Data & Knowledge Engineering 128 (2020) 101822

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams

Paulius Danenas[a], Tomas Skersys[a,b,*], Rimantas Butleris[a,b]

[a] Center of Information Systems Design Technologies, Kaunas University of Technology, K. Barsausko str. 59, Kaunas, Lithuania
[b] Department of Information Systems, Kaunas University of Technology, Studentu str. 50, Kaunas, Lithuania

ARTICLE INFO

Keywords:
SBVR business vocabulary and rules
UML use case diagram
Model-to-model transformation
Controlled natural language
Natural language processing
Information extraction

ABSTRACT

Discovery, specification and proper representation of various aspects of business knowledge plays crucial part in model-driven information systems engineering, especially when it comes to the early stages of systems development. Being among the most applicable and advanced features of model-driven development, model transformation could help improving one of the most time- and resource-consuming efforts in this process, namely, discovery and specification of business vocabularies and business rules within the problem domain. One of our latest developments in this area was the solution for the automatic extraction of SBVR business vocabularies and business rules from UML use case diagrams, which was arguably one of the most comprehensive developments of this kind currently available in public. In this paper, we present an enhancement to our previous development by introducing a novel natural language processing component to it. This enhancement provides more advanced extraction capabilities (such as recognition of entities, entire noun and verb phrases, multinary associations) and better quality of the extraction results compared to our previous solution. The main contributions presented in this paper are pre- and post-processing algorithms, and two extraction algorithms using custom-trained POS tagger. Based on the related work findings, it is safe to state that the presented solution is novel and original in its approach of combining together M2M transformation of UML and SBVR models with natural language processing techniques in the field of model-driven information systems engineering.

1. Introduction

In information systems development, textual specifications are used as either an alternative or a supplementary means to knowledge specification and representation. Of course, when it comes to *model-driven* information systems development, graphical models are the ones playing the most important role. Nonetheless, even in model driven approaches, textual specification of business domain knowledge, system design and other aspects of the project remains a crucial part of the overall documentation. This is due to the fact that any business-oriented development project will always involve *business people* who will tend to communicate their business knowledge using natural language (NL) and will be quite cautious about any kind of formal models; even if graphical diagrams are read by business people, there will always be a risk of them misinterpreting things.

Based on our extensive research work carried out in VEPSEM[1] and VeTIS[2] projects, we argue that one of the ways of dealing with such situation is to use both formal graphical models and their mirrored representation in textual format based on so-called

* Corresponding author at: Department of Information Systems, Kaunas University of Technology, Studentu str. 50, Kaunas, Lithuania.
E-mail addresses: paulius.danenas@ktu.lt (P. Danenas), tomas.skersys@ktu.lt (T. Skersys), rimantas.butleris@ktu.lt (R. Butleris).

[1] "Integration of Business Processes and Business Rules on the Basis of Business Semantics (VEPSEM)", 2013–2015. The project was funded by the European Social Fund (ESF).

[2] "Business Rule Solutions for Information Systems Development (VeTIS)", 2007–2009. The project was funded by the High Technology Development Program, Lithuanian State Science and Studies Foundation.

https://doi.org/10.1016/j.datak.2020.101822
Received 16 June 2019; Received in revised form 10 February 2020; Accepted 2 May 2020
Available online 6 May 2020

controlled NL. Our current focus is on the integration of formal graphical and textual models of business knowledge and user requirements residing within the early stages of information systems development. In there, we use UML use case diagrams as means of graphical representation of a functional requirements model, and for the textual representation of business knowledge we adopted SBVR standard [1]. Now, while UML requires no further introduction, the acronym of SBVR is known to a much smaller audience and therefore calls for more elaborate overview. SBVR refers to the *Semantics of Business Vocabulary and Business Rules*, which is a formally sound standard for the specification of virtually any kind of knowledge using controlled NL. An SBVR model is composed of one or more *business vocabularies* and may contain sets of *business rules*, which are based on the concepts defined in the aforementioned business vocabularies. Textual representation of SBVR models has the "look and feel" of a natural language and at the same time these are the specifications based on *Meta Object Facility* (MOF) and therefore interpretable by computers. Moreover, standardization of business knowledge in a form of NL-based and manageable concepts enables reuse of that knowledge throughout the whole system development life cycle and beyond, e.g. in other projects within the same business domain.

Nevertheless, our personal experience from participating in actual systems development projects and also the conducted research show that projects quite often lack well-structured vocabularies of business concepts, not to mention business rules, as traceable and manageable artefacts, even though the importance and usefulness of these artefacts is perceived quite well [2]. One of the main reasons explaining the situation is obvious – without proper tooling, the development of such vocabularies from scratch will always require considerable amount of additional effort, which in some cases may be a decisive negative factor against such approach. In our most recent development [3], we used model-to-model (M2M) transformation technology to extract well-structured SBVR business vocabularies and business rules from formal UML use case models represented through a set of UML use case diagrams. The proposed solution consists of two concurrent approaches, namely, automatic and semi-automatic, which may be used selectively to achieve the best expected result. Through experimenting [3], we proved that our transformation tool could fully reproduce benchmark results in wizard-based *semi-automatic* mode; however, fully *automated* transformation approach was less accurate due to its limitations while dealing with so-called bad modeling practices and other natural language related issues. While some of these issues are also addressed within the scope of interpreting conceptual model labels (Leopold, 2013), they tend to focus on processing at a single label level and do not consider composite artefacts, such as business rules. Moreover, recent achievements in natural language processing enable more advanced textual analysis, parsing, extraction and normalization further improving the results obtained during the initial automated procedures. These include, but are not limited to, part-of-speech tagging, entity extraction, relation extraction and named entity recognition which have shown to be influential in the context of our research. The most recent achievements in these fields present promising results [4–7].

Research objectives. We argue that recent developments in natural language processing (NLP), and more specifically – entity extraction, can be used to deliver certain improvement to the automatic model transformation. Indeed, natural language processing has been researched quite actively in the area of requirements engineering, and beyond. Nevertheless, at the time of writing, we could not find a single paper describing any tangible results of performing automated extraction of natural language concepts from formal graphical models. In this paper, we present the main results of the developed NLP enhancement to our previous M2M transformation solution [3]. In addition, we show that the achieved result is generalizable to a broader context of model transformations, not limiting itself to only the extraction of SBVR business vocabularies and rules from UML use case models.

Hence, this paper provides the following contributions to the domain of model-driven information systems development:

- Discusses premises and issues of entity extraction from graphical models;
- Provides an NLP-based approach to enhance the already existent model transformation solution for the extraction of SBVR business vocabularies and business rules from UML use case diagrams;
- Identifies certain premises for the further research in this topic.

In the next sections of this paper, the main results of our research are presented: Section 2 describes main concepts and definitions within the relevant topics; Section 3 discusses certain modeling and entity naming practices, as well as the main challenges and specificities of grammatical processing; Section 4 presents the developed algorithms for text processing with Section 5 providing their experimental evaluation; main threats to construct, internal and external validity are defined in Section 6; Section 7 is an overview of the related work; finally, Section 8 concludes the paper summarizing the results and outlining future research.

2. Basic definitions

This section provides definitions of certain core concepts and some other background information relevant to the research. Section 2.1 provides definitions of a set of core SBVR concepts related to the research, while Section 2.2 is dedicated to the brief coverage of entity extraction topic.

2.1. SBVR business vocabulary and business rules

Both business and technical people benefit from using well-structured business vocabularies not only during system development project, but in other real-world business cases as well. Business vocabularies and business rules are used to explicitly define the core concepts used for expressing, communicating and exchanging virtually any kind of knowledge. *Semantics of Business Vocabulary and Business Rules* [1] is an OMG's attempt to introduce a formalized way of dealing with the meaning, representation and structure of concepts, thus enabling automated parsing and interpretation of these concepts throughout various practical applications.

An SBVR model is organized in two parts: business vocabulary (BV) and business rules (BR). A *business vocabulary* consists of a set of glossary-like entries representing noun concepts and verb concepts (see their definitions further on) that are relevant to a specific business domain. Each entry is for a single concept. The entry starts with a primary presentation denoting the name of a concept; optionally, a concept can be defined by additional fields (properties), e.g. *Definition*, *Concept Type*, *Synonym, See*. In its turn, *business rules* are organized in rulesets and go alongside business vocabularies.

In our work, we use a subset of business vocabulary concepts:

- *General concept* is a noun concept, which classifies things based on their common properties.
- *Individual concept* is a noun concept corresponding to an instance or a single object.
- *Verb concept* is some type of relationship between two or more noun concepts or a characteristic of the noun concept. Following the definition, verb concepts are specified using noun concepts that were previously defined in BV. SBVR defines several types of verb concepts; however, that taxonomy is not relevant to this research and therefore will not be elaborated further on.

Further, SBVR defines a *business rule* as a rule that is under business jurisdiction [1]. However, we consider this definition being rather implicit and requiring more in-depth explanation. According to von Halle [8], a *business rule* is a logical statement that captures the nature of an enterprise's business model, defines or constrains some business aspect of particular business situation or context and ensures that one or more business goals are achieved. SBVR specification supports both structural and operational business rules, which represent structural and behavioral assertions and constraints, respectively. It must be noted that the development of SBVR model is based on the so-called business rules "mantra" [9]. It states that business rules are built on facts (or verb concepts, according to SBVR terminology) and facts are built on terms (noun concepts in SBVR terminology). Verb concepts and general concepts form the basis of any business vocabulary; successively, one must have a business vocabulary to properly specify and manage business rules.

Even though SBVR is not restricted to only textual representation of SBVR concepts, the main representation form is a so-called controlled natural language, which is also adopted in our approach. SBVR standard has explicitly defined formatting notation for this representation form [1]:

- 'term' formatting is used to depict general concepts, e.g. 'marketing specialist'. Best practices require that general concepts are defined in singular form using lower case letters.
- 'Name' formatting represents individual concepts that usually are proper nouns and represents names, towns, countries, companies, etc. (e.g., 'EUR', 'Lithuania'). The first letter of an individual concept is capitalized.
- '*verb*' formatting designates a verb, a preposition, or a combination of these two. Verbs are used in singular active or passive forms as synonymous forms (e.g., 'customer *makes* car booking' and its corresponding synonymous form 'car booking *is made by* customer');
- 'keyword' font represents linguistic symbols that are used to construct statements and definitions, such as quantifiers, 'if', 'It is obligatory that', 'greater than', 'that', etc.

The presented formatting is used for the representation of both SBVR business vocabulary concepts as well as business rules.

