Assessing the Performance of Automated Model Extraction Rules

Jorge Echeverría, Francisca Pérez, Óscar Pastor and Carlos Cetina

Abstract. Automated Model Extraction Rules take as input requirements (in natural language) to generate domain models. Despite the existing work on these rules, there is a lack of evaluations in industrial settings. To address this gap, we conduct an evaluation in an industrial context, reporting the extraction rules that are triggered to create a model from requirements and their frequency. We also assess the performance in terms of recall, precision and F-measure of the generated model compared to the models created by domain experts of our industrial partner. Results enable us to identify new research directions to push forward automated model extraction rules: the inclusion of new knowledge sources as input for the extraction rules, and the development of specific experiments to evaluate the understanding of the generated models.

Keywords: Conceptual Models, Natural Language Requirements, Model Extraction.

A prior version of this paper has been published in the ISD2017 Proceedings (http://aisel.aisnet.org/isd2014/proceedings2017).

Jorge Echeverría (✉)
Universidad San Jorge, Zaragoza, Spain
e-mail: jecheverria@usj.es

Francisca Pérez
Universidad San Jorge, Zaragoza, Spain
e-mail: mfperez@usj.es

Óscar Pastor
Universitat Politècnica de València, Valencia, Spain
e-mail: opastor@pros.upv.es

Carlos Cetina
Universidad San Jorge, Zaragoza, Spain
e-mail: ccetina@usj.es

N. Paspallis et al. (eds.), Information Systems Development Methods, Tools and Management
Lecture Notes in Information Systems and Organisation, XX
DOI XX.XXXX/XXXXXXX

1 Introduction

Software requirements specifications are prevalently expressed using Natural Language (NL) [13]. The transition from requirements expressed in NL to a domain model is an important step to obtain a precise and analyzable specification [20]. Automated model extraction from NL requirements has been studied for a long time, with a large body of literature already existing in the area such as [7], [8], [11], [14], [18], [21].

Automated model extraction applies model extraction rules. Nevertheless, crucial aspects about the existing Automated Model Extraction Rules (AMER) remain underexplored such as the AMER that are triggered to build a domain model, and the differences between the model generated by applying the AMER and the models generated by domain experts for a given NL requirements specification. These differences can be accentuated in many industrial situations [4].

However, the large majority of existing work on model extraction is evaluated over exemplars and in artificial settings. Evaluations on model extraction in real settings remain scarce. This work, which is conducted in an industrial context, takes a step towards addressing this gap by assessing the performance of the AMER. This allows us to evaluate whether the result obtained from the AMER is closer to the results obtained from the domain experts.

In this work, we design a process made up of four steps in order to compare the model generated according to the AMER with the models generated by the domain experts of our industrial partner, which is a worldwide provider of railway solutions. First of all, a model is generated from a requirements specification using AMER. In the second step each one the domain experts of our industrial partner generated a model from a requirements specification. Next, in the third step, some Natural Language Processing (NLP) techniques are applied to homogenize the words used in all models. Finally, we obtain as results both a report with the occurrences of each AMER triggered by the requirements, and a report with the performance measurement in terms of precision, recall and F-measure values.

Our results show that 10 of 18 AMER are triggered, providing insights about the rules that are capable of deriving a model from NL requirements in realistic settings.

Moreover, our results of performance show an average value of 78.75% in terms of recall and 75.55% in terms of precision. Furthermore, results enable us to identify new research directions to push forward the AMER: It is necessary to consider new knowledge sources that can play the role of tacit knowledge, and it is necessary to perform specific experiments to evaluate the understanding of models generated by the AMER.

The paper is structured as follows: Section 2 provides the required background on the AMER. Section 3 describes our process. Section 4 shows the results, and Section 5 presents a discussion of the results. Section 6 deals with the threats to validity. Section 7 summarizes the works related to this paper. Finally, Section 8 concludes the paper.

2 Background

The AMER used in this work appear in [5]. The authors summarize the literature on model extraction from unrestricted NL requirements and identify a set of extraction rules. These AMER are shown in Fig. 1. These AMER are organized into four categories based on the nature of the information they extract: concepts, associations and generalizations, cardinalities, and attributes. These categories are defined as follows:

- **Concepts** are the items in the real world that the domain experts are trying to discover for building a domain model.
- **Associations and generalizations** describe a naturally occurring relationship between specific concepts.
- **Cardinalities** are measures of the number of links between one concept and another concept in a relationship.
- **Attributes** are defined as descriptive pieces of information about concepts.

The above AMER have two limitations: (1) they do not cover link paths [2], these rules enable the extraction of relations between concepts that are only indirectly related, and (2) they do not fully exploit the results from NLP tools, these tools provide detailed information about the dependencies between different segments of sentences.

