

Formalising Natural Language Specifications using a Cognitive Linguistic/Configuration Based Approach

Matt Selway^{1,*}, Georg Grossmann, Wolfgang Mayer, Markus Stumptner

*Knowledge and Software Engineering Laboratory, School of Information Technology &
Mathematical Sciences, University of South Australia
University Blvd, Building D Level 1, Room 19, Mawson Lakes, SA, Australia, 5095*

Abstract

This paper addresses the problem of transforming business specifications written in natural language into formal models suitable for use in information systems development. It proposes a method for transforming controlled natural language specifications based on the Semantics of Business Vocabulary and Business Rules standard. This approach is unique in combining techniques from Model-Driven Engineering (MDE), Cognitive Linguistics, and Knowledge-based Configuration, which allows the reliable semantic processing of specifications and integration with existing MDE tools to improve productivity, quality, and time-to-market in software development. The method first learns the vocabulary of the specification from glossary-like definitions then parses the rules of the specification and outputs the resulting formal SBVR model. Both aspects of the method are tested separately, with the system correctly learning 98% of the vocabulary and correctly interpreting 98% of the rules of an SBVR SE based example. Finally, the proposed method is compared to state-of-the-art approaches for creating formal models from natural language specifications, arguing that it meets the criteria necessary to fulfil the three goals of: (1) shifting control of specification to non-technical business experts, (2) reducing the manual effort involved in formalising specifications, and (3) supporting business experts in creating well-formed sets of business vocabularies and rules.

Keywords: Business rules, Business vocabulary, SBVR, Controlled natural language, Natural language processing

*Principal corresponding author

Email addresses: matt.selway@mymail.unisa.edu.au (Matt Selway),
georg.grossmann@unisa.edu.au (Georg Grossmann), wolfgang.mayer@unisa.edu.au
(Wolfgang Mayer), markus.stumptner@unisa.edu.au (Markus Stumptner)

¹Corresponding author Tel.: +61 8 8302 3943

This manuscript has been accepted for publication in Information Systems. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all disclaimers that apply to the journal apply to this manuscript. A definitive version was subsequently published in Information Systems, [VOL#, ISSUE#, (DATE)] doi: 10.1016/j.is.2015.04.003

1. Introduction

The transformation from natural language business specifications into formal models is an ongoing problem in the development of information systems. While it is desirable to have domain experts develop and maintain the business vocabulary and logic themselves, there is a gap between the natural language descriptions preferred by business people (the domain experts) and the formal models required by technical experts for requirements analysis, consistency checking, compliance checking, model transformations, etc. Manual translation and verification of the natural language text, even by technical experts, is time consuming and error-prone. Furthermore, the resultant formal model requires validation by the domain experts who most likely do not know the formal notation and must be trained to use it.

To address these issues and improve the communication of specifications to technical experts, standards such as the Semantics of Business Vocabulary and Business Rules (SBVR) [1] have been proposed. SBVR attempts to reach a middle ground by providing a controlled English (SBVR Structured English) representation for business people and a metamodel, with a basis in formal logic, for technical experts. SBVR represents a shift of focus from a technical perspective of business rules (see [2, 3]) to a business perspective as ‘it is conceptualized optimally for business people ... and is designed to be used for business purposes, independent of information systems designs’ [1, p. 3]. However, when using SBVR, it is still a difficult task to define complete sets of well-formed concepts and rules governing a business [4], so tools that can assist the user in formalising their concepts and rules are a necessity.

Various tools and approaches have been proposed for creating models from natural language specifications. Early work (e.g. [5, 6]) focused on identifying the object-oriented elements (i.e. classes, associations, attributes, and methods) of a software specification given in unrestricted natural language for use in information systems design. More recently, SBVR-based tools (e.g. [7, 8, 9]) have been proposed for the extraction or formalisation of business rules at a higher level. Yet other approaches focus on extracting specific business aspects, such as business processes, from unrestricted texts (e.g. [10]). Each of these approaches have their pros and cons. Approaches for processing unrestricted text are predominantly automated, but produce incomplete or inaccurate models that require extensive manual revision and validation. On the other hand, more formal approaches (including many of those based on SBVR Structured English) often provide supporting tools and editors, but require manual interpretation and translation (usually by a technical expert) of the business specification into the formal notation of the tool.

40 To address this, our aim is to provide a method of semi-automating the translation of business specifications into formal models that fulfils three goals:

1. allow business people to develop, maintain, and validate their business specifications with minimal intervention from technical experts;
2. support business users in *formalising* their business specifications as well-
45 formed sets of business vocabularies and rules; and
3. reduce the amount of manual effort involved in formalising business specifications.

The first goal is particularly important as, according to SBVR, a business rule is ‘a rule that is under business jurisdiction’ [1, p. 8]. This means that business
50 rules can be added, removed, or modified at the discretion of the business [7]. Therefore, the focus of tools needs to shift from allowing technical experts to have business people validate the formal models, to allowing business people to create and maintain formal models themselves.

This leads naturally to the second goal. If we want business people, who are
55 used to specifying things informally, to formalise their specifications themselves, then we need to be able to support them in creating explicit and unambiguous formal models. Consequently, high quality feedback on ambiguities, inconsistencies, and errors is important to enable domain experts to revise the models until they represent the intended meaning.

60 The third goal aims to minimise the manual effort of existing processes as much as possible. Therefore, in achieving the first two goals, existing business specifications should not have to be completely rewritten, new specifications should not have to be written in a highly particular or technical form, and the analysis and translation of the text into formal models should be highly
65 automated.

In [11], we presented an approach to fulfil these goals that focused on the transformation from (controlled) natural language business specifications into their formal representation in SBVR. We used SBVR as the starting point for both the natural language business specifications and the formal representation
70 as it is intended for use by business people and helps to reduce ambiguity, ensure strict semantics, and provide a more direct logical interpretation of the text. We then introduced a novel combination of techniques from Cognitive Linguistics, Knowledge-based Configuration, and Model-Driven Engineering (MDE) to iteratively build a formal SBVR model of the vocabulary and rules
75 contained within the business specification. Finally, we argued that this approach combines the advantages of alternative approaches and meets specific criteria (such as naturalness, expressiveness, parsing completeness, etc.) required to achieve the goals mentioned above.

In this paper we revise and extend the work presented in [11]. We have
80 introduced a clearer overview of approaches to creating models from textual
specifications in Section 3, including additional approaches. Furthermore, we
have expanded the description of the approach (Section 5) with additional
examples and incorporated multiple new heuristics that help to optimise the
parsing process. Finally, a new evaluation section has been included (Section 6)
85 that presents experimental results on the application of our prototype for SBVR
to the EU-Rent case study discussed in the motivating example.

The contributions of our approach include the following:

1. Flexible syntactic analysis to allow the processing of less restricted text
than many controlled natural languages;
90 2. Direct integration of semantic analysis into the parsing process enabling:
 - (a) detailed feedback on errors and inconsistencies in the specification,
 - (b) non-technical users to revise and formalise the specification themselves,
 - and (c) the identification of missing or incorrect vocabulary and the
suggestion of corrections during processing;
95 3. Specification of formal vocabulary in a glossary-like format without the
need for technical notations.

The remainder of the paper is organised as follows: Section 2 introduces a
motivating example that is based on the EU-Rent case study and used throughout
the remainder of the paper; Section 3 provides an overview of existing techniques
100 for generating models from natural language specifications; Section 4 briefly
introduces SBVR, Knowledge-based Configuration, and Cognitive Grammar;
Section 5 describes our approach to parsing natural language business specifica-
tions; Section 6 presents an evaluation of the proposed approach on the EU-Rent
example included in the SBVR specification; and Section 7 discusses directions
105 for future work and concludes the paper.

2. Motivating Example

Business specifications are documents detailing the terms and requirements
of the business in the form of business rules [2]. While we assume that an
organisation already has some existing documentation or specification, it is not a
110 requirement. The process is equally applicable to an organisation that wants to
formalise their business specification from scratch. Moreover, while we focus on
business users, our approach does not preclude a collaborative or agile approach
involving multiple stakeholders. It is likely that the process will still require a
business analyst or requirements engineer who can ‘ask the right questions’ to
115 elicit requirements that the business users take for granted (or are otherwise

EU-Rent is a car rental company owned by EU-Corporation. It is one of three organisation units—the other two being hotels and an airline—that each has its own business and IT systems, but with a shared customer base. Many of the car rental customers also fly with EU-Fly and stay at EU-Stay hotels.

EU-Rent has 1000 branches in towns in several countries. At each branch cars are available for rental to customers. Each branch is part of a local area responsible for managing their respective branches. The local area managers form part of a company operating out of a specific country.

The company operating out of Canada is EU-Rent CA.

Figure 1: EU-Rent case study overview based on [2]

EU-Rent Business Rules

Structural Rules

Each branch is part of a local area.

Each local area is included in an operating company.

Each operating company operates out of a specific country.

...

Figure 2: Semi-formalised EU-Rent business rules

left implicit). Moreover, since our approach works by incrementally building and checking the formal model, it fits naturally into an iterative process (such as the Agile Business Rules Approach [3]) where the vocabulary and rules are incrementally defined, refined and extended in scope.

120 In the remainder of the paper we consider the EU-Rent case study, developed by the Business Rules Group, of a fictitious car rental company with global presence [2, Annex D]. Figure 1, describes aspects of EU-Rent’s business structure and locations.

125 The overview in Figure 1 is an informal summary. In contrast, it is expected that business rules are documented in a more structured fashion: for example, by dividing the document into sections and separating individual rules. Such semi-formalisation can regularly be seen in legal, policy, guideline, and technical documents in which sections, lists, etc. are used to structure the information. This document structure can be used to help process the business specification. 130 Figure 2 illustrates a semi-formal specification for the EU-Rent case study.