2.2. Entity extraction

Entity extraction (sometimes also referred as "*entity chunking*") is one of the core research subjects of NLP standing next to relation extraction, ontology generation and enrichment, and other relevant subjects within the domain of NLP. The main task of entity extraction is recognition of presence of a word or phrase as named entity (NE) in a given natural language text, e.g., sentence, document. Depending of the level of complexity, this task may interface with other NLP-related problems, which in their turn can be combined to form pipelines to solve the specified task:

- *General pre-processing* – a subtask dealing with automated text cleansing, removal of invalid symbols, resolution of abbreviations, lemmatization, etc.
- *Error correction* is performed to identify and correct typographical, grammatical and semantic errors, e.g., accidental use of homophones. Multiple approaches exist for solving this problem, including multiple string distance metrics [10,11], noisy-channel models [12,13], feature-based classification [14], statistical machine translation [15] or neural machine translation [16] techniques. However, the latter techniques are tested on or address error correction at a sentence level, while this paper focuses on short phrases; therefore, suitability of such techniques should be verified. In our research, grammatical error processing is not applied.
- *Part of speech (POS) tagging* is an automated identification of the part of speech tags for each identified word. The existing POS tagging solutions tend to rely on statistical and machine learning approaches, such as maximum entropy classifiers [17], conditional random fields [18], or deep learning [19,20].
- *Named entity recognition (NER)* is relatively similar to POS tagging, but focuses more on the specific categories, such as organizations, locations, expressions of date/time. In the context of the specification of SBVR vocabularies, this subtask can be used for the identification of categories. Modern NLP tools, such as Stanford CoreNLP [21], OpenNLP [22], or Spacy [23], provide pre-trained POS taggers and customization tools.

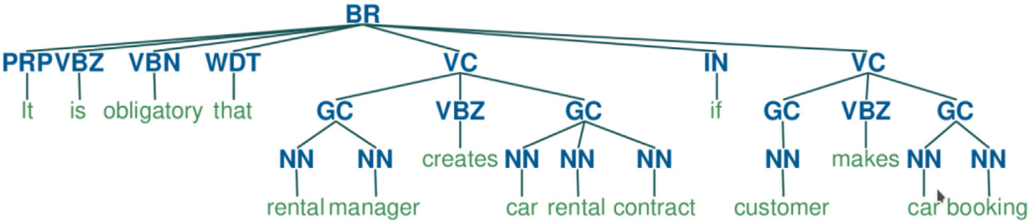

Fig. 1. Sample parse tree for the text rumbling representing a business rule.

- *Semantic analysis* is focused on the identification of synonyms, homonyms, antonyms, hyponyms, hypernyms, and other semantic relations.
- *Dependency resolution* finds dependencies between different entities within the given text.
- *Role extraction* subtask classifies extracted NER to determine the roles of those entities.
- *Post-processing* provides additional procedures, which can be applied to extracted entities: normalizing replacements to singular forms (for nouns or noun phrases), unifying verb tenses, etc.

The approach introduced in this research currently applies preprocessing, tokenization and POS tagging; yet, it can be extended, if more advanced features are required in the future.

At the first look, general entity extraction shares similar objectives as the extraction of business vocabularies, i.e., the identification of noun phrases and verb phrases. However, there are certain differences between those two:

- Model-specific semantics and metadata must be considered during the extraction of business vocabularies from graphical models, e.g., formal definitions of relationships between specific elements.
- The graphical language describing models is formalized, and the contextual information is very concise. As a result, some textual expressions in graphical models may be grammatically incorrect. Due to this, additional text processing approaches (e.g., grammatical error correction) must be applied to ensure correct performance of POS and NER tools.

Fig. 1 presents a sample parse tree of some business rule. This parse tree represents syntactic structure along with the tagged entities including the identified general concepts and verb concepts, which are included in the SBVR business vocabulary. The meaning of the labels presented in Fig. 1 are as follows (in alphanumeric order): BR — business rule, GC — general concept, IN — preposition, NN — singular noun, PRP — pronoun, VBN — past participle, VBZ — 3rd person singular present tense verb, VC — verb concept, WDT — WH-determiner.

3. Modeling practices and concept extraction challenges

In modeling, application of best modeling practices is of the means to improve quality, readability and common understanding of the models. In automated model processing, these practices (especially, for naming model concepts) play critical role. Section 3.1 describes a set of general modeling and naming conventions, which should be taken into account when dealing specifically with use case models. In its turn, Section 3.2 presents potential issues with the existing entity extraction applications, which may also could affect the results of automated extraction, if not taken into consideration.

3.1. Recommended practices for use case modeling

In our work, we share and support a common understanding that a set of common standards and guidelines should be used in any project to ensure effective communication among all interested parties. Further, we will present a set of practices for UCM modeling, which are relevant to the research presented in this paper:

- Name of an actor should be a noun or a noun phrase. In case there is more than one noun used in the name, adjectives and articles may be used in between those nouns (e.g., "manager of a rental"). However, we recommend using these parts of speech in the names of UCM concepts as little as possible and prefer so called "compact" representation of concepts (e.g., the latter example could be refactored to "rental manager"). This recommendation holds true for all the noun phrase usages within UCM naming scheme;
- Name of a subject (system boundary) should be a noun or a noun phrase;
- Name of a use case starts with a verb and is always followed by one or more nouns (verb-object label pattern). Furthermore, the name should be kept as simple as possible (e.g., without cascading noun phrases, or logical operators). In a broader context, this naming convention is also embraced for activity naming and explicitly imprinted as one of the seven modeling guidelines in process modeling [24];
- Condition of an extension point should be defined as a Boolean condition or a set of Boolean conditions connected using Boolean operators 'AND', 'OR'. Ideally, the formulated expression should be compliant with SBVR;

- Association is the only allowed type of a UML relationship used between an actor and a use case. Also, associations do not convey any conditions in UCM;
- There should be no redundant actors, use cases, and system boundaries with no relationships to any other concept within the UCM;
- One should avoid using duplicate concepts representing different forms of the same entity (e.g., '*Client*' and '*Clients*').

3.2. Potential issues with entity extraction

This section presents a non-finite set of potential issues with entity extraction applications, which may negatively affect the results of model transformation, if not dealt with properly. The illustrative examples in this section (and further on) are represented using formalism defined in our previous paper [3].

Part-of-speech tagger performance:

- Grammatical forms, which are not aligned with the defined naming conventions of the concepts of a use case model. Let us consider triplets of the use case model elements *<Actor::'manager', UseCase::'sign contract', Association('manager', 'sign contract')>* and *<Actor::'manager', UseCase::'contract creation', Association('manager', 'contract creation')>*. Then, the text rumblings[3] extracted from these triplets would be "*manager sign contract*" and "*manager contract creation*". In the first case, POS tagger might fail to tag words appropriately by tagging all three of them as nouns, while the second case might result in a combined noun-phrase, which would also be an incorrect result from the point of view of our M2M transformation. For the variety of possible use case naming cases, we refer to Pittke [25] which provides a set of activity labeling styles together with POS-based syntactical structures. While those styles were originally developed for process model elements, they are also easily applicable for use case elements as well.
- False identification of nouns and verbs having homonymous forms. This issue is highly relevant to our research; moreover, it is a quite frequent one as well. The coincidence of the infinitive form, recommended by the naming convention, and the same representation of a noun may lead to incorrect tagging. In such cases POS taggers tend to misidentify verbs, e.g., in the text rumbling "*administrator log system performance*", the "*log*" will be identified as a noun even though it was clearly assumed as a verb by the modeler.

Modeling quality:

- Ambiguous semantical meaning of certain model elements, which cannot be automatically identified, and, therefore, processed correctly. A good example of such ambiguous cases is a use case model element *Actor* – it can be named either starting with a capital letter (e.g., '*Manager*') or with a low case letter (e.g., '*manager*'). Both cases are allowed and used equally frequently in use case modeling. However, SBVR makes it different – nouns (or noun forms) starting with a capital letter are identified as *individual concepts* (or *unitary concepts*), and *general concepts* if otherwise.
- Sometimes assigning ambiguous names to model elements makes it virtually impossible to identify their true meaning not only for an automated system, bus for the human as well. For example, let us consider an extracted text rumbling "*analyst writing mapping*" – it can be interpreted both as "*analyst writes mapping*" and "*analyst maps writing*" (or even "*someone maps writing of the analyst*"). In this case, the result would be a natural false positive, which can be correctly interpreted and validated only by a human analyst after examining the context.

Semantic ambiguity:

- Non-trivial identification of synonymous forms in the text. While recognition of synonymous forms can significantly improve any entity extraction task, it should be noted that a safe replacement of any noun with its synonymous form may not be that safe at all. In our practice, we found many situations, where synonymous forms were actually homonyms depending on the context, e.g., in certain situations a '*client*', may be synonymous to both '*customer*' and '*desktop application*'; however, those two usually represent completely different kinds of a '*client*'.

Text normalization:

- So called "compact" representation of the names of use cases while using specific symbols and abbreviations. One of the most common cases is representing multiple elements separated by specific separators (e.g., use cases '*register/unregister client*', '*open *.gif/*.jpeg*'). While in many cases it will pre-processed correctly, there will be exemptions that might be taken into consideration (e.g., '*measure by miles/hour*').
- Various concept normalization issues, especially when converting singular and plural forms, e.g., should a "*start up*" be replaced with "*start ups*" or "*starts up*".

Classifier development:

[3] A term "*text* rumbling" represents an unstructured piece of textual information in a problem domain and is derived from a term "*business* rumbling", which was first introduced by B. von Halle in her paper "Back to Business Rule Basics" (in Database Programming & Design, 1994) and had more specific scope than a "*text* rumbling" (i.e. *business* domain).

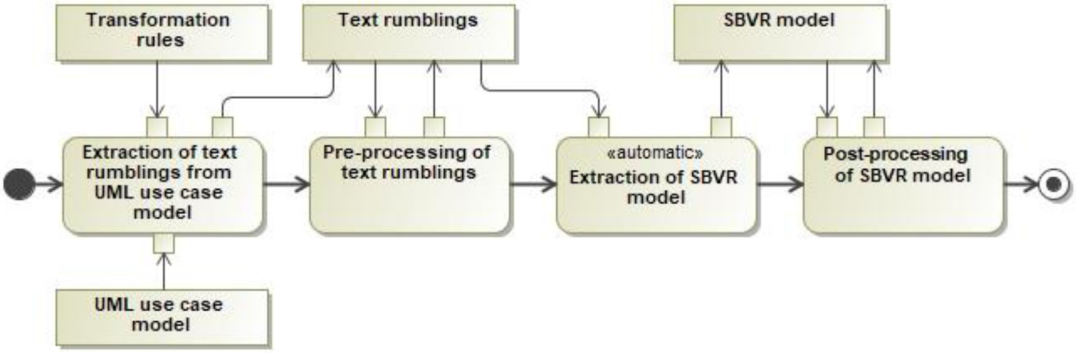
Transformation rules
Text rumblings
SBVR model
Extraction of text rumblings from UML use case model
Pre-processing of text rumblings
«automatic» Extraction of SBVR model
Post-processing of SBVR model
UML use case model

Fig. 2. The process of NLP-enhanced extraction of SBVR business vocabulary and business rules from UML use case diagrams.

– Issues related to training classifiers to tag instances of specific concepts, such as n-ary associations, which can be on virtually unlimited length. The lack of corpora with both grammatically and syntactically valid and invalid verb phrases is also considered to be an issue, because current corpora are optimized for training classifiers, which do not differentiate between homonymous verb and noun forms.

4. NLP-enhanced extraction of SBVR business vocabulary and business rules from UML use case diagrams

This section introduces the principles of NLP-enhanced extraction of SBVR business vocabulary concepts and business rules from UML use case diagrams. The proposed algorithms improve the automatic model transformation approach presented in our previous paper [3].

