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

Matt Selway, Georg Grossmann, Wolfgang Mayer, Markus Stumptner
Advanced Computing Research Centre
University of South Australia, Adelaide
{⟨firstname⟩.⟨lastname⟩}@unisa.edu.au

***Abstract*—This paper addresses the problem of transforming natural language business specifications into formal models suitable for use in information systems. In particular, the Semantics of Business Vocabulary and Business Rules (SBVR) standard is used as a starting point for both the natural language specifications and the formal representation. In recent years, SBVR-based approaches have been proposed for transforming natural language business rules into models, such as UML Class Diagrams; however, most focus on the transformations performed *after* the SBVR models have been created and, therefore, simplify or entirely neglect the natural language to SBVR transformation.**

There are a number of difficulties in transforming natural language into formal models, such as ambiguity and inconsistency. This paper presents a solution based on a unique combination of techniques from Cognitive Linguistics and Knowledge-based Configuration in order to transform natural language business specifications into SBVR models. We present a comparative survey of state-of-the-art approaches and argue that current solutions do not fulfil the criteria necessary to meet our goals. We demonstrate our approach and show how it improves translation of natural language business specifications into formal models and increases the level of automation for Model-Driven Engineering.

***Index Terms*—SBVR; business rules; MDA; MDE; natural language;**

I. 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 control 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. Manual translation and verification of the natural language text, even by technical experts, is time consuming and error-prone. Furthermore, the formal model requires validation by the domain experts who most likely do not know the formal notation.

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 meta-model with a basis in formal logic for technical experts. However, a technical expert is still required to manually translate the controlled English specifications into the formal model; even when using existing tools such as [2]–[4]. Moreover, when using SBVR, it still a difficult task to define complete sets of well-formed concepts and rules governing a business [4].

In the last few years, a number of SBVR-based tools and approaches to automate the transformation from natural language business specifications into models have been proposed (e.g. [2]–[15]). However, most approaches focus on the transformations from SBVR models into others, such as UML Class and Activity diagrams, or BPMN diagrams, etc. As a result, the transformation from natural language (even SBVR Structured English) into SBVR models is often trivialised or neglected entirely so that manual interpretation of the business specification by technical experts remains necessary.

In this paper we present a solution that focuses on automating the transformation from (controlled) natural language business specifications into their formal representation. It uses SBVR as the starting point for both the natural language business specifications and the formal representation, reducing ambiguity and ensuring strict semantics. Our approach utilises a unique combination of techniques from Cognitive Linguistics, Knowledge-based Configuration, and Model-Driven Engineering in order to iteratively build a formal SBVR model of the vocabulary and rules of a business specification. The SBVR model can then undergo further transformations into the specific models required by technical experts.

There are several advantages to our approach. Firstly, we support the processing of a more natural style of language than many other approaches while maintaining strict semantics for reduced ambiguity. Moreover, we provide the ability to customise the document format as well as some aspects of the language. This allows business people to use language and formatting that they are comfortable with or that is required by organisational policies. Secondly, we perform semantic analysis of the business specification in order to improve the quality of the business specification and ensure

its correctness. Furthermore, our approach requires only a well defined business vocabulary and set of business rules, without the need for additional UML- or ontology-based domain models. Finally, we take a Model-Driven Engineering approach, enabling successive transformations to executable models and code through the use of MDE tools.

We demonstrate our approach by applying it to the EU-Rent case study defined in Annex E of the SBVR specification [1]. In addition, we present a comparative survey of current state-of-the-art approaches and show how our approach fulfils the necessary combination of criteria to achieve our goals.

The remainder of the paper is organised as follows: Section II introduces a motivating example, based on the EU-Rent case study, used throughout the remainder of the paper; Section III briefly introduces SBVR, MDE, Knowledge-based Configuration, and Cognitive Linguistics; Section IV describes our approach to parsing natural language business specifications; Section V presents our comparison to related work; and Section VI discusses directions for future work and concludes the paper.

II. Motivating Example

In this section we introduce a motivating example that identifies the limitations of existing approaches. A detailed comparison of approaches is provided in Section V.

