Deriving Domain Models from User Stories: Human vs. Machines

Maxim Bragilovski
Ben-Gurion University of the Negev
Israel
Email: maximbr@post.bgu.ac.il

Ashley T. van Can and Fabiano Dalpiaz
Utrecht University
The Netherlands
Email: {a.t.vancan, f.dalpiaz}@uu.nl

Arnon Sturm
Ben-Gurion University of the Negev
Israel
Email: sturm@bgu.ac.il

***Abstract*—Domain models play a crucial role in software development, as they provide means for communication among stakeholders, for eliciting requirements, and for representing the information structure behind a database scheme or at the basis of model-driven development. However, creating such models is a tedious activity and automated support may assist in obtaining an initial domain model that can later be enriched by human analysts. In this paper, we propose an experimental comparison of the effectiveness of various approaches for deriving domain models from a given set of user stories. We contrast human derivation with machine derivation; for the latter, we compare (i) the Visual Narrator: an existing rule-based NLP approach; (ii) a machine-learning classifier that we feature engineered; and (iii) a generative AI approach that we constructed via prompt engineering. Based on a benchmark dataset that consists of nine collections of user stories and corresponding domain models, the evaluation indicates that no approach matches human performance, although a tuned version of the machine learning approach comes close. To better understand the results, we qualitatively analyze them and identify differences in the types of false positives as well as other factors that affect performance.**

I. Introduction

Domain models provide a holistic view of the environment where the system to-be will operate. These models serve multiple purposes, as they facilitate communication with developers and stakeholders, may be used to uncover missing requirements, and are a blueprint for the design of a database scheme or for adopting model-driven development.

The construction of domain models is usually a human activity. Analysts create domain models on the basis of domain knowledge and documentation, interview notes, existing requirements and, in general, any information that an analyst possesses about the domain at hand [1].

The systematic mapping by Zhao *et al.* [2] shows the existence of many NLP-powered approaches for constructing conceptual models from written text, including documentation, domain descriptions, and requirements. We refer to this process as *domain model derivation*, and this paper focuses on the derivation of such domain models *from a set of user stories.*

The existing literature on deriving domain models from requirements comprises both *manual* approaches that provide guidelines for extraction models [3], [4], [5] as well as *automated* approaches that include rule-based solutions [6], [7] and machine learning solutions [8].

On the one hand, deriving domain models automatically can bring several benefits, as it saves time and effort. Furthermore, it might improve consistency and completeness, as manual derivation may lead to omitting certain concepts, especially when starting from a large collection of requirements.

On the other hand, deriving domain models automatically is challenging due to the ambiguity and impreciseness of requirements expressed in natural language. Moreover, as observed by Arora and colleagues, the notion of relevance is inherently subjective [5], although this can be partially mitigated via clear guidelines [3], [4].

In this paper, we investigate the relative effectiveness of several automated approaches, compared to human performance, for the task of deriving domain models that capture the core structural concepts and relationships from a set of user stories [9]. Our research question (RQ) is: *How do automated approaches for the derivation of domain models from user stories compare to manual derivation by human analysts?*

To answer the RQ, we develop a machine learning approach to derive domain models and we construct a prompt procedure that allows ChatGPT to derive domain models. We then compare the results with the state-of-the-art rule-based approach from the literature (Visual Narrator [10]) and with human performance (manually derived domain models).

Thus, the paper includes the following contribution:

1) We propose a machine-learning-based approach and prompts to generative models (ChatGPT 4.0) for the derivation of domain models from user stories;
2) We assess the relative performance of the approaches to derive domain models by measuring their completeness and validity [11], [12];
3) We conduct a qualitative analysis of the results to better understand the strengths and weaknesses of the approaches, beyond the metrics;
4) We construct and share a dataset of 487 user stories from nine cases and their derived domain models; these can be used as a benchmark for future research.

Organization. In Sect. II, we set the background and review related studies. We present the benchmark dataset in Sect. III. In Sect. IV, we introduce the two approaches we developed as well as the rule-based baseline. Our evaluation is presented in Sect. V, including the experiment design and the results. In Sect. VI we qualitatively analyze the results and their implications and discuss the threats to validity. We conclude and set plans for future research in Sect. VII.

II. Background and Related Work

Requirements describe the stakeholders' needs, the desired functionality, and the constraints under which the software should operate, making them one of the most critical phases of the software development process [13]. Requirements expressed in natural language may suffer from several defects [14], including ambiguity and vagueness [15], [14]. Furthermore, requirements keep evolving and they undergo a number of changes alongside their evolution [16], [17].

Domain models support analysts in addressing these challenges, as these models deliver an overview of the key entities in the domain [1], facilitate the identification of ambiguity and incompleteness [18], and serve as the *starting point* for design and implementation [3]. We group the existing literature into two lines of work: the essence of domain models (Sect. II-A), and approaches to extract or derive such models (Sect. II-B).

A. Domain Models

According to Broy [1], a "domain model identifies fundamental business- and application-specific entity types and relationships between them, including business processes". Blaha and Rumbaugh [3] propose three types of domain models: class models, state models, and interaction models. They are constructed based on *user interviews*, *domain knowledge*, *real-world experience*, and the analysis of *related systems*.

Domain class models include [3] classes (entities), attributes, and various types of relationships: association, aggregation, and inheritance. Relationships have names and may express cardinality constraints [3].

In this paper, we focus on domain class models (domain models from here on) that only consist of *classes* and *associations*. This choice is motivated by our RQ: (1) user stories are a user-oriented notation for expressing requirements that usually misses information such as attributes and cardinality constraints [7], [9]; and (2) rather than a complete domain model, we aim to obtain a high-level domain class model that serves as a first step in analyzing the requirements for understanding the static structures of a domain for both professional and non-professional stakeholders [3], [19].

Domain models can be constructed in many ways. They can be built manually from scratch or from informal domain knowledge, or they can be derived from a textual artifact such as domain documents, business processes, or an existing set of requirements. In this paper, we rely on the guidelines for deriving models for human analysts proposed by Blaha and Rumbaugh [3], which suggest identifying all candidate elements and then removing those that are not relevant for the model, based on some exclusion criteria.

B. Model Extraction and Derivation

Generating models from requirements descriptions is a well-studied problem [20], [21], [22]. Yue *et al.*'s [23] systematic analysis on the generation of models from requirements revealed that, to generate complete and consistent models, either human intervention or artificially restricted natural language, such as controlled vocabulary and grammar, are required.

Most of the approaches for the extraction of models from requirements rely on rule-based NLP. Although many studies exist [2], only a few provide in-depth empirical evaluations that deliver solid evidence of their effectiveness.

Arora *et al.* [6] propose a rule-based approach that extracts domain concepts, associations, generalizations, cardinalities, and attributes *from 'shall' requirements*. A questionnaire was given to practitioners with 50 random requirements to assess the perceived usefulness of their approach. According to the results, the relevance of the elements within a domain model extracted by their approach lies in the range of 29%–43% with a 95% confidence level.

Lucassen *et al.* [10], [24] propose the Visual Narrator, a rule-based NLP tool that extracts structural elements *from user stories*. The output consists of entities, hierarchical and non-hierarchical relationships. Using precision, recall, and F_1-score metrics, the Visual Narrator achieved precision and recall scores above 90%. These results, however, are obtained by assessing the tool's performance against a human execution of the same algorithm the tool implements, rather than against models that are created by humans based on their own rationale for the inclusion of entities and relationships.

Bragilovski *et al.* [8] propose a machine learning algorithm to recommend relationships between entities in conceptual models created from user stories. Although their reported F_1-score is better than the Visual Narrator's [24], they did not compare it to human performance and they only provided a limited analysis of the results.