4.1. Roadmap process of NLP-enhanced extraction

The overall process of NLP-enhanced extraction of SBVR specifications from UML use case diagrams is composed of four consequent stages (Fig. 2):

- The process starts with the extraction of text rumblings from UML use case model represented via one or more UML use case diagrams. Text rumblings are extracted using a set of extraction rules fully specified in [3], therefore, we will not elaborate on this activity;
- The second stage is for the pre-processing of the extracted text rumblings. The pre-processing algorithm together with other relevant details is described in Section 4.2;
- Next, the extraction of SBVR model from the pre-processed set of text rumblings takes place. This stage is presented in Section 4.3;
- The last step is for the post-processing of the generated SBVR model. The post-processing algorithm is also presented in Section 4.2.

4.2. Pre-processing and post-processing algorithms

Before discussing the developed algorithms, let us introduce certain notation elements used in those algorithms:

- First, let us denote the extracted SBVR business vocabulary as *SBV* and extracted SBVR business rules as *SBR*. Then, SBV_{GC},SBV_{IC} and SBV_{VC} are the subsets of *SBV* representing general concepts, individual concepts and verb concepts respectively;

TR is a set of text rumblings, or *candidates*, acquired from the use case model using a set of predefined model transformation rules during the pre-processing stage:

TR = {(r, conceptType(r)) | conceptType(r) in {GC, IC, VC, BR}},

where *r* is a result (text rumbling) of the particular model transformation rule, and *conceptType(r)* is an SBVR type of that result: *GC, IC, VC,* or *BR*. For example, the given extraction result (''*manager create contract*'', VC) specifies that the text rumbling ''*manager create contract*'' is a candidate for an SBVR verb concept. Such formalization is required for proper identification of expected SBVR concept types and the reduction of various ambiguities. For example, let us get back to our text rumbling ''*manager sign contract*''. In general, this text expression could be identified as either a single general concept consisting of three nouns or a verb concept with the ''*sign*'' representing a verb; unfortunately, in cases like this, POS taggers tend to produce false positives by tagging such verbs as nouns, confused by the homonymous infinitive form of those verbs. In addition, we consider that traditional text preprocessing, such as removing punctuation signs, cleansing operations, etc., is already performed on the acquired *TR*.

- We define $TR_{GC} \subset TR$ as the subset of text rumblings, which are candidates for valid general and individual concepts according the corresponding transformation rules; similarly, $TR_{VC} \subset TR$ and $TR_{BR} \subset TR$ are the subsets of text rumblings, which contain candidates for valid verb concepts and business rules, respectively.
- Lastly, *TR'* is a set of text rumblings, which were obtained after applying specifically the proposed algorithm, with $TR'_{VC} \subset TR'$ and $TR'_{BR} \subset TR'$ representing text rumbling sets updated after the pre-processing step.

Algorithm 1. preprocessRumblings ($\boldsymbol{TR_{GC}}$,$\boldsymbol{TR_{VC,}}$$\boldsymbol{TR_{BR}}$)

```
for $\forall r \in TR_{GC}$:
  applyPosTagging(r)
  $S_r^{NER} \leftarrow extractNamedEntities(r)$
  if $S_r^{NER} \neq \emptyset$ then
      $SBV_{IC} \leftarrow SBV_{IC} \cup S_r^{NER}$
  else
      $SBV_{GC} \leftarrow SBV_{GC} \cup r$
for $\forall r \in TR_{GC}$:
  $replaced \leftarrow \emptyset$
  $S_r^{SYN} \leftarrow identifySynonyms(r)$
  for $\forall s \in S_r^{SYN}$
    if s does not have homonymous forms then
      for $\forall gc \in SBV_{GC} \cap S_r^{SYN}$
          $gc \leftarrow addSynonym(s)$
          $SBV_{GC} \leftarrow SBV_{GC} - s$
          $replaced \leftarrow replaced \cup (s, gc)$
$TR'_{VC} \leftarrow \emptyset,\ TR'_{BR} \leftarrow \emptyset$
for $\forall r \in TR_{VC}$
  $TR'_{VC} \leftarrow TR'_{VC} \cup replaceAll(r, replaced)$
for $\forall r \in TR_{BR}$
  $TR'_{BR} \leftarrow TR'_{BR} \cup replaceAll(r, replaced)$
for $\forall r \in TR_{GC}$
  $g \leftarrow getGeneralConcept(r)$
      if $g \neq null$ then
         $SBV_{GC} \leftarrow SBV_{GC} \cup g$
      else
        $i \leftarrow getIndividualConcept(r)$
       if $i \neq null$ then
          $SBV_{IC} \leftarrow SBV_{IC} \cup i$
Output: $SBV_{GC}, SBV_{IC}, TR'_{VC}, TR'_{BR}$.
```

Algorithm 1 describes the implemented pre-processing algorithm. In this algorithm, the operator *replaceAll* performs replacement of certain text rumblings *r* with values from the *replaced* set. Basically, it extracts named entities, synonymous forms, and identifies individual concepts; however, their identification is a non-trivial task and should be considered as optional. The output of the algorithm are the sets of extracted SBVR general concepts and individual concepts together with the updated text rumblings for further processing and acquisition of SBVR verb concepts and business rules.

One of the issues with entity extraction is the resolution of conjunction and disjunction cases (e.g., "*manager or VIP manager*", "*administrator start/restart/shutdown system*"). Pittke et al. [26] identified 9 anti-patterns of activity naming in the context of business process modeling, application of which would result in multiple elements after preprocessing, with 4 of them related to extra information coming from conjunctive or disjunctive-like clauses. Hence, multiple noun forms or verb phrases contained inside given text rumblings must be split into separate atomic phrases. Moreover, when forming an SBVR vocabulary, these changes must be propagated to other interrelated concepts as well, e.g., changes made to general concepts must be reflected in corresponding verb concepts and business rules.

Algorithm 2 represents an implemented algorithm for the post-processing of the generated business vocabulary concepts and business rules by propagating modifications after the re-tagging step in the extracted noun phrases and verb phrases throughout the whole sets of extracted verb concepts and business rules. There, the notion $c_1 \overset{part}{\Longrightarrow} c_2$ means that a concept c_1 is contained in (*is-part-of*) a concept c_2.

Algorithm 2. postprocessConcepts ($\boldsymbol{SBV_{GC}}$, $\boldsymbol{SBV_{VC}}$, $\boldsymbol{SBV_{BR}}$)

```
for $\forall gc \in SBV_{GC}$:
    $SN$ ← extracted atomic noun phrases
    if $|SN| > 1$:
        $SBV_{GC} \leftarrow (SBV_{GC} - gc) \cup SN$
        for $\forall vc \in \{vc | vc \in SBV_{VC}, gc \overset{part}{\Longrightarrow} vc\}$:
            $SBV_{VC} \leftarrow SBV_{VC} - vc$
            for $c' \in SN$:
                vc' ← verb concept obtained by changing c to c'
                $SBV_{VC} \leftarrow SBV_{VC} \cup vc'$
                for $\forall br \in \{br | br \in SBV_{BR}, vc \overset{part}{\Longrightarrow} br\}$:
                    $SBV_{BR} \leftarrow SBV_{BR} - br$
                    for $c'' \in SN$:
                        br' ← business rule obtained by changing vc to vc'
                            $SBV_{BR} \leftarrow SBV_{BR} \cup br'$
for $\forall vc \in SBV_{VC}$:
    $SV$ ← extracted atomic verb phrases
    if $|SV| > 1$:
        $SBV_{VC} \leftarrow (SBV_{VC} - vc) \cup SV$
        for $\forall br \in \{br | br \in SBV_{BR}, vc \overset{part}{\Longrightarrow} br\}$:
            $SBV_{BR} \leftarrow SBV_{BR} - vc$
            for $c' \in SV$:
                br' ← business rule obtained by changing vc to vc'
                $SBV_{BR} \leftarrow SBV_{BR} \cup br'$
Output: $SBV_{VC}, SBV_{BR}$.
```

It should be emphasized that the Algorithm 2 as well as the algorithms presented in Section 4.3 are consistent with the aforementioned Business Rules Mantra, and closely follow the methodology described in [3].

4.3. Extraction algorithms

The main idea of the NLP-enhanced extraction algorithm is the multistage application of a parser using part of speech patterns on the specific chunks of text. We assume that the output of the parser is a set of tokens (words) labeled using Penn Treebank notation [27]. The grammar for these patterns is defined using ABNF notation [28]:

```
POS_GC_MULTIPLE = ⟨NNS⟩
POS_IC_MULTIPLE = ⟨NNPS⟩
CHAR_COND = ⟨JJ⟩|⟨JJR⟩|⟨JJS⟩
PREP_COND = ⟨RB⟩|⟨RBR⟩|⟨RP⟩|⟨TO⟩|⟨IN⟩|⟨PREP⟩
CHAR_COND = ⟨JJ⟩|⟨JJR⟩|⟨JJS⟩
PREP_COND = ⟨RB⟩|⟨RBR⟩|⟨RP⟩|⟨TO⟩|⟨IN⟩|⟨PREP⟩
ADJ_COND = ⟨ADJ⟩|⟨CHAR_COND⟩
IF_COND = 'if'
GC_SINGLE_COND = ⟨NN⟩|⟨VBG⟩|⟨POS⟩|⟨RBS⟩|⟨FW⟩
IC_SINGLE_COND = ⟨NNP⟩|⟨POS⟩|⟨CD⟩
GC_COND = ⟨GC_SINGLE_COND⟩|⟨POS_GC_MULTIPLE⟩
IC_COND = ⟨IC_SINGLE_COND⟩|⟨POS_IC_MULTIPLE⟩
VERB_FORM = ⟨VB⟩|⟨VBD⟩|⟨VBG⟩|⟨VBN⟩|⟨VBP⟩|⟨VBZ⟩
VERB_COND = ⟨VERB_FORM⟩|⟨PREP_COND⟩
GC_COND = ⟨GC_SINGLE_COND⟩|⟨POS_GC_MULTIPLE⟩
IC_COND = ⟨IC_SINGLE_COND⟩|⟨POS_IC_MULTIPLE⟩
VERB_COND = ⟨VERB_FORM⟩|⟨PREP_COND⟩
GC = [⟨ADJ_COND⟩], [⟨IC_COND⟩], {⟨GC_COND⟩}
IC = [⟨ADJ_COND⟩], [⟨GC_COND⟩], {⟨IC_COND⟩}
GC_OR_IC = [⟨ADJ_COND⟩], {⟨GC_COND⟩|⟨IC_COND⟩}
UNARY_VC = ⟨GC_OR_IC⟩, {⟨VERB_COND⟩}, [⟨CHAR_COND⟩]
BINARY_VC = ⟨UNARY_VC⟩, ⟨GC_OR_IC⟩
NARY_VC = 2 * ⟨UNARY_VC⟩, 1 * ⟨GC_OR_IC⟩
ANY_VC = 1 * ⟨UNARY_VC⟩, [⟨UNARY_VC⟩], 1 * ⟨GC_OR_IC⟩
RULE_TAG = "It is obligatory"|"It is possible"|"It is permitted"|"It is required"
BR = 1 * ⟨RULE_TAG⟩, ⟨ANY_VC⟩, 1 * (⟨IF_COND⟩, ⟨ANY_VC⟩), 1 * (⟨CC⟩, ⟨ANY_VC⟩)
```

4.3.1. Simple cascaded extraction algorithm

Algorithm 3 represents a simple grammar-based chunking algorithm, which sequentially processes text rumblings and identifies instances of general, individual, verb concepts, and business rules in each of them. Here, *tagAndExtract...(r)* operators perform tagging of identified concepts in actual text rumblings; the identified concepts are added to the corresponding sets. The advantage of this algorithm over generic formal grammar-based taggers is that it unifies various wording forms (singular/plural, tenses, etc.), which do not coincide in different text rumblings. Additionally, it can be extended to identify multiplicity cases and to insert general quantifiers (e.g., "*one or many*"), at the same time propagating them to the identified SBVR model elements.