A glossary of terms containing the definitions of the business vocabulary is also helpful in processing the specification. Although [2] does not include a glossary for the EU-Rent case study, glossaries are a common element of business and technical documentation, and an important aspect of formalising a business specification (as evidenced by its incorporation into the SBVR standard). A 135 glossary to accompany the EU-Rent extract may appear similar to that displayed

Glossary of Terms

branch an organisational unit responsible for renting cars to customers

local area an organisational unit responsible for managing a group of branches

operating company

an EU-Rent member company that performs EU-Rent business in a specific country

Figure 3: Semi-formalised EU-Rent business vocabulary

in Figure 3.

The goal, then, is to translate these sets of natural language rules and their accompanying glossary (i.e. the business specification) into formal models. The models can then be used to support the development of information systems and the improvement of the specifications themselves.

3. Related Work

In this section we provide an overview of approaches to transforming natural language specifications into formal models. We have identified three main categories of approaches: (1) those that perform a complete parse of a specification using a formal grammar, (2) those that process unrestricted texts using Information Extraction (IE) techniques, and (3) those that perform a complete parse of the sentence without necessarily using formal grammars. Most approaches fall into the first and second categories, while only a few fall into the third (including our own).

3.1. Formalist Approaches

The first category is characterised by parsing controlled natural language using a formalist approach, in which a formal language is simplified to make it more accessible to non-experts [12]. Formalist approaches provide complete parsing of a specification with respect to their formal grammar, which results in reduced naturalness of the text. Due to the strict requirements of the grammar and its basis in a formal language, greater manual effort is necessary to translate specifications into the language and the user must have more technical knowledge of the underlying formalism.

In an attempt to overcome the knowledge and effort requirements of the formalism, Formalist approaches are often tool-driven. They provide editors that embed parsers generated from their formal grammar (see [7, 13, 4, 14, 9]) or graphical user interfaces that guide the user through the creation of rules (see [15, 16]). One advantage of such approaches is that they often include text prediction or ‘code-completion’ features. Moreover, while they perform

no semantic analysis to parse a sentence, they can often *validate* the produced model either after each rule is entered or at the request of the user (e.g. [7, 4, 9]). In the case of [17], the authors use an OWL reasoner on the transformed model to check its consistency.

170 The more general knowledge representation approaches, PENG Light [18, 19] and Attempto Controlled English (ACE) [20], also provide predictive editors. Where they differ is in the complexity of the language that they process, allowing elements such as anaphoric reference in the text. Moreover, they take some semantic information into account to produce their output formalisms, but
175 leave the complete semantics for further processing and reasoning. As general knowledge representation languages, PENG Light and ACE are not particularly suited to business specifications (even though ACE was initially applied to software specifications, see [21]). In particular, PENG Light lacks expressiveness with respect to modality, while ACE has the characteristic that all sentences
180 are unambiguous even if deemed ambiguous to the user. This characteristic requires the user to learn a specific set of interpretation rules to ensure they are interpreting the sentences in the same way as ACE. Whereas we prefer an approach that identifies the ambiguity and reformulates it into an unambiguous form that is intuitive to the user.

185 Another approach in the formalist category is NL-OOPS [5], which attempted to produce object-oriented models from unrestricted text. It is considered formalist as the underlying natural language processing engine, LOLITA, depended primarily on a very large formal grammar (rather than the Information Extraction approaches of the following section). Where NL-OOPS differs from the other
190 approaches is that it performed a deep semantic analysis of the text (courtesy of LOLITA’s extensive semantic network containing many lexical and semantic relations), from which it extracted the object-oriented elements. However, the approach was never fully evaluated and, along with the underlying NLP engine, is no longer available to the best of our knowledge.

195 3.2. Information Extraction Approaches

The second category encompasses approaches that use IE techniques to process natural language specifications. These approaches utilise rules, patterns, or templates to identify objects and relations in unrestricted textual specifications, usually to produce initial models that can be used as a basis of understanding
200 and refinement (see [6, 22, 23, 24, 25, 10, 26, 27, 28, 29, 30]). While the exact rules vary, they typically recognise common nouns or noun phrases as candidate types, proper nouns as individuals, verbs as possible relations, and adjectives as possible attributes. Moreover, the approaches tend to differ on how they filter the candidates to remove spurious types and relations. Although the exact

205 models produced are varied, many approaches utilise intermediate models from which they produce the final models (see [6, 23, 24, 10, 26]). This is important as the final model, for example UML Class or Activity models do not necessarily capture all of the information.

The advantage of IE approaches is that they perform completely automated
210 analysis of the documents (except [24] and the semi-automatic mode of [6]), so there is no manual translation required on the part of the user. However, the IE approaches typically perform shallow and incomplete parsing using rules that search for syntactic patterns and ignore parts of a sentence to extract elements. As a result, it is difficult to provide in-depth feedback to the user about possible
215 errors, ambiguities, or inconsistencies in the documents or produced models.

To circumvent this, some approaches require that the user provide a domain model (e.g. [31, 8, 29] or glossary (e.g. [26, 27, 32])). In particular, the work of [29] uses a domain model to provide deep analysis; however, it requires increased effort and technical knowledge to provide a complete and consistent domain
220 model in the first place, thereby eliminating the benefits gained by using IE techniques.

One particularly promising approach is that of [23], in which structural and behavioural models are created from a pair of intermediate models: a tabular representation of subject, predicate, object triples and a semantic network of
225 directed relations. While their approach appears to utilise the desired information of rules (e.g. modality, quantification, etc.) in order to identify them, it appears that they are not formalised in the produced models. For example, their Hybrid Activity Diagrams (a cross between UML Activity and Sequence diagrams) appear to include conditions only as simple text. As such it is useful as a simple
230 analysis tool, but not a method of formalising business specifications.

Finally, the major drawback for IE-based approaches is their variable performance. For example, in [6] precision and recall across several simple case study texts ranged between 40-100% for recall and 57-80% for precision, while in [25] ranges of 75-93% recall and 82-93% precision were reported. While such
235 measures are a contentious issue, since there is usually not a single correct model, it highlights the difficulty of acquiring complete and precise models from text. If an approach aims to use the models for further automated transformations, it is perhaps better to ensure that precise and correct models are first created that more directly represent the logical meaning of the text.

240 3.3. *Naturalist Approaches*

The third category contains only a few approaches, of which we consider ours to be one. They are characterised by performing complete parsing while not relying on a typical formal grammar. To various degrees, the approaches of this

category attempt to gain the benefits of both the formalist approaches (precise
245 analysis) and the IE approaches (ability to process less restricted documents).

The approach of [33] produces SBVR models and UML Class diagrams through a series of transformations starting from natural language text. Like our approach, their parsing is performed using configuration; however, they utilise configuration as a more flexible means of generating traditional syntactic
250 parse trees while performing semantic analysis based on the SBVR model as an additional step. In contrast, we use configuration on the domain knowledge stored in the SBVR model itself, effectively integrating the knowledge into the parser to perform more direct semantic analysis of a sentence. Moreover, the method of processing the text into the models for configuration is left unspecified
255 in [33], while we utilise flexible expectation-based parsing for that purpose. Lastly, their approach requires a domain specific lexicon containing detailed linguistic information that goes beyond what a business user should be expected to provide.

The CPL language [12] is similar to ACE and PENG Light in that it is more
260 of a general knowledge representation language. In contrast to those formalist approaches, it surrounds a formalist core with a naturalist shell to gain the benefits of both languages. As a result it performs complete parsing, but can have reduced writing naturalness if the user must resort to using the core language. Like the other knowledge representation languages, it does not support the level
265 of expressiveness required for business specifications. Finally, CPL relies on a pre-defined ontology to correctly interpret sentences, rather than attempting to acquire the vocabulary and relations from the documentation itself.

An approach that is very similar to our own in both motivation and technique is CIRCE [34]. Their approach uses matching rules augmented with semantic tags
270 to process natural language specifications; however, it provides complete parsing in that it reports unmatched portions of text. Furthermore, they aim to provide significant feedback to support the user in developing a complete, correct, and unambiguous specification. While their software environment is more extensive, supporting various output models and analyses to provide feedback, CIRCE
275 requires a domain specific glossary and rules (which they called *definitions*) provided in a technical notation to perform the parsing of natural language. Therefore, it is deemed inappropriate for use by our target audience of business and domain experts.

3.4. Summary

280 In summary, to the best of the authors' knowledge, there is no existing approach to transforming natural language business specifications into formal models that can achieve our aims. The formalist approaches require too much

rework to be done on behalf of the business experts we aim to support. On the other hand, Information Extraction approaches are only suitable for the creation of “first-cut” models that must be manually validated and refined as they could be incomplete, incorrect, or imprecise.

The approach that comes closest to our aims is CIRCE; however, their focus is on assisting technical users in specifying *software* specifications. As a result, CIRCE is not suitable for use by business users since it requires technical knowledge to be used effectively.

4. Preliminaries

In the following, we briefly introduce SBVR, Cognitive Grammar, and Knowledge-based Configuration as the foundations of our approach.

4.1. Semantics of Business Vocabulary and Business Rules

The Semantics of Business Vocabulary and Business Rules [1] is a standard developed by the Object Management Group (OMG) for the documentation of business specifications in natural languages and their exchange between different organisations and software tools. It is intended for use by business people to improve communication between stakeholders and bridge the gap between business people and the IT people (technical experts) who develop information systems for them. Moreover, SBVR is an ideal intermediate model for several reasons: (1) it supports information on both business structure and its operations (specified declaratively rather than the imperative style of an explicit process); (2) it can be used to create or complement other models (such as UML Class Diagrams, BPMN models, etc.) through model transformations; and (3) it partly retains the textual structure of the specification, improving traceability to the original text.

The SBVR specification includes two important aspects: (1) a conceptual model, and (2) a controlled English representation called SBVR Structured English.

4.1.1. SBVR Metamodel

The conceptual model (and associated MOF-based metamodel) of SBVR standardises a set of concepts for the definition of business vocabularies and rules. It constitutes the semantics of a controlled language, allowing the defined business specifications to be interpreted with reduced ambiguity. In particular, SBVR can be interpreted in formal logic; primarily first-order logic with an extension in modal logic (necessity, obligation, permission, possibility), and higher-order logic using Henkin semantics [1]. This basis in formal logic supports various forms of reasoning, consistency and conformance checking.