Saini *et al.* propose DoMoBOT [25], a tool that aids modelers in generating class diagrams from requirements descriptions. DoMoBOT combines rules, machine learning, and deep learning techniques. This bot takes requirements descriptions rather than user stories, and it is designed to *interactively* assist users throughout the process rather than autonomously producing a comprehensive and validated model. Nevertheless, we tested the performance of the class identification task on two datasets from our benchmark – Planningpoker and School (see Table I). For Planningpoker, the precision and recall were 0.121 and 0.666, respectively. For School, the precision and recall were 0.211 and 0.611. In view of the limited precision and recall and the different application scenario, we decided not to compare this approach in our paper.

Cámara *et al.* [26] explore the capabilities of ChatGPT to develop UML class diagrams with OCL constraints to assist modelers. In their experiments, they provide a description of the domain in a conversational manner to ChatGPT and request a PlantUML as a result. Their results show that ChatGPT's performance depends heavily on domain knowledge. Our objective and approach is different, as we are asking ChatGPT to identify entities and relationships from a given set of user stories and providing guidance for their identification.

Arulmohan *et al.* [27] study how LLMs (ChatGPT 3.5) can extract domain concepts from user stories: personas, actions, and entities. They argue these can then be further transformed into a domain representation. We take it a step further and derive relevant classes (a type of entity) and associations, excluding irrelevant entities and associations. While doing so, we use the prompt patterns proposed by White *et al.* [28] and the strategies introduced by OpenAI [29].

III. A Benchmark for Domain Model Derivation from User Stories

Due to the lack of benchmarks that contain user stories and corresponding domain models, we set off to develop a new one. We selected three collections of user stories from an online repository of user stories [30] and six sets from projects of master's level students taking a course on requirements engineering. The process for selecting these collections (*projects*, from here on) involved analysis to ensure they met two criteria: (i) they contain mainly functional requirements, and (ii) they refer to domains that are relatively easy to understand with limited domain knowledge. These criteria were established to isolate factors that could potentially affect the outcomes of the evaluated approaches. The objective of this research is to ascertain the optimal capabilities of the existing approaches within a controlled, "ideal" scenario. For choosing the projects in our dataset, we took into account the available resources after applying the inclusion/exclusion criteria, the time-consuming nature of constructing a gold standard, and the minimum number of projects (at least five) required for conducting a Friedman test to statistical analysis (which we describe in the evaluation part). In the following we describe the process of creating such a benchmark dataset.

TABLE I: Overview of our benchmark dataset of user stories and domain models. FK is the Fleiss' Kappa among the three taggers.

Project	#US	#Class	Class FK	#Assoc.	Assoc. FK
Camperplus	55	17	0.768	23	0.558
Fish&Chips	50	9	0.557	7	0.732
Grocery	49	9	0.724	8	0.892
Planningpoker	53	6	0.723	6	0.528
Recycling	51	9	0.599	6	0.634
School	61	17	0.601	23	0.589
Sports	63	13	0.694	12	0.398
Supermarket	51	12	0.627	11	0.851
Ticket	54	10	0.657	13	0.730

A. Classes Identification

Three authors of this paper identified the classes for each project independently using TAGRAM[1] [31], a web platform designed as an infrastructure for processing, tagging, and modeling sets of user stories. The tagging has been done according to the guidelines for class identification by Blaha and Rumbaugh [3]. The first step was the identification of all nouns that become tentative classes and the removal of spurious classes. A class was then eliminated when falling into one of the following cases: redundant class, irrelevant class, vague class, attribute, operation, role, implementation construct, and derived class. Next, we measured the inter-coder reliability via Fleiss' Kappa [32] (FK). Table I shows that for most of the projects, we achieved *substantial* agreement (FK of at least 0.6). To finalize the list of classes, we discussed the differences and reached an agreement on the gold-standard classes for each project. The process of identifying classes and establishing the gold-standard classes through discussion occurred in three rounds. The initial round involved one project (School), the second round encompassed two projects (Fish&Chips and Ticket), and the third round covered all remaining projects. The purpose of this process was to assess and align on the understanding of the task.

[1]https://github.com/ShahafErez/Tagram

B. Associations Identification

Associations denote relationships between classes that are at the same level of abstraction [3], thereby excluding hierarchical relationships. After agreeing on the classes for each project, we created a spreadsheet that included all possible binary associations between pairs of classes in the gold standard. As a next step, the three authors independently tagged each candidate association. The taggers were instructed discard unnecessary and incorrect associations when they fall into one of the following categories: irrelevant or implementation associations, actions, ternary associations, and derived associations, as suggested by [3]. Table I shows the inter-coder reliability, measured using Fleiss' Kappa for the associations. In most projects, we achieved *moderate to substantial* agreement (FK of at least 0.55). Here again, we resolved the disagreements via three rounds of discussion, following the same process described for the classes.

IV. Domain Model Derivation Approaches

We describe the approaches that we used to automatically derive domain models from user stories. The first approach consisted of the Visual Narrator, a rule-based-NLP based tool [10] (Sect. IV-A). Next, we developed a machine learning model based on feature engineering (Sect. IV-B). Finally, we devised an LLM-based approach by setting a prompt for ChatGPT 4.0 (Sect. IV-C).

A. Rule-based: Applying the Visual Narrator

The Visual Narrator [10] was designed by Robeer, Lucassen, and colleagues to automatically generate conceptual models from a collection of user stories. This tool contains an algorithm composed of several heuristic rules from the NLP and conceptual modeling domain to determine classes and relationships. Visual Narrator builds on a traditional NLP pipeline that uses the spaCy toolkit for Python: after tokenizing the user stories, the part-of-speech tagger is used to separate the parts of a user story (role, action, benefit), and then various heuristics are applied (e.g., every noun is a candidate class) to identify classes and relationships.

B. Feature Engineering a Machine-Learning-based approach

We started from the approach by Bragilovski *et al.* [8], which identifies associations from a pre-defined set of classes, and further extend it. Specifically, we conducted feature engineering in order to build an end-to-end machine-learning approach that also identifies classes from a collection of user stories, leading to an approach that is able to output both classes and associations.

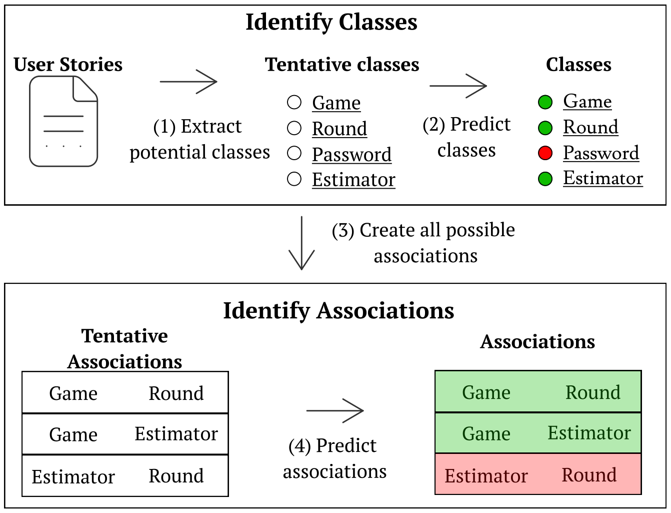

Fig. 1: An illustration of the process of deriving a domain model using a machine-learning based approach.

The pipeline of this approach is illustrated in Fig. 1 and consists of (1) extracting potential classes; (2) identifying non-spurious classes; (3) assembling all possible associations between the identified classes; (4) filtering those associations by selecting the relevant ones. The identification of classes and associations is mapped into a prediction task using Random Forest (RF), a state-of-the-art technique that achieved the best results in many software-engineering-related tasks [33].