Algorithm 3. extract ($\boldsymbol{TR_{GC}}$, $\boldsymbol{TR'_{VC}}$, $\boldsymbol{TR'_{BR}}$,)

```
$for\ \forall r \in TR_{GC} \cup TR'_{VC} \cup TR'_{BR}$:
    $SBV_{GC(r)}, SBV_{IC(r)} \leftarrow tagAndExtractGC\_IC(r)$
    $SBV_{GC} \leftarrow SBV_{GC} \cup SBV_{GC(r)}$
    $SBV_{IC} \leftarrow SBV_{IC} \cup SBV_{IC(r)}$
    $SBV_{VC} \leftarrow SBV_{VC} \cup tagAndExtractMultinaryVC(r)$
    $SBV_{VC} \leftarrow SBV_{VC} \cup tagAndExtractBinaryVC(r)$
    $SBV_{VC} \leftarrow SBV_{VC} \cup tagAndExtractUnaryVC(r)$
    $SBV_{BR} \leftarrow SBV_{BR} \cup tagAndExtractBR(r)$
Output: $SBV_{GC}, SBV_{IC}, SBV_{VC}, SBV_{BR}$.
```

The main issue with Algorithm 3 is that it is biased by its direct dependency on POS tagging performance, and fails to identify multiple cases, where noun and verb forms are identical. This issue is addressed in the modified algorithm presented in Section 4.3.2.

4.3.2. Stepwise cascaded extraction algorithm

To overcome the issue of misidentified POS tags, we introduced certain modifications to our simple cascaded extraction algorithm, which include additional retagging steps after each stage (Algorithm 4).

Algorithm 4. extractStepwise ($\boldsymbol{TR_{GC}}$, $\boldsymbol{TR'_{VC}}$, $\boldsymbol{TR'_{BR}}$,)

```
$TR' \leftarrow TR_{GC} \cup TR'_{VC} \cup TR'_{BR}$
$for\ \forall r \in \{r \in TR' | conceptType(r) \neq BR\}$:
    $SBV_{GC(r)}, SBV_{IC(r)} \leftarrow tagAndExtractGC\_IC(r)$
    $SBV_{GC} \leftarrow SBV_{GC} \cup SBV_{GC(r)}$
    $SBV_{IC} \leftarrow SBV_{IC} \cup SBV_{IC(r)}$
replaceUpdated(TR')
$for\ \forall r \in \{r \in TR' | conceptType(r) = BR\}$:
    $SBV_{GC(r)}, SBV_{IC(r)} \leftarrow tagAndExtractGC\_IC(r)$
    $SBV_{GC} \leftarrow SBV_{GC} \cup SBV_{GC(r)}$
    $SBV_{IC} \leftarrow SBV_{IC} \cup SBV_{IC(r)}$
$for\ \forall r \in TR'$:
    $SBV_{VC} \leftarrow SBV_{VC} \cup tagAndExtractMultinaryVC(r)$
    $SBV_{VC} \leftarrow SBV_{VC} \cup tagAndExtractBinaryVC(r)$
    $SBV_{VC} \leftarrow SBV_{VC} \cup tagAndExtractUnaryVC(r)$
$for\ \forall r \in \{r \in TR' | conceptType(r) = BR\}$:
    $SBV_{BR} \leftarrow SBV_{BR} \cup tagAndExtractBR(r)$
replaceUpdated(TR')
$for\ \forall r \in \{r \in TR' | conceptType(r) = BR\}$:
    $SBV_{BR} \leftarrow SBV_{BR} \cup tagAndExtractBR(r)$
    $SBV_{BR} \leftarrow SBV_{BR} \cup tagAndExtractBR2(r)$
Output: $SBV_{GC}, SBV_{IC}, SBV_{VC}, SBV_{BR}$.
```

In Algorithm 4, several moments need to be underlined. First, POS retagging updates are propagated through the entire set of text rumblings. Second, the extraction of business rules is performed in several stages; additional extraction procedure *tagAndExtract2* is performed to ensure that the parser recognizes a complete business rule concept instead of partial, which might have been extracted in previous step (the latter instance is removed from SBV_{BR}). In the current implementation, retagging procedure applies a single heuristic rule, which selects the first candidate verb coming after the identified general/individual concept. As discussed further, one may also employ additional, more sophisticated techniques to improve the quality of the results in this stage.

Further, Algorithm 4 is extended to support multinary (or *n*-ary) associations (Algorithm 5). Here, *identifiedGC* and *identifiedIC* are the sets of previously identified general and individual concepts, *sorted* in a descending order by their length (i.e., the number of contained words) to satisfy the "matching by the longest suffix" principle. Note, that this extension of the main algorithm will add new general concepts, provided they follow the pattern, where nouns can be combined with the previously identified general concepts.

Algorithm 5. retagVerbCandidate(r, POS$_r$, verbTags, identifiedGC, identifiedIC)

Input: rumbling r with POS tags vector POS$_r$

```
if conceptType(r) = GC
    return POS_r
matchGC ← check if POS_r can be mapped to GC
matchIC ← check if POS_r can be mapped to IC
if matchGC or (not matchGC and matchIC) or (not matchGC and not matchIC and $POS_r \cap verbTags = \emptyset$)
       Search for concepts from identifiedGC and identifiedIC  in r
       startIndex ← indices of first nouns which are not parts of existing GC or IC in r
       for $ind \in startIndex$
           pos_loop: for i = ind to |POS_r|
                $r_i$ ← i-th token of rumbling r
                if $r_i$ is also a verb:
                      $POS_r(i)$ ← VB
                      break pos_loop
                if $i \neq ind$
                      gc ← create new GC from r subset $r_{\{k\}}$, $ind \le k \le i$
                      $SBV_{GC} \leftarrow SBV_{GC} \cup gc$   # Add newly identified concept
```

Output: *POS$_r$*

As mentioned previously in this section, the retagging procedure can be enhanced in multiple ways, e.g. by exploiting external corpora using heuristic likelihood measure. Let $S_{POS}(w_t)$ be the set of all possible part-of-speech tags for word w in position t. Then, identification of the actual part of speech of the given word w is context-dependent on the k preceding and l succeeding words. Therefore, the identification of an actual POS of w_t would be defined as likelihood maximization:

$$\underset{pos_k}{\operatorname{argmax}} \frac{Count(pos(w_{t-k}), \ldots, pos(w_{t-1}), pos_k(w_t), pos(w_{t+1}), \ldots, pos(w_{t+l}))}{Count(pos(w_{t-k}), \ldots, pos(w_{t-1}), pos(w_{t+1}), \ldots, pos(w_{t+l}))}$$

where $pos_k(w_t)$ is the kth candidate POS of w_t from $S_{POS}(w_t)$, $pos(w_{t-1})$ is the actual POS of the preceding word $w_{t-1,}$ $pos(w_{t+1})$ is the actual POS of the succeeding word w_{t+1}. The objective is to obtain the most frequently used POS of w_t, when w_t appears in the predefined context. For example, let us consider the text rumbling "*manager sign contract*", and define context with $k = 1$ and $l = 0$; hence it would be simplified to obtaining the most frequent n-gram. If "*NN VB*" is identified as the most frequent 2-gram POS for "*manager sign*", then it will be preferred over "*NN NN*", which is preferred by POS tagger by default, and the "*sign*" will be retagged to "*VB*".

5. Experiment

This section gives an overview of the experiment performed to validate the presented algorithms. Section 5.1 describes the experiment settings and the dataset used in the experiment; Section 5.2 defines a set of metrics used for the evaluation of the acquired experiment results; Section 5.3 presents the main results of the experiment; lastly, certain conclusions on the obtained results are drawn in Section 5.4.

5.1. Experiment settings

The experiment was performed using a set of both our own developments, as well as tools developed by other parties:

- All of the presented pre-processing, post-processing and extraction algorithms were coded in Java programming language.[4]

[4] The latest version of the implementation can be found at: https://github.com/paudan/sbvr-extraction.

- Stanford CoreNLP toolkit [21] was used to implement the functionality of named entity recognition, tokenization, and POS tagging. The core features of this tool are consistency with the Penn TreeBank tags, and dependency parser for the recognition of grammatical relationships between words. In addition to that, CoreNLP's POS tagger makes use of log-linear conditional Markov models (CMM) [17], and its named entity recognition engine uses conditional random fields [18]. During our initial experimenting, Stanford CoreNLP performed better than its close competitor Apache OpenNLP, therefore, we chose it as baseline tool in our development.
- WordNet lexical database [29] was used to identify synonyms or check for existing verb forms.
- SimpleNLG tool [30] was used for the normalization of verb forms to simple present tense. The retagging procedure applied a single heuristic rule.

The initial experimental dataset consisted of 10 UML use case models represented using UML use case diagrams. These models were taken from various relevant sources, such as, OMG UML specification, popular UML books, and research papers. Before proceeding further, some of the models were refactored to meet common naming and modeling practices, e.g., renaming ill-named use cases. In addition, we decided to ignore all capital letters in the models in order to avoid confusion between actual proper nouns and general nouns written with the first capital letter (e.g., an actor named as "Manager", instead of "manager"); if not dealt with, this would result in false identification of SBVR individual and general concepts during the extraction process.

After that, sets of raw text rumblings were manually extracted from the reviewed UML models. Next, corresponding sets of *benchmark* results (SBVR concepts and business rules) were manually acquired from these text rumblings. This whole manual process was guided by the predefined model transformation rules presented in [3] and our own expert knowledge to get the best possible result. Lastly, to automate the experiment, each set of text rumblings together with corresponding benchmark results was stored as an XML document, which then could be passed as an input to the developed extraction system

5.2. Evaluation metrics

The developed extractors were evaluated in terms of precision, recall, and F-measure. Precision is the ratio of correctly recognized concepts to the number of total extracted concepts; recall is a ratio of correctly recognized concepts to a number of correct concepts. By defining type $t \in \{GC, VC, BR\}$ and C_t as t-type of a concept, these results were calculated for each C_t, as follows:

$$precision_t = \frac{number\ of\ C_t\ correctly\ identified\ by\ extractor}{number\ of\ C_t\ identified\ by\ extractor}$$

$$recall_t = \frac{number\ of\ C_t\ correctly\ identified\ by\ extractor}{number\ of\ manually\ identified\ C_t}$$

While these two metrics try two capture slightly different measures, it is preferred to use harmonic mean of these two measures:

$$F_{\beta_t} = \frac{(1+\beta^2)(precision * recall)}{\beta^2 \times precision + recall}, \beta = 1$$

A general observation about the performance of any extractor, including our own developments, is that the extraction results of one stage might be influenced by the results of the preceding stages. In other words, failing to recognize some general concept may lead to false extraction of certain verb concepts (which are based on that general concept) in the next stage, and so on.

