From User Stories to Domain Models: Recommending Relationships between Entities

Maxim Bragilovski[1,*], Fabiano Dalpiaz[2] and Arnon Sturm[1]

[1]Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel
[2]Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

User stories are a common notation for expressing requirements, especially in agile development projects. While user stories provide a detailed account of the functional requirements, they fail to deliver a holistic view of the domain. As such, they can be complemented with domain models that not only help gain this comprehensive view, but also serve as a basis for model-driven development. We focus on the task of recommending relationships between entities in a domain model, assuming that these entities were previously extracted from a user story collection either manually or through an automated tool. We investigate whether an approach based on supervised machine learning can recommend essential relationships in a domain model more accurately than state-of-the-art rule-based methods. Based on a collection of datasets that we manually labeled and a set of 32 features we engineered, we train a machine learning model by using a random forest classifier. The results indicate that our approach has higher precision and F_1-score than the baseline rule-based methods. Our findings provide preliminary evidence of the suitability of using machine learning to support the development of domain models, especially in recommending relationships between related entities.

Keywords

Requirements Engineering, Conceptual Modeling, Domain Models, Machine Learning, Model Derivation

1. Introduction

User stories are a widespread notation for expressing functional requirements from the perspective of a user [1]. Despite their popularity and simplicity, each user story describes an individual feature of the system, thereby making it hard for an analyst to obtain a holistic view of the system domain. As a solution, researchers have investigated the automated and manual derivation of different types of conceptual or domain models from user stories [2, 3].

A conceptual model is a graphical representation of static phenomena (such as entities and relationships) as well as dynamic phenomena (such as events and processes) in some domain [4]. Conceptual models can be used to illustrate the functionality of a system, such as *use case diagrams*. Furthermore, they may be used to provide a holistic view of the main entities and relationships that appear in the requirements [5, 2]. These models can be used as a basis for identifying ambiguities [6], for analyzing qualities such as security and privacy [7], and as a starting point for model-driven engineering.

*Corresponding author.
maximbr@post.bgu.ac.il (M. Bragilovski); f.dalpiaz@uu.nl (F. Dalpiaz); sturm@bgu.ac.il (A. Sturm)
0000-0002-4778-7897 (M. Bragilovski); 0000-0003-4480-3887 (F. Dalpiaz); 0000-0002-4021-7752 (A. Sturm)

Conceptual and domain model development is a challenging activity, which requires the identification of the important concepts (in the case of a structural model, *entities*) and their relationships. To do so, it is important to distinguish between the essential concepts in a domain and the secondary ones. Furthermore, the resources used to develop the conceptual model (i.e., the requirements) make use of ambiguous terms [6]. Moreover, as the complexity of the system increases, it becomes more time consuming for humans to derive these models.

To address the challenges of developing conceptual and domain models, several solutions exist, including guidelines [8] and automatic approaches [2]. The existing automated methods are rule-based; this limits their effectiveness to those linguistic patterns that the researchers encoded into the rules. In contrast, methods that rely on guidelines for humans [8] are time consuming and do not achieve perfect accuracy either.

In our research agenda, we aim to build machine and deep learning models for deriving a domain model from a collection of user stories. A domain model should contain the entities and relationships that represent the domain of the system that implements the user stories. This model can serve as a basis for model-driven development, e.g., via low-code development platforms. Thus, the automated derivation could increase the usefulness of user stories by reducing the gap between requirements and the following development activities.

In this paper, we present initial results on the automated derivation of a conceptual model. As the current automated state-of-the-art method, the Visual Narrator [2], is more effective at identifying entities than relationships, we choose the *relationship identification task* as our first research step. We propose a machine learning-based model that recommends essential relationships between the entities that are derived from a set of user stories. Our research question is as follows: *Does a machine-learning-based approach outperform rule-based state-of-the-art methods for identifying relationships between the entities extracted from user stories?*

The results reported in this paper positively answer that question and demonstrate the advantages of using machine learning for the task at hand. In particular, we make the following contributions: (i) we describe a novel approach, based on 32 features, for recommending essential relationships using a machine learning model; and (ii) we compare our machine learning model to current automated models.