1) Identifying Classes: The first step is the identification of the classes from the user stories. To do so, we first extract all nouns and compound nouns (up to three words) through the noun-phrase identification component of spaCy[2]. These extracted phrases are tentative classes [3]. Next, to eliminate spurious classes, we define a vector of features for each tentative class that is used by the RF model to identify the classes of the domain model. We engineered features based on rules for class identification [24], [3] as well as from additional insights we gained after exploring the data. The features are listed in Table II, and they are based on the following rationale. Feature 1 assumes that if a phrase appears frequently, it is likely to be an important phrase. Features 2–5 determine if the noun phrase should be a class based on part-of-speech features: is it a noun, the subject of a sentence, a compound noun? Feature 6 refers to the cases where a noun co-occurs at least once with other ones. Finally, features 7–9 relate to the frequency of occurrence in the various parts of the user story template (according to the Connextra format [9]).

[2]https://spacy.io/

TABLE II: Features used by the machine-learning model for class identification. Each feature characterizes a noun phrase.

ID	Feature	Description
1	count	The number of times the phrase appears in the user stories.
2	noun	The number of times the phrase appears as a noun in the user stories.
3	subject	The number of times the phrase appears as a subject in the user stories.
4	compound	The number of times the phrase appears as a compound in the user stories.
5	gerund	The number of times the phrase appears as a gerund in the user stories.
6	part_of_frequent_ngram	1 if the class is in an N-Gram ($N \in 2, 3, 4$) that occurs ≥ 2 times, else 0.
7	user_role	The number of times the phrase appears in the role part of a user story.
8	action	The number of times the phrase appears in the action part of a user story.
9	benefit	The number of times the phrase appears in the benefit part of a user story.

2) Identifying Associations: The next step is to identify associations between the classes. Here, we employ the features engineered by Bragilosvki *et al.* [8], which we report and briefly comment on for completeness.

These features can be categorized into three groups: external knowledge, internal knowledge, and requirements-type knowledge. The inspiration for this categorization stems from [3], which emphasizes the importance of considering domain knowledge and real-world experience when building a model.

a) External knowledge: concerns the existing knowledge in other domains (we used the ModelSet repository [34]). It involves features that calculate the similarity between a tentative association and other associations that have already been used in models from different domains. The rationale is that if an association appears in other projects, it may be relevant to the new project as well.

b) Internal knowledge: concerns the information obtained from the given requirements. It involves examining features that count how frequently classes co-occur in the user stories and how often both classes occur individually. The rationale behind this is that if classes frequently occur together, they may have an association.

c) Requirements-type knowledge: concerns the information derived from the structure of the requirements. Our benchmark consists of user stories that follow the Connextra format. Thus, features that examine in which part each class of the association occurs may aid in the identification of the association. For example, if two classes only appear in the role part, it is unlikely for an association between them to exist.

C. An LLM-based Approach based on ChatGPT

Following the guidelines of [3], we also built a derivation approach using ChatGPT 4.0 (API). We have defined two initial prompts (one for classes, one for associations) that are intended to be useful for any collection of user stories. In unusual cases, however, ChatGPT may produce a list of classes that is clearly too long or too short. In such situations, limited follow-up questions may be required to indicate that the final list of classes can be made stricter or more flexible.

We used the user stories from the School project to experiment with the initial prompt(s). To refine this initial prompt, we attempted several enhancements derived from OpenAI's strategies [29] (also considered by, e.g., [27]) and the community-posted prompt patterns from White *et al.* [28]. Both resources include strategies that are independent from our specific task. In the following, we discuss the rationale for the various parts of our prompt:

- *General structure.* Especially for more complex prompts, it is important to remove ambiguity in the formulated task by structuring the prompt [29]. For example, we add delimiters (e.g., triple quotation marks, colons) and divide the query into separate sections. As OpenAI [29] suggests, it is also valuable to divide complex tasks into simpler subtasks. Consequently, we divide the task into two subtasks: (1) identifying relevant classes, and (2) finding associations between these classes. Furthermore, writing the task as a sequence of steps makes it easier for ChatGPT to accomplish it [29]. We split both subtasks into two steps based on the guidelines by Blaha and Rumbaugh [3]: 1) define classes/associations, 2) remove irrelevant classes/associations.
- *Persona.* Specifying a persona ensures that the LLM takes into account a specific perspective when generating output, which can help focus on the correct details [29], [28]. For this reason, we specify that the model should behave like a requirements engineer specializing in domain modeling.
- *Context.* As indicated by the guidelines [29], [28], it is relevant to provide additional details to the query and indicate what can be discarded from the context to avoid irrelevant output. Thus, our prompt includes more context about which classes can be selected for objects (with some examples) and which classes should be excluded. For associations, we follow the same procedure.
- *Reasoning.* Both OpenAI and [28] emphasize the importance of giving the model time to reason, which can help understand how the model reached the answer [28] and help the model reason more reliably toward correct answers. Therefore, we explicitly ask the model to explain how it arrived at the output.
- *Template.* The user can instruct the LLM to produce the output of the LLM to follow a specific template. Although we do not require a specific template, for ease of use, we specify that the end of the output should contain a final list of classes/associations.
- *Testing.* We added some details to the prompt using the user stories from the School project, as advised by [29]. For example, the statement to exclude subclasses was added because we noticed that ChatGPT would otherwise include them. However, we opted not to test an exhaustive list of prompt options to avoid overfitting.

Although the guidelines [28], [29] list additional patterns, we excluded those that were unrelated to our problem statement or that we found irrelevant after experimentation. For example, the Question Refinement Pattern [28] could be added to almost any problem statement, but this resulted in reformulated questions that no longer met the guidelines of [3]. The resulting prompts are in the online appendix [35].

V. Evaluation

We present the results obtained by applying the approaches discussed in Sect. IV on the benchmark dataset that we assembled (Sect. III). We compare these to the so-called Human Achievable Performance (HAP), described below.

A. Experimental Settings

Using the approaches outlined in Section IV, we describe how the domain models were derived. To counter potential bias in the evaluation process, the evaluation of *VN* and *ChatGPT* was carried out by an author who was not involved in the preparation of, nor aware of, the gold-standard domain models. *The experimental materials, along with the ML implementation, the prompt, and the benchmark are in the appendix [35].*

a) Tasks: We report on three evaluations, one of which ($Eval_C$) measures the ability of the approaches to derive classes, while the other two ($Eval_A^{Adj}$ and $Eval_A^{Gold}$) determine the ability to derive the associations.

In $Eval_C$, we compare the performance of *HAP*, *VN*, *ML*, and *ChatGPT* in deriving classes for each of the nine projects in the benchmark dataset.

In $Eval_A^{Adj}$, we perform a similar comparison for associations identification. Since this step depends on the preceding stage (identifying classes), we *adjusted* the way we calculate the metrics. In particular, we do not consider associations that include one class that a specific approach has not identified: if that approach fails to identify a class, it obviously cannot identify any associations involving that class. Similarly, we do not consider associations where the model identified a class that is not present in the gold standard. We report the percentage of omitted associations in the appendix [35]. We also omit *HAP* from $Eval_A^{Adj}$; unlike the automated approaches, the human taggers identified the associations only after agreeing on the gold standard.