For aggregate measures, we use simple average and weighted average scores. Weighted average is expected to represent more precisely the complexity of different Use Case models in terms of the number of elements and concepts, which can be extracted. In this context, weighted average is calculated as follows:

$$WA_t = \frac{\sum_{i=1}^{N} score_t^i * Total_t^i}{\sum_{i=1}^{N} Total_t^i}$$

where N is the total number of models used in evaluation, $score_t^i$ is the score obtained for concept of type t in the ith model, and $Total_t^i$ is the actual number of manually extracted concepts for concept of type t in the ith model (also presented in Table 1).

5.3. Experiment results

Table 1 depicts the benchmark extraction results, which were acquired by manually extracting SBVR concepts and business rules from the given dataset of text rumblings. *Total* columns represent the actual numbers of particular concepts (including duplicates), while columns under *Distinct* exclude all the repetitions obtained at each particular step. Values of *Total* are also used to calculate weighted aggregate scores in Section 5.2.

Table 2 presents the precision and F-Measure scores for the acquired extraction results, when a generic POS tagger was used.

The results of using a generic POS tagger are quite satisfactory. However, as discussed in previous section, a generic POS tagger is biased towards so-called regular language, while various modeling practices tend to contain numerous infinitive forms. To mitigate the impact of this issue, we trained our own POS tagger focussing on the recognition of infinitive forms. Several well-known resources were used to form a corpus for the training of our POS tagger, specifically, Brown and CONLL 2000 text corpora together with a dataset of reported issues; all these sources can be found within resources provided by NLTK toolkit [31]. The selected corpora best met our requirements with respect to formal language, content of incorporated vocabularies, style of writing, and already assigned

Table 1
The benchmark (manual) extraction results.

	Total			Distinct		
	GC	VC	BR	GC	VC	BR
Model 1	19	18	4	9	11	4
Model 2	39	34	8	11	19	8
Model 3	35	37	11	14	15	11
Model 4	37	49	16	14	17	16
Model 5	32	17	2	8	13	2
Model 6	41	31	6	11	19	6
Model 7	9	16	3	8	10	3
Model 8	33	41	13	11	16	13
Model 9	15	19	6	7	7	6
Model 10	24	19	4	8	11	4
Model 11	33	48	17	13	14	17
Model 12	53	30	4	19	23	4
Model 13	23	17	3	9	11	3
Model 14	70	67	14	17	35	14
Model 15	47	68	20	15	24	20
Model 16	33	15	–	9	15	0
Model 17	33	38	13	12	17	13
Model 18	21	17	4	10	10	4
Model 19	47	98	38	11	22	38
Model 20	65	183	78	15	31	78
Model 21	20	8	–	8	8	0
Model 22	20	7	–	9	7	0
Model 23	21	20	4	6	9	4
Model 24	23	14	2	10	10	2
Model 25	45	78	29	13	23	29

Table 2
Performance of the extraction algorithms using generic POS tagger.

	Simple cascaded extractor						Stepwise cascaded extractor					
	GC		VC		BR		GC		VC		BR	
	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score
Model 1	0,63	0,67	0,61	0,63	0,75	0,75	0,79	0,79	0,72	0,72	0,25	0,25
Model 2	0,56	0,62	0,27	0,29	0,13	0,17	0,41	0,58	0,27	0,28	0,13	0,17
Model 3	0,60	0,64	0,57	0,61	0,27	0,33	0,46	0,56	0,62	0,64	0,27	0,33
Model 4	0,57	0,70	0,61	0,70	0,50	0,59	0,62	0,72	0,71	0,75	0,50	0,59
Model 5	0,31	0,47	0,06	0,09	0,00	–	0,69	0,82	0,71	0,75	0,00	–
Model 6	0,83	0,86	0,77	0,87	0,50	0,50	0,95	0,92	0,58	0,61	0,00	–
Model 7	0,67	0,50	0,19	0,24	0,00	–	0,78	0,48	0,25	0,27	0,00	–
Model 8	0,58	0,68	0,68	0,78	0,46	0,57	0,68	0,74	0,68	0,73	0,39	0,48
Model 9	0,67	0,74	0,53	0,65	0,33	0,40	0,93	0,93	0,16	0,16	0,00	–
Model 10	0,67	0,80	0,68	0,81	0,50	0,50	0,71	0,83	0,90	0,94	0,50	0,50
Model 11	0,36	0,49	0,38	0,51	0,18	0,26	0,64	0,71	0,54	0,67	0,18	0,26
Model 12	0,39	0,50	0,27	0,40	–	–	0,44	0,50	0,47	0,47	–	–
Model 13	0,39	0,50	0,29	0,46	0,00	–	1,00	0,90	0,77	0,87	0,00	–
Model 14	0,60	0,68	0,55	0,71	0,17	0,25	0,97	0,86	0,72	0,78	0,17	0,25
Model 15	0,57	0,66	0,44	0,54	0,20	0,29	0,40	0,53	0,49	0,57	0,20	0,29
Model 16	0,58	0,63	0,33	0,36	–	–	0,39	0,48	0,47	0,47	–	–
Model 17	0,91	0,87	0,86	0,86	0,83	0,83	0,85	0,86	0,67	0,67	0,50	0,50
Model 18	0,48	0,50	0,29	0,33	0,00	–	0,43	0,46	0,29	0,33	0,00	–
Model 19	0,45	0,57	0,38	0,49	0,14	0,18	0,57	0,66	0,38	0,46	0,05	0,07
Model 20	0,52	0,60	0,33	0,37	0,05	0,07	0,52	0,60	0,35	0,39	0,05	0,07
Model 21	0,55	0,61	0,38	0,46	–	–	0,90	0,88	1,00	1,00	–	–
Model 22	0,74	0,78	0,57	0,67	–	–	1,00	0,95	0,71	0,71	–	–
Model 23	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Model 24	0,48	0,55	0,50	0,64	0,50	0,67	0,91	0,84	0,57	0,62	0,50	0,67
Model 25	0,44	0,57	0,42	0,51	0,14	0,17	0,53	0,62	0,49	0,54	0,14	0,17
Average:	**0,58**	**0,65**	**0,48**	**0,56**	**0,32**	**0,44**	**0,70**	**0,73**	**0,58**	**0,62**	**0,23**	**0,37**
Weighted average:	**0,57**	**0,64**	**0,46**	**0,54**	**0,22**	**0,26**	**0,67**	**0,71**	**0,52**	**0,57**	**0,17**	**0,21**

POS tags, which could be used to train our POS tagger. All verbs in the selected corpora were converted into their infinitive forms using NLTK and its internal WordNet lemmatization functionality; the resulting corpora were combined with the original texts into a single corpus, which was then used to train the POS tagger. Bidirectional feature extraction was also used to make the tagger more dependent on the context before and after each word, which corresponds to heuristics described in Section 4.3.2. The tagger was then applied in our development. The results of the extraction enhanced with custom-trained POS tagger are presented in Table 3.

Table 3
Performance of the extraction algorithms using custom-trained POS tagger.

	Simple cascaded extractor						Stepwise cascaded extractor					
	GC		VC		BR		GC		VC		BR	
	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score
Model 1	0,58	0,63	0,50	0,55	0,75	0,75	0,79	0,79	0,61	0,65	0,25	0,25
Model 2	0,62	0,64	0,38	0,39	0,13	0,13	0,56	0,60	0,24	0,24	0,13	0,13
Model 3	0,71	0,77	0,78	0,82	0,73	0,76	0,57	0,67	0,78	0,80	0,73	0,76
Model 4	0,73	0,78	0,71	0,73	0,63	0,67	0,49	0,62	0,74	0,74	0,63	0,67
Model 5	0,53	0,63	0,71	0,83	0,50	0,50	0,63	0,71	0,77	0,77	0,00	–
Model 6	0,73	0,79	0,58	0,74	0,17	0,25	0,95	0,92	0,68	0,75	0,00	–
Model 7	0,78	0,54	0,63	0,74	0,00	–	0,67	0,46	0,38	0,40	0,00	–
Model 8	0,68	0,75	0,80	0,84	0,62	0,70	0,48	0,59	0,75	0,77	0,54	0,61
Model 9	0,53	0,62	0,53	0,69	0,17	0,22	0,93	0,90	0,32	0,33	0,00	–
Model 10	0,96	0,98	0,95	0,95	1,00	1,00	0,58	0,74	0,95	0,95	1,00	1,00
Model 11	0,36	0,46	0,58	0,65	0,35	0,44	0,46	0,56	0,73	0,77	0,41	0,52
Model 12	0,18	0,28	0,13	0,22	–	–	0,23	0,34	0,87	0,87	–	–
Model 13	0,57	0,65	0,53	0,69	0,00	–	1,00	0,90	0,71	0,77	0,00	–
Model 14	0,68	0,74	0,67	0,80	0,50	0,60	0,97	0,89	0,78	0,83	0,50	0,60
Model 15	0,75	0,81	0,57	0,65	0,40	0,50	0,43	0,58	0,57	0,66	0,40	0,50
Model 16	0,82	0,87	0,87	0,90	–	–	0,27	0,43	0,73	0,73	–	–
Model 17	0,88	0,88	0,75	0,79	0,58	0,58	0,82	0,84	0,56	0,59	0,33	0,33
Model 18	0,52	0,58	0,35	0,43	0,00	–	0,43	0,50	0,35	0,41	0,00	–
Model 19	0,79	0,88	0,88	0,93	0,78	0,88	0,62	0,75	0,73	0,76	0,46	0,52
Model 20	0,59	0,67	0,44	0,47	0,20	0,23	0,39	0,48	0,46	0,48	0,20	0,23
Model 21	0,60	0,67	0,50	0,62	–	–	0,85	0,85	1,00	1,00	–	–
Model 22	0,74	0,78	0,57	0,67	–	–	1,00	0,95	0,71	0,71	–	–
Model 23	0,95	0,95	0,90	0,90	0,75	0,75	0,95	0,95	0,90	0,90	0,75	0,75
Model 24	0,48	0,55	0,50	0,64	0,50	0,67	0,91	0,84	0,57	0,62	0,50	0,67
Model 25	0,69	0,81	0,87	0,90	0,89	0,91	0,49	0,65	0,74	0,75	0,57	0,58
Average:	**0,66**	**0,71**	**0,63**	**0,70**	**0,46**	**0,59**	**0,66**	**0,70**	**0,66**	**0,69**	**0,35**	**0,54**
Weighted average:	**0,65**	**0,71**	**0,63**	**0,69**	**0,47**	**0,52**	**0,62**	**0,68**	**0,64**	**0,67**	**0,37**	**0,39**

Finally, our last goal in the experiment was to make comparison between the *original* solution featuring no NLP capabilities [3] with our newly developed version introducing the NLP-enhanced extraction. It is observed, that, for a given set of UML use case models, the original solution provided somewhat similar results as compared to the results of NLP-enhanced solution. However, such comparison is fair only in the case of non-normalized output (i.e., output that does not contain any text normalization or NLP preprocessing), and the performance of the original solution would drop significantly if normalized concepts were used for the evaluation. It is much more difficult to match directly outputs of automated extraction, which does not use any natural language processing, with the compared artefacts containing normalized forms of particular words; hence, if normalized outputs are used, the original solution would fail beyond any comparison. For objectivity reasons, we provide the extraction results of the original solution for both cases in Table 4 as well.