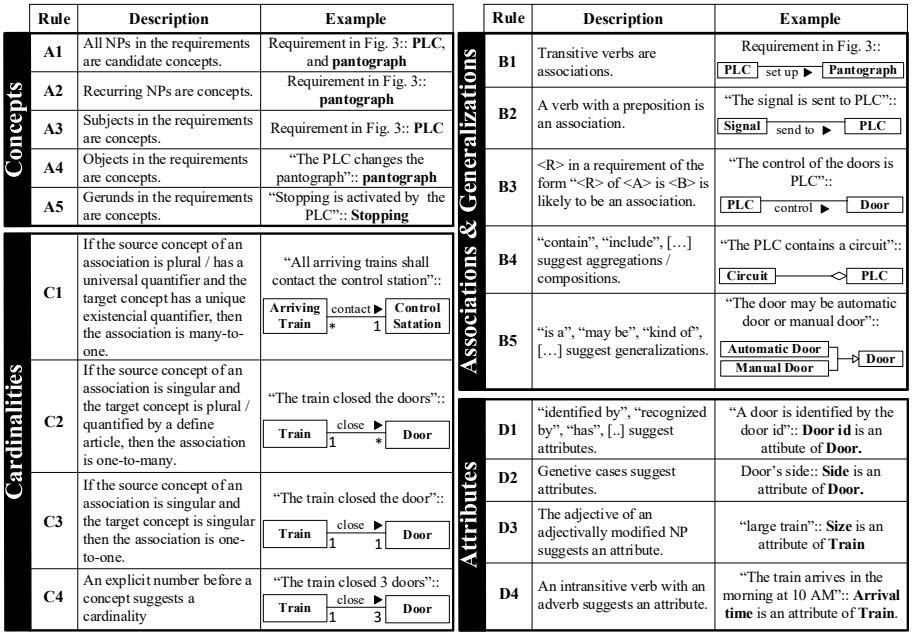

Fig. 1. Automated Model Extraction Rules

There is a large body of literature about the automated extraction of models from NL requirements, the large majority of existing work on model extraction is evaluated

over no real environments. Thus, there is a need to conduct evaluations in industrial contexts. For this reason, our work aims to cover the lack of evaluations to analyze the models generated from requirements specifications written in NL in real contexts. Our aim is to compare the models generated using the AMER with the models generated by the domain experts in an industrial context.

3 Process

In order to compare the model generated according to the AMER with the models generated by the domain experts, we design a process made up of four steps, marked I-IV in Fig. 2. First, the requirements specification in NL is taken as input to generate a model by applying the AMER. Second, the domain experts take as input the requirements specification to generate a model for each domain expert. Third, NLP techniques (e.g., Parts-Of-Speech Tagging and root reduction) are applied to homogenize the words used in both the model generated according to the AMER and the model generated by the domain experts. Finally, we obtain as results both a report with the occurrences of each extraction rule triggered by the requirements, and a report with the performance measurement in terms of precision and recall values by comparing the natural language processed model obtained from the AMER with the natural language processed model obtained from each domain expert.

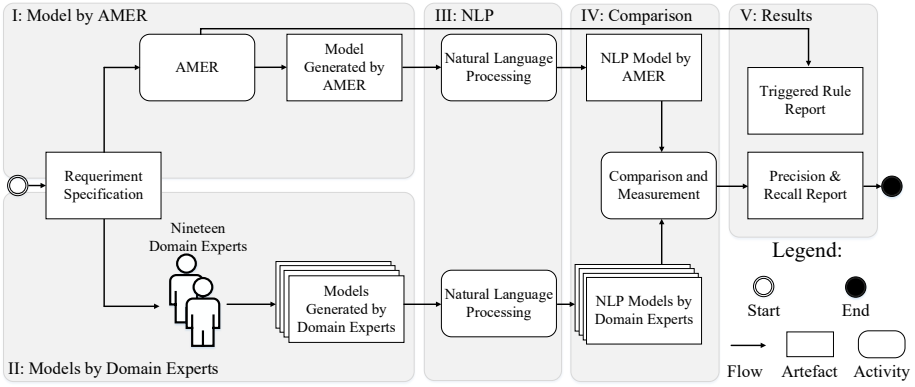

Fig. 2. Process overview

3.1 Model generated by the AMER

To generate the model by applying the AMER (see Fig. 1) it is necessary to provide as input a requirements specification. In this work, the requirements specification provided as input was stated by a domain expert, who is not involved in this paper. The requirements specification is made up of four requirements, which have an average length of 28 words. In general, requirements are expressed using NL text in a large number of software projects, and the railway domain is no exception [16]. NL is

used to specify requirements due to its high degree of understandability among all of the stakeholders in industrial projects [9].
In the requirements, we identify the units of interest that are noun phrases and verbs. A noun phrase (NP) is a unit that can be the subject or the object of a verb. A verb (VB) appears in a verb phrase (VP) alongside any direct or indirect objects, but not the subject. Verbs can have auxiliaries and modifiers (typically adverbs) associated with them. After the NPs and VBs are identified, we find grammatical dependencies between individual words in a sentence, e.g., the subject and the object. Finally, we apply the AMER shown in Fig. 1, which are organized in four categories (concepts, associations and generalizations, cardinalities, and attributes), in order to construct the model. The model obtained as a result of applying the AMER to the requirements specification has 67 elements.