In $Eval_A^{Gold}$, we compare the approaches using the *same test set* (unlike $Eval_A^{Adj}$, where each approach relies on its own generated list of classes). To do so, we compare the approaches for association identification starting from the classes in the gold standard. We excluded *VN* due to its inability to predict associations only between a defined set of classes. Thus, the comparison is limited to *HAP*, *ML*, and *ChatGPT*.

b) Metrics (dependent variables): The results are reported using $F_{0.5}$-score, F_1-score, and F_2-score, which are different harmonic means of precision and recall [36]. These metrics integrate the notions of *Validity* (precision) and *Completeness* (recall), drawn from previous conceptual modeling research [11], [37], [12]. We employed three distinct F-scores, each tailored to emphasize a specific facet of the domain models. The $F_{0.5}$-score assigns greater weight to validity, catering to users who prioritize domain models where nearly all classes are relevant, even if a few are omitted. Conversely, the F_2-score places more emphasis on completeness, catering to users who favor domain models that contain many relevant classes, even if some irrelevant ones are included. Lastly, the F_1-score strikes a balance by assigning equal weight to both validity and completeness, recognizing the comparable importance of both aspects.

c) Compared approaches (independent variables):

- **Human Achievable Performance.**
  For analyzing HAP, we first calculated the precision and recall regarding class identification for each author involved in creating the gold standard. This involved comparing their outputs *prior* to the discussion and establishing the gold standard. Note that there were cases where, after the discussion, classes that none of the authors initially identified were included in the gold standard After reaching a consensus on the gold standard classes, each author independently collected a list of associations between these classes. Precision and recall were calculated for each author by comparing their list to the agreed-upon gold standard, established through discussion among the authors. The HAP is then computed based on the *average precision and recall of the three authors* for both classes and associations.
- **VN-based approach.**
  When executing the Visual Narrator, a threshold can be defined to help filtering out the least frequent classes [10]. This threshold considers a ranking based on both the frequency of occurrence and the weight (i.e., relative importance) of each class [10]. To mimic a real-world situation in which an analyst could experiment with multiple thresholds before continuing the modeling, the threshold value was chosen separately for each project.
  We post-processed some of the outputs to enable a fair comparison. First, the VN can derive multiple associations between two specific classes. Since we are interested in a domain model that only indicates if there is an association between two classes, we consider whether or not the VN suggests at least one association. Also, the VN derives parent classes from noun-noun combinations [10]. It defines the parent class as the prefix of the compound and creates a hierarchical relationship between this parent class and the actual compound. We include both the original compounds and the parent classes and do treat the resulting hierarchical relationship between these classes as a standard association. Second, when the *VN* cannot find a main object in the user story, it is replaced by the *VN* with the dummy word "System". If this class type appears in the final model, we remove it along with its associations[3]. Third, the VN includes all functional roles from the user stories (even the infrequent ones) as a class in the final model. When these functional roles were not selected based on the threshold and appeared to have no associations in the final model, we removed them as well.
- **ML-based approach.** To obtain a domain model using the ML approach outlined in Sect. IV-B, we employed the widely used Leave-One-Out Cross-Validation (LOOCV) technique [38]. In software engineering research, this is also referred to as the leave-one-project-out (LOPO) or *p-fold*. In this approach, data from a single project is set aside for testing purposes, while the data from the remaining projects is utilized for training. This technique leads to more generalizable results than cross-validation, as model training is conducted on projects that differ entirely from the one that is used for prediction [39]. For class identification, we trained the model on all projects except the one designated for testing. This process was repeated nine times, corresponding to the number of projects in the experiment. For association identification, we initiated the process by creating a candidate list that includes all possible associations. This involved generating all combinations of associations among the classes in the gold-standard model for each project. Subsequently, similar to the class identification, we trained nine distinct models, where each model was specifically designed to predict a project that was excluded from the training phase.
  For each project, we therefore had two distinct ML models, one for class identification and the other for association identification, trained on the other projects. In order to determine which elements to retain, the author who was not involved nor aware of the gold standard made a choice by determining an optimal threshold by considering the prediction probability set by the ML model. This was done in two rounds: first, the author identified the relevant classes by tuning the threshold for the classes. Then, the ML model for association identification was executed to connect those classes, and the author considered different threshold values for the associations.
- **LLM-based approach with ChatGPT 4.0.** For the LLM-based approach, we set the temperature to 0 to keep the output as deterministic as possible. We also attempted to produce confidence levels per generated class and association, but since the model tended to be 100% certain in its selection (presumably due to the temperature value), we excluded this option. Thus, for the LLM-based approach that uses ChatGPT, we simply rely on the resulting lists of classes and associations.
- **Precision- and recall-oriented variants of VN and ML.** Given our decision of considering, in addition to the classic F_1-score, two variants that favor precision ($F_{0.5}$) and recall (F_2), we established that the author who would define the thresholds for VN and for ML would actually make two choices. The first threshold should favor a model that, in the eyes of the author, prioritizes precision, adding little noise to the model. The second threshold

[3]If "System" is a real domain class, we retain the class, but remove the associations with this class that refer to the dummy word "System"

should focus on building a model that contains as many relevant classes (and associations) as possible, without compromising precision too much. This leads to the precision- and recall-oriented variants of VN and ML (see the results in Sect. V-B and in Table III).

d) Hypothesis: After defining the approaches, and the metrics, we define the following (null) hypotheses:

- The class prediction performance ($Eval_C$) from user stories results in equal $F_{0.5}$ ($H_0^{F_{0.5}\text{-}class}$), F_1 ($H_0^{F_1\text{-}class}$), F_2 ($H_0^{F_2\text{-}class}$) when comparing the approaches.
- The association prediction performance ($Eval_A^{Adj}$) from user stories results in equal $F_{0.5}$ ($H_0^{F_{0.5}\text{-}association}$), F_1 ($H_0^{F_1\text{-}association}$), F_2 ($H_0^{F_2\text{-}association}$) when comparing the approaches.

B. Quantitative Results

We compare HAP with the automated approaches for the identification of classes and associations. Table III shows the performance of the approaches for deriving classes ($Eval_C$), organized by project. We report on the results of the six alternatives: *HAP*, two versions of *VN* (precision- and recall-oriented), two versions of *ML* (precision- and recall-oriented), and *ChatGPT*. We present $F_{0.5}$-score, F_1-score, and F_2-score, and report macro-average and standard deviation for each approach. We highlight in green the best-automated approach per project and macro-average.

It appears that *HAP* consistently outperforms the automated approaches in terms of $F_{0.5}$-score, F_1-score, and F_2-score across all projects. Among the automated approaches, *ML precision-oriented* has the highest average $F_{0.5}$-score and F_1-score at 0.625 and 0.550, respectively. Regarding F_2-score, *ChatGPT* takes the lead, with a slight margin over *ML recall-oriented* (0.608 versus 0.599).

To determine if the differences are statistically significant, we conducted statistical tests with $\alpha = 0.05$. We applied the Friedman test [40], a non-parametric statistical test to compare more than two treatments. We found a statistically significant difference among the compared approaches with $p < 0.01$. To check the differences between pairs of approaches, we applied Nemenyi's post-hoc test [41]. The results, visualized in Fig. 2, show that the differences between some automated approaches and HAP are not statistically significant: (1) ML-precision in terms of $F_{0.5}$-score; (2) ML-precision, ML-recall, and ChatGPT in terms of F_1-score; and (3) ML-recall and ChatGPT in terms of F_2-score.

Additionally, we calculated the effect size using Cohen's d. We refer to Cohen [42] where $d > 0.2$ indicates small effect, $d > 0.5$ denotes medium effect, and large effect holds when $d > 0.8$. From the results, HAP consistently achieved a *large effect size* compared to all other alternatives in all metrics. The full results are shown in the appendix [35].

In Table V, the performance of the approaches for deriving associations is presented ($Eval_A^{Gold}$) against the classes from the gold standard (GS). *HAP* seems to outperform the automated approaches in terms of all average metrics. The closest approach is *ML* recall-oriented for the case of F_2-score,

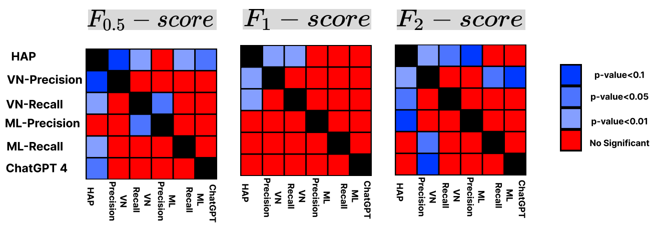

Fig. 2: Classes ($Eval_C$) post-hoc analysis.