Paper organization. In Section 2, we discuss the background and related studies. In Section 3, we present the research method and we describe our proposed approach. In Section 4, we report on the preliminary results and we discuss the limitatinos. Finally, in Section 5 we conclude and set plans for future research.

2. Related Work

Deriving conceptual models automatically from natural language requirements has been a research topic for quite some time [9]. Even so, despite Mike Cohn's book on user stories [10] that contributed to the popularity of user stories, only in 2016 Robeer *et al.* [11, 2] performed the first major attempt at extracting conceptual models from user stories.

Since then, research about deriving models from user stories started to emerge. In this section, we review related studies by referring, when applicable, to the different types of models that are extracted, the method through which the model is derived, the experimental setting and

datasets, the metrics, and the performance that was achieved.

Elallaoui *et al.* [12] use part-of-speech tagging to identify whether certain keywords should represent entities or relationships, and this information is used to generate use case diagrams. Their approach is evaluated via precision and recall. They compare the outcomes with models that were created manually from the WebCompany dataset [11]. The results demonstrate that their plugin has acceptable precision and recall for detecting actors, and high results (above 0.85 for both metrics) for detecting use cases and relationships. While they derive a use case diagram, we are interested in generating domain models that require a holistic view.

Similarly, the recent studies that extract class diagrams automatically also rely on the part-of-speech tagging of terms within user stories. Lucassen *et al.* [11, 2] propose an automated approach, based on the Visual Narrator tool, for extracting structural conceptual models (i.e., class diagrams) from a set of user stories. The Visual Narrator was used to generate conceptual models from user stories based on 11 out of 23 identified heuristics from the literature. Using precision, recall and F_1-score metrics, they determined whether their tool was successful in identifying entities and relationships compared to gold-standard models that were created by the authors of the paper. The approach achieved good precision (97%) and recall (98%), with a lower bound of 88% recall and 92% precision. These results, however, are obtained by assessing the tool's performance against a human execution of the algorithm, rather than against models that are created by humans based on their own rationale.

Typically, automated model derivation from user stories is done using rule-based methods based on natural language processing heuristics. Although these works achieved good precision and recall despite limitations of user stories like ambiguity [13], they cannot be perfectly accurate due to the variety of linguistic patterns that natural language allows. Furthermore, they are limited to the lexicon they identified and cannot perform the abstraction process that is crucial to conceptual models.

Approaches that derive domain models from other formats of requirements also exist. The most relevant work is that by Arora and colleagues [5], who use heuristics to create a first version of a domain model and then apply active learning to remove superfluous elements. We also use machine learning, but rather than pruning elements, we focus on enriching a model by suggesting essential relationships.

3. Research Method and Proposed Approach

The task of this research – illustrated by the mock-up of Figure 1 – consists of recommending relationships among the entities in a domain model. We assume these entities have been extracted previously from a collection of user stories, either manually [8] or through an automated tool such as the Visual Narrator [2]. Given a collection of user stories (in the figure, regarding Planning Poker), selected entities, and a probability threshold, the tool suggests relationships whose probability to exist is higher than the set threshold, and then visualizes the resulting domain model with those relationships.

To develop our ML technique for such a tool, we followed a common machine learning method [14], which consists of five steps. *Dataset preparation*, based on which we created a gold standard model for each set of user stories. *Feature engineering*, where features are