The upper part of Fig. 3 shows an example of a requirement in which the main units of interest to apply the AMER are highlighted (e.g., nouns and verbs), whereas the lower part of Fig. 3 shows the model obtained as a result of applying the AMER.

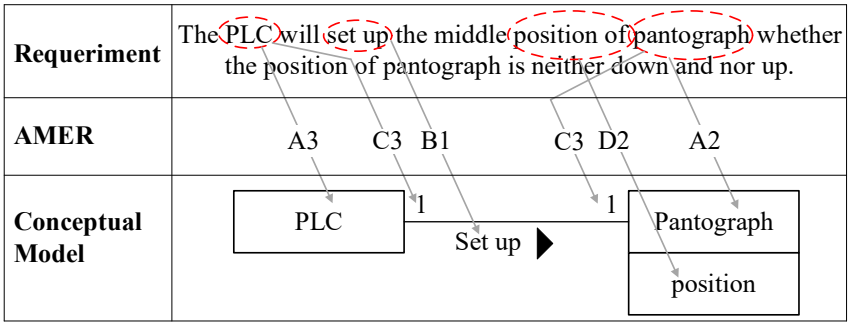

Fig. 3. Example of model generation by applying the AMER

The AMER are applied to generate the model associated to the requirement of Fig. 3 as follows:

- A3: The statement of A3 claims *"Subjects in the requirements are concepts",* then **PLC** is a concept.
- A2: The statement of A2 claims *"Recurring NPs are concepts",* then **Pantograph** is a concept.
- B1: The statement of B1 claims "*Transitive verbs are associations",* then **set up** shows an association between **PLC** and **Pantograph**.
- C3: The statement of C3 claims *"If the source concept of an association is singular and the target concept is singular then the association is one-to-one",* then the association between **PLC** and **Pantograph** is one-to-one.
- D2: The statement of D2 *claims "Genitive cases suggest attributes",* then **position** is an attribute of **Pantograph**.

3.2 Models generated by domain experts

To generate models by domain experts, this step involved 19 domain experts from our industrial partner. They are experts in developing software and requirements. In their daily work, these experts develop software from requirements. They have spent a mean of 6.65 years working as software engineers. The domain experts stated that they spent a mean of 3.36 hours per day interpreting requirements.

We involved 19 domain experts rather than one because it would not be fair to consider only one domain expert as the oracle (the ground truth). According to the literature [12], [22], several different solutions (models) can be provided for the same problem (requirements specification). Hence, we compare the model generated by several domain experts with the model generated by the AMER. This comparison will allow us to evaluate whether the result of the AMER is close to some of the models generated by the domain experts. In addition, we perform the comparison in a real world industrial context, which is a step towards addressing the existing gap of obtaining results in an industrial context (the large majority of existing work on model extraction is evaluated over samples or artificial settings).

In this step, each domain expert had to interpret each of the requirements in NL provided as input. As a result of this interpretation, the subjects had to build a software model that captures all the ideas articulated in the requirements. To avoid a possible ceiling effect, there was no time limit in interpreting requirements. As a result of this step, 19 different software models were obtained. These models required an average of 62 minutes to be built and they have an average of 72.94 elements. The Fig. 4 shows an example of a requirement and its corresponding model generated by a domain expert. All requirements and the software models generated by domain experts are available at http://svit.usj.es/requerimentinfluenceexperiment.

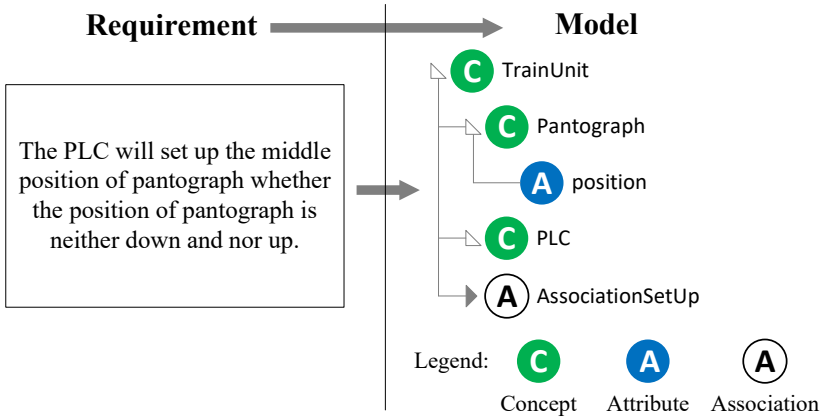

Fig. 4. Example of a model generated by a domain expert from a requirement

3.3 Natural Language Processing

Once the models generated by the AMER and the models generated by domain experts are obtained, we apply to them NLP techniques to homogenize the words used in the models with the aim of comparing them. The Fig. 5 shows the process to homogenize the words used in the models.