Our aim is to provide a method of automating the translation of business specifications into formal models that fulfils three goals: (1) allows business people to control and maintain their own business vocabulary and rules; (2) supports business users in formalising their business specifications as well-formed sets of business vocabularies and rules; and (3) reduces the amount of manual effort involved in their formalisation.

The first goal is important, particularly in the SBVR context, as the SBVR specification defines a business rule as "a rule that is under business jurisdiction" [1, p. 8]. This means that business rules can be added, removed, or modified at the discretion of the business [2]. Therefore, the focus 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 to the second goal, as we need to be able to support business people, who are typically used to specifying things informally, in creating these explicit and unambiguous formal models. Therefore, high quality feedback is important to enable domain experts to revise the models appropriately until they correctly represent the intended meaning.

Finally, we want to avoid adding manual labour to existing processes as much as possible. Therefore, in order to achieve the last two goals, the analysis and translation of the text into formal models should be highly automated.

Consider the case study of EU-Rent, a fictitious car rental company with global presence [1, Annex E]. Fig. 1 shows an extract of an informal EU-Rent business specification describing the structure and locations of the company.

Many existing approaches and tools (e.g. [2]–[4]) do not allow formal business vocabulary and rules to be under the control of business people as, although they provide an SBVR-based notation to more easily allow business people to validate the formal model, they are intended to be used by technical experts. As such, a business person may develop an informal specification in plain natural language and provide that to a technical expert. The technical expert then reads the specification to identify rules and translate them into a more formal form that the tool can translate into a formal model, such as:

> 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.

Fig. 1. EU-Rent case study extract based on [16]

each branch *is_included_in* exactly one local_area

The technical expert then provides this more formal form to the business person to have them validate that it means the same as the original specification.

While this may still be relatively easy for the business person to read, and hence validate, it requires much more technical knowledge to write. Moreover, having to do such things as replace spaces with underscores is a menial task that makes writing the specifications less natural. While this more formal style might be natural to a technical expert, it inhibits the development of formal specifications by business people.

In contrast, bringing the formal business specification under the control of business people will mean a shift in the relationship between business person and technical expert. Rather than providing informal specifications, business people will be able to provide formal, validated, models of their requirements that are immediately usable by technical experts. If additional requirements are elicited by the technical experts, the changes will easily be integrated by a business person and the updated formal model returned to the technical experts.

Apart from being intended for technical experts, existing approaches may not support business people in formalising their specifications for the following reasons:

- greater effort and technical knowledge is required to translate existing vocabulary and/or rules into a formal form, for example, restricted text or UML Class Diagrams ([6]–[8], [10], [11])
- feedback is limited as it is based primarily on whether or not the notation has been used correctly ([3], [9]), or
- feedback is non-existent due to attempting to process only relevant information, potentially missing important errors ([5], [8], [10], [11], [13])

In contrast, our approach aims at satisfying all three goals by, firstly, allowing the use of language more natural to a business person, although not completely unrestricted to reduce ambiguity. This allows business people to more readily

maintain their business vocabulary and rules and helps to minimise the overhead involved in translating their documentation into a formal form. Although some additional effort is required, we see it as adding information to better formalise the vocabulary, rather than needing to expend a large amount of effort rewriting an entire specification. Finally, the formalisation of specifications by business people is supported by an approach that analyses specifications in their entirety to improve feedback on errors, inconsistencies, and ambiguities.

III. Preliminaries

In the following, we briefly introduce SBVR, MDE, Knowledge-based Configuration, and Cognitive Linguistics.

A. 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 (i.e. vocabulary, facts, and rules) in natural languages and their exchange between different organisations and software tools. It is intended for use by business people and can improve the communication between stakeholders; bridging the gap between business people and IT people (the technical experts). The SBVR specification includes two important aspects: a conceptual model and a controlled English representation.

1) SBVR Meta-model: The conceptual model (and associated MOF-based meta-model) 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, permissibility, possibility) [1]. Within the meta-model, meanings and representations form the basis for defining the vocabulary as a set of interrelated *object types*, *individual concepts*, and *fact types*. The meanings are separated from representations, allowing a single concept to be represented by multiple words (in the same or different languages), images, or sounds.