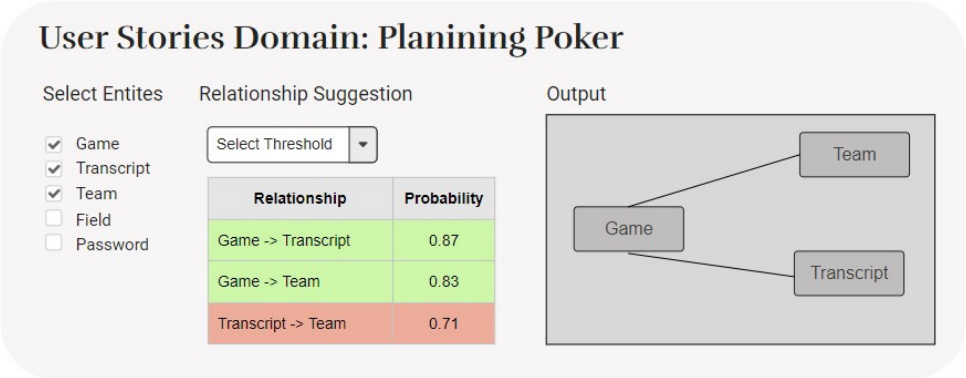

Figure 1: A mock-up that illustrates the task where this research fits: the recommendation of relationships between entities extracted from a collection of user stories.

created to facilitate the recommendation of relationships between two entities. *Baselines and alternatives selection,* in order to compare our method to current state-of-the-art approach. *Choice of a machine learning algorithm* from the current state-of-the-art families of machine learning models (e.g., decision trees, random forest). *Metrics selection*, to determine against which criteria we compare the performance of different approaches.

3.1. Data preparation

As there is no benchmark dataset of user stories with an associated class diagram, we developed such a dataset. Indeed, the existing gold standards for the datasets used with the Visual Narrator [2] are not suitable, as the identified relationships are meant to navigate through the user stories, rather than for representing the domain). Therefore, we first selected 7 sets of user stories from an online collection of user stories data sets [15]. Next, for each set of stories, we developed a conceptual model. During this process, we had to answer the following questions: (1) *What entities are of interest to find relationships between?* (2) *What are the relationships that we want the model to recommend?* Table 1 shows descriptive information about the seven datasets.

3.1.1. Entities Extraction

The first step is the identification of the entities from the user stories. To do so, we first used the Visual Narrator tool [2] with its default parameters. Since the entities that the Visual Narrator returns include both domain terms as well as technical concepts that would not be part of a domain model, we filtered manually its outputs by retaining only those entities that we considered to be part of the domain, thereby excluding technical terms that pertain to the solution. We acknowledge that some entities may have been overlooked because of this filtering. However, as this paper focuses on detecting the relationship between pre-defined entities, the omission of entities should not affect our analysis of the recommended relationships between the existing entities.

Table 1
Descriptive information about the gold standards for the employed datasets, showing the number of user stories, entities, relationships, and the percentage of essential relationships ($\frac{\#Rel}{(\#Ent \times (\#Ent-1))/2}$), and percentage of entities that co-occur in at least one user story.

DataSet	# US	# Ent	# Rel	% Essential Relationships	% Co-Occurring Entities
planningpoker	53	8	6	21.4%	60.7%
datahub	67	13	7	8.9%	34.6%
camperplus	55	19	17	9.9%	27.4%
zooniverse	60	20	6	3.3%	24.2%
recycling	51	6	3	20.0%	46.6%
frictionless	67	17	18	13.2%	23.5%
unibath	53	14	14	15.3%	20.8%

3.1.2. A gold standard for relationships between entities

After extracting the entities from the set of user stories, we developed a dataset that contains all the possible relationships that might exist (i.e., all pairs of entities). As a next step, each author of this paper tagged each relationship independently as follows:

1. **Essential:** it is required in the domain model for implementing the user stories in the collection;
2. **Optional:** it may or may not exist because of the existence of other relationships;
3. **Unnecessary:** it should not be part of the domain model.

Next, we measured the inter-rater agreement using the Fleiss Kappa [16], which is a statistical measure specifically designed to handle categorical data and to handle more than two raters. We checked the agreement in two ways: (i) binary, where we consider only strong disagreements if the three tags include at least one *essential* and at least one *unnecessary*; and (ii) multi-class, where we consider disagreements even when considering the *optional* class.

Afterward, we held a discussion that eventually led to the gold standards which can be found in [17]. We decided that the gold standard should include relationships on which we have a high agreement. This is intended to minimize the chance of false positives in our gold standard. Because of our high agreement of more than 0.6, we chose the binary classification as the gold-standard, as this improves our identification of true positives: essential relationships (class 1) and all others (class 0).