320 Within the metamodel, ‘Meanings and Representations’ form the basis for
 defining the vocabulary as a set of interrelated **object types**² (e.g. ‘branch’,
 ‘local area’), **individual concepts** (e.g. ‘EU-Rent’, ‘Canada’), and **fact types** (e.g.
 ‘includes’/‘is included in’). The meanings are separated from representations
 325 to allow a single concept to be represented by multiple words (in the same or
 different languages), images, or sounds. The core elements of the ‘Meanings and
 Representations’ are shown in Figure 4; lines with an empty triangle are
subclass-of relationships, while lines with an open arrow are associations between
 concepts.

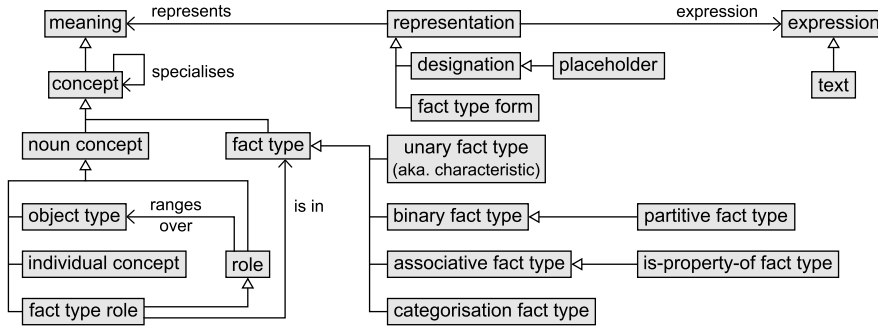


Figure 4: Extract of the SBVR metamodel [1]: Meanings and Representations

330 The ‘Logical Formulations’ aspect of the metamodel forms the semantic
 structure of business rules as a conceptualisation of formal logic. The metamodel
 includes concepts for first-order logical operators (e.g. **conjunction**, **implication**),
 quantification (e.g. **universal quantification**, **existential quantification**), and modal
 operators (e.g. **necessity**, **obligation**). An extract of the ‘Logical Formulations’
 aspect of the metamodel is shown in Figure 5 (the dotted lines indicate that
 335 elements have been omitted).

4.1.2. SBVR Structured English

340 SBVR Structured English (SBVR SE) is a non-normative notation for ex-
 pressing business vocabularies and rules with SBVR semantics. It constitutes
 the syntax of a controlled language that ‘... maps mechanically to SBVR con-
 cepts’ [1, p. 237]; however, it does not eliminate ambiguity. SBVR SE is mainly
 described by restrictions to standard English and, from the linguistic point of
 view, SBVR SE has a number of features, including:

1. verbose expressions with deeply nested clauses and long dependencies
 between linguistic elements;

²We use **sans-serif** font to denote concepts from the SBVR metamodel.

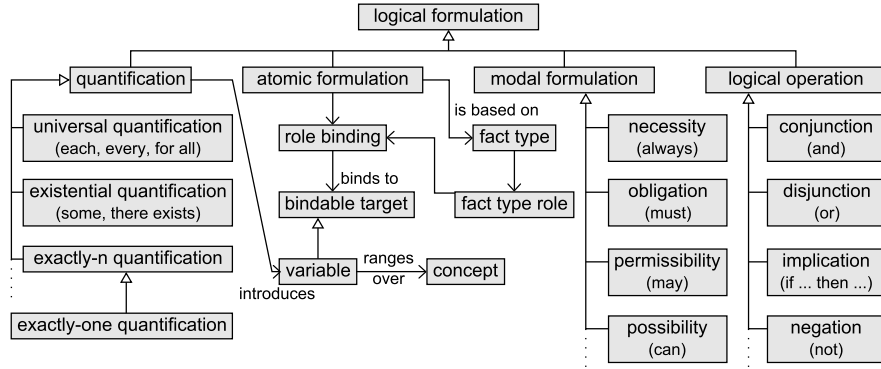


Figure 5: Extract of the SBVR metamodel [1]: Logical Formulations

2. a high degree of polysemy (e.g. ‘local area *includes* branch’ and ‘company *includes* local area’ are considered semantically different relations);
3. negation always has local scope;
4. modalities always have wide scope;
5. quantification scope is restricted, but not deterministic, and must be resolved;
6. anaphoric reference (i.e. forward or backward references to the mention of an object) is limited to within each rule and to typed references (e.g. ‘the branch’ can be used to refer back to a previous mention of ‘branch’, while ‘it’ is not allowed);
7. repeated subjects and verbs can be omitted from logical expressions (e.g. in ‘a branch has a name and [the branch has] a country’ the expression in square brackets is optional).

Many of these features reduce ambiguity; however, other features and their combinations may lead to increased ambiguity or other issues. For example, restrictions to the scope of negations help reduce ambiguity, while the distant dependencies created by deeply nested clauses are problematic in natural language processing. Furthermore, combining a high degree of polysemy (2) with omitted verbs (7) leads to a complicated situation where the verb is considered the same from the point of view of the user, but semantically different from that of the computer.

An SBVR SE specification is separated into sections for vocabulary and rules. The vocabulary section takes the form of a detailed glossary with elements such as notes, examples, synonyms, general concept, concept type, etc.; many of which are optional. A vocabulary entry can also include structural rules directly related to the given term. The rules section consists of individual rule statements,

which can include additional guidance information, notes, etc. These additional elements can help in correctly processing the specification; however, they are not strictly necessary as many are intended as additional information for the people reading the specification, not the computer.

375 SBVR SE uses text styling (bold, italics, colour, etc.) in order to differentiate elements of a business specification that map to different elements of an SBVR model. When performed automatically by the system, this can aid a user in determining whether or not the vocabulary and rules have been correctly interpreted. The following styles are used in this paper to demonstrate the
380 intended SBVR interpretation of statements. Although similar to the styles defined in [1], we forgo the use of colour and more clearly delineate keywords with dotted underlining.

- underlined terms represent the designations of object types
- **Names** represent the designations of individual concepts
- 385 • italicised *verbs* represent the designations of fact types
- keywords represent words forming statements when combined with other words and that typically map to logical formulations, and

We use SBVR SE as a starting point for processing natural language into SBVR models for several reasons: (1) it is intended for use by business users;
390 (2) it has a relatively straightforward mapping to SBVR semantics, described in [1, Annex C]; (3) it is not overly restricted, allowing rules to be expressed quite freely in a number of ways; and (4) it is the notation used by the SBVR specification itself, providing access to a reasonably sized case study in [1, Annex E].

395 4.2. Cognitive Grammar

For our natural language processing approach, we make use of a theory from the field of Cognitive Linguistics called Cognitive Grammar [35]. This theory merges syntax, semantics, and pragmatics into a holistic view of language and provides a uniform treatment of linguistic elements that are traditionally treated
400 separately. As a result, traditional syntactic elements (e.g. the part-of-speech categories noun, verb, adjective, etc.) and grammar rules emerge from semantics as regions of a multi-dimensional continuum.

The emergence of traditional syntactic elements from meaning provides the advantage of making semantic analysis the primary aspect of the process, while
405 syntactic analysis is reduced to word-order information evoking possible meanings. Therefore, in Cognitive Grammar-based language processing traditional syntax is

taken into account (directly or indirectly) in parallel with other disambiguation tasks. This allows us to avoid the problem of propagating errors from, for example, part-of-speech tagging (a traditional syntactic analysis step) throughout further analysis [36]. Finally, this approach allows the definition of vocabulary to be simplified for domain experts as complex linguistic information is not required.

4.3. Knowledge-based Configuration

Knowledge-based Configuration is a constraint-based search method traditionally used to compose a customised system from generic components [37]. A configurator requires two things: (1) a description of the problem and its constraints, i.e. the configuration model; and (2) user preferences and constraints on the desired configuration, i.e. the configuration goal. Using this information the configurator can then produce one or more configurations that satisfy the constraints, or a description of why one cannot be found. As such, configuration is a natural fit for Cognitive Grammar and searching for the meaning(s) of natural language sentences, which allows the embedding of knowledge in the parser and building on it during processing.

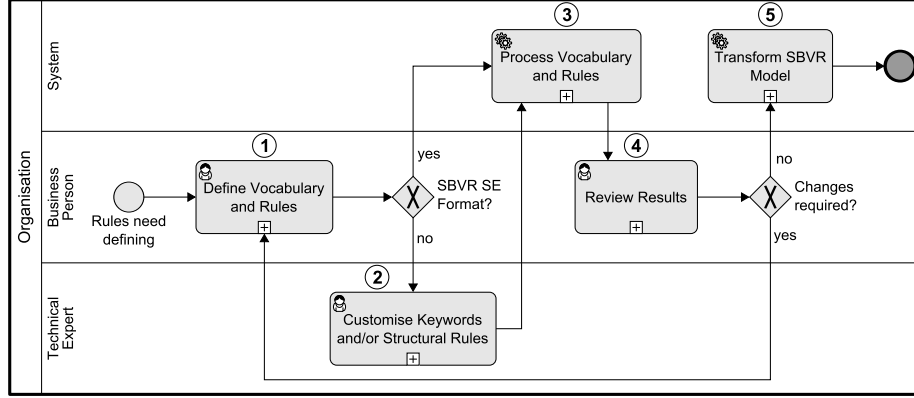
In the context of MDE, configuration can be seen as an advanced model transformation that requires searching for the target model, as opposed to deterministically generating it [33]. This technique, called *model search* in [38], utilises two versions of the metamodel; the original and a metamodel where all of the constraints have been relaxed (i.e. minimum cardinalities are reduced to zero, predicates are removed, etc.). The model transformation is then the process of searching for a terminal model that satisfies the constraints of the constrained metamodel based on an input, relaxed terminal model. In this case the description of the problem and its constraints is provided by the metamodel(s), while the configuration goal is provided by a partial configuration that is the relaxed terminal model.