Logical Formulations form the semantic structure of business rules as a conceptualisation of formal logic. The meta-model includes concepts for first-order logical operators (e.g. *conjunction*, *implication*, etc.) as well as modal operators (e.g. *necessity*, *obligation*, etc.).

2) SBVR Structured English: SBVR Structured English (SBVR-SE) is a non-normative notation for expressing business vocabularies and rules with SBVR semantics. It constitutes the syntax of a controlled language that '... maps mechanically to SBVR concepts' [1, p. 237].

The structure of an SBVR-SE specification is separated into sections for vocabulary and rules. The vocabulary section takes the form of a detailed glossary. Many elements of a vocabulary entry are optional and can include structural rules directly related to a given term. The rules section consists of the individual rule statements and can include additional guidance information, notes, etc.

Vocabulary
rental organisation unit
  Definition: organisation unit that operates part of EU-Rent's car rental business
  Concept Type: role
rental organisation unit *having rental responsibility*
  Definition: the rental organisation unit is responsible for the operation of customer-facing rental business
rental organisation unit *having area responsibility*
  Definition: the rental organisation unit includes organisation units for which it has the responsibility to coordinate operations and ensure resources
local area
  Definition: rental organisation unit that *has area responsibility*
branch
  Definition: rental organisation unit that *has rental responsibility*
branch *is included in* local area
  Synonymous Form: local area *includes* branch
branch *has* country
EU-Rent operating company
  Synonym: operating company
EU-Rent operating company *is located in* country
EU-Rent operating company *includes* local area
country
  Source: MWU(1, 2b)["country"]
Canada
  General Concept: country
EU-Rent CA
  Definition: the EU-Rent operating company that *is located in* **Canada**

Rules
Each branch *is included in* exactly one local area.
Each local area *is included in* exactly one operating company.
Each operating company *is located in* exactly one operating country.
The country *of* a branch *is* the country that *includes* the operating company that *includes* the local area that *includes* the branch.

Fig. 2. EU-Rent case study, Locations extract [1, Annex E]

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. The following styles are used in this paper to demonstrate the 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 underlining.

Underlined terms represent the designations of object types; **Names** represent the designations of individual concepts; *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, informal text is shown in normal font, which allows for additional notes and examples that are not intended to be interpreted formally.

An SBVR-SE version of the EU-Rent case study extract is displayed in Fig. 2.

This paper uses SBVR-SE as a starting point for processing natural language into SBVR models. We do so for several reasons: (1) it has a relatively straightforward mapping to SBVR semantics, (2) it is not overly restricted, allowing rules to be expressed quite freely in a number of ways, and (3) it is the notation used by the SBVR specification itself.

B. Model-Driven Engineering

Model-Driven Engineering (MDE) is an approach to Software Engineering that promotes models (abstracted representations of a system) from a documentation role to the primary

artefacts of the software development process, raising the level of abstraction and improving separation of concerns, productivity, time-to-market, and software quality [17]. The guiding principle underlying this promotion of models to key artefacts is that of "Everything is a model" [18], which allows for the uniform treatment of many aspects of MDE.

The primary concepts associated with this principle are that a system is *represented by* a model that *conforms to* its metamodel [18]. The notion of conformance is that the model is written in the language defined by the metamodel. Moreover, models do not feature every element of the system, but an abstracted view of it. Therefore, multiple models are used to represent a system, each conforming to their own metamodel.

The Model Driven Architecture (MDA) [19] realisation of MDE, developed by the OMG, standardises three levels of abstraction that separate the specification of a system from the implementation details of that system: (1) the Computation Independent Model (CIM) layer, which models the environment and requirements of a system without regard to how the system is implemented; (2) the Platform Independent Model (PIM) layer, which models the functionality of the system independent of any particular platform; and (3) the Platform Specific Model (PSM) layer, which models the implementation with respect to a particular platform.