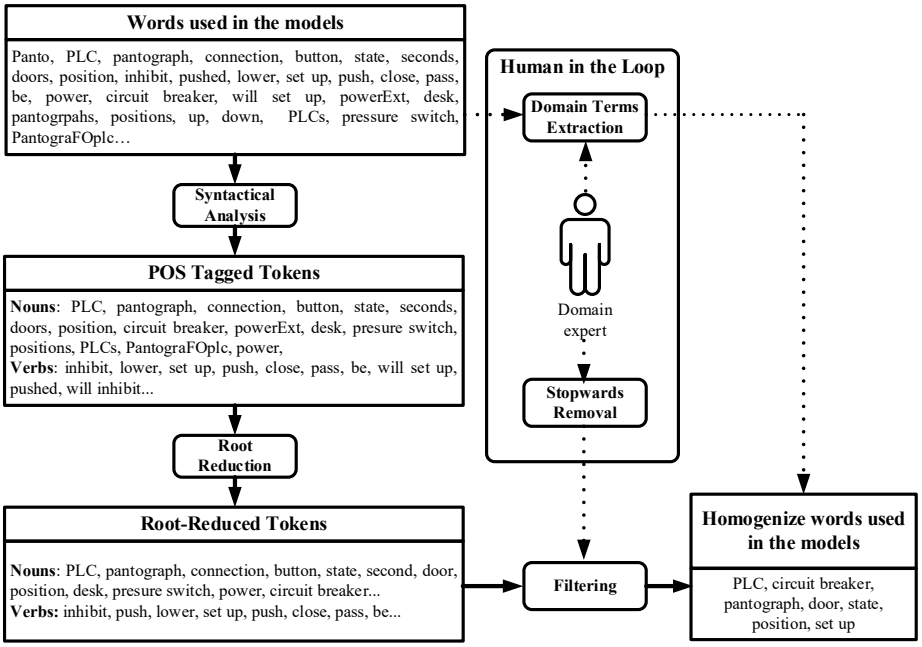

Fig. 5. NLP techniques to homogenize the words used in the models

The whole compendium of NLP techniques used in this work are syntactical analysis, root reduction, and human in the loop as follows:

1. **Syntactical Analysis**. Syntactical Analysis (SA) techniques split the words used in the models, analyzing the specific roles of each one of them and determining their grammatical load. In other words, these techniques determine the grammatical function of each word (e.g.: nouns, verbs, adjectives, adverbs, etc.). These techniques, often referred to as Parts-Of-Speech Tagging (POS Tagging) techniques allow engineers to implement filters for words that fulfill specific grammatical roles in a sentence, usually opting for nouns, since these words are the ones that carry the relevant information about descriptions of features and actions [6]. Words like verbs, adverbs, and adjectives are often filtered out and disregarded. For example, some of the POS Tagged Tokens obtained as outcome of syntactically analyzing a model are the nouns PLC, seconds, button and doors; and the verbs pushed and close.

2. **Root Reduction.** Through the usage of semantic techniques such as Lemmatizing, words can be reduced to their semantic roots or lemmas. Thanks to lemmas, the language of the models is unified, avoiding verb tenses, noun plurals, and strange word forms that interfere negatively with the comparison process. Prior to carrying out Root Reduction (RR) techniques, it is imperative to use SA techniques, due to the fact that RR techniques are based on word dictionaries that are built upon the grammatical role of words in a sentence. The unification of the language semantics is an evolution over pure syntactical role filtering that allows for a more advanced filtering of words in models. For example, some of the Root-Reduced tokens obtained as outcome of the previously POS Tagged tokens are the nouns PLC, second, button and door; and the verbs push and close.
3. **Human-In-The-Loop.** The inclusion of domain experts is a widely discussed topic within the SE community since it is often regarded as beneficial to have some sort of domain knowledge embedded. Some of the techniques derived from humans are Domain Terms Extraction, Stopwords Removal and Equivalence of Terms. In order to carry out these techniques, domain experts provide three separate lists of terms: one list of terms (both single-word terms and multiple-word terms) that belong to the domain and that must be always kept for analysis, a list of irrelevant words that can appear throughout the models and that have no value whatsoever for the analysis, and a list of words that are equivalent and can be unified in models. Both kinds of terms can be automatically filtered in or out of the final query, depending on the needs of the domain experts. For example, the domain experts provide the word door as a word that belong to the domain and must be always kept for analysis, the word second as irrelevant word, and the word system as a equivalent term of PLC that must be unified for analysis.

3.4 Comparison of models

The model generated by the AMER and the models generated by the domain experts are then compared in order to get a confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model on a test data (the model generated by the AMER) for which the true values are known (from each model generated by a domain expert). In our case, each solution outputted is a model composed of a subset of the model elements. Since the granularity will be at the level of model elements, each model element presence or absence for each category of the AMER (concepts, associations and generalizations, cardinalities, and attributes) will be considered as a classification. The confusion matrix distinguishes between the predicted values and the real values classifying them into four categories:

- True Positive: values that are predicted as true (in the model generated by the AMER) and are true in the real scenario (the model generated by a domain expert). That is, True Positive are the model elements included in both, the model generated by the AMER and the domain experts.
- False Positive: values that are predicted as true (in the model generated by the AMER) but are false in the real scenario (the model generated by a domain expert).