5. Cognitive Linguistic Configuration

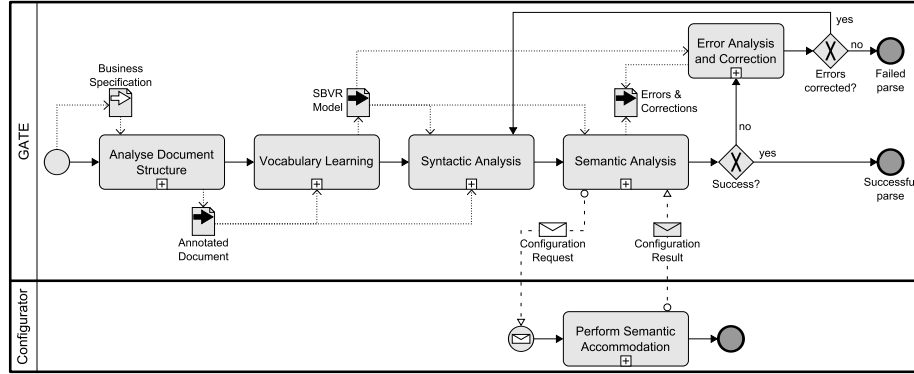
In this section we describe the process and architecture of Cognitive Linguistic Configuration as it is realised in our prototype tool CLUE. We first provide an overview of our approach, followed by a more detailed description of the different stages of the analysis.

5.1. Process Overview

A view of our process as part of a possible organisational workflow is summarised in Figure 6. The workflow is shown in Figure 6a, while the translation



(a) Overall workflow



(b) Details of subprocess (3) 'Process Vocabulary and Rules'

Figure 6: A possible workflow using our approach and details of the parsing process

process of CLUE is detailed in Figure 6b. The activities are numbered for reference purposes.

The process starts whenever the organisation has some rules they wish to specify; this could be due to changes to policies or the business environment, or due to wanting to formalise their rules for the first time. A business person then goes about defining (or updating) the business vocabulary and rules of the organisation (1). We do not require the use of SBVR SE structure, rather, we allow the use of the organisation's preferred style and format. This helps to minimise the manual work required in formalising a business specification. If the document format varies too widely, however, a technical expert³ may be

³While we show the technical expert as part of the organisation, this is not necessarily the case.

required to assist in customising the rules that identify the different elements of the document (2).

Once formatting issues are dealt with, the vocabulary and rules are processed
455 by our system (3), which is expanded in Figure 6b. First, the different parts of the business specification are identified using tokenisation, sentence splitting, and rules for identifying structural elements of the document (e.g. vocabulary and rule entries). The tokenisation and sentence splitting are performed using standard components of GATE (General Architecture for Text Engineering) [39],
460 while the customised rules identify the document structure.

Next, a component of our prototype (realised as a GATE plugin) creates a preliminary SBVR model by processing the vocabulary entries of the specification. This SBVR model allows the syntactic and semantic processing of the rules to occur and is incrementally revised as rules are parsed. Our GATE plugin then
465 performs syntactic analysis of the document using the expectation-based parsing approach described in Section 5.3.1.

Following the syntactic analysis process, the semantic analysis sub-process takes the syntactic analysis and partial SBVR model and calls the configurator, a Smalltalk implementation of a “generative constraint satisfaction” solver [40],
470 to perform model search. If successful, the result is an interpretation of the rule(s) as a valid SBVR model. However, if the configuration fails, the errors and inconsistencies are analysed for any that can be resolved automatically. If possible, the errors are corrected and the specification is reprocessed to ensure consistency.

475 After processing the specification, the results (including any errors that could not be automatically resolved and any corrections that were made) are presented to the user for review; stage (4) of Figure 6a. The user can then make changes if necessary and reprocess the document, returning to (1), which updates the generated SBVR model and provides further feedback. This cycle will continue
480 until no more changes need to be made.

After the business user has resolved any issues with the specification, the SBVR model will be provided to the technical experts to support the development of an information system (5). It can then undergo automated model transformations or otherwise be used to develop Platform Independent Models
485 (PIMs) in accordance with the Model-Driven Architecture [41]. For example, model transformations could be used to create UML models, rules for execution on a rules engine, etc.

5.2. Vocabulary Learning

The “Vocabulary Learning” process creates a preliminary SBVR model and
490 lexicon from the glossary. Like a standard glossary, a basic vocabulary entry

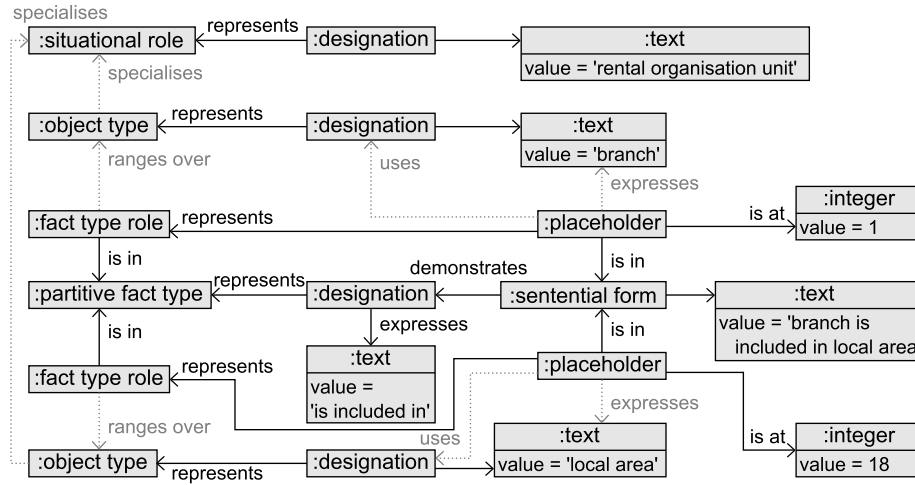


Figure 7: SBVR model of the example vocabulary. Grey dotted arrows indicate relations between different vocabulary entries.

includes the word (or phrase) and its definition. The initial lexicon is acquired as follows:

1. **Add temporary lexical entries:** For each vocabulary entry identified in the document a temporary lexical entry is created without any associated semantics.
2. **Disambiguate concept types:** For each temporary lexical entry, it is determined which main category of SBVR concept type it is most likely to be, i.e. object type, individual concept, or fact type (or one of their subtypes). The heuristics used to make these determinations and the potential issues involved in using them are discussed below.
3. **Create the semantic representations:** Templates for SBVR concept types are then used to create the model elements for each concept. The separate vocabulary entries are also linked appropriately, for example, the roles of fact types are connected to the object types that they *range over*. Figure 7 shows an example of vocabulary defined as an SBVR model.
4. **Add mappings to the lexicon:** Each temporary lexical entry is then linked to its semantic representation. Synonyms are mapped to the same semantic representation.

A number of features of the lexical entries are taken into account when disambiguating which concept type(s) apply to them, including:

- capitalisation

- the number of placeholders/roles identified
- the positions of any identified placeholders in the expression
- any explicitly specified concept types
- 515 • any general concept; either specified as part of a definition or defined using an explicit field of the entry
- whether or not the entry has been specified as the concept type for another entry (this occurs when categorising vocabulary entries)

Differentiating between object types and individual concepts is primarily
 520 performed through the use of capitalisation consistent with normal English. Ambiguity may arise if the name of an individual concept is included in the term of an object type. For example, ‘EU-Rent operating company’ may be considered an individual concept, as it begins with a capital, when it is actually an object type representing an EU-Rent owned company operating in a particular country. This
 525 ambiguity can be resolved using other fields, if present. For example, the type could be disambiguated through a synonym that is clearly an object type, such as ‘operating company’ for ‘EU-Rent operating company’. Finally, subsequent parsing of the definitions and rules may identify an inconsistency in the term’s usage and update the lexical entry accordingly. For example, the definition of
 530 ‘EU-Rent CA’ is ‘the EU-Rent operating company that is located in Canada’, which only makes sense if ‘EU-Rent operating company’ is interpreted as an object type.

To distinguish fact types, the placeholders (i.e. terms for object types) included in them are identified. For example, the fact type ‘branch is included in local
 535 area’ contains two placeholders: ‘branch’ and ‘local area’. A fact type can include one, two, or more placeholders for characteristics (i.e. unary fact types), binary fact types, and other fact types, respectively.

This method may introduce an ambiguity between characteristics and object types where a prefix matches another term. For example, the object type ‘EU-Rent
 540 operating company’ could be identified as the characteristic ‘EU-Rent operating company’. Conversely, if a term has been omitted from the vocabulary, a fact type could be identified as an object type or as a fact type with one less placeholder. In these situations, other fields may clarify the intended type of the vocabulary entry. Otherwise, the processing of the definition and/or rules may identify
 545 inconsistencies in its usage and update the lexical entry accordingly. For example, if ‘local area’ were not included in the vocabulary, ‘branch is included in local area’ would be identified as a characteristic. This would lead to an inconsistency with the rule ‘Each branch is included in exactly one local area’, where the

‘exactly one’ causes a separation in the **characteristic** and indicates that ‘local
550 area’ should be defined as an **object type**.

This approach does not depend on any external lexical resources, which provides two benefits: (1) the process focuses on the intended, domain specific meaning of terms rather than the potentially dozens of meanings in a general purpose resource, and (2) domain specific terms that do not exist in the general
555 resources are covered. The limitation, though, is that **fact types** must be expressed in the vocabulary using the placeholder form (e.g. ‘term *verb* term’) as opposed to only the verb. However, this is considered beneficial in the development of a formal vocabulary and is simpler than other lexical approaches that require complex linguistic information to be specified in the lexicon.

560 5.3. Rule Parsing

Once the lexicon has been populated, the definition statements and rules can be parsed. Our parsing process is based on Holmqvist’s computational model of Cognitive Grammar [42]. One aspect of this approach is a word-order only based syntactic analysis that we refer to as *expectation-based parsing*. *Grammatical*
565 *expectations* (or simply *expectations*) of lexical entries are used to propose parses of a sentence. The expectations are paired with elaboration sites of semantic structures; that is, locations in the structure where additional information can be attached. Therefore, when applied to SBVR, **object types**, **fact types**, and the other elements of the SBVR metamodel are the semantic structures; **fact type**
570 **roles** represent the elaboration sites; and the **placeholders** associated with the **roles** represent expectations.