3.2. Feature engineering

We engineered a set of features ($[x_i]$) to characterize each pair of entities ((e_{i_1}, e_{i_2}), that the ML model uses to learn which relationships are essential and which are unnecessary (y_i). Based on this, the trained ML model can recommend essential relationships on unseen pairs of entities.

We engineered features based on rules for relationship identification [2] as well as from additional insights we gained after exploring the data. We denote *sim* as a function that calculates the similarity between two words or sentences. sim_g (global similarity) implements that function

using pre-traind model from `nltk`[1], and sim_l (local similarity) implements that function using word2vec model from `gensim`[2].

The user story datasets were used as our corpus to train the gensim model, which resulted in an embedding vector for each entity that can be used to calculate cosine-similarity (the code to train the model appears in our online appendix [17]). We represent the dataset as follows: $\mathcal{D} = \{(r_0, \mathbf{x_0}, y_0), ..., (r_n, \mathbf{x_n}, y_n)\}$ where $r_i = (e_{i_1}, e_{i_2})$ is a relationship between two entities e_{i_1} and e_{i_2}, $\mathbf{x_i}$ is a vector of the features' values, and y_i is the target label (essential or unnecessary relationship). We also denote $\mathcal{D}' = \{r'_0, ..., r'_n\}$ as an external dataset that contains all the relationships between two entities $r'_i = (e'_{i_1}, e'_{i_2})$ from different existing domain model repositories (we used the `ModelSet` repository [18] [3]). The similarity between two relationships $r_i = (e_{i_1}, e_{i_2})$ and $r_j = (e_{j_1}, e_{j_2})$ is calculated as follows:

$$rel\_sim_x(r_i, r_j) = \frac{sim_x(e_{i_1}, e_{j_1}) + sim_x(e_{i_2}, e_{j_2})}{2} \tag{1}$$

where x is the way the similarity *sim* is calculated: $x \in \{g, l\}$. Each $r_i = (e_{i_1}, e_{i_2}) \in \mathcal{D}$ is associated with a vector of values for the engineered features listed in Table 2.

These features were assigned based on the following rationale. Each of Features 1–3 considers external sources; we search in ModelSet and define features with the similarity value of those relationships that have the highest similarity with the examined relationship, applying Equation 1. We expect that if other models link entities that are similar to ours, our approach will also recommend a relationship. Each of Features 4–9 does a similar analysis but based on each individual entity. Feature 10 calculates the average of Features 1–3. Each of Features 11–18 characterizes individual entities by counting how many times an entity appears in the user stories (11–12) and whether it appears in the actor, action, or benefit part of at least one user story (13–18). Each of Features 19–20 calculates the similarity between the entities using `gensim` and NLTK. Feature 21 determines the number of user stories where both entities co-occur. Each of Features 22–23 does a similar calculation but considers only co-occurrences where at most, 3 or 5 words exist between the entities. Each of Features 24–27 is a binary value that is true when both entities are identified as either subject or object in at least one user story. Feature 28 is true if there is a user story where there is an 'and' or an 'or' word between the two entities. Feature 29 counts the nouns that appear in a user story that includes e_{i_1} and in a user story that includes e_{i_2}. Features 30–32 normalize the number of user stories where at least one entity occurs over the number of user stories where both occur.

3.3. Evaluation Settings

To select relevant machine learning models, we distinguish between two types of models: *shallow* and *deep*. Shallow models such as decision trees are better suited for small, structured datasets. In contrast, deep models are better suited for large NLP and vision datasets. We do not select deep models because NLP-for-RE tasks like ours rely on small datasets [19]. Thus, we

[1] https://nltk.org
[2] https://pypi.org/project/gensim/
[3] https://modelset.github.io/

Table 2
Features used by our machine-learning model for a relationship $r_i = (e_{i_1}, e_{i_2}) \in \mathcal{D}$