False Positive are the model elements included in the model generated by the AMER and not included in the model generated by domain experts.

- True Negative: values that are predicted as false (in the model generated by the AMER) and are false in the real scenario (the model generated by a domain expert). True Negative are the model elements included in neither the model generated by the AMER nor the domain experts.
- False Negative: values that are predicted as false (in the model generated by the AMER) but are true in the real scenario (the model generated by a domain expert). That is, False Negative are the model elements not included in the model generated by the AMER and included in the model generated by the domain experts.

From the values in the confusion matrix we create a report including three performance metrics (Recall, Precision, and F-measure) of both, from the model generated by the AMER and from each model generated by a domain expert for each category of the AMER (concepts, associations and generalizations, cardinalities, and attributes).

Recall is the number of model elements retrieved (True Positive) divided by the number of the model elements generated by the domain experts (True Positive + False Negative):

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (1)$$

Precision is the number of model elements retrieved (True Positive) divided by the elements of the models generated by the AMER (True Positive + False Positive):

$$Precision = \frac{True\ Postive}{True\ Positive + False\ Positive} \quad (2)$$

F-measure combines Precision and Recall to obtain the harmonic mean, the value of F-measure is defined as follows:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3)$$

A Recall value of 0% means that there are no model elements of a category obtained from the model generated by the domain expert that matched those of the model generated by the AMER. On the other hand, a Recall value of 100% means that all the model elements of a category from the model generated by the domain expert are present in the model generated by the AMER.

A Precision value of 0% means that there are not model elements of a category obtained from the model generated by the AMER that matched those of the model generated by the domain expert. On the other hand, a Precision value of 100% means that

all the model elements of a given extraction rule category from the model generated by the AMER are present in the model generated by the domain expert. Finally, a value of 100% Precision and 100% Recall for a category of the AMER implies that the same model has been generated by both the domain expert and the AMER.

4 Results

In this section, we present both the results with the occurrences of each extraction rule from the requirements, and the results of performance measurement in terms of precision and recall values for each domain expert and for each category of the AMER (Concepts, Associations and Generalizations, Cardinalities, and Attributes).

Fig. 6 shows a chart with the 18 different the AMER in the x axis and the occurrences of each extraction rule in the y axis that have been triggered to obtain the model generated by the AMER. As the graph shows, 11 rules from the four categories have been triggered in total. The rules with more occurrences since they have been triggered in all the requirements are: A1 (all NPs in the requirements are candidate concepts), A3 (subjects in the requirements are concepts), and B1 (transitive verbs are associations). This makes the category Concepts as the most applied in requirements even it is achieved by using only 60% of the rules. By contrast, the categories that have triggered the maximum number of different rules (75%) are Cardinalities and Attributes.

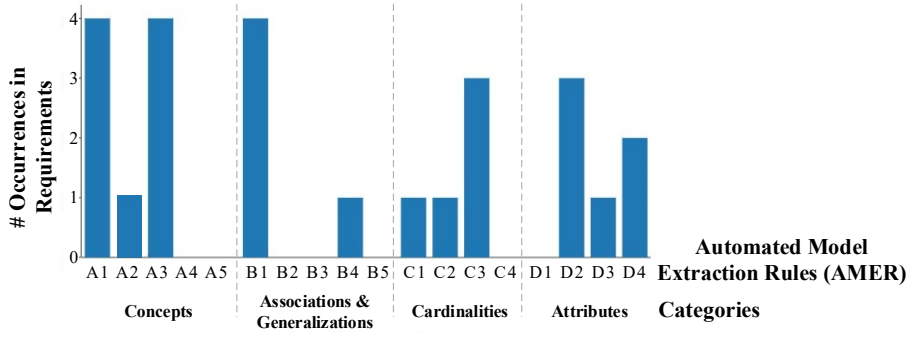

Fig. 6. Automated Model Extraction Rules (AMER) in requirements

Fig. 7 shows four charts with the results of the performance measurement in terms of recall and precision values after comparing the model obtained by the exaction rules with each model obtained by a domain expert. Each chart represents a category of the AMER (*Concepts, Associations and Generalizations, Cardinalities*, and *Attributes*), whereas each point in the charts represents the value of the two performance indicators (recall on the y axis and precision on the x axis) for each domain expert.

Fig. 7 a) related to *Concepts* shows that when the domain experts generated the models from requirements, most of Recall values are higher than Precision values (13 of 19). The ranges of values related to *Concepts* are [63.16%,100%] for Recall and [58.82,94.12%] for Precision. Similarly, Fig 7 b) related to *Associations and Generalizations* shows that the domain experts achieved higher Recall values than Precision values in their models. This fact appears in 12 models of 19. The ranges of values related to *Associations and Generalizations* are [47.62%,100%] for Recall and [57.14,92.86%] for Precision. The results related to *Cardinalities* are similar to previous ones, Fig. 7 b) shows that when the domain experts generated the models, most of Recall values are higher than Precision values (12 of 19). Related to *Cardinalities*, the range of values of Recall is [43.75%,100%] and the range of Precision is [50%,89.29%]. Finally, the Fig. 7 d) shows that the tendency is opposite to the previous categories. The number of models with higher values of Precision is 12. The ranges values related to *Cardinalities* are [31.25%,85.71%] for Recall and [57.14,92.86%] for Precision.