Expectation-based parsing is performed iteratively with semantic accommodation, which combines the semantic aspects of the suggested parses into a complete structure using model search. As such, the parsing process is non-
575 deterministic, which is important when dealing with controlled languages that allow the ambiguity of multiple interpretations for a sentence, or for natural language in general. The result of parsing a specification is a progressively more detailed SBVR model containing the concepts, their definitions, and associated rules.

580 5.3.1. Syntactic Analysis

The syntactic analysis is performed incrementally using the **placeholders** of lexical entries as expectations to identify word combinations in a sentence and propose possible parses (termed *suggestions*). In [42], only *left* and *right* expectations were introduced, which search *backward* or *forward* for another
585 expression to fill it, respectively. However, since SBVR fact types can have any number of **roles/placeholders**, we introduced *internal* expectations that search

within the span of the expression. This allows a more natural handling of **fact** types with **placeholders** that appear between words as well as some keywords.

When an expectation suggests a combination with another suggestion (a single
 590 word or a larger composition that was proposed previously), it is said to *catch*
 the suggestion. This catching information constitutes a parse tree, albeit not a
 traditional one. Figure 8 shows an example parse tree produced by our syntactic
 analysis (8a) compared to two (simplified) traditional syntactic parse trees: a
 constituent tree (8b), and a dependency tree (8c). Like a constituency tree, an
 595 expectation-based tree shows how words and phrases combine to form larger
 phrases, but like a dependency tree it more directly represents the relationships
 between words. Unlike a constituency tree, the constituents are the words and
 phrases themselves⁴ rather than an abstract representation based on parts-of-
 speech. In contrast to dependency trees, the relationships between words are
 600 of a single type (catches) with the actual relationships determined by semantic
 analysis, which may consist of more than just grammatical relations. While
 parts-of-speech and grammatical relations can be represented in the semantics,
 this information is not required for the construction of expectation-based parse
 trees.

The suggested parses are kept track of in a suggestion list. Since the number
 605 of possible word combinations is exponential in the length of the sentence, the
 suggestion list is kept small by ordering it using several heuristics to identify
 the *best* suggestion and pruning off any suggestions over a maximum limit or
 that are not considered good enough [42]. The initial metrics used by the
 610 ordering heuristics include: (1) catching distance, the linear distance to the
 word/suggestion caught by an expectation; (2) binding energy, the summation
 of all catching distances of a suggestion; (3) local coverage, the ratio of words in
 the suggestion to words spanned by the suggestion; and (4) global coverage, the
 ratio of words in the suggestion to all words of the expression encountered up to
 615 the current point.

We extend these heuristics with two more lexical heuristics: (1) catch weight,
 and (2) caught weight. These weights are functions associated with lexical entries
 and, when a catch occurs, are evaluated by the lexical entry of the suggestion that
 catches another (catch weight) and by the lexical entry of the suggestion being
 620 caught (caught weight). Due to the semantics of SBVR, certain combinations
 are more likely and others should not occur; therefore, the catch and caught
 weights can promote or demote a suggestion and improve the ability for the
 correct interpretation to be found early.

⁴The ellipses in the expectation-based parse tree represent omission of the caught expression in the larger phrase and are used for brevity.

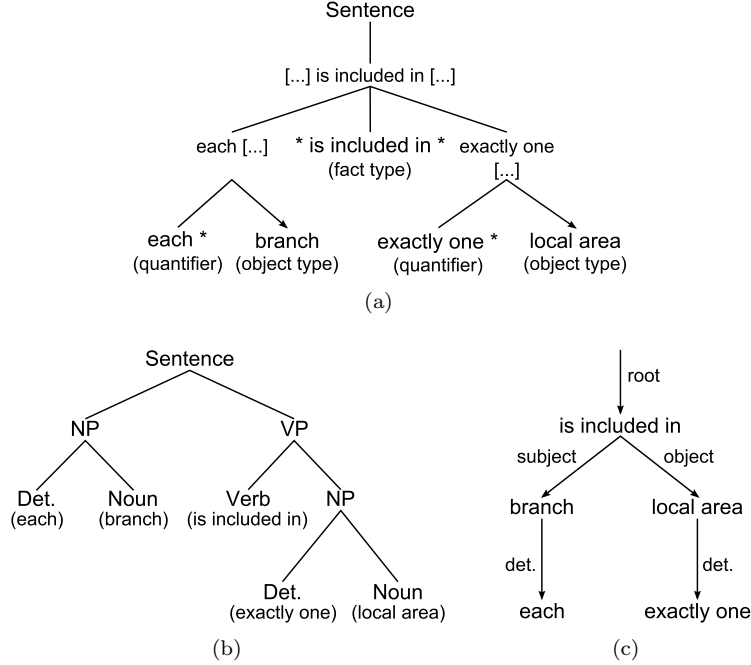


Figure 8: Comparison of an expectation-based parse tree (a), a traditional constituency tree (b), and a dependency tree (c).

Another extension we introduced is the explicit specification of *required* and *optional* expectations. If an expectation is required, a suggestion with that expectation cannot be caught unless the expectation is filled (i.e. catches another suggestion), while optional expectations do not have that restriction. Holmqvist [42] discusses the topic of optionality of expectations in relation to completely natural language, which leads to an indeterminate conclusion. However, in the context of controlled languages, specifying required and optional expectations is practically useful to help limit the size of the suggestion list. For SBVR, the expectations of keywords are required, while those of fact types are optional. This has the practical benefit of allowing ellipsis and incomplete statements to be parsed while not allowing, for example, ‘is included in’ to catch ‘exactly one’ before ‘exactly one’ catches ‘local area’ (in the rule ‘Each branch is included in exactly one local area’).

Whether or not a suggestion is pruned from the suggestion list is determined by the *pruning strategy*. This takes into account information such as the metrics used in sorting the suggestion list, the distance from a suggestion to the current word being processed, the length of the suggestion list, and whether or not the

suggestion has been successfully accommodated. The pruning strategy currently in use is based on the suggestions made by Holmqvist [42] with extensions to our catch and caught weight functions and to the concept of *important* suggestions. A suggestion is considered important if it is large (which we define as consisting
645 of six or more elements) and has unfilled right expectations. This is necessary to cope with rules consisting of a large number of nested restrictions since, without it, suggestions early on may be pruned before they can catch the complete nested structure. The pruning strategy is as follows:

- Suggestions that fail semantic accommodation are pruned.
- 650 • Unimportant suggestions with a global coverage < 0.5 and distance to the current word $> 3-9$ (scales up as the length of the sentence increases) are pruned.
- Suggestions whose catch or caught weight function results in a value < 0.1 are pruned.

655 This approach provides a flexible means of performing syntactic analysis that is generic with respect to the semantics. Since it is driven purely by word order and the grammatical expectations of the known lexical entries, it can handle a variety of inputs including fragments of text that are not complete sentences or phrases. Whether or not any meaning can be derived from such fragments is left
660 to the semantic analysis. Due to the incremental nature of the analysis, such fragments are routinely provided with (partial) meaning that is combined with other suggestions in later iterations.

5.3.2. Semantic Analysis

The suggestions derived by the syntactic analysis are forwarded to the
665 semantic accommodation process. This process determines whether or not a suggestion is admissible in the SBVR semantics. The parse tree and the partial SBVR model for each suggestion is provided to the semantic analysis. The SBVR model contains the evoked semantic structures, while the parse tree adds constraints to reduce the search space.

670 Rather than the numerous processes for accommodation discussed in [42], we utilise knowledge-based configuration as model search, which subsumes many of the disambiguation processes required for accommodation such as word-sense disambiguation and co-reference resolution. Using configuration for parsing has been shown to be an effective and flexible technique by [43] and [33], in which
675 traditional parse trees were generated by the configuration of property grammars. However, we go further by performing configuration on the SBVR model directly, which ensures that parsed sentences are semantically sound based on the existing

domain knowledge, rather than just syntactically or grammatically correct. For example, the configuration of rules using the fact type ‘branch *has* country’ is only successful if branches (including more specific types of branches, e.g. receiving branches) and countries are referred to in the appropriate roles.

In our approach, the SBVR metamodel acts as the configuration model and the configuration goal is provided by the partial SBVR models and parse trees produced by the expectation-based syntactic analysis. The configurator can then search for and generate new model elements and relationships as necessary to complete the transformation from the text to the SBVR model. For example, the best suggested parse of the rule ‘Each branch *is included in* exactly one local area’ evokes the vocabulary elements as follows:

branch & local area

object types are evoked directly

branch is included in local area

creates an atomic formulation based on the fact type, with role bindings for the roles branch and local area

each

creates a universal quantification and its variable

exactly one

creates an exactly-one quantification, the variable that it introduces, and the mandatory *cardinality* 1

The partial SBVR model created by the evocation of the vocabulary is displayed in Figure 9. For brevity, only the relevant elements of the vocabulary are included. The constraints provided by the parse tree restrict the search space for the configurator and allows it arrive at the desired configuration(s). The complete configuration for the example is shown in Figure 10.

5.4. Error Analysis and Correction

So far we have only discussed successful configuration, which results in incrementally building a complete and (internally) consistent semantic representation of a business specification as each rule is processed. However, if the configuration fails then the suggested parse is considered either *ungrammatical* in terms of SBVR’s semantics or inconsistent with the current domain knowledge contained in the model. If all suggested parses fail then feedback is provided to the user as to what caused the error or inconsistency. Errors may occur for one of three reasons: (1) there is an error in the rule or definition, for example, an incorrect term has been used in the role of a fact type; (2) there is an error in the lexicon

for the erroneous rule ‘Each rental organisation unit *is included in* exactly one local area’, the configuration of the model with relaxed parse tree constraints would suggest that ‘rental organisation unit’ be changed to ‘branch’ instead (due to the fact type ‘branch *is included in* local area’). Since not all of the parse tree constraints would be relaxed ‘local area’ would not be suggested because there is no fact type ‘local area *is included in* local area’.