ID	Feature	Description
1–3	Ext._rel_sim$_g$_k	The k highest ($k \in \{1, 2, 3\}$) relationship similarity values between r_i and any of the relationships in ModelSet ($r'_j \in \mathcal{D}'$)
4–9	Ext._entity$_t$_sim$_g$_k	For both entities in r_i, the global similarity with the corresponding entity in the k most similar relationships in $\mathcal{D}'$: $sim_g(e_{i_t}, e'_{j_t})$ ($t \in \{1, 2\}$)
10	sim_avg_3_rel	average(Ext._rel_sim$_g$_1 , Ext._rel_sim$_g$_2, Ext._rel_sim$_g$_3)
11–12	appear$_t$	Number of appearances of e_{i_t} in the user stories, for each $t \in \{1, 2\}$
13–14	actor$_t$	1 if e_t appears in the role part of at least one user story, otherwise 0
15–16	action$_t$	1 if e_t appears in the action part of 1+ user story, otherwise 0
17–18	benefit$_t$	1 if e_t appears in the benefit part of 1+ user story, otherwise 0
19–20	sim$_x$	$sim_x(e_{i_1}, e_{i_2})$, for $x \in \{g, l\}$
21	both	The number of user stories in which e_{i_1} and e_{i_2} co-occur
22–23	window$_z$	The number of user stories where e_{i_1} and e_{i_2} co-occur with less than $z \in \{3, 5\}$ words in between them
24-27	sub/obj_sub/obj	1 if e_{i_1} is identified as $\{subject, object\}$ and e_{i_2} is identified as $\{subject, object\}$ in 1+ user story, otherwise 0
28	and_or_btw	1 if there is a user story where 'and' or 'or' appeared between e_{i_1} and e_{i_2}
29	common_friends	Number of different nouns that appear both in a user story where e_{i_1} occurs and in a user story where e_{i_2} occurs
30–32	both$_w$	Number of user stories where $w \in \{e_{i_1},\ e_{i_2}, e_{i_1} \vee e_{i_2}\}$ occur divided by feature 21

opt for a shallow model in the form of Random Forest (RF), a state-of-the-art technique that achieved the best results in some software-engineering-related tasks [20].

We report the results in terms of the commonly used metrics of precision, recall, and F_1 score. Our statistical analysis, however, focuses only on precision and F_1 score. We choose *precision* as we assume it might be more helpful to humans than recall in a recommendation scenario like the one sketched in Figure 1, where having a smaller set of essential links without many unnecessary relationships creates less noise for the analyst than having all the essentials with many unnecessary ones. We also analyze the F_1-score because it balances both precision and recall, thereby penalizing recommendations that provide a too limited number of essential links. We acknowledge that these are preliminary metrics that we use for an early assessment of our approach; future work should determine the most suitable metric based on an in-practice analysis of the impact of different types of errors [21].

We compare the performance of the *RF classifier* against the Visual Narrator [2] and a naive approach in which an essential relationship is suggested every time two entities appear in the same user story. As the Visual Narrator did not identify all the entities in the gold standard model, we omitted these entities from the evaluation. This is done since we are only assessing the ability to predict relationships between entities. Also, we defined the threshold that the ML model uses to discriminate between the two classes: essential or unnecessary. Since the *RF*

classifier returns a probability of a relationship being essential and the dataset is unbalanced, it is not reasonable to set the threshold to 0.5. After checking several thresholds, we found that a threshold of 0.8 provides reliable results.

We evaluate the performance of the *RF classifier*, the *Visual Narrator*, and the *Naive* approach using the seven datasets presented in Table 1. We apply the leave-one-out evaluation method: all datasets except one are used for training the model, and we report the performance on the remaining dataset.

To compare if the differences in the metrics are significant between the approaches (*Independent Variables*), we selected F_1-score and precision (*Dependent Variables*) as metrics to check the significance. We set the following (null) hypothesis:

- The three F_1-scores/precision of the naive approach, the Visual Narrator and the RF classifier are the same ($H_0^{EXP\text{-}F\text{-}score}$ and $H_0^{EXP\text{-}Precision}$).

The experiment materials can be found in an online appendix [17].

4. Preliminary Validation

In this section, we report on the results of the preliminary validation we conducted according to the method described in Section 3.3.