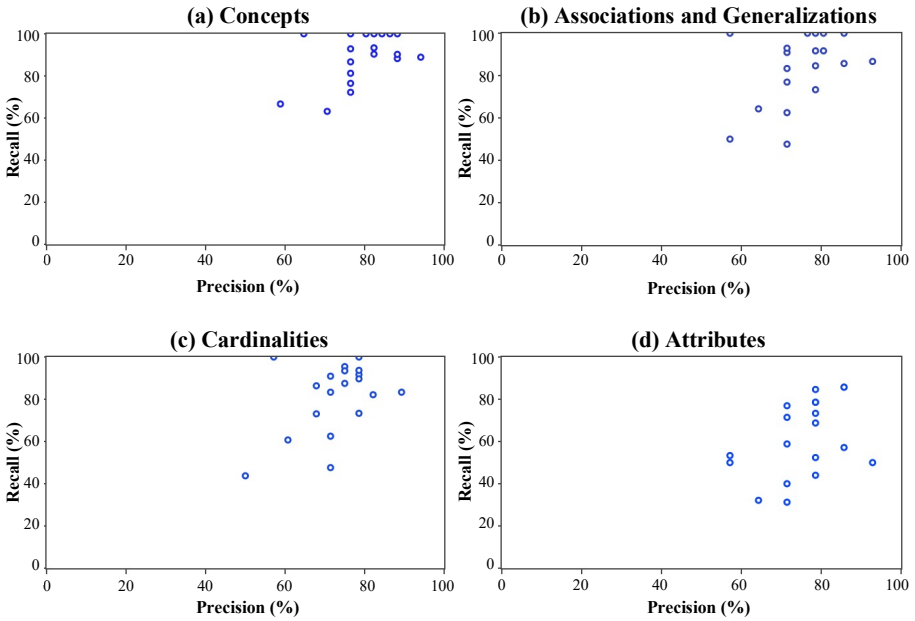

Fig. 7. Recall and Precision values for each category and domain expert

Table 1 shows the mean values of recall, precision and F-measure of the graphs for each category of the AMER. The category *Concepts* obtains the best results in recall and precision, providing an average value of 79.26% in precision and 89.02% in recall. In recall, the next best result is obtained by *Associations & Generalizations* (83.17%) followed by *Cardinalities* (81.08%) and *Attributes* (61.72%). In precision, the next best result is obtained by *Attributes* (75.56%) followed by *Associations & Generalizations* (74.81%) and *Cardinalities* (72.56%).

Table 1. Mean Values and Standard Deviations for Precision, Recall, and the F-Measure.

	Recall±(σ)	Precision±(σ)	F-measure±(σ)
Concepts	89.02±11.95	79.26±8.40	83.53±8.66
Associations & Generalizations	83.17±16.64	74.81±9.02	78.07±11.10
Cardinalities	81.08±16.73	72.56±9.15	75.90±11.19
Attributes	61.72±17.96	75.56±9.32	66.75±13.37

5 Discussion

The recall values of 100% (see the points in the 100% line of Fig. 7) indicate that the AMER has fully covered the model elements created by a domain expert. However, recall values lower than 100% indicate that the AMER has not covered all the model elements of the domain expert. By analyzing these results, we detected that in cases where the AMER does not reach 100% of recall it is because the domain experts create the models using as input both the requirements specification and their own tacit knowledge about the domain.

Domain experts leverage tacit knowledge to specify: concepts in 12/19 models, associations in 14/19 models, cardinalities in 17/19 models, and attributes in 19/19 models. Especially in the case of attributes, research in the AMER should consider as input to its rules other knowledge sources (such as domain ontologies or reference architectures) in order to achieve the performance of domain experts.

For the categories *Concepts*, *Associations & Generalizations*, and *Cardinalities* in the models generated by the domain experts the most values have higher values of recall than values of precision. On the contrary, the values in the category *Attributes* are higher for precision than recall. These data and the achieved values corroborate the need to use ontologies, reference architectures or similar techniques. This is especially relevant in the case of category *Attributes*.

For the same requirements specification, several different models may be considered solutions equally valid by different domain experts [12], [22]. For example, the same domain expert can use more model elements than another domain expert to specify the same requirement. Precision values below 100% may reveal that models created by the AMER have a different modeling style than domain experts. However, precision values lower than 100% may also reveal that the models created by the AMER specify aspects that domain experts considered non-relevant to be specified.

By analyzing the comparisons between the model created by the AMER and the models created by domain experts, we detected that precision values lower than 100% were produced by differences in modeling style. Neither concepts (79.26%), associations (74.81%), cardinalities (72.56%), nor attributes (75.56%) can exactly match the modeling style of any of the 19 domain experts.