Types of errors that might occur in the learnt lexicon have been discussed in Section 5.2, many of which can be identified or corrected by using a combination of configuration and analysis of the parse tree. By relaxing the parse tree constraints and adding constraints to prevent connections to existing vocabulary elements, a semantic structure can be generated that will suggest what type of lexical entry should occur as part of the rule. For example, if the object type ‘local area’ were incorrectly learnt as an individual concept, the rule ‘Each local area *is included in* exactly one operating company’ would not be configured correctly. However, by removing the constraints associated with ‘local area’ and preventing connections to existing vocabulary, we can use the configurator to generate a possible solution including an object type appropriately connected to other elements of the SBVR model. This “new” vocabulary element could then be suggested to the user as a correction to ‘local area’.

Analysis of the parse trees and suggestion list itself may also be necessary to identify and correct errors. If, for example, the fact type ‘*is included in*’ has been learnt with the incorrect number of roles and placeholders, then the suggestion list for the rule ‘Each local area *is included in* exactly one operating company’ could be analysed to help determine where the placeholders should be. In this rule, the keywords quantify the object types that are the placeholders for the fact type and clearly delineate the second placeholder. Therefore, if the second role were not present then the suggestion list would appear as in Figure 11, where there is no suggestion that covers the entire sentence. Instead there are two “large”, disjoint suggestions. This indicates that there should be a second fact type role and associated object type for ‘operating company’. If, on the other hand, ‘local area’ were omitted from the vocabulary then it would be uncertain as to whether the placeholder between ‘Each’ and ‘exactly one’ is ‘local’, ‘local area’, ‘local area is’, or ‘local area is included’. In that case, automatic correction is unlikely; however, the user can still be given advice that a role or object type may be missing. Any changes or suggested corrections are marked and presented for the user to review.

Finally, a definition may be specified informally, while rules should always be specified formally. Therefore, an informal rule indicates missing vocabulary and we can automatically attempt to create lexical entries for unknown words using the configurator as discussed above. However, since definitions can be

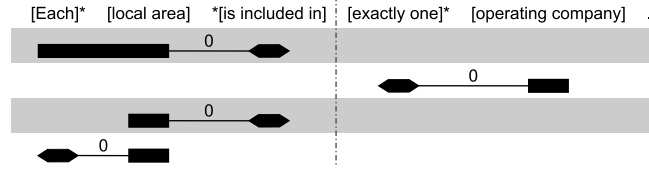


Figure 11: Example suggestion list with disjoint suggestions

intentionally informal we do not automatically generate lexical entries. Instead, informal definitions can be presented to the user for review, with the option to mark them as intentionally informal or to create the lexical entry for them (if the suggested lexical entry is correct).

765 This capacity to provide detailed feedback to the user on errors and inconsistencies is an important feature of our approach, which is not present in many other approaches. It is critical for the formalisation of specifications by business users as technical modelling knowledge is not required. Moreover, it supports the revision of less restricted language into a more restricted, less ambiguous, controlled form: SBVR SE in this case.

6. Evaluation

In this section we present an experimental analysis of our process on part of the EU-Rent case study extracted from the SBVR specification [1, Annex E]. The analysis investigates several aspects. First, we evaluate the accuracy of the vocabulary learning process. Secondly, we evaluate the parsing accuracy on the structural rules of the included sections. This is followed by execution time experiments on a pair of prototypical rules, for the configuration separately and the process as a whole. Finally, we perform an analysis of the errors in the output.

780 Of the EU-Rent case study we used the first 6 sections of vocabulary (out of 11) for the experiments described here. This decision was based primarily on the time taken to create the gold standard annotations, which is an ongoing process. However, the sections used for these experiments demonstrate a variety of phenomena occurring in SBVR-based specifications, including: rules of varying length and complexity; a number of different logical operations (e.g. ‘and’, ‘if and only if’); and higher-order logic through the categorisation of concepts by others. The experiments are performed on each section individually and then as a combined document. This provides some indication of the performance on business specifications of varying sizes and level of ambiguity. The documents are provided in plain-text, i.e. no markup and no special formatting; however,

the SBVR SE structure is preserved. Each test is run without any automatic correction of the vocabulary or rules in order to get a baseline of CLUE for SBVR. Future work will perform experiments on error handling and correction of the specification.

795 In the evaluation we use the standard measures of precision, recall, and F1-score (the harmonic mean of precision and recall) to determine the accuracy of our approach. These measures are calculated by comparing the annotations of the document produced by CLUE against manually created gold standard annotations.

800 The gold annotations were manually created by a single person based on the EU-Rent case study, where the text styling indicates the correct interpretation of a term. In the syntactic analysis, multiple correct parses are possible, which may lead to different, but equally correct, semantic compositions. In these cases, a single candidate was chosen for the gold standard that reflected the desired
805 properties: for example, certain parses are expected to be preferred due to the catch/caught weights of keywords. In certain cases, SBVR permits the existence of different models representing the same meaning, in these cases only one was chosen for inclusion in the gold standard.

810 The comparison is performed automatically by the evaluation framework of the GATE tool. Precision (P), recall (R), and F1 are defined as follows:

$$P = \frac{n_c}{n_p}, \quad R = \frac{n_c}{n_g}, \quad F1 = \frac{2n_c}{n_p + n_g},$$

where n_c is the number of correct annotations, n_p is the number of produced annotations, and n_g is the number of annotations in the gold standard. Partially correct annotations are considered incorrect.

6.1. Vocabulary Learning

815 The results of the vocabulary learning are summarised in Table 1 and show the number of gold standard annotations, precision, recall, and F1-score of each of the major categories (**object type**, **individual concept**, and **fact type**) as well as the total number of concepts, precision, recall, and F1-score.

There are exactly 200 terms in the vocabulary of the sections processed;
820 almost half of which are **object types**, approximately one third are **fact types**, with the remainder **individual concepts**. The overall accuracy (F1-score) of 98%, with the lowest scoring category (**fact types**) at 96%, is excellent considering verbs (which roughly correspond to **fact types**) are typically more difficult to learn than nouns (i.e. **object types** and **individual concepts**). The high accuracy of
825 **fact types** is primarily due to the explicit method of representing them in SBVR by including the terms they relate, rather than the verb alone. This highlights

Table 1: Vocabulary Learning Results

category/section	2.2.1.X	1	2	3	4	5	6	1-6
object types	#	14	10	20	11	20	19	94
	P	1.00	1.00	1.00	1.00	0.95	0.86	0.99
	R	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	F1	1.00	1.00	1.00	1.00	0.97	0.93	0.99
individual concepts	#	2	0	17	14	0	4	37
	P	1.00	N/A	1.00	1.00	N/A	1.00	1.00
	R	1.00	N/A	1.00	1.00	N/A	1.00	1.00
	F1	1.00	N/A	1.00	1.00	N/A	1.00	1.00
fact types	#	8	8	25	7	14	7	69
	P	1.00	0.75	1.00	0.86	0.86	1.00	0.97
	R	1.00	0.75	1.00	0.86	0.80	0.57	0.96
	F1	1.00	0.75	1.00	0.86	0.83	0.73	0.96
total	#	24	18	62	32	34	30	200
	P	1.00	0.89	1.00	0.97	0.91	0.90	0.98
	R	1.00	0.89	1.00	0.97	0.91	0.90	0.98
	F1	1.00	0.89	1.00	0.97	0.91	0.90	0.98

the benefits of taking such an explicit approach to glossary definitions; however, future work aims to achieve equivalent or better results while relieving the user from the burden of such strict definitions. The lower accuracies of individual sections are due to referencing vocabulary terms defined in other sections, which is rectified once they are included in the combined sections.

Overall, there are only three incorrectly learnt vocabulary entries, which are all due to missing vocabulary. Although this missing vocabulary is defined in the currently unprocessed sections, one of the advantages of our approach is that it has the ability to identify missing vocabulary and suggest suitable vocabulary entries during the processing of definitions and rules, as discussed in Section 5.4. Therefore, if some vocabulary is completely left out of the glossary, it does not result in a complete failure of the parsing; the two aspects work together to improve the completeness and correctness of the specification.

6.2. Rule Parsing

There are two important aspects of rule parsing that need to be taken into account for the approach to be able to meet our goals. The first is parsing accuracy, since we aim to achieve complete and precise interpretation of the specification. The second is execution time; since the process is viewed as semi-automated, the parser must not take too long to parse an individual rule or specification.

The parsing accuracy results are summarised in Table 2, broken down by category of annotation. The ‘syntax tree’ category denotes the annotations that define the catching relations or parse tree of each rule, while the remaining categories denote semantic information that identifies the correct sense of each word as a reference to an **object type**, **individual concept**⁵, **fact type**, or keyword. Only the intended interpretation is included in the gold standard as annotations. The table shows the precision, recall, and F1-score for each category and section, as well as the combined result. In addition, the table displays the number of rules and the number of expected annotations for each category.