4.1. Descriptive Statistics

Table 3 presents the results of the experiment. The Dataset column refers to the set of user stories. We report on the results of the three alternatives: the *Naive*, *Visual Narrator*, and *RF classifier*. For each alternative, we present the precision, recall, and the F_1-score. The bottom row of the table represents the macro-average for each column. The numbers in bold indicate the best results of the F_1-score for a given user story dataset.

Table 3
Results of the experiment

Dataset	Naive			Visual Narrator			RF classifier		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score
Planing Poker	0.357	0.833	0.500	0.500	0.167	0.250	0.800	0.667	**0.727**
DataHub	0.500	1.000	0.667	0.571	1.000	0.727	0.750	0.750	**0.750**
Campers	0.280	0.875	0.424	0.167	0.125	0.143	0.444	0.500	**0.471**
Zooniverse	0.150	0.750	0.250	0.143	0.250	0.182	0.250	0.500	**0.333**
Recycling	0.400	1.000	**0.571**	0.333	0.500	0.400	0.333	0.500	0.400
frictionless	0.667	1.000	**0.800**	0.000	0.000	0.000	0.750	0.750	0.750
unibath	0.500	1.000	**0.667**	0.000	0.000	0.000	1.000	0.333	0.500
Average	0.408	0.923	0.565	0.245	0.292	0.266	0.654	0.536	**0.589**

In most datasets, using the RF classifier leads to better F_1-scores. Particularly, it achieved superior results in 4 out of 7 datasets. The *RF classifier* achieved an average F_1-score of 0.589, *Naive Approach* achieved 0.565 and *Visual Narrator* only achieved 0.266. In addition, we observe that the RF classifier has better precision over the other alternatives in 6 out of 7 datasets.

We conducted statistical tests with $alpha = 0.05$ to determine if the differences are statistically significant. We applied the Friedman test [22], a non-parametric statistical test, to compare more than two methods. We found statistically significant differences among the related approaches with $p = 0.01$ for both F_1-score and precision. Therefore we can reject the $H_0^{EXP1\text{-}F\text{-}score}$ and $H_0^{EXP1\text{-}Precision}$ hypotheses. To check which alternative is better, we applied Nemenyi's post-hoc test [23], and we calculated effect size using Cohen's d. We found that: (1) the *RF classifier* is statistically better than the *Visual Narrator* ($p = 0.042$ and $p = 0.02$) with effect sizes of 1.564 and 1.591; (2) there is no statistically significant difference between the *RF classifier* and *Naive* ($p = 0.9$ and $p = 0.608$) with effect sizes of 0.042 and 0.997; (3) there is no statistically significant difference between *Naive* and the *Visual Narrator* ($p = 0.111$ and $p = 0.191$) with effect sizes of 1.518 and 0.877 for F_1-score and precision, respectively.

4.2. Discussion and Limitations

The results answer positively our research question as they indicate that using an ML-based model (*RF classifier*) for the relationship recommendation task leads to higher F_1-score and precision than the rule-based alternatives (*Naive* and *Visual Narrator*). Furthermore, the RF classifier also returns the probabilities for occurrences of relationships, providing extra information for the user to make the final decision. The preliminary results require additional validation, such as defining the most suitable metrics by analyzing the relative impact of Type 1 and Type 2 errors [21], by estimating human achievable performance, and to assess the necessary effort (time). We could not estimate the human achievable performance on our datasets as we were already familiar with some of those from previous research, and due to an iterative approach for the construction of the gold standard. Lastly, the selection of the datasets may be biased; although they differ in the number of samples (pairs of entities) and in the distribution of features and classes, we need to experiment with other datasets to draw more robust conclusions.

5. Conclusions and Future Work

We have presented an ML-based model for recommending relationships between the entities of conceptual models that are derived from a set of user stories.

Rule-based approaches and guidelines were suggested for deriving conceptual models from user stories. They achieve good accuracy in recognizing entities but fall short in finding relationships between these entities. Here, we provide initial evidence that an ML-based approach improves the current state-of-the-art methods for recommending relationships between entities.