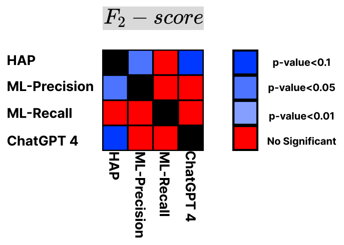

Fig. 3: Associations ($Eval_A^{Gold}$) post-hoc analysis. Only the results for F_2 are shown, as no differences could be found for the other metrics.

where the automated approach achieved 0.800 while the HAP achieved 0.850. Unlike class identification, there are instances where automated approaches outperform *HAP*. For instance, in the Grocery project, *ML* recall-oriented achieved the highest F_2-score with a value of 0.952 (while HAP is 0.917). Another noteworthy example is the Sport project, where *ChatGPT* achieved an F_2-score of 0.820 (while HAP is 0.745).

Here again, we apply the Friedman test and Nemenyi's post-hoc test to check the statistical significance. The results appear in Fig. 3. We found that: (1) there is no statistical significant difference between the approaches for *p-value* < 0.05 in terms of $F_{0.5}$ and F_1 (in the appendix, not shown in the figure); (2) for F_2-score, there is no statistical significant difference between *HAP* and *ML* recall-oriented.

We also calculated the effect size using Cohen's d. Except for ML recall-oriented, HAP consistently outperformed all other alternatives. Based on F_2, it had a medium effect size of 0.404. The full results are shown in the appendix [35].

VI. Discussion

We first present insights obtained from the results presented in Sect. V-B. Then, we discuss threats to validity.

A. Results Analysis

We analyze the results from two perspectives: qualitative and qualitative.

1) Quantitative Analysis: We derive quantitative insights from the tables and figures in Sect. V-B.

Finding 1. For class identification, HAP outperforms automated approaches in terms of all metrics. However, for each F-score, there is at least one version of ML that has no statistically significant differences from HAP.

Examining the outcomes presented in Figure Fig. 2, which depicts the post-hoc analysis of $Eval_C$, we observe that for

TABLE III: F-scores for class identification ($Eval_C$). The cells in green highlight the best results per project and overall.

Project	HAP			VN						ML						ChatGPT		
				Precision-oriented			Recall-oriented			Precision-oriented			Recall-oriented					
	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2
Camperplus	0.784	0.818	0.856	0.444	0.333	0.267	0.426	0.489	0.573	0.610	0.435	0.338	0.556	0.619	0.699	0.588	0.588	0.588
Fish&Chips	0.708	0.676	0.648	0.333	0.333	0.333	0.294	0.357	0.455	0.714	0.737	0.761	0.507	0.583	0.686	0.472	0.500	0.532
Grocery	0.778	0.849	0.933	0.405	0.375	0.349	0.337	0.414	0.536	0.909	0.800	0.714	0.449	0.552	0.714	0.439	0.476	0.521
Planningpoker	0.767	0.809	0.855	0.370	0.444	0.556	0.370	0.444	0.556	0.652	0.750	0.882	0.349	0.462	0.682	0.305	0.400	0.581
Recycling	0.680	0.714	0.751	0.283	0.300	0.319	0.260	0.308	0.377	0.366	0.353	0.341	0.370	0.444	0.556	0.472	0.500	0.532
School	0.704	0.745	0.790	0.566	0.462	0.390	0.489	0.565	0.670	0.732	0.522	0.405	0.495	0.526	0.562	0.537	0.605	0.691
Sports	0.745	0.754	0.763	0.455	0.333	0.263	0.410	0.400	0.391	0.377	0.348	0.323	0.362	0.370	0.379	0.591	0.667	0.764
Supermarket	0.775	0.749	0.724	0.424	0.435	0.446	0.422	0.483	0.565	0.741	0.533	0.417	0.391	0.486	0.643	0.374	0.457	0.588
Ticket	0.666	0.704	0.746	0.595	0.556	0.521	0.278	0.333	0.417	0.526	0.471	0.426	0.403	0.435	0.472	0.479	0.560	0.673
Mean	0.734	0.757	0.785	0.431	0.397	0.383	0.365	0.421	0.504	0.625	0.550	0.512	0.431	0.498	0.599	0.473	0.528	0.608
Std-dev	0.045	0.057	0.085	0.101	0.083	0.105	0.078	0.083	0.099	0.178	0.172	0.213	0.074	0.079	0.116	0.094	0.083	0.084

TABLE IV: F-scores for association identification ($Eval_A^{Adj}$) starting from the classes identified by each approach.

Project	VN						ML						ChatGPT		
	precision-oriented			recall-oriented			precision-oriented			recall-oriented					
	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2
Camperplus	0.455	0.400	0.357	0.426	0.348	0.294	0.938	0.857	0.789	0.263	0.174	0.130	0.500	0.364	0.286
Fish&Chips	-	-	-	0.435	0.500	0.588	0.588	0.500	0.435	0.270	0.308	0.357	0.714	0.800	0.909
Grocery	1.000	1.000	1.000	0.800	0.800	0.800	0.000	0.000	0.000	0.370	0.333	0.303	1.000	1.000	1.000
Planningpoker	0.333	0.333	0.333	0.333	0.333	0.333	0.385	0.364	0.345	0.385	0.364	0.345	0.714	0.667	0.625
Recycling	0.833	0.667	0.556	0.833	0.667	0.556	-	-	-	0.417	0.333	0.278	0.625	0.727	0.870
School	0.469	0.545	0.652	0.390	0.414	0.441	0.250	0.182	0.143	0.377	0.348	0.323	0.530	0.467	0.417
Sports	-	-	-	0.294	0.400	0.625	0.556	0.667	0.833	0.833	0.889	0.952	0.652	0.750	0.882
Supermarket	0.417	0.444	0.476	0.429	0.429	0.429	0.652	0.750	0.882	0.652	0.632	0.612	0.795	0.824	0.854
Ticket	0.385	0.286	0.227	0.294	0.250	0.217	0.250	0.250	0.250	0.385	0.286	0.227	0.526	0.471	0.426
Mean	0.556	0.525	0.515	0.470	0.460	0.476	0.452	0.446	0.460	0.439	0.407	0.392	0.673	0.674	0.696
Std-dev	0.255	0.246	0.258	0.204	0.173	0.184	0.292	0.299	0.337	0.185	0.217	0.247	0.158	0.204	0.263

TABLE V: F-scores for association identification ($Eval_A^{Gold}$) starting from the set of classes in the gold standard.

Project	HAP			ML						ChatGPT		
				precision-oriented			recall-oriented					
	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2	$F_{0.5}$	F_1	F_2
Camperplus	0.851	0.870	0.891	0.862	0.769	0.694	0.840	0.851	0.862	0.588	0.596	0.603
Fish&Chips	0.855	0.889	0.926	0.897	0.933	0.972	0.556	0.667	0.833	0.364	0.421	0.500
Grocery	0.917	0.917	0.917	0.972	0.933	0.897	0.833	0.889	0.952	0.729	0.778	0.833
Planningpoker	0.869	0.855	0.842	0.455	0.400	0.357	0.463	0.556	0.694	0.735	0.769	0.806
Recycling	0.844	0.860	0.877	0.962	0.909	0.862	0.882	0.923	0.968	0.364	0.421	0.500
School	0.765	0.718	0.677	0.339	0.250	0.198	0.408	0.444	0.488	0.488	0.500	0.513
Sports	0.699	0.721	0.745	0.515	0.538	0.565	0.536	0.600	0.682	0.781	0.800	0.820
Supermarket	0.931	0.923	0.914	0.652	0.667	0.682	0.644	0.743	0.878	0.738	0.720	0.703
Ticket	0.921	0.891	0.864	0.702	0.667	0.635	0.674	0.750	0.845	0.725	0.741	0.758
Mean	0.850	0.849	0.850	0.706	0.674	0.651	0.648	0.714	0.800	0.612	0.638	0.671
Std-dev	0.076	0.077	0.085	0.233	0.243	0.253	0.173	0.161	0.153	0.168	0.156	0.143

each F-score measure, at least one version of ML model demonstrates no statistically significant difference compared to the HAP. Specifically, regarding the $F_{0.5}$-score, we find no basis to reject the null hypothesis $H_0^{F_{0.5}\text{-}class}$ that posits no difference between HAP and ML precision. Similarly, for the F_1-score, the evidence does not allow us to reject the null hypotheses $H_0^{F1\text{-}class}$ for the differences between HAP and both ML precision and recall, as well as chatGPT. Lastly, concerning the F_2-score, the null hypotheses $H_0^{F2\text{-}class}$ cannot be rejected, as the results indicate no statistical difference between HAP and both ML recall and chatGPT. These findings suggest that automated methods may achieve results comparable to human performance, indicating potential savings in time and effort.