It can be observed that automated extraction resulted in much higher performance, when verb concepts or business rules were extracted. This is clearly explained by the fact that general concepts or verb phrases were formed using less strict normalization requirements, which resulted in phrases that could be easily matched with original phrases without normalization (currently, matching was performed using plain string matching). However, matching with normalized output failed almost completely, considering that verbs were not properly matched or normalized.

5.4. Discussion

Automatic extraction of SBVR business vocabulary and business rules from labels in UML use case model is a challenging task highly influenced by the ambiguity coming from multiple interpretations of certain terms. It can be easily observed from the results in Tables 2 to 4 that the straightforward extraction (Algorithm 3) did not always produce output as precisely as the improved stepwise cascaded extraction algorithm (Algorithm 4). However, it is assumed that simple extraction algorithm might work better with models comprised of elements with relatively simple label names; such assumption is supported by the acquired experiment results. For the future, it might be a viable option to introduce some level of automation to heuristically selecting the technique which is expected to provide better extraction results for the specific case.

Part-of-speech (POS) tagger performance plays a major role in the current implementation. As stated before, generic tagger tools are trained for tagging well-formed and more complete documents which provide more context required to resolve ambiguous situations. However, they are less suitable for tagging short phrases and do not always produce the expected output. Custom trained taggers could be a natural option to improve results. Indeed, the results show that the application of custom-trained POS tagger often resulted in improved performance, and aggregate measures indicate overall better performance as well. Yet, there were multiple cases, when it performed worse than the generic tagger. Such inconsistency can be explained by the fact that our custom-trained tagger, while it is biased towards the identification of infinitive forms of homonymous verbs, is now sometimes failing to correctly

Table 4
Performance results of the original solution (non-normalized and normalized output).

	Non-normalized output						Normalized output					
	GC		VC		BR		GC		VC		BR	
	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score	Prec	F-score
Model 1	0,63	0,62	0,83	0,83	0,75	0,75	0,63	0,62	0,00	–	0,00	–
Model 2	0,10	0,18	0,77	0,77	0,50	0,50	0,10	0,18	0,00	–	0,00	–
Model 3	0,20	0,33	1,00	1,00	1,00	1,00	0,14	0,24	0,00	–	0,00	–
Model 4	0,14	0,23	1,00	1,00	1,00	1,00	0,14	0,23	0,00	–	0,00	–
Model 5	0,17	0,29	1,00	1,00	1,00	1,00	0,16	0,27	0,00	–	0,00	–
Model 6	0,90	0,90	1,00	1,00	1,00	1,00	0,85	0,85	0,07	0,07	0,00	–
Model 7	0,33	0,29	0,63	0,63	0,00	–	0,33	0,29	0,00	–	0,00	–
Model 8	0,18	0,29	1,00	1,00	1,00	1,00	0,16	0,26	0,00	–	0,00	–
Model 9	1,00	1,00	0,90	0,90	0,67	0,67	0,67	0,67	0,00	–	0,00	–
Model 10	0,08	0,15	1,00	1,00	1,00	1,00	0,08	0,15	0,00	–	0,00	–
Model 11	0,18	0,29	1,00	1,00	1,00	1,00	0,18	0,29	0,00	–	0,06	0,06
Model 12	0,19	0,32	0,67	0,67	0,00	–	0,26	0,41	0,07	0,07	–	–
Model 13	0,91	0,91	1,00	1,00	1,00	1,00	0,91	0,91	0,00	–	0,00	–
Model 14	0,93	0,93	1,00	1,00	1,00	1,00	0,87	0,87	0,10	0,10	0,00	–
Model 15	0,06	0,11	0,81	0,82	0,75	0,75	0,06	0,11	0,02	0,02	0,00	–
Model 16	0,09	0,17	1,00	1,00	–	–	0,09	0,17	0,00	–	–	–
Model 17	0,90	0,84	0,97	0,97	0,73	0,73	0,79	0,74	0,19	0,19	0,00	–
Model 18	0,52	0,67	0,77	0,77	0,50	0,50	0,57	0,73	0,06	0,06	0,00	–
Model 19	0,06	0,11	0,92	0,92	0,88	0,88	0,06	0,12	0,00	–	0,00	–
Model 20	0,19	0,30	1,00	1,00	1,00	1,00	0,19	0,27	0,01	0,01	0,00	–
Model 21	0,80	0,82	1,00	1,00	–	–	0,80	0,82	0,13	0,13	–	–
Model 22	0,84	0,78	1,00	1,00	–	–	0,84	0,78	0,00	–	–	–
Model 23	0,84	0,82	0,95	0,95	0,75	0,75	0,84	0,82	0,05	0,05	0,00	–
Model 24	0,78	0,78	0,86	0,86	1,00	1,00	0,78	0,78	0,07	0,07	0,00	–
Model 25	0,11	0,20	0,84	0,84	0,69	0,69	0,09	0,16	0,01	0,01	0,00	–
Average:	**0,45**	**0,49**	**0,92**	**0,92**	**0,78**	**0,86**	**0,42**	**0,47**	**0,03**	**0,07**	**0,00**	**0,06**
Weighted average:	**0,40**	**0,46**	**0,93**	**0,93**	**0,87**	**0,87**	**0,38**	**0,44**	**0,03**	**0,03**	**0,00**	**0,00**

tag other words that were handled correctly by the original tagger. Nevertheless, considering certain limitations of the corpus used for training, specificity of input models, as well as the tagger itself, we consider the achieved results as a considerable improvement over the previously presented case with generic POS tagger. To achieve more stable and consistent performance, the developed POS tagger will require further investigation and tuning.

Lastly, the most obvious shortcoming of the original solution introduced in [3] was its inability to extract correct verb concepts from the text rumblings containing noun phrases, verb phrases or verbs followed by prepositions. Given a tuple <Actor::*'manager',* UseCase::'*sign with digital signature', Association('manager', 'sign with digital signature')*>, the simple heuristic transformation rule would provide the following incorrect result: 'manager *sign* with digital signature' (the correct non-normalized result would be: 'manager *sign with* digital signature'). This shortcoming was overcome in our NLP-enhanced solution. However, in our experimental dataset, the great amount of the extracted use cases' names comprised single verbs; hence, a relatively simple heuristics used in the original solution was also good enough to show satisfactory extraction results. In this regard, it is highly expected that experimenting with larger sets of use case models representing higher variability of elements' naming cases would indicate even greater gap in terms of quality of the results acquired by the original solution and its NLP-enhanced version.

6. Threats to validity

Basic threats to the validity of our experiment results and the approach itself are categorized as follows [32]: (a) threats to construct validity, (b) threats to internal validity, and (c) threats to external validity. All three categories of threats are assessed in the following subsections.

6.1. Threats to construct validity

To apply any approach properly, one must ensure common understanding of things among all interested parties. However, the interpretability and common understanding of concepts throughout the problem domain has the potential to be handled improperly in certain settings – this is called a threat to construct validity. This threat was minimal in our research, because: (1) all the developed algorithms were automated and did not leave any space for misinterpreting things; (2) non-automated parts of the approach were performed exclusively by the authors, who naturally shared common understanding of things.

6.2. Threats to internal validity

Internal validity considers internal factors and how they can affect the results, e.g., human error, motivation, subjectivity. Even though the automated nature of the experiment helped to reduce the influence of human factor, some threats to internal validity remained.

Manual selection of source models for the experiment might be considered as one of such threats, because it holds certain degree of subjectivity, especially when considering a relatively small volume of the experimental dataset acquired from these models. Nonetheless, the models were carefully selected considering the variety of possible use case modeling cases to test various capabilities of the NLP-enhanced model transformation approach. Therefore, we state that in this case the subjectivity in the selection of source models was justified.

Another threat to internal validity, which is a human error, is also related to the set of source models. The thing is that the use case models were selected from different sources, neither of which were ready to be processed automatically by our experimental system. Human errors could appear during this preparation phase. However, all the inputs for the experiment were double-checked and agreed upon by all authors, and therefore, we assume this threat to be minimal.

6.3. Threats to external validity

External validity can be biased by the non-rigorous generalization or failure to generalize the developed approach to a larger extent. As it was stated previously, the authors believe that the developed algorithms could be applied to a wider range of UML models, as well as other ECore-based models. Here, the main threat to external validity is the fact that a relatively small number of visual models was selected for the experiment – this objectively lessens the reliability of the acquired experiment results to some extent.

Another threat is the lack of standardized modeling practices, in particular, using certain conventions of naming model elements. In addition, there is always a chance that some practitioners will disagree on certain modeling practices used in this research, e.g., use of *n*-ary associations or hierarchical relationships. All this could lead to poorer results compared to the ones presented in this paper.

7. Related work

In this research, we outline the three major areas of related work, namely, natural language processing, conceptual models' interpretation, and model-driven systems engineering.

Natural language processing. Due to the large amount of research carried out in the NLP field, we will distinguish only certain publications presenting most relevant research trends. For more in-depth information on the subject one can consult various survey papers of recent research, such as [7,33–35].

In the context of our research, we recognize entity extraction, relation extraction and named entity recognition as the most relevant approaches used in NLP. While earlier developments tend to apply complex machine learning techniques, such as maximum entropy optimization [36], conditional random fields [4,19], the most recent developments focus on contextual representation extraction and its extension for different NLP tasks [5,37,38]. As more and more computational and data resources become available, recent trends focus on research in the state-of-the-art deep learning techniques, which tend to provide better performance compared to statistical or feature-based machine learning approaches. Such internal representation learning can be applied for multi-word term representation [39], grouping of term pairs [40], embed various lexical resources, like WordNet [41] or lexical dictionaries, such as Oxford, Collins or Cambridge dictionaries [42] to estimate semantical relation between pairs of words. Relation extraction is also relevant in this context, relying on linguistic grammatical, syntactic dependency structure based pattern extraction [43], semantic information feature based extraction [44] or using kernel based classifiers, such as SVM, with tree-based kernels which incorporate syntactic structure or semantic relation features [45,46]. Recent findings in deep learning-based research include, but are not limited to, applications of recurrent neural networks [47,48], convolutional neural networks [49], graph convolutional networks [50] or transformer language models [51]. Whereas current implementation of our approach does not apply the cutting-edge developments of this field, they are positioned high in our future research list as means to further improving the overall performance of our solution.