Table 2: Parsing Accuracy Results

category/section 2.2.1.X		1	2	3	4	5	6	1-6
rules	#	13	8	15	8	16	20	80
object types	#	17	36	24	11	75	43	206
	P	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	R	1.00	1.00	1.00	1.00	0.97	1.00	0.99
	F1	1.00	1.00	1.00	1.00	0.99	1.00	0.99
individual concepts	#	17	0	20	10	1	26	74
	P	1.00	N/A	1.00	1.00	1.00	1.00	1.00
	R	1.00	N/A	1.00	1.00	0.00	1.00	0.99
	F1	1.00	N/A	1.00	1.00	0.00	1.00	0.99
fact types	#	16	23	19	10	52	33	153
	P	1.00	1.00	0.90	1.00	1.00	1.00	0.99
	R	1.00	1.00	1.00	1.00	0.96	1.00	0.99
	F1	1.00	1.00	0.95	1.00	0.98	1.00	0.99
keywords	#	23	51	32	15	111	62	294
	P	1.00	0.84	1.00	1.00	0.86	1.00	0.91
	R	1.00	1.00	1.00	1.00	0.97	1.00	0.99
	F1	1.00	0.91	1.00	1.00	0.91	1.00	0.95
syntax tree	#	13	8	15	8	16	20	80
	P	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	R	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	F1	1.00	1.00	1.00	1.00	1.00	1.00	1.00
total	#	86	118	110	54	255	184	807
	P	1.00	0.92	0.98	1.00	0.93	1.00	0.96
	R	1.00	1.00	1.00	1.00	0.97	1.00	0.99
	F1	1.00	0.96	0.99	1.00	0.95	1.00	0.98

As shown in Table 2, the accuracy of the system is excellent (overall F1-

⁵In the interpretation of a sentence, **individual concept** references include references to **object types** where the concept itself is the referent, rather than **things** of that type.

score of 98%), with the correct interpretation of almost every rule being found (99% recall). Although the results show a reduction in precision (96% overall), this is due to finding multiple semantic solutions for some rules and, therefore, demonstrates a desired behaviour of our approach. In a complete system, the alternative interpretations would be presented to the user, allowing them to select the intended interpretation with which to update the specification. It should be reiterated that these results are achieved by assuming a correct lexicon and that future work will investigate the error handling capabilities of the system. However, it shows that if the lexicon can be learnt correctly, our approach is capable of determining the correct formal model of rules using that vocabulary.

The one exception preventing a result of 100% recall occurred in section 2.2.1.5 of the case study. As can be seen from the 100% recall of the syntax annotations for that section, the rule was correctly parsed syntactically; however, the configuration process was unable to find a semantically sound interpretation of the rule. This is due to the rule using an amount with a unit of measure, where the amount is incompatible with the type of the relationship, which is defined in relation to the unit of measure itself. Further investigation is needed to determine whether or not this is an issue with our mappings to SBVR, or if it is a problem with SBVR itself⁶.

Although our focus is currently on accuracy rather than the efficiency of our implementation, it is important to consider the execution time of our approach. For these experiments we evaluate CLUE on two example rules: a short, simple rule with limited ambiguity (1), and a longer more complex rule with nested restrictions and greater ambiguity (2). In addition, the execution time is reported on a minimal vocabulary needed to parse each rule, as well as a combined vocabulary to estimate the effects of a larger vocabulary model and the added ambiguity due to polysemy.

- (1) *Each branch is included in exactly one local area.*
- (2) *The country that is of a branch is the country that is of the operating company that includes the local area that includes the branch.*

There are three main factors influencing the execution time:

1. the time to perform the configuration, which is in turn affected by the size of the model being configured, the number of constraints, and the number of backtracks performed before reaching a valid configuration,

⁶A similar example in the SBVR specification [1, p. 71] seems to circumvent the issue by defining the relationship wrt. thing (the most general concept of SBVR), allowing any type (whether it be an amount or the unit of measure) to be referenced in the relationship. Obviously, this goes against our goal of being able to perform semantic consistency/conformance checking.

Table 3: Configuration Time

	Rule (1)	Rule (2)
# Variable Assignments	267	859
# Constraints	114	434
Min. # Backtracks	9	93
Max. # Backtracks	11	118
Ave. # Backtracks	10	106
# Components Generated	6	24
Ave. Config. Time (sec)	0.08	0.42

2. the number of suggestions in the suggestion list (controlled through pruning and early accommodation), and
3. the number of suggestions with equal ranking (caused by various ambiguities, primarily polysemy).

895 The execution times of the two rules and related statistics are displayed in Tables 3 and 4, which focus on the configuration itself and the overall parsing process, respectively. The tests were performed on a worst-case situation, where accommodation is left until the end of the rule is reached, rather than performing it iteratively. This means there are more suggestions in the suggestion list than
900 necessary. There is one optimisation, however, that is polysemous lexical entries (i.e. same form but different meaning) are treated the same for the purpose of generating suggestions, but different during the configuration phase. This is shown in Table 4 through the different statistics ‘Max # Suggestions’, which is the maximum number of suggestions that were in the suggestion list at any one
905 time, and ‘# Suggestions Configured’, which is the number of semantic variants of the top-ranked suggestion configured at the end.

The configuration results differ from a similar evaluation we performed in previous work [44] in several respects. While the simple rule remains the same, here we have included a second rule and an evaluation of the overall process
910 including the syntactic analysis rather than only the configuration. In addition, the configuration results show improvement due to implementation changes that have resulted in an approximately 19-fold reduction of the number of variable assignments required. As a result, the number of constraints evaluated and the number of backtracks (maximum and average) required to find a solution have
915 decreased, ultimately leading to a large reduction in the configuration time.

These statistics show some interesting results. First of all, the configuration itself executes quite quickly, while the entire process takes quite a large amount of time for the second rule. This shows that repeated configurations of the entire sentence cause significant overhead, which incremental accommodation

Table 4: Process Execution Time

	Individual		Combined	
	Rule (1)	Rule (2)	Rule (1)	Rule (2)
Suggestion Time (sec)	0.01	1.34	0.01	1.55
Max. # Suggestions	11	134	11	136
# Suggestions Configured	1	16	3	20
Total Time (sec)	0.93	9.73	3.15	13.83
Seconds per Suggestion	0.93	0.61	1.05	0.69

920 should help to improve. Furthermore, it indicates that future effort should be put into minimising the number of suggestions that are accommodated. Since many of the additional suggestions are created by treating polysemous lexical entries as different suggestions come accommodation, the overall time could be reduced by relaxing the *catching* constraint on appropriate lexical entries and
925 by allowing the configurator to find the appropriate one. Another possibility is to use parallelisation to accommodate multiple possibilities at the same time.

Secondly, the time per suggestion statistics show that less time is spent per suggestion on the longer example than the shorter one (0.61s vs. 0.93s). At first glance this seems odd since we know that the configuration takes longer on the
930 larger models. The apparent contradiction is caused by many of the alternative suggestions creating malformed SBVR models, which fail the configuration process quickly.

Finally, the large discrepancy between the actual time to configure rule (1) and the time taken for the overall processing of the rule (0.08s vs. 0.93s)
935 is an indication that there is a large amount of overhead in the prototype implementation. Approximately 60% of the time to perform the configuration is used for the serialisation and de-serialisation of large SBVR models, due to the loose coupling between the GATE components and the Configurator. Therefore, future work will need to look at more tightly integrating the two aspects of the
940 process to reduce the overhead and obtain acceptable performance.

6.3. Summary & Outlook

In summary, our approach shows promising results with high accuracy in both the vocabulary learning and parsing processes. Moreover, most errors that appear in the vocabulary learning process could be solved by analysis of
945 the rules and definitions, which will be investigated more fully in future work. Moreover, the additional interpretations lowering the precision of the parsing results actually demonstrate a desired behaviour of our approach. Although it is evident that the execution time needs to be improved for a full-scale system,

there are a number of avenues that can be investigated to optimise the system
950 and overcome this issue.

As the experiments are performed on SBVR SE-based specifications it is expected that the accuracy will be reduced when performed on more unrestricted natural language. However, one of the aims of our approach is to provide user feedback that actively supports them in revising the specification into a more
955 formal form. Therefore, once the revisions have been performed, processing the revised specification will be consistent with the results reported here. Moreover, as SBVR SE has been developed to be quite natural to business users, many natural language expressions are processable in the current state of the prototype. One notable exception is the handling of adjectives to describe properties;
960 however, future work will investigate the use of WordNet [45] and/or other lexical resources to automatically create lexical variants and mappings into the SBVR semantics.

In performing the vocabulary learning separately from the parsing of rules we have made the assumption that the vocabulary was correct when parsing the
965 rules. Obviously this is not the case as we did not achieve perfect results in the vocabulary learning experiments; however, we have discussed several ways in which the errors are potentially auto-correctable and intend to experiment with and refine this capability in the future.

Finally, we assume the existence of a well-defined glossary from which to
970 learn the vocabulary. While glossaries are quite common as part of business and technical documentation, they often include only the most important terms but not the extra information supported by SBVR SE. This will hamper the ability of our approach to be applied to specifications without significant effort on the part of the user to define such a glossary. However, we believe such effort
975 to be beneficial considering the positive results that can be achieved through the automated analysis of the rules. Moreover, future work will investigate the integration of Information Extraction approaches similar to those of [6, 22, 25, 32, 30] to suggest candidate vocabulary entries. This will reduce the manual effort involved in defining the glossary.

980 7. Conclusions and Future Work

In this paper we have demonstrated an approach to transforming natural language business specifications into formal SBVR models keeping three goals in mind: (1) allowing business people to create and maintain their business vocabulary and rules using a flexible natural language and document structure, (2)
985 reducing the amount of manual effort involved in formalising the business specifications into models suitable for use in and for the development of information

systems, and (3) supporting business users in creating well-formed sets of business vocabularies and business rules, with resulting benefits in communicating requirements, software quality, time-to-market, and productivity.

990 We argued that the unique combination of MDE, Cognitive Linguistics,
and Knowledge-based Configuration techniques meets the criteria necessary to
achieve the stated goals. To that end, we have shown that our approach is able to
reliably produce highly accurate and complete SBVR-based formal models; firstly
by learning the vocabulary from an SBVR SE style glossary, and then by parsing
995 the rules using the acquired vocabulary. In the vocabulary learning experiments
we demonstrated an F1-score of 98%, with the predominant error type able to
be identified and corrected during the parsing phase. In the parsing experiments
(using a correct vocabulary) we demonstrated an almost perfect recall of 99%,
meaning that our approach was able to produce the correct formal SBVR model
1000 for almost every rule in the test set. For those that are not correct, the approach
can be used to infer repair suggestions. Moreover, the lower precision score (96%)
is a result of identifying multiple possible interpretations and demonstrates a
feature of our approach: the ability to report ambiguities to the business user
who can then select the intended interpretation, thereby, improving the clarity
1005 of the specification.

A subject of future work will be to optimise the run-time performance of the
prototype. Other future work will perform experiments on the complete EU-
Rent case study; test the error-handling/-correction capabilities of our approach;
investigate expanding the controlled natural language to beyond that of SBVR
1010 SE in order to improve its naturalness further; developing the tool and user
interfaces to allow customisation by non-technical experts; and investigate the
suggestion of vocabulary for specifications that do not already provide a glossary.