Achieving that the modeling style of the AMER is aligned with the modeling style of a domain expert can be beneficial to facilitate the domain expert's understanding. However, in industrial environments, the same model is consumed by multiple actors (such as domain experts, engineers or testers) and each actor may have different mod-

eling preferences. Therefore, it is necessary to perform specific experiments in order to evaluate the understanding of the AMER modeling style for the different actors that consume the models in an industrial environment.

6 Threats to Validity

This section describes the threats that we have avoided, the threats that we could not avoid but that we mitigated, and the threats that we could not tackle. We use the classification of threats to validity of [17]; this classification distinguishes four aspects of validity:

Construct validity: The first identified threat of this type was the author bias, this threat means that the people that define the artifacts can subjectively influence the obtainment of the results that they are looking for. In order to mitigate this threat, the requirements specification was designed by a domain expert who was external to the design of the experiment and who was not involved in this paper. The second threat was the task design, this threat appears when the tasks can be correctly performed just by chance. To mitigate this threat the requirements specification did not have a true/false answer; the domain experts had to generate a model; this is very difficult for them to answer correctly if they do not understand the requirements. The third identified threat was the hypothesis guessing, this threat means that the subject may guess the hypotheses and work to fulfill them. To mitigate this, we did not talk with the domain experts about the evaluation goals.

Internal validity: The first identified threat of this type was the history, this threat appears when different treatments are applied to the same object at different times. We mitigated this threat by applying the AMER to the requirements specification without knowledge about the models generated by domain experts. The second identified threat was the subject motivation, this threat appears when the subjects are not motivated to participate in the experiment. The experiment was affected by this threat since the domain experts were recruited as part of their daily work (they were not volunteers).

External validity: The first identified threat of this type was the statistical power, this threat appears when the number of subjects is not enough to generalize results. Our experiment was affected by this threat, because the number of subjects (19) was not high enough to generalize results. However, it is important to note that the role of the subjects (domain experts in an industrial environment) makes an interesting contribution in an area where most experiments are conducted with students or artificial problems. The second identified threat was the object dependency, this threat appears when the results may depend on the objects used in the experiment and they cannot be generalized. We mitigated somewhat by using requirements specification were real requirements that were extracted from the company's catalog.

Reliability: The first identified threat of this type was the data collection, the data collection was not always done in the same way. This was mitigated by applying the same mechanized procedure. In addition, we tested the data coherence when the domain experts finished each generated model. Finally, the last identified threat was

the reliability of measures, this threat appears when there is no guarantee that the outcomes will be the same if a phenomenon is measured twice. To mitigate this threat, we used measurements accepted by the research community such as precision and recall.

7 Related Work

Several works have dealt with processing requirements specifications for model building. These works aim to extract conceptual models from texts with NL requirements. One example of these works was developed by Robeer et al. [15], who propose to automatically derive conceptual models from user stories that are written in NL. Bhala and Abirami [18] also proposed an automatic transformation from functional specifications in NL to conceptual models. The proposal is based on the analysis of grammatical constructs. The result of the transformation is the construction of an entity-relationship diagram with notations. Ferrari et al. [10] conducted an evaluation of a tool (named CAR) that supports a textual definition of requirements. The evaluation was done using metric completeness, where the experiments compare the completeness of requirements using CAR versus using no tool. The authors of that paper are also the subjects of the study. These empirical studies had not been conducted in an industrial context involving real domain experts as our work does.

In [5] Arora et al. present an automated approach based on NLP for extracting domain models from unrestricted requirements. This approach is developed by bringing together existing extraction rules in the software engineering literature, extending these rules with complementary rules from the information retrieval literature, and proposing new rules to better exploit results obtained from modern NLP dependency parsers. In [3] Ambriola et al. present the tool CIRCE, an environment for the analysis of NL requirements. The tool is based on a transformational paradigm. The result of all the transformations is a set of models for the requirements document, for the system described by the requirements, and for the requirements writing process. Furthermore, Yue et al. [21] propose a method and a tool called aToucan to automatically generate a UML analysis model comprising class, sequence and activity diagrams from a use case requirements and to automatically establish traceability links between model elements of the use case requirements and the generated analysis model. Even though these works provide empirical data on building models from textual requirements, they do not address the performance of the AMER as our work does.

Ben Abdessalem Karaa et al. [1] explain their vision of an approach for class diagram generation from user requirements expressed in NL. Their approach amalgamates the statistical and pattern recognition properties of NLP techniques. To validate their approach the authors implemented a tool named ABCD. Elbendak et al. [8] present a tool, Class-Gen, which can partially automate the identification of objects/classes from NL requirements specifications for object identification. Ibrahim et al. [11] propose a method and a tool to facilitate requirements analysis process and class diagram extraction from textual requirements supporting NLP techniques. They propose a tool (named RACE) that assists analysts by providing a way to produce the

class diagram from their requirements. Thakur et al. [19] propose a systematic, automated approach to identify the domain elements from textual specifications. The approach uses a language model to interpret the sentences, and identifies the domain elements using the semantic relationships between the words in the sentences obtained from Type Dependencies. These works do not evaluate their approach in a real context with real requirements as our work does.