Conceptual models' interpretation. Another important trend of related research is focused on the interpretation of labels in models using natural language techniques, which is one of the key points in this research. A solid stream of work is published related to similar issues in business process modeling. While the application domain is somewhat different compared the research presented in this paper, it is safe to state that many of the following techniques are transferable to our solution to a certain extent. Leopold et al. [52] tries to overcome problem of activity naming misalignment (discussed in Section 4.2) by proposing a sequential pipeline of four algorithms for activity labeling style identification. Based on such information, *action* and *business subject* is extracted from the label. While it follows grammatical pattern-based approach, which is similar to the one in this paper, it does not consider exact relation extraction, nor deals with multinary associations. Naming conventions in process model elements were also addressed in multilanguage settings [53], where the authors claim of designing a technique transferable to different languages and independent of resources, such as WordNet, while at the same time making extensive use of text corpora and linguistic knowledge, which also resonates with our ideas to apply contextual knowledge-based heuristics. At the same time, one must admit that obtaining suitable corpora might be problematic, if it concerns languages, which are overall less widely used and researched. Further, Leopold et al. [54] introduced the notion of canonicity (which could be interpreted similarly to atomicity) to describe business processes consisting of one action, one business object, and no more than one addition. This notion is used to describe refactoring for activities with labels that conform to activity naming antipatterns in [26]; hence, the principles in this paper could be one of the extension points in the post-processing step, as use case modeling also suffers from similar problems (Section 4.2). Additionally, Pittke [25] presents algorithms to detect homonyms and synonyms in the context of process models, which combine semantic vectors with information obtained from BabelNet resource. Previous attempts to integrate other sources of semantic information [55], as well as attempts to integrate them in-between in order to obtain a robust semantic parser [56] are also notable.

As it was already mentioned, recent advancements in NLP attract attention of researchers from the field of model-driven systems engineering as well. Yet, in contrast to our research, most authors tend to choose model transformations to the *opposite direction*, i.e., they use natural language specifications as a source to generate visual models in UML, BPMN, etc. Early works mainly considered generating conceptual data models from natural language specifications; as a result, a number of pioneering NL-based CASE tools for data modeling were introduced [57–59]. After the introduction of UML, research trended to the generation of UML class model [60,61]. While research on data models was, and still remains, the most extensive, other UML models and modeling notations were also investigated. Deeptimahanti and Sanyal [62] tried to generate use case models mainly focusing on role extraction and use case identification. In their approach, sentences were decomposed to atomic "subject-predicate" phrases; further, nouns were mapped to actors, and verb phrases – to use cases, while using prepositions to recognize associations between the created concepts. Generation of UCM from NL specifications is explored by other researchers as well [63,64]. Some other works concentrate on the transformations between use case models and other UML models, while utilizing various NLP techniques, e.g., Christiansen et al. [65] and Sharma et al. [66] generate class model, activity models are generated in [67], state machine models – in [68]. Next to UML, researchers also investigated mining of BPMN process models from different NL-based sources, such as group stories [69], use case specifications [70] or generic text [55]. Only one research considered *SBVR specifications*, which is of the most interest to us, as a source model for the generation of UML use case models (Thakore, Upadhyay, 2013); however, it did not cover a full set of UCM elements, only the basic ones, such as *Actor, Use Case*, and *Association*. A somewhat larger cluster of related research considers SBVR standard as a source for the transformation to other UML models. Yet again, UML class model remains a preferred target model in the analyzed works [71–73].

Generation of natural language specifications from UML models has been researched to a much smaller extent, and yet again, putting emphasis on UML class model [74,75]. Cabot et al. [76] published a comprehensive paper – the presented approach described different aspects of UML/OCL transformation to SBVR, also providing a set of verb patterns and guidelines. Several other works were aimed at the extraction of SBVR specifications from BPMN process models [77,78]. Recently, alignments between process models and their textual descriptions were also investigated to detect automatically inconsistencies between model and text [79]. Nevertheless, at the moment of writing, we could not find any published research directly dealing with M2M transformation-based development of SBVR business vocabularies and business rules from UML use case models, except for our own early research findings on semi-automatic model transformation [80], and the latest research results presented in [3]. The later research presents conceptual and engineering aspects on semi-automatic and automatic generation of SBVR specifications from UML use case models and forms the basis for our research presented in this paper.

8. Conclusions and future work

While SBVR-based specifications of business vocabularies and business rules provide certain benefits to all of the interested parties, manual development of these artefacts is tedious and time-consuming task. On the other hand, our previous experimenting with M2M transformations showed that the *automatic* transformation speeds up this task to almost nothing, however, it is very sensitive to certain modeling practices applied to source models, e.g., inconsistent naming of use case concepts resulted in a considerably greater number of erroneous SBVR model elements [3]. The application of NLP is one of the ways of diminishing the impact of bad modeling practices and thus, improving the results of model transformation results. This paper contributes to the research aimed at intensifying the application of NLP techniques in model-driven information systems development. It extends our previously developed M2M transformation approach [3] by presenting NLP-enhanced, pattern-based algorithms for the automatic extraction of SBVR specifications from UML use case diagrams.

In this paper, we reported experimental findings on the NLP-enhanced extraction of SBVR business vocabularies and business rules from real world use case models. It provides certain advantages over purely template-based rules (e.g., "the first word from a use case's name is a verb"). Using NLP-enhanced model transformation approach, one can now recognize entities, entire noun and verb phrases, and multinary associations acquiring more complete results, and of better quality when compared to the results reported in our previous work on model transformation.

In addition, the developed algorithms improved entity classification accuracy, when compared to the results obtained with generic NLP tagging. Still, there are several other NLP approaches, which have not yet been considered in our research – this includes, but not limited to the use of external sources, such as external ontologies, lexical databases or simple corpora to implement the heuristics described in Section 4. Moreover, novel machine learning approaches suggest other advanced ways for improvement, e.g., self-learning, which would be capable to identify and learn from various errors. Unfortunately, actual application of such advances in our approach is currently limited by the absence of actual datasets required to test and improve such extractors. At the same time, corpora, which are used for similar NLP-based research, do not incorporate specificity, nor errors coming from entity naming practices. In our research, we tried to overcome this problem by augmenting existing corpora with modifications that allow biasing trained tagger towards such specificity and demonstrated that this could make a positive effect on the extraction performance. The next steps would be implementation of rule-based algorithms to improve performance, as well as to compare them with machine-learning based tagger used in this paper; results in [52] indicate that application of such techniques have potential to increase performance. We also expect to overcome this barrier by extending our previously developed tool to enable the recording of actual modifications made by the tool users – this information could then be turned into features to train the tagger. It is expected that such improvement would also provide more flexibility to incorporating different contextual and semantical features.

It is safe to state that the developed approach can be generalized over a wider range of other types of UML models, and beyond. During the aforementioned VEPSEM research project, we successfully developed model transformation rules from several UML model types, as well as BPMN process model to SBVR model – all these developments could be enhanced with the presented NLP development.

CRediT authorship contribution statement

Paulius Danenas: Conceptualization, Investigation, Software, Writing - original draft, Writing - review & editing. **Tomas Skersys:** Conceptualization, Investigation, Methodology, Writing - original draft, Writing - review & editing. **Rimantas Butleris:** Investigation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] OMG, Semantics of Business Vocabulary and Business Rules (SBVR) V.1.4, OMG Doc. No.: formal/2017-05-05, 2017.

[2] K. Kapocius, T. Skersys, R. Butleris, The need for business vocabularies in BPM or ISD related activities: survey based study, in: 2014 IEEE International Conference on Computer and Information Technology: 11–13 September 2014, Xi'an, Shaanxi, China: Proceedings, IEEE Computer Society, Los Alamitos, Ca, ISBN: 9781479962389, 2014, pp. 622–629.

[3] T. Skersys, P. Danenas, R. Butleris, Extracting SBVR business vocabularies and business rules from UML use case diagrams, J. Syst. Softw. 141 (2018) 111–130.

[4] E. Yan, Y. Zhu, Identifying entities from scientific publications: A comparison of vocabulary- and model-based methods, J. Infometr. 9 (3) (2015) 455–465.

[5] M. Peters, M. Neumann, M. Iyyer, M. Gardner, Ch. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), New Orleans, Louisiana, 2018.

[6] A. Ghaddar, Ph. Langlais, Robust lexical features for improved neural network named-entity recognition, in: Proceedings of the 27th International Conference on Computational Linguistics (COLING), Association for Computational Linguistics, 2018, pp. 1896–1907.

[7] M.M. Mirończuk, J. Jarosław Protasiewicz, A recent overview of the state-of-the-art elements of text classification, Expert Syst. Appl. 106 (2018) 36–54.

[8] B. von Halle, Business Rules Applied: Building Better Systems using the Business Rules Approach, John Wiley and Sons, 2002.

[9] OMG, The business rules manifesto, 2003, businessrulesgroup.org/brmanifesto.htm.

[10] K. Kukich, Techniques for automatically correcting words in text, ACM Comput. Surv. 24 (4) (1992) 377–439.

[11] W.W. Cohen, P. Ravikumar, S.E. Fienberg, A comparison of string distance metrics for name-matching tasks, in: Subbarao Kambhampati, Craig A. Knoblock (Eds.), Proceedings of the 2003 International Conference on Information Integration on the Web (IIWEB'03), AAAI Press, 2003, pp. 73–78.

[12] M.D. Kernighan, K.W. Church, W.A. Gale, A spelling correction program based on a noisy channel model, in: Proceedings of the 13th Conference on Computational Linguistics, Vol. 2, 1990, pp. 205–210.

[13] K. Toutanova, R.C. Moore, Pronunciation modeling for improved spelling correction, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 144–151.

[14] A. Rozovskaya, K. Chang, M. Sammons, D. Roth, The university of illinois system in the CoNLL-2013 shared task, in: Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, 2013, pp. 13–19.

[15] M. Felice, Z. Yuan, Ø. Andersen, H. Yannakoudakis, E. Kochmar, Grammatical error correction using hybrid systems and type filtering, in: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, 2014, pp. 15–24.

[16] T. Ge, F. Wei, M. Zhou, Fluency boost learning and inference for neural grammatical error correction, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 1055–1065.

[17] K. Toutanova, D. Klein, Ch. Manning, Y. Singer, Feature-rich part-of-speech tagging with a cyclic dependency network, in: Proceedings of HLT-NAACL 2003, 2003, pp. 252–259.

[18] J.R. Finkel, T. Grenager, Ch. Manning, Incorporating non-local information into information extraction systems by gibbs sampling, in: Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), 2005, pp. 363–370.

[19] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2011) 2493–2537.

[20] M. Yasunaga, K. Jungo, D. Radev, Robust multilingual part-of-speech tagging via adversarial training, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 976–986.

[21] Ch.D. Manning, S. Mihai, J. Bauer, J. Finkel, S.J. Bethard, D. McClosky, The stanford CoreNLP natural language processing toolkit, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55–60.

[22] Apache OpenNLP 2010. https://opennlp.apache.org/.

[23] Spacy – Industrial-strength natural language processing in Python, https://spacy.io/.