References

- [1] OMG, Semantics of Business Vocabulary and Business Rules (SBVR), v1.0,
1015 Object Management Group, 2008.
- [2] D. Hay, K. A. Healy, Defining business rules: What are they really?, Tech.
rep., Business Rules Group (Jul. 2000).
- [3] J. Boyer, H. Mili, Agile Business Rule Development, Springer Berlin Heidel-
berg, 2011. doi:10.1007/978-3-642-19041-4.
- 1020 [4] 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: Proc. ICIST 2010, 2010, pp. 377–384.

- 1025 [5] L. Mich, NL-OOPS: From natural language to object oriented requirements using the natural language processing system LOLITA, *Natural Language Engineering* 2 (02) (1996) 161–187.
- [6] H. M. Harmain, R. Gaizauskas, CM-Builder: A natural language-based CASE tool for object-oriented analysis, *Automated Software Engineering* 10 (2) (2003) 157–181. doi:10.1023/A:1022916028950.
- 1030 [7] M. De Tommasi, A. Corallo, SBEAVER: A tool for modeling business vocabularies and business rules, in: *Proc. KES 2006*, Vol. 4253 of LNCS, 2006, pp. 1083–1091. doi:10.1007/11893011_137.
- [8] I. Bajwa, M. Lee, B. Bordbar, SBVR business rules generation from natural language specification, in: *Proc. 2011 AAAI Spring Symposium Series*, 2011, pp. 2–8.
- 1035 [9] P. B. Feuto, S. Cardey, P. Greenfield, W. El Abed, Domain specific language based on the SBVR standard for expressing business rules, in: *Proc. EDOC Workshops 2013*, 2013, pp. 31–38. doi:10.1109/EDOCW.2013.11.
- 1040 [10] F. Friedrich, J. Mendling, F. Puhmann, Process model generation from natural language text, in: H. Mouratidis, C. Rolland (Eds.), *Proc. CAiSE 2011*, Vol. 6741 of LNCS, Springer Berlin Heidelberg, 2011, pp. 482–496. doi:10.1007/978-3-642-21640-4_36.
- 1045 [11] M. Selway, G. Grossmann, W. Mayer, M. Stumptner, Formalising natural language specifications using a cognitive linguistic/configuration based approach, in: *Proc. EDOC 2013*, IEEE, 2013, pp. 59–68. doi:10.1109/EDOC.2013.16.
- [12] P. Clark, W. R. Murray, P. Harrison, J. Thompson, Naturalness vs. predictability: A key debate in controlled languages, in: *Proc. Workshop on CNL 2009*, Vol. 5972 of LNCS, 2010, pp. 65–81. doi:10.1007/978-3-642-14418-9_5.
- 1050 [13] A. Raj, T. V. Prabhakar, S. Hendryx, Transformation of SBVR business design to UML models, in: *Proc. ISEC’08*, ACM, 2008, pp. 29–38. doi:10.1145/1342211.1342221.
- 1055 [14] B. Steen, L. Pires, M.-E. Iacob, Automatic generation of optimal business processes from business rules, in: *Proc. EDOCW 2010*, 2010, pp. 117–126. doi:10.1109/EDOCW.2010.40.
- [15] M. H. Linehan, Semantics in model-driven business design, in: *Proc. 2nd International Semantic Web Policy Workshop*, 2006, pp. 86–93.

- [16] W. Roover, F. Caron, J. Vanthienen, A prototype tool for the event-driven enforcement of SBVR business rules, in: Proc. Business Process Management Workshops, Vol. 99 of Lecture Notes in Business Information Processing, 2012, pp. 446–457. doi:10.1007/978-3-642-28108-2_43.
- [17] A. Sukys, L. Nemuraite, B. Paradauskas, E. Sinkevicius, Transformation framework for SBVR based semantic queries in business information systems, in: Proc. BUSTECH 2012, 2012, pp. 19–24.
- [18] R. Schwitter, Working for two: A bidirectional grammar for a controlled natural language, in: W. Wobcke, M. Zhang (Eds.), AI 2008: Advances in Artificial Intelligence, Vol. 5360 of LNCS, Springer Berlin / Heidelberg, 2008, pp. 168–179. doi:10.1007/978-3-540-89378-3_17.
- [19] R. Schwitter, Creating and querying formal ontologies via controlled natural language, Applied Artificial Intelligence 24 (1&2) (2010) 149–174. doi:10.1080/08839510903448700.
- [20] N. E. Fuchs, K. Kaljurand, T. Kuhn, Attempto controlled english for knowledge representation, in: Proc. Reasoning Web 2008 — 4th International Summer School, Vol. 5224 of LNCS, 2008, pp. 104–124. doi:10.1007/978-3-540-85658-0_3.
- [21] N. E. Fuchs, R. Schwitter, Attempto: Controlled natural language for requirements specifications, in: Proc. 7th Workshop on Logic Programming, 1995.
- [22] G. Anandha Mala, G. Uma, Automatic construction of object oriented design models [UML diagrams] from natural language requirements specification, in: Proc. PRICAI 2006, Vol. 4099 of LNCS, 2006, pp. 1155–1159. doi:10.1007/978-3-540-36668-3_152.
- [23] M. G. Ilieva, O. Ormandjieva, Models derived from automatically analyzed textual user requirements, in: Proc. Fourth International Conference on Software Engineering Research, Management and Applications, 2006., 2006, pp. 13–21. doi:10.1109/SERA.2006.51.
- [24] G. Fliedl, C. Kop, H. C. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, C. Winkler, Deriving static and dynamic concepts from software requirements using sophisticated tagging, Data & Knowledge Engineering 61 (3) (2007) 433–448. doi:10.1016/j.datak.2006.06.012.
- [25] H. Afreen, I. Bajwa, B. Bordbar, SBVR2UML: A challenging transformation, in: Proc. FIT 2011, 2011, pp. 33–38. doi:10.1109/FIT.2011.14.

- [26] D. de Almeida Ferreira, A. R. da Silva, RSLingo: An information extraction approach toward formal requirements specifications, in: Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE, 2012, pp. 39–48. doi:10.1109/MoDRE.2012.6360073.
- [27] P. B. F. Njonko, W. El Abed, From natural language business requirements to executable models via SBVR, in: Proc. ICSAI 2012, 2012, pp. 2453–2457. doi:10.1109/ICSAI.2012.6223550.
- [28] K. Letsholo, L. Zhao, E.-V. Chioasca, TRAM: A tool for transforming textual requirements into analysis models, in: Proc. ASE 2013, 2013, pp. 738–741, tool demonstration. doi:10.1109/ASE.2013.6693146.
- [29] D. Sadoun, C. Dubois, Y. Ghamri-Doudane, B. Grau, From natural language requirements to formal specification using an ontology, in: Proc. ICTAI 2013, 2013, pp. 755–760. doi:10.1109/ICTAI.2013.116.
- [30] V. B. R. V. Sagar, S. Abirami, Conceptual modeling of natural language functional requirements, Journal of Systems and Software 88 (0) (2014) 25–41. doi:10.1016/j.jss.2013.08.036.
- [31] I. Bajwa, B. Bordbar, M. Lee, OCL constraints generation from natural language specification, in: Proc. EDOC 2010, 2010, pp. 204–213. doi:10.1109/EDOC.2010.33.
- [32] S. Sarkar, V. S. Sharma, R. Agarwal, Creating design from requirements and use cases: Bridging the gap between requirement and detailed design, in: Proc. of the 5th India Software Engineering Conference, ACM, 2012, pp. 3–12. doi:10.1145/2134254.2134256.
- [33] M. Kleiner, P. Albert, J. Bézivin, Parsing SBVR-based controlled languages, in: Proc. MODELS 2009, Vol. 5795 of LNCS, 2009, pp. 122–136. doi:10.1007/978-3-642-04425-0_10.
- [34] V. Ambriola, V. Gervasi, On the systematic analysis of natural language requirements with CIRCE, Automated Software Engineering 13 (1) (2006) 107–167. doi:10.1007/s10515-006-5468-2.
- [35] R. W. Langacker, Cognitive grammar : a basic introduction, Oxford University Press, Oxford, New York, 2008.
- [36] J. Naradowsky, T. Vieira, D. Smith, Grammarless parsing for joint inference, in: Proc. of COLING 2012, 2012, pp. 1995–2010. URL <http://www.aclweb.org/anthology/C12-1122>

- [37] U. Junker, Configuration, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006, Ch. 24, pp. 837–873.
- 1130 [38] M. Kleiner, M. D. Del Fabro, P. Albert, Model search: Formalizing and automating constraint solving in MDE platforms, in: Proc. ECMFA 2010, Vol. 6138 of LNCS, 2010, pp. 173–188.
- [39] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, GATE: a framework and graphical development environment for robust NLP tools and applications, in: Proc. ACL’02, 2002, pp. 168–175.
- 1135 [40] M. Stumptner, G. E. Friedrich, A. Haselböck, Generative constraint-based configuration of large technical systems, AI EDAM 12 (04) (1998) 307–320.
- [41] J. Miller, J. Mukerji (Eds.), Technical Guide to Model Driven Architecture: MDA Guide Version 1.0.1, Object Management Group (OMG), 2003.
- 1140 [42] K. B. I. Holmqvist, Implementing cognitive semantics: image schemata, valence accommodation and valence suggestion for AI and computational linguistics, Ph.D. thesis, Dept. of Cognitive Science Lund University, Lund, Sweden (1993).
- [43] M. Estratat, L. Henocque, Parsing languages with a configurator, in: Proc. ECAI’2004, Vol. 16, 2004, pp. 591–595.
- 1145 [44] M. Selway, W. Mayer, M. Stumptner, Configuring domain knowledge for natural language understanding, in: Proc. 15th International Configuration Workshop (ConfWS’13), Vienna, Austria, 2013, pp. 63–70.
URL <http://ceur-ws.org/Vol-1128/paper9.pdf>
- 1150 [45] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.