.29
DH	One	0.65	**0.73**	0.50	**0.33**	0.75	0.50
	Multiple	0.63	0.33	0.33	0.17	0.60	0.50

In Table III, *Ex.* and *Scn.* refer to the number of examples and covered scenarios, respectively. Regarding the augmentation of examples to encompass more scenarios in attribute identification, there is an observed improvement in completeness for class identification. However, this augmentation does not yield a comparable improvement in attribute and relationship identifications for Project PP. Conversely, in Project DH, introducing more scenarios enhances the capacity to identify attributes and relationships.

TABLE III: Effectiveness of Number of Examples

Project	Ex.	Scn.	Cls		Att		Rel	
			val	com	val	com	val	com
PP	5	3/2/2	0.83	0.56	1.00	0.50	1.00	0.40
	6	3/3/2	0.78	**0.78**	1.00	0.50	1.00	0.40
DH	5	3/2/2	0.59	0.67	0.50	0.17	0.83	0.42
	6	3/3/2	0.65	**0.73**	0.50	**0.33**	0.75	**0.50**

VI. Discussion

To address RQ3, we conducted case studies on the sub-tasks to provide an in-depth analysis of the potential causes.

A. Case Study on Class Identification

Table IV highlights the distinction between classes identified solely through CoT prompting (in red) and those identified exclusively through zero-shot and few-shot prompting (in blue).

It reveals summarization-related entities that are ascertainable only through CoT prompting. Figure 2 presents a detailed user story delineating two policy types, which remain unidentified by the LLM when employing zero-shot and few-shot promptings. This observation aligns with constrained human extraction reported in [3], primarily due to their absence in explicit terms within user stories and their singular occurrence. Integrating with CoT prompting, the LLM elucidates the reasoning steps that encapsulate the summarization, denoting them as classes *EstimatePolicy1* and *EstimatePolicy2*.

Finding 1: Incorporating CoT prompting improves the proficiency of LLMs in identifying classes associated with summarization.

TABLE IV: Partial / Un-identified Classes

Project	Partial Identified	Unidentified
PP	Round; EstimatePolicy1; EstimatePolicy2	EstimationPolicy
DH	Tag; PublishedDataPackage; Example	SiteDeployment

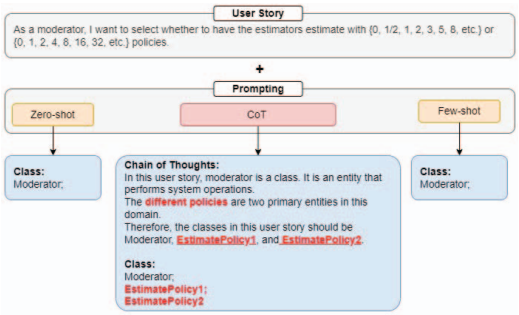

Fig. 2: Case study on class identification (PP)

Reviewing unidentified classes in Table IV, *EstimationPolicy* encompasses both *EstimatePolicy1* and *EstimatePolicy2*, demanding a more nuanced summarization of knowledge, posing challenges for LLM-based methodologies. Similarly, *SiteDeployment* aggregates deployment behaviors derived from user stories, necessitating domain knowledge. Notably, these classes were also identified to a restricted extent through human extraction, as reported in [3].

Finding 2: The current prompting methodologies utilized in the LLM demonstrate a limitation in their capacity to perform aggregation summarization.

B. Case Study on Attribute Identification

Table V presents the erroneously identified attributes. In Project DH, the description-related attributes are easily incorrectly identified as classes. However, such entities contain less information, which is better to be defined as attributes to reduce the class diagram complexity. Figure 3 demonstrates a detailed instance. The CoT generated in this case reveals that the LLM initially misidentified classes and subsequently, based on this erroneous identification, inaccurately identified the attributes. Despite providing examples specifically tailored to description-related scenarios in examples, the LLM appears unable to assimilate the information.

TABLE V: Erroneously / Un-Identified Attributes

Project	Erroneously Identified	Unidentified
PP	N/A	Item (isEstimated); Estimate (isAgreedUpon)
DH	DataPackage (descriptor, resource, isValidate)	PublishedDataPackage (isSharedPublicly)

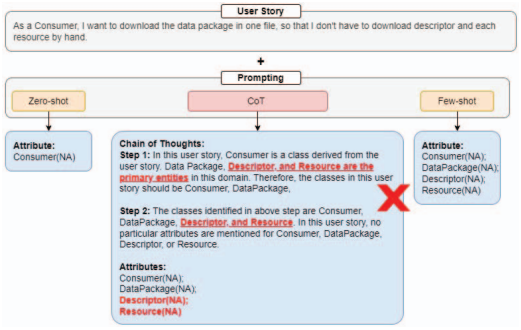

Fig. 3: Case study on attribute identification (PP)

Finding 3: LLM-based approaches tend to misidentify information categorized under the description type as classes instead of attributes, leading to erroneous classification.

Reviewing both projects, attributes were identified to a limited extent. As presented in Table V, the unidentifiable attributes are mainly associated with those related to status (e.g., *isEstimated*, and *isSharedPublicly*). Despite explicitly mentioning a singular type of status in the user stories, the LLM encounters difficulties comprehending them as attributes signifying various states. Even integrating with proposed CoT examples, where specific instances were introduced to guide the LLM in determining whether the entity possesses different statuses, such information is still overlooked.

Finding 4: When employing LLM-based approaches, the description related to the status can often be easily overlooked during attribute identification.