Finding 2. For association identification, HAP generally outperforms the automated approaches. However, tuning the ML for recall shows an effect, however, it is not statistically significantly different from HAP in terms of F_2-score.

Observing Fig. 3, the post-hoc analysis comparing the HAP and ML-recall reveals a p-value exceeding 0.1. Consequently, we cannot reject the null hypothesis $H_0^{F_2\text{-}association}$, indicating that the difference between HAP's average F_2-score of 0.85 and ML-recall's 0.80 is not statistically significant.

2) Qualitative Analysis: We complement conventional statistical evaluations with a qualitative analysis of the generated domain models in order to obtain a more nuanced understanding of the differences between the approaches beyond the F-scores.

We derive qualitative insights via a thorough examination of the alignment of elements (classes and association) across the domain models generated by automated approaches and against the gold standard.

Classes: For this element, we observe the following:

Finding 3. The most dominant features that influence the identification of classes by the *ML* are: 'action', 'noun', and 'count', indicating that frequency of occurrence, by itself, as a noun, and in the action part of a user story, contributes to identifying a noun phrase as a class.

We ran SHAP [43] on each *ML* model for class identification (due to the LOOCV evaluation) and found that the SHAP scores were consistent across all approaches (appears in the appendix [35]). Fig. 4 shows the bar plot of SHAP scores of the *ML* achieved when the Planningpoker is used as a test set. Upon scrutinizing the instances of false positives and false negatives, it becomes evident that these prevailing features heavily influence misidentification. For example, in Planningpoker, the class 'account' was identified as a viable class (with a probability of 0.891) while it was not included in the gold standard since it is an implementation detail in the context of the domain. This outcome was influenced by features such as 'action', 'noun', and 'count', which contributed more significantly to such identification than the 'compound' feature that contributed negatively to that identification (following the SHAP analysis). Thus, the ML approach favors frequently occurring noun phrases, especially those occurring in the action part of the user stories. The users of this approach could therefore search carefully into the role and benefit parts for potentially omitted classes.

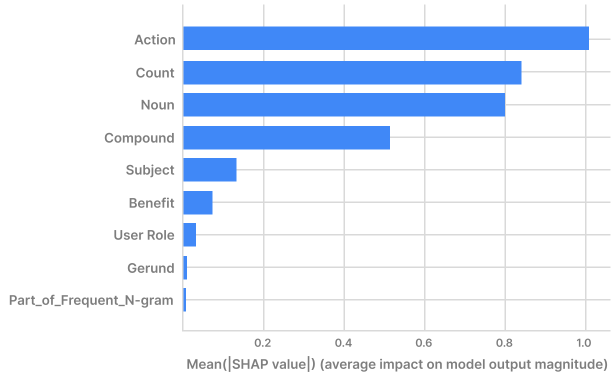

Fig. 4: SHAP values of the *ML* model; Planningpoker as a test set.

Finding 4. There are differences between the types of false positives that each approach generates.

In order to qualitatively compare the treatments and the differences across domains, we have conducted an analysis of the false positives according to the categories outlined in our reference book [3]: (i) redundant or derived (we merged those); (ii) vague; (iii) role; (iv) attribute; (v) implementation construct; (vi) operations; and (vii) irrelevant to the domain. To do so, two authors of this paper have independently examined the false positives that each treatment produced per each collection of user stories and assigned one of the six labels (the operation category was never selected) to each false positive. This led to Cohen's kappa of 0.63 (six categories, two raters), indicating *good* alignment. The disagreements were discussed between the two taggers and the conflicts were resolved. To facilitate the discussion, the taggers constructed a decision tree (Fig. 5) to determine which class the false positive should belong to. Following this tree, no false positives were left as 'undefined' (the tagging outcomes are in the appendix [35]).

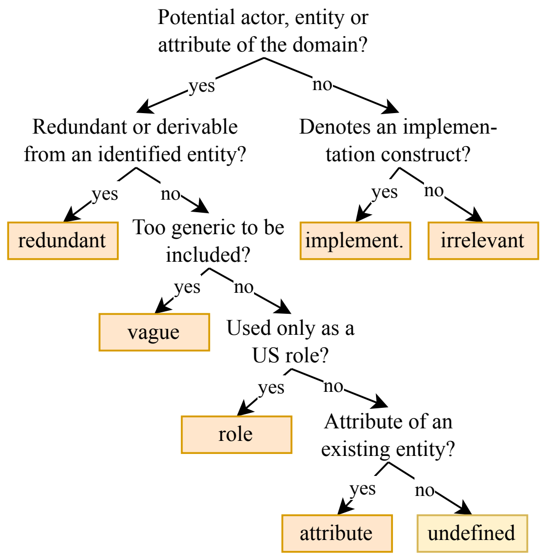
Potential actor, entity or attribute of the domain?
yes
no
Redundant or derivable from an identified entity?
Denotes an implementation construct?
yes
no
redundant
Too generic to be included?
yes
no
implement.
irrelevant
yes
no
vague
Used only as a US role?
yes
no
role
Attribute of an existing entity?
yes
no
attribute
undefined

Fig. 5: Decision tree for the analysis of false positives.

The results of the analysis appear in Fig. 6. Unsurprisingly, the *VN* identifies roles as classes since it is implemented in that manner. *ChatGPT* faces challenges in distinguishing between classes and attributes. This difficulty may arise because the GPT model is trained on extensive data, where it has probably encountered the same phrase either as an attribute or as a class, depending on the domain (e.g., 'password' might be treated as a class in a security system context). In contrast, *ChatGPT* adeptly handles irrelevant classes, perhaps due to the model's exposure to a large number of data it was trained on (the irrelevant classes infrequently appear in the model's Markov Decision Process). On the other hand, the *ML* approach struggles with irrelevant classes. This challenge stems from the fact that the model heavily relies on word statistics within the data, lacking a deeper understanding of the domain context. This is in contrast to LLMs, which, we believe, possess sufficient contextual knowledge.

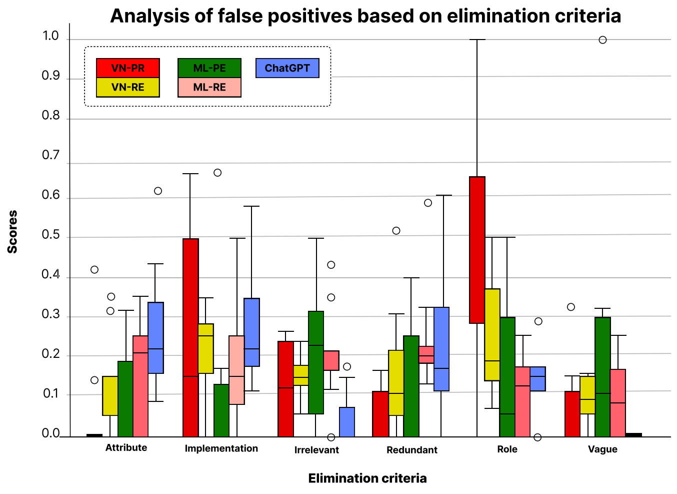

Fig. 6: Analysis of false positives based on elimination criteria by approach. The FP type operation is omitted as it was never selected.