8 Concluding Remarks

The transition from a requirements specification expressed in NL to a domain model is an important step that can be performed using the AMER. However, crucial aspects remain under-explored in real settings. To address this gap, we have designed a process to assess the AMER performance in terms of recall, precision and F-measure by comparing the model generated by the AMER with the model generated by different domain experts of our industrial partner.

In contrast to current research efforts in the AMER (which develop more and more rules to create models by means of processing the NL of requirements), our results suggest new research directions to push forward the AMER:

- Especially in the case of attribute extraction, it is necessary to consider new knowledge sources (such as domain ontologies or reference architectures) that can play the role of tacit knowledge about the domain, which is not explicit in the requirements.
- It is necessary to perform specific experiments to evaluate the understanding of the AMER modeling style for the different actors (such as domain experts, engineers or testers), who consume the models in an industrial environment.

Acknowledgements

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the project Model-Driven Variability Extraction for Software Product Line Adoption (TIN2015-64397-R).

References

1. Ben Abdessalem Karaa, W., Ben Azzouz, Z., Singh, A., Dey, N., S. Ashour, A., Ben Ghazala, H.: Automatic Builder of Class Diagram ABCD: An Application of UML Generation from Functional Requirements. Softw. Pr. Exper. 46 (11), 1443–1458 (2016)
2. Akbik, A., Broß, J.: Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns. In: WWW Workshop.

(2009)

3. Ambriola, V., Gervasi, V.: On the Systematic Analysis of Natural Language Requirements with CIRCE. Autom. Softw. Eng. 13 (1), 107–167 (2006)

4. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Automated checking of conformance to requirements templates using natural language processing. IEEE Trans. Softw. Eng. 41 (10), 944–968 (2015)

5. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Extracting Domain Models from Natural-language Requirements: Approach and Industrial Evaluation. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. pp. 250–260. ACM, New York, NY, USA (2016)

6. Capobianco, G., Lucia, A. De, Oliveto, R., Panichella, A., Panichella, S.: On the role of the Nouns in IR-based Traceability Recovery. In: Proc. of the Int'l Conf. on Program Comprehension. pp. 148–157. IEEE (2009)

7. Deeptimahanti, D.K., Sanyal, R.: Semi-automatic Generation of UML Models from Natural Language Requirements. In: Proceedings of the 4th India Software Engineering Conference. pp. 165–174. ACM, New York, NY, USA (2011)

8. Elbendak, M., Vickers, P., Rossiter, B.N.: Parsed use case descriptions as a basis for object-oriented class model generation. J. Syst. Softw. 84 1209–1223 (2011)

9. Fanmuy, G., Fraga, A., Lloréns, J.: Requirements Verification in the Industry. In: Proceedings of the Second International Conference on Complex Systems Design & Management, CSDM 2011, Paris, 7-9 December 2011. pp. 145–160. (2011)

10. Ferrari, A., Dell'Orletta, F., Spagnolo, G.O., Gnesi, S.: Measuring and improving the completeness of natural language requirements. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 8396 LNCS 23–38 (2014)

11. Ibrahim, M., Ahmad, R.: Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques. In: 2010 Second International Conference on Computer Research and Development. pp. 200–204. (2010)

12. Lucas, F.J., Molina, F., Toval, A.: A Systematic Review of UML Model Consistency Management. Inf. Softw. Technol. 51 (12), 1631–1645 (2009)

13. Pohl, K., Rupp, C.: Requirements Engineering Fundamentals, 1 sl ed. Rocky Nook. (2011)

14. Popescu, D., Rugaber, S., Medvidovic, N., Berry, D.M.: Reducing ambiguities in requirements specifications via automatically created object-

oriented models. In: Monterey Workshop. pp. 103–124. (2007)

15. Robeer, M., Lucassen, G., v. d. Werf, J.M.E.M., Dalpiaz, F., Brinkkemper, S.: Automated Extraction of Conceptual Models from User Stories via NLP. In: 2016 IEEE 24th International Requirements Engineering Conference (RE). pp. 196–205. (2016)

16. Rosadini, B., Ferrari, A., Gori, G., Fantechi, A., Gnesi, S., Trotta, I., Bacherini, S.: Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain. In: Requirements Engineering: Foundation for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings. pp. 344–360. (2017)

17. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164 (2009)

18. Sagar, V.B.R.V., Abirami, S.: Conceptual modeling of natural language functional requirements. J. Syst. Softw. 88 25–41 (2014)

19. Thakur, J.S., Gupta, A.: Identifying Domain Elements from Textual Specifications. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. pp. 566–577. ACM, New York, NY, USA (2016)

20. Yue, T., Briand, L.C., Labiche, Y.: A systematic review of transformation approaches between user requirements and analysis models. Requir. Eng. 16 (2), 75–99 (2011)

21. Yue, T., Briand, L.C., Labiche, Y.: aToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models. ACM Trans. Softw. Eng. Methodol. 24 (3), 13:1--13:52 (2015)

22. Zave, P.: Classification of Research Efforts in Requirements Engineering. ACM Comput. Surv. 29 (4), 315–321 (1997)