This work calls for further improvements. The ML-based models can be extended to suggest a complete conceptual model (entities, attributes, and relationships) as well as performing a better evaluation that compares the tool's performance with that of human analysts.

References

[1] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, The use and effectiveness of user stories in practice, in: Proc. of REFSQ, Springer, 2016, pp. 205–222.

[2] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, Extracting Conceptual Models from User Stories with Visual Narrator, Req. Eng. 22 (2017) 339–358.
[3] F. Dalpiaz, P. Gieske, A. Sturm, On deriving conceptual models from user requirements: An empirical study, Inf. Softw. Technol. 131 (2021) 106484.
[4] Y. Wand, R. Weber, Research commentary: Information systems and conceptual modeling–a research agenda, Information Systems Research 13 (2002) 363–376.
[5] C. Arora, M. Sabetzadeh, S. Nejati, L. Briand, An Active Learning Approach for Improving the Accuracy of Automated Domain Model Extraction, ACM TOSEM 28 (2019).
[6] F. Dalpiaz, I. van der Schalk, S. Brinkkemper, F. B. Aydemir, G. Lucassen, Detecting Terminological Ambiguity in User Stories: Tool and Experiment., Inf. Soft. Tech. (2019).
[7] P. X. Mai, A. Goknil, L. K. Shar, F. Pastore, L. C. Briand, S. Shaame, Modeling Security and Privacy Req.: a Use Case-Driven Approach, Inform. Soft. Tech. 100 (2018) 165–182.
[8] M. Bragilovski, F. Dalpiaz, A. Sturm, Guided derivation of conceptual models from user stories: A controlled experiment, in: Proc. of REFSQ, 2022, pp. 131–147.
[9] T. Yue, L. C. Briand, Y. Labiche, aToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models, ACM TOSEM 24 (2015).
[10] M. Cohn, User stories applied: For agile software develop., Addison-Wesley Prof., 2004.
[11] M. Robeer, G. Lucassen, J. M. E. Van Der Werf, F. Dalpiaz, S. Brinkkemper, Automated extraction of conceptual models from user stories via nlp, in: RE, IEEE, 2016, pp. 196–205.
[12] M. Elallaoui, K. Nafil, R. Touahni, Automatic transformation of user stories into UML use case diagrams using NLP techniques, Procedia computer science 130 (2018) 42–49.
[13] A. R. Amna, G. Poels, Ambiguity in user stories: A systematic literature review, Information and Software Technology 145 (2022) 106824.
[14] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms, Cambridge University Press, 2014.
[15] F. Dalpiaz, Requirements data sets (user stories), Mendeley Data, V1, doi: 10.17632/7zbk8zsd8y.1, 2018.
[16] J. L. Fleiss, Measuring nominal scale agreement among many raters., Psychological bulletin 76 (1971) 378.
[17] M. Bragilovski, F. Dalpiaz, A. Sturm, Experimental material - from user stories to domain models: Recommending relationships between entities, 2023. URL: http://dx.doi.org/10.17632/tvjyw4pzsk.1, Mendeley Data, v1.
[18] J. A. H. López, J. L. Cánovas Izquierdo, J. S. Cuadrado, Modelset: a dataset for machine learning in model-driven engineering, Soft. and Systems Modeling 21 (2022) 967–986.
[19] F. Dalpiaz, A. Ferrari, X. Franch, C. Palomares, Natural Language Processing for Requirements Engineering: The Best Is Yet to Come, IEEE Software 35 (2018) 115–119.
[20] D. Falessi, J. Roll, J. L. Guo, J. Cleland-Huang, Leveraging historical associations between requirements and source code to identify impacted classes, IEEE TSE 46 (2018) 420–441.
[21] D. M. Berry, Empirical evaluation of tools for hairy requirements engineering tasks, Empirical Software Engineering 26 (2021) 111.
[22] D. W. Zimmerman, B. D. Zumbo, Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks, The Journal of Exper. Education 62 (1993) 75–86.
[23] D. G. Pereira, A. Afonso, F. M. Medeiros, Overview of Friedman's test and post-hoc analysis, Communications in Statistics-Simulation and Computation 44 (2015) 2636–2653.