Finding 5. The characteristics of the user story collections seem to affect the performance of the approaches.

The distribution of false positive types per project (Table VI) shows some differences. Fish&Chips and Planningpoker have the highest number of roles that were incorrectly classified as classes. To assess if a correlation exists between the number of roles (characteristic of the project) and this error type, we computed the Pearson correlation between the number of roles and the mean of role misidentification ('Role' column in Table VI), revealing a statistically significant coefficient of 0.591 (p-*value* $<$ 0.05). Furthermore, *ML-P*, *ML-R*, and *ChatGPT* demonstrate that the 'Role' type misidentification occurs roughly twice as often as their average misidentifcation. For example, in the Planningpoker project (0.5 vs 0.172, 0.214 vs 0.103, 0.285 vs 0.155, respectively, data in our online appendix). Considering these findings, coupled with the fact that, on average, the 'Role' misidentification is the most prevalent (0.228), we can infer a need for improvement in distinguishing whether roles should be integrated into the system or not in all automated approaches.

TABLE VI: Analysis of the false positives, organized by project, highlighting the most common error types per project.

Project	Red./der.	Irrel.	Vague	Attribute	Role	Imp. const.
Camperplus	0.308	0.045	0.245	0.069	0.264	0.069
Fish&Chips	0.260	0.154	0.117	0.120	0.317	0.033
Grocery	0.060	0.158	0.057	0.134	0.193	0.399
Planningpoker	0.121	0.221	0.000	0.107	0.350	0.200
Recycling	0.220	0.162	0.112	0.074	0.259	0.172
School	0.277	0.176	0.031	0.145	0.016	0.355
Sports	0.117	0.217	0.029	0.229	0.151	0.257
Supermarket	0.060	0.127	0.088	0.428	0.148	0.150
Tickets	0.047	0.027	0.130	0.111	0.287	0.398
Mean	0.166	0.142	0.092	0.152	0.228	0.221

Another example regards the 'Attribute' error type; when user stories describe many attributes, the automated approaches seem incapable of distinguishing between attributes and classes. For example, in the Supermarket project, 42.8% of the false positive mistakes identified an attribute as a class, while the mean across all projects is 15.2%. In case stakeholders prefer to filter out secondary important elements (i.e., attributes), an approach that excels at distinguishing between attributes and classes, such as *VN* and *ML*, may be preferable, as depicted in Fig. 6.

Finding 6. Our analysis found no significant correlation between the project's complexity and the effectiveness of automated tools.

We used the level of agreement among taggers as a proxy measure for project complexity. A lower agreement is expected to indicate higher complexity, as it suggests that taggers could not reach consensus, leading to more extensive discussions to determine the 'gold standard'. To investigate this, we looked at the correlation between the performance scores of HAP — specifically, its $F_{0.5}$, F_1, and F_2 scores (Table III) — and the complexity of the project as indicated by the Class FK metric shown in Table I. We found significant positive correlations with values of 0.674, 0.857, and 0.811, respectively. These findings suggest that our approach to measuring project complexity is valid, as there's a clear link between these scores and the measured complexity.

However, when comparing the performance of all automated approaches against the Class FK metric, we found weak correlations (lower than 0.25). This initial evidence supports the conjecture that automated tools could be particularly beneficial in projects where achieving consensus among human annotators is challenging.

Adding to this, the results underline that there are no prominent issues impacting the outcomes, suggesting that tacit knowledge and abstraction capabilities significantly contribute to the superior performance of human experts. Nonetheless, this superiority is accompanied by considerable demands on time and effort, which may be particularly high for less experienced analysts.

Associations We observe the following:

Finding 7. The features that influence the most in the identification of association by the *ML* are related to features regarding the *Internal knowledge*, indicating that the most important factor to determine whether two classes should be linked by an association is the co-occurrence of the noun phrases in the same user story.

We executed SHAP across all the ML models. Unlike the ML-model for classes, the SHAP values were not always the same, although almost always all the features that appeared in the top 5 were from the *Internal knowledge* category. These results can be interpreted in view of classic heuristics [44] such as the one stating that noun phrases acting as subject and object in the same sentence are connected by an association labeled with the verb phrase. However, this is not enough to match human performance, as shown in Table V and Fig. 3.

Finding 8. We found a significant Pearson correlation of 0.95 (*p-value* < 0.05) between the recall of the HAP and the ML recall-oriented results when classes from the gold standard are used to generate associations ($Eval_{A}^{Gold}$).

This result indicates that the *ML* approach (recall-oriented) seems to be able to utilize features considered by humans when performing the association identification task. Our observations indicate that when the *HAP* exceeds 0.88, in 4 out of 5 instances, the *ML* recall-oriented approach achieved a perfect recall score of 1. *ML* may therefore be capable of identifying associations that humans could overlook. Obviously, this should be tested more thoroughly on a larger scale.

B. Threats to Validity

We discuss the relevant threats to validity according to the framework by Wohlin *et al.* [45], which includes four categories: conclusion, internal, external, and construct.
Conclusion validity concerns the relationship between the treatments (HAP, and the VN, ML, ChatGPT approaches) and the experiment outcome (i.e., their comparison). The major threat in this category concerns the selection of the user story sets. To avoid the risk of biased selection from a single source, we sampled nine sets of user stories from two sources (see Sect. III). Within these sources, we made a random selection of the sets while excluding those with user stories that were highly focused on solution-oriented details (a defect of user stories according to the Quality User Story framework [46]). Analyzing our treatments with low-quality user stories should be considered for more general conclusions; here, we decided to control this factor to reduce heterogeneity between the user story collections. To mitigate overfitting in the ML model, we used the LOOCV technique, using eight projects for training and the remaining one for evaluation purposes.

To evaluate the differences among the approaches we applied the related statistical checks and analysis.
Internal validity concerns external threats that may affect the independent variable with respect to causality. Three authors of this paper constructed the domain models from selected user story sets. To mitigate bias, we built these models separately, discussed the results, and agreed on the gold-standard model. Additionally, one of the authors, not involved in constructing the gold-standard models, independently executed and constructed the domain models using the tools. Although this approach may yield sub-optimal results, since that researcher relied on her intuition on what would be the best thresholds for inclusion of elements in the domain model, this mirrors a real-world scenario where users are not aware of the gold-standard model when utilizing automated tools.
External validity concerns the generalization toward industrial practice. These threats primarily stem from limitations in the representativeness of the projects, either in terms of number of user stories, writing style, organization, quality, and corresponding model size. To mitigate this threat, our experimentation involved nine sets with different numbers of user stories from two distinct sources. Additionally, the domain models generated exhibit differences in size concerning the number of classes and associations. Despite that, experiments need to be repeated with more projects to gain more robust conclusions. Moreover, for the ML model, we did not conduct hyper-parameters optimization to reduce overfitting.
Construct validity concerns generalizing the experiment results to the theory/construct behind it. Metrics and measurement are a major factor. Finding the right balance between precision and recall is challenging. One may prefer a domain model with almost all relevant classes included, but this leads to selecting a few irrelevant classes, while another may prefer a diagram with almost all relevant classes, with some important classes missing. To mitigate the risk, we used three common types of F-scores from the literature, each of which gives different weights to precision and recall. However, further validation is necessary, and one important direction is the identification of the most suitable metric, such as F_{β}, by analyzing the relative impact of Type 1 and Type 2 errors [36].

VII. Conclusion and Future Work

We experimentally compared three approaches for automatically deriving domain models from user stories: a rule-based approach (Visual Narrator), a machine learning approach that we feature engineered, and a generative AI approach based on ChatGPT. We contrasted them to human achievable performance (HAP) to answer our RQ.

None of the approaches could outperform HAP in identifying classes and associations (Findings 1–2). However, both ChatGPT and the ML-based approach outperformed the Visual Narrator in identifying classes and associations, showing the potential of learning-based approaches. For class identification, ML precision-oriented has the highest precision, while ChatGPT is better in terms of recall. In identifying associations, in certain cases, the automated approaches came close to the results obtained by HAP, especially with the version of the ML approach optimized for recall (Finding 2), which correlated strongly with HAP (Finding 7).