C. Case Study on Relationship Identification

Table VI delineates either undefined or erroneously identified relationships. The validity of relationship identification for Project PP closely approximates 1, indicating accurate identification. In Project DH, user stories pertaining to the *Consumer* and *Publisher* classes suggest that the *Consumer* can manipulate the profile data of the *Publisher*. However, in the gold standard, the *Consumer* and *Publisher* are intended to operate through the class *Account*. Such an operational association is latent within the messaging and lacks overt noun references, resulting in an erroneous identification. The LLM-generated solutions directly connect the *Consumer* and the *Publisher*.

Finding 5: The interpretation of associations between classes, particularly in situations where explicit noun references for intermediate operations are not provided, can result

TABLE VI: Erroneously / Un-Identified Relationships

Project	Erroneously Identified	Unidentified
PP	N/A	Game-EstimatorPolicy (Association)
DH	Publisher-Consumer (Association)	Account-Consumer (Association); DataPackage-PublishedDataPackage (Inheritance)

in misinterpretation when employing LLM-based methods for relationship identification.

In Project DH, the LLM-generated solutions fail to identify the inheritance relationship. Figure 4 illustrates CoT in detail. While *PublishedDataPackage* is identifiable, the LLM struggles to integrate contextual cues to recognize the inheritance between *DataPackage* and *PublishedDataPackage*, where the latter inherits from the former. As a result, this inheritance relationship remains unidentified. Furthermore, the LLM erroneously associates *Publisher* with *PublishedDataPackage* instead of *DataPackage*.

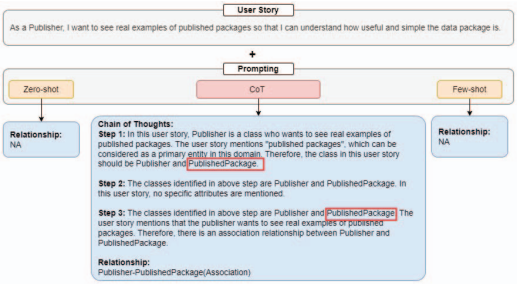

Fig. 4: Case study on relationship identification (DH)

Finding 6: During the class identification step, the contextual information often remains elusive, consequently resulting in the subsequent omission of the associated relationship.

VII. Conclusion

In this paper, we have proposed a CoT prompting for deriving conceptual class diagram components and investigated the efficacy of LLMs utilizing different prompting techniques in this task. The experimental results showed that the CoT prompting improved the accuracy and completeness of class diagram derivation compared to traditional prompt-based methods introduced in LLMs. The CoT prompting technique demonstrated better performance in terms of completeness compared to guided and unguided manual extraction for class identification. However, there were limitations observed in the attribute identification. Through the qualitative case studies, we gained deeper insights into the strengths and limitations of regular prompting and CoT prompting techniques. Overall, this research offers valuable insights to automate class derivation from user stories, utilizing the LLM-based technique.

Acknowledgment

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (6000796, 9229109, 9229098, 9220103, 9229029).

References

[1] G. Lucassen, F. Dalpiaz, J. M. E. v. d. Werf, and S. Brinkkemper, "The use and effectiveness of user stories in practice," in *Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings 22*. Springer, 2016, pp. 205–222.

[2] L. Müter, T. Deoskar, M. Mathijssen, S. Brinkkemper, and F. Dalpiaz, "Refinement of user stories into backlog items: Linguistic structure and action verbs: Research preview," in *Requirements Engineering: Foundation for Software Quality: 25th International Working Conference, REFSQ 2019, Essen, Germany, March 18–21, 2019, Proceedings 25*. Springer, 2019, pp. 109–116.

[3] M. Bragilovski, F. Dalpiaz, and A. Sturm, "Guided derivation of conceptual models from user stories: A controlled experiment," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2022, pp. 131–147.

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

[5] J. Zhang, Y. Chen, N. Niu, and C. Liu, "A preliminary evaluation of chatgpt in requirements information retrieval," *arXiv preprint arXiv:2304.12562*, 2023.

[6] A. Fantechi, S. Gnesi, L. Passaro, and L. Semini, "Inconsistency detection in natural language requirements using chatgpt: a preliminary evaluation," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, 2023, pp. 335–340.

[7] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 697–12 706.

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[9] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Improving agile requirements: the quality user story framework and tool," *Requirements engineering*, vol. 21, pp. 383–403, 2016.

[10] F. Dalpiaz and S. Brinkkemper, "Agile requirements engineering with user stories," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*. IEEE, 2018, pp. 506–507.

[11] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–41, 2021.

[12] J. Berends and F. Dalpiaz, "Refining user stories via example mapping: an empirical investigation," in *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 2021, pp. 345–355.

[13] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. Van Der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with visual narrator," *Requirements Engineering*, vol. 22, pp. 339–358, 2017.

[14] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[15] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.

[16] F. Dalpiaz, P. Gieske, and A. Sturm, "On deriving conceptual models from user requirements: An empirical study," *Information and Software Technology*, vol. 131, p. 106484, 2021.

[17] F. Dalpiaz, I. Van Der Schalk, S. Brinkkemper, F. B. Aydemir, and G. Lucassen, "Detecting terminological ambiguity in user stories: Tool and experimentation," *Information and Software Technology*, vol. 110, pp. 3–16, 2019.

[18] T. Cao, M. Law, and S. Fidler, "A theoretical analysis of the number of shots in few-shot learning," *arXiv preprint arXiv:1909.11722*, 2019.

[19] O. I. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE software*, vol. 11, no. 2, pp. 42–49, 1994.

[20] S. España, M. Ruiz, and A. González, "Systematic derivation of conceptual models from requirements models: a controlled experiment," in *2012 Sixth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2012, pp. 1–12.