The key to MDE is model transformation; the process of converting one model into another model of the same system [19]. A model transformation can be applied between layers, or within the same layer. The mappings between models, i.e. transformation rules, can be specified using a model transformation language. A transformation engine can then be used to execute the mappings, automating the transformation process. A number of requirements of transformation languages were identified in [20], and include the ability to: (1) match sets of source model elements, their associations and attribute values, (2) match an element by direct and indirect types, (3) (implicitly) create traceability links between source and target elements, (4) define multiple target elements in a single rule, and (5) support multi-step transformations through intermediate transformations. Many of the requirements identified by [20] are fulfilled by current transformation languages such as MOF Query/View/Transformation, the transformation language of the MDA.

Business models, including those defined in SBVR, exist at the CIM layer as they are descriptions of the environment and requirements within which the system must operate without regard to its implementation. Therefore, our overall approach constitutes a text-to-model transformation from the natural language representation of a business model to its formal SBVR model representation. Internally, we use model transformations to bridge the different technologies of our implementation.

C. Knowledge-based Configuration

Knowledge-based configuration is a constraint-based search method traditionally used to compose a customised system from generic components [21]. 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.

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 [7]. This technique, called *model search* in [17], utilises two versions of the meta-model; the original and a meta-model 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 meta-model based on an input relaxed terminal model. This a key technique used in our approach.

D. Cognitive Linguistics

Cognitive Linguistics is a framework of disparate theories, unified by one basic principle and four characteristics [22]:

P) Language is about meaning
C1) Meaning is based on perspective
C2) Meaning is dynamic and flexible
C3) Meaning is encyclopedic and non-autonomous
C4) Meaning is based on usage and experience

Therefore, Cognitive Linguistic theories often encompass methods for the representation and processing of concepts and knowledge.

Of particular interest is Cognitive Grammar [23], which merges syntax, semantics, and pragmatics into a holistic view of language. This allows a uniform treatment of linguistic elements that are traditionally treated separately. For example, parts-of-speech are considered as semantic structures with certain arrangements and complex linguistic structures are combinations of semantic structures learnt by convention.

In Cognitive Grammar, the reading of a sentence evokes the *semantic structures* related to each word. These semantic structures contain *elaboration sites*; locations in the structure where they can be linked to others in order to derive further meaning. Using various mental processes, the evoked structures are then *accommodated* by connecting the sites into a single structure that is the meaning of the entire sentence. We base our language processing on Cognitive Grammar due to its focus on meaning and knowledge representation, and its holistic treatment of various linguistic elements.

IV. Cognitive Linguistic Configuration

An overview of our process is summarised in Fig. 3. The overall workflow is shown in Fig. 3a, while the translation process and architecture of our approach is detailed in Fig. 3b.

The process starts with the definition of the business vocabulary and rules by a business person. We do not require that they use the SBVR-SE structure, rather, we allow the use of their preferred style and format. This helps to minimise the manual work required in formalising the business specification. However, if the document format varies too widely, a technical