Our qualitative analysis reveals that the types of false positives are affected by the chosen approach (Finding 4) and that the characteristics of the user story collections also seem to affect the results (Finding 5). These two empirical findings hint that we could not determine an automated approach that is always the best, thereby opening avenues for future research.

In our research agenda, we plan to test and further improve the automatic approaches. For example, the features we engineered for the ML models could be revised based on Findings 3 and 6, which show how features that measure the co-occurrence of noun phrases in the user stories are the predominant factor. We also aim to explore new alternatives, such as a deep-learning approach. Finally, we aim to study the preferences of domain modelers using automatic approaches with respect to tradeoff between precision and recall.

References

[1] M. Broy, "Domain modeling and domain engineering: Key tasks in requirements engineering," *Perspectives on the Future of Software Engineering: Essays in Honor of Dieter Rombach*, pp. 15–30, 2013.

[2] 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*, vol. 54, no. 3, pp. 1–41, 2021.

[3] M. Blaha and J. Rumbaugh, *Object-Oriented Modeling and Design with UML*. Pearson, 2005.

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

[5] C. Arora, M. Sabetzadeh, S. Nejati, and L. Briand, "An active learning approach for improving the accuracy of automated domain model extraction," *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 1, pp. 1–34, 2019.

[6] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: Approach and industrial evaluation," in *Proc. of the International Conference on Model Driven Engineering Languages and Systems*. ACM/IEEE, 2016, pp. 250–260.

[7] 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.

[8] M. Bragilosvki, F. Dalpiaz, and A. Sturm, "From user stories to domain models: Recommending relationships between entities," in *Proc. of the Workshop on Natural Language Processing in Requirements Engineering*, vol. 3378. CEUR Workshop Proceedings, 2023, pp. 1–11.

[9] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[10] M. Robeer, G. Lucassen, J. M. E. van der Werf, F. Dalpiaz, and S. Brinkkemper, "Automated extraction of conceptual models from user stories via NLP," in *Proc. of the International Requirements Engineering Conference*. IEEE, 2016, pp. 196–205.

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

[12] 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.

[13] K. E. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.

[14] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi, "Detecting requirements defects with nlp patterns: an industrial experience in the railway domain," *Empirical Software Engineering*, vol. 23, pp. 3684–3733, 2018.

[15] A. R. Amna and G. Poels, "Ambiguity in user stories: A systematic literature review," *Information and Software Technology*, vol. 145, p. 106824, 2022.

[16] D. Zowghi and V. Gervasi, "On the interplay between consistency, completeness, and correctness in requirements evolution," *Information and Software technology*, vol. 45, no. 14, pp. 993–1009, 2003.

[17] Y. Levy, R. Stern, A. Sturm, A. Mordoch, and Y. Bitan, "An impact-driven approach to predict user stories instability," *Requirements Engineering*, vol. 27, no. 2, pp. 231–248, 2022.

[18] F. Dalpiaz, I. van der Schalk, and G. Lucassen, "Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and NLP," in *Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2018, pp. 119–135.

[19] D. Bjørner, *Software Engineering 3: Domains, requirements, and software design*. Springer Science & Business Media, 2006.

[20] L. Mich, "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA," *Natural Language Engineering*, vol. 2, no. 2, pp. 161–187, 1996.

[21] V. Ambriola and V. Gervasi, "On the systematic analysis of natural language requirements with CIRCE," *Automated Software Engineering*, vol. 13, no. 1, pp. 107–167, 2006.

[22] T. Yue, L. C. Briand, and Y. Labiche, "aToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models," *ACM Transactions on Software Engineering and Methodology*, vol. 24, no. 3, 2015.

[23] ——, "A systematic review of transformation approaches between user requirements and analysis models," *Requirements Engineering*, vol. 16, pp. 75–99, 2011.

[24] 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, no. 3, pp. 339–358, 2017.

[25] R. Saini, G. Mussbacher, J. L. Guo, and J. Kienzle, "Automated, interactive, and traceable domain modelling empowered by artificial intelligence," *Software and Systems Modeling*, pp. 1–31, 2022.

[26] J. Cámara, J. Troya, L. Burgueño, and A. Vallecillo, "On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml," *Software and Systems Modeling*, vol. 22, no. 3, pp. 781–793, 2023.

[27] S. Arulmohan, M.-J. Meurs, and S. Mosser, "Extracting domain models from textual requirements in the era of large language models," in *Companion Proc. of the International Conference on Model Driven Engineering Languages and Systems*. ACM/IEEE, 2023, pp. 580–587.

[28] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with ChatGPT," *arXiv preprint arXiv:2302.11382*, 2023.

[29] OpenAI, "Prompt engineering," https://platform.openai.com/docs/guides/prompt-engineering?ref=blef.fr, [Accessed 20-12-2023].

[30] F. Dalpiaz, "Requirements data sets (user stories)," Mendeley Data, V1, doi: 10.17632/7zbk8zsd8y.1, 2018.

[31] M. Bragilovski, S. Erez, C. Mordehai, S. Rahamim, N. Shpack, and A. Sturm, "TAGRAM: A framework for tagging user stories," in *Proc. of the International Requirements Engineering Conference Workshops*. IEEE, 2023, pp. 62–66.

[32] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica*, vol. 22, no. 3, pp. 276–282, 2012.

[33] D. Falessi, J. Roll, J. L. Guo, and J. Cleland-Huang, "Leveraging historical associations between requirements and source code to identify impacted classes," *IEEE TSE*, vol. 46, no. 4, pp. 420–441, 2018.

[34] J. A. H. López, J. L. Cánovas Izquierdo, and J. S. Cuadrado, "ModelSet: a dataset for machine learning in model-driven engineering," *Software and Systems Modeling*, vol. 21, no. 3, pp. 967–986, 2022.

[35] M. Bragilovski, A. T. van Can, F. Dalpiaz, and A. Sturm, "Material of the paper "Deriving Domain Models from User Stories: Human vs. Machines"," Jul. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.12740518

[36] D. M. Berry, "Empirical evaluation of tools for hairy requirements engineering tasks," *Empirical Software Engineering*, vol. 26, no. 6, p. 111, 2021.

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

[38] T.-T. Wong, "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation," *Pattern Recognition*, vol. 48, no. 9, pp. 2839–2846, 2015.

[39] D. Dell'Anna, F. B. Aydemir, and F. Dalpiaz, "Evaluating classifiers in SE research: the ECSER pipeline and two replication studies," *Empirical Software Engineering*, vol. 28, no. 1, p. 3, 2023.

[40] D. W. Zimmerman and B. D. Zumbo, "Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks," *The Journal of Experimental Education*, vol. 62, no. 1, pp. 75–86, 1993.

[41] D. G. Pereira, A. Afonso, and F. M. Medeiros, "Overview of Friedman's test and post-hoc analysis," *Communications in Statistics-Simulation and Computation*, vol. 44, no. 10, pp. 2636–2653, 2015.

[42] J. Cohen, "Statistical Power Analysis," *Current Directions in Psychological Science*, vol. 1, no. 3, pp. 98–101, 1992.

[43] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. of the International Conference on Neural Information Processing Systems*. ACM, 2017, pp. 4765–4774.

[44] H. Harmain and R. Gaizauskas, "CM-Builder: A natural language-based case tool for object-oriented analysis," *Automated Software Engineering*, vol. 10, pp. 157–181, 2003.

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[46] G. Lucassen, F. Dalpiaz, J. M. E. Van Der Werf, and S. Brinkkemper, "Forging high-quality user stories: Towards a discipline for agile requirements," in *Proc. of the International Requirements Engineering Conference*. IEEE, 2015, pp. 126–135.