[24] J. Mendling, H.A. Reijers, W.M.P. van der Aalst, Seven process modeling guidelines (7PMG), Inf. Softw. Technol. 52 (2) (2010) 127–136.

[25] F. Pittke, Linguistic Refactoring of Business Process Models (Doctoral thesis), WU Vienna University of Economics and Business, 2015.

[26] F. Pittke, H. Leopold, J. Mendling, When language meets language: Anti patterns resulting from mixing natural and modeling language, in: Proceedings of BPM 2014: Business Process Management Workshops, 2014, pp. 118–129.

[27] A. Taylor, M. Marcus, B. Santorini, The penn treebank: An overview, in: Treebanks: Building and using Parsed Corpora, 2003, pp. 5–22.

[28] The Internet Engineering Task Force (IETF), Augmented BNF for syntax specifications: ABNF, 2008, https://tools.ietf.org/html/rfc5234.

[29] G. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.

[30] A. Gatt, E. Reiter, SimpleNLG: a realisation engine for practical applications, in: Proceedings of the 12th European Workshop on Natural Language Generation (ENLG '09), Association for Computational Linguistics, Stroudsburg, PA, USA, 2009, pp. 90–93.

[31] S. Bird, E. Loper, E. Klein, Natural Language Processing with Python, O'Reilly Media Inc., 2009.

[32] P. Runeson, M. Höst, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, John Wiley & Sons, 2012.

[33] S. Sarawagi, Information extraction, Found. Trends Databases 1 (3) (2007) 261–377.

[34] J.G. Enriquez, F.J. Dominguez-Mayo, M.J. Escalona, M. Ross, G. Staples, Entity reconciliation in big data sources: a systematic mapping study, Expert Syst. Appl. 80 (2017) 14–27.

[35] B. Altınel, M.C. Ganiz, Semantic text classification: a survey of past and recent advances, Inf. Process. Manage. 54 (6) (2018) 1129–1153.

[36] H.L. Chieu, H.T. Ng, Named entity recognition with a maximum entropy approach, in: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4 (CONLL '03), Vol. 4, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003, pp. 160–163.

[37] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13), Vol. 2, pp. 3111–3119.

[38] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, 2018, CoRR abs/1810.04805.

[39] S. Henry, C. Cuffy, B.T. McInnes, Vector representations of multi-word terms for semantic relatedness, J. Biomed. Inform. 77 (2018) 111–119.

[40] B.T. McInnes, T. Pedersen, Evaluating semantic similarity and relatedness over the semantic grouping of clinical term pairs, J. Biomed. Inform. 54 (2015) 329–336.

[41] R. Bartusiak, L. Augustyniak, T. Kajdanowicz, P. Kazienko, M. Piasecki, WordNet2Vec: Corpora agnostic word vectorization method, Neurocomputing 326–327 (2019) 141–150.

[42] J. Tissier, C. Gravier, A. Habrard, Dict2vec: learning word embeddings using lexical dictionaries, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017.

[43] D.-Th. Vo, W. Bagheri, Self-training on refined clause patterns for relation extraction, Inf. Process. Manage. 54 (4) (2018) 686–708.

[44] G. Zhou, M. Zhang, Extracting relation information from text documents by exploring various types of knowledge, Inf. Process. Manage. 43 (4) (2007) 969–982.

[45] M. Zhang, G. Zhou, A. Aw, Exploring syntactic structured features over parse trees for relation extraction using kernel methods, Inf. Process. Manage. 44 (2) (2008) 687–701.

[46] G. Zhou, J. Li, J. Fan, Q. Zhu, Tree kernel-based semantic role labeling with enriched parse tree structure, Inf. Process. Manage. 47 (3) (2011) 349–362.

[47] M. Xiao, C. Liu, Semantic relation classification via hierarchical recurrent neural network with attention, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, pp. 1254–1263.

[48] S. Zheng, J. Xu, P. Zhou, H. Bao, Z. Qi, B. Xu, A neural network framework for relation extraction: learning entity semantic and relation pattern, Knowl.-Based Syst. 114 (2016) 12–23.

[49] L. Wang, Zh. Cao, G. de Melo, Zh. Liu, Relation classification via multi-level attention CNNs, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 2016, pp. 1298–1307.

[50] Y. Zhang, P. Qi, Ch.D. Manning, Graph convolution over pruned dependency trees improves relation extraction, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, pp. 2205–2215.

[51] Ch. Alt, M. Hubner, L. Hennig, Improving relation extraction by pre-trained language representations, Proc. Autom. Knowl. Base Constr. (2019).

[52] H. Leopold, S. Smirnov, J. Mendling, On the refactoring of activity labels in business process models, Inf. Syst. 37 (5) (2012) 443–459.

[53] H. Leopold, E.-S. Rami-Habib, J. Mendling, L. Guerreiro Azevdo, F.A. Baião, Detection of naming convention violations in process models for different languages, Decis. Support Syst. 56 (2013) 310–325.

[54] H. Leopold, F. Pittke, J. Mendling, Ensuring the canonicity of process models, Data Knowl. Eng. 111 (2017) 22–38.

[55] F. Friedrich, J. Mendling, F. Puhlmann, Process model generation from natural language text, in: International Conference on Advanced Information Systems Engineering (CAiSE 2011), 2011, pp. 482–496.

[56] L. Shi, R. Mihalcea, Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing, in: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. CICLing 2005, in: Lecture Notes in Computer Science, vol. 3406, Springer, Berlin, Heidelberg, 2005.

[57] S.P. Overmyer, L. Benoit, R. Owen, Conceptual modeling through linguistic analysis using LIDA, in: Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, 2001, pp. 401–410.

[58] H.M. Harmain, R. Gaizauskas, CM-Builder: A natural language-based CASE tool, J. Autom. Softw. Eng. 10 (2003) (2003) 157–181.

[59] A. Oliveira, N. Seco, P. Gomes, A CBR approach to text to class diagram translation, in: TCBR Workshop at the 8th European Conference on Case-Based Reasoning, Turkey, 2006.

[60] I.S. Bajwa, A. Samad, S. Mumtaz, Object oriented software modeling using NLP based knowledge extraction, Eur. J. Sci. Res. 35 (1) (2009) 22–33.

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

[62] D.K. Deeptimahanti, R. Sanyal, Semi-automatic generation of UML models from natural language requirements, in: Proceedings of the 4th India Conference on Software Engineering (ISEC '11), ACM Press, New York, USA, 2011, pp. 165–174.

[63] T. Karkkainen, M. Nurminen, P. Suominen, T. Pieniluoma, I. Liukko, UCOT: semiautomatic generation of conceptual models from use case descriptions, in: C. Pahl (Ed.), Proceedings of the IASTED International Conference on Software Engineering (SE 2008), 2008, pp. 171–177.

[64] M.G. Georgiades, A.S. Andreou, Formalizing and automating use case model development, Open Softw. Eng. J. 6 (2012) 21–40.

[65] H. Christiansen, C. Have, K. Tveitane, From use cases to UML class diagrams using logic grammars and constraints, in: RANLP'07: Proceedings of Recent Advances in Natural Language Processing, 2007, pp. 128–132.

[66] R. Sharma, P.K. Srivastava, K.K. Biswas, From natural language requirements to UML class diagrams, in: 2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), 2015, pp. 25–32.

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

[68] A. Fatwanto, Software requirements translation from natural language to object-oriented model, in: Proceedings of 2012 IEEE Conference on Control, Systems and Industrial Informatics, ICCSII 2012, 2012, pp. 191–195.

[69] J.C. Goncalves, F.M. Santoro, F.A. Baião, Business process mining from group stories, in: Proceedings of the 13th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2009, pp. 161–166.

[70] A. Sinha, A. Paradkar, Use cases to process specifications in business process modeling notation, in: ICWS 2010-2010 IEEE 8th International Conference on Web Services, 2010, pp. 473–480.

[71] L. Nemuraite, T. Skersys, A. Sukys, E. Sinkevicius, L. Ablonskis, VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML & OCL models, in: International Conference on Information and Software Technologies IT2010, 2010, pp. 377–384.

[72] I.S. Bajwa, M.G. Lee, Transformation rules for translating business rules to OCL constraints, in: ECMFA'11 Proceedings of the 7th European Conference on Modeling Foundations and Applications, Springer Berlin Heidelberg, 2011, pp. 132–143.

[73] M. Bonais, W. Rahayu, E. Pardede, Integrating information systems business rules into a design model, in: 2012 15th International Conference on Network-Based Information Systems, 2012, pp. 104–111.

[74] F. Meziane, N. Athanasakis, S. Ananiadou, Generating Natural Language specifications from UML class diagrams, Requir. Eng. 13 (1) (2007) 1–18.

[75] H. Burden, R. Heldal, Natural language generation from class diagrams, in: Stephan Weißleder, Levi Lúcio, Harald Cichos, Frédéric Fondement (Eds.), Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa), ACM, New York, NY, USA, 2011, 8.

[76] J. Cabot, R. Pau, R. Raventós, From UML/OCL to SBVR specifications: A challenging transformation, Inf. Syst. 35 (4) (2010) 417–440.

[77] S. Malik, I.S. Bajwa, Back to origin: Transformation of business process models to business rules, in: M. La Rosa, P. Soffer (Eds.), BPM 2012 Workshops, Springer-Verlag Berlin Heidelberg, 2012, pp. 611–622.

[78] T. Skersys, R. Butleris, K. Kapocius, T. Vileiniskis, An approach for extracting business vocabularies from business process models, Inf. Technol. Control 42 (2013) 178–190.

[79] H. van der Aa, H. Leopold, H.A. Reijers, Comparing textual descriptions to process models - the automatic detection of inconsistencies, Inf. Syst. 64 (2017) 447–460.

[80] T. Skersys, P. Danenas, R. Butleris, Approach for semi-automatic extraction of business vocabularies and rules from use case diagrams, in: Proceedings of Enterprise Engineering VIII - 4th Enterprise Engineering Working Conference (EEWC 2014), pp. 182–196.

Paulius Danenas is a scientific researcher at the Center of Information Systems Design Technologies, Kaunas University of Technology in Lithuania. He obtained a PhD degree in Informatics from Vilnius University in 2013. His research interests cover various topics in software engineering and model-driven development, as well as artificial intelligence, machine learning, business intelligence and decision support systems (particularly for business and financial domains). He has published several papers in high rated academic journals and in a number of international conferences.

Tomas Skersys is a scientific researcher at the Center of Information Systems Design Technologies and an Associate Professor at the Department of Information Systems (Kaunas University of Technology). His research interests and practical experience cover various aspects of business process management and model-driven information systems development. On these topics, he published several articles in high rated academic journals and in a number of international conferences. He is also a co-editor of three books of international conferences published by Springer Verlag.

Rimantas Butleris is a Director of the Center of Information Systems Design Technologies and a full-time Professor at the Department of Information Systems (Kaunas University of Technology). His main area of research is requirements engineering. In his career, he was a co-author of more than seventy research papers and also participated in more than fifteen national and international research projects; in many of those he was a leading researcher. Up to the year 2013, Rimantas Butleris was a long-time program chair and co-chair of the International Conference on Information and Software Technologies (ICIST).