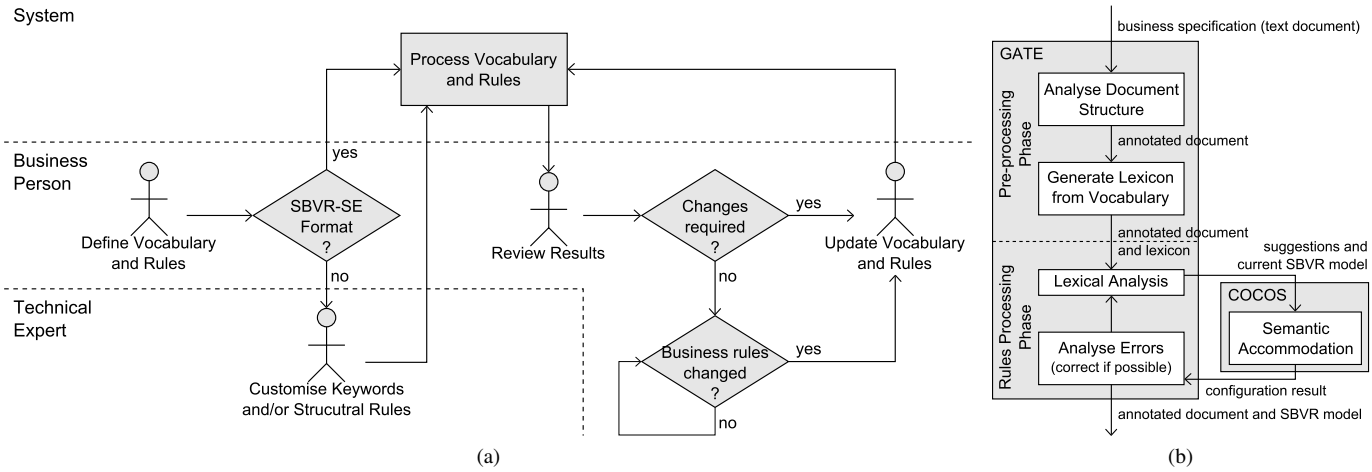

Fig. 3. The overall workflow of our approach (a) and details of the translation process/architecture of the 'Process Vocabulary and Rules' step (b).

expert may be required in order to assist in customising the rules that identify the different elements of the document.

Once formatting issues are dealt with, the vocabulary and rules are processed. First, the different parts of the business specification are identified, for example, the vocabulary entries and rule entries, based on the SBVR-based rules or any customised rules that were provided. Next, a preliminary SBVR model of the vocabulary is created, which is then revised as the rules are parsed and new information is added. Finally, any errors or inconsistencies identified during the processing are presented to the user. The user can then make any changes and reprocess the document, which will update the generated SBVR model and provide feedback. This cycle will continue until no more changes are required to be made.

At any time after the SBVR model has been created, business rules may change due to changes in the environment, new procedures, changes to existing procedures, etc. If any of these occur, the domain expert may update the specification and re-process it to update the SBVR model.

The SBVR model produced by the processing of the business specification can then be used to develop PIMs or PSMs, in accordance with the MDA. For example, an additional model transformation could immediately transform the rules into a format suitable for execution on a rules engine, or the model could be provided to IT experts who can then use it to support the development of an information system for the business by transforming it into UML diagrams.

In order to implement this process, a number of software technologies and tools were integrated. The textual processing is performed by GATE (General Architecture For Text Engineering) [24], which provides a development environment for the annotation of text and the validation and verification of our approach. The Eclipse Modelling Framework [25], which provides an implementation of MOF, is used to provide the MDE capabilities. Model transformations are provided by the Epsilon Eclipse plug-in [26]. The configuration process uses a "generative constraint satisfaction" solver [27], which is based on a non-deterministic, backtracking, depth-first search-based constraint solving algorithm. These components are bound together using model transformations.

A. Structural Analysis

The first step of parsing is to identify the different aspects of the business specification; chiefly the vocabulary and rule entries. The GATE tool is used to perform standard tokenisation and sentence splitting, followed by the identification of the vocabulary and rule entries. By default the rules for an SBVR-SE structured document are used; however, rules for different document styles can be provided to support the processing of other business specifications. Once these elements have been identified the language processing can begin.

B. Lexical Acquisition

The lexical acquisition process creates a preliminary SBVR model/lexicon of the vocabulary entries. A basic vocabulary entry includes the word (or phrase) and its definition. However, the definition can only be parsed once we have a lexicon, so we use heuristics and the structural elements to make a best guess of the type of lexical entry a word might be and then refine it as definitions, rules, etc. are processed. This allows our approach to maintain its flexibility with respect to document format, where the different SBVR elements for a vocabulary entry (concept type, etc.) may not be included.

The initial lexicon is acquired as follows:

1) **Add temporary lexical entries**: For each vocabulary entry identified in the document we create a temporary lexical entry without any associated semantics.
2) **Disambiguate concept types**: For each temporary lexical entry, we determine which main category of SBVR concept type it is most likely to be, i.e. object type, individual concept, or fact type. 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 the main SBVR concept types are then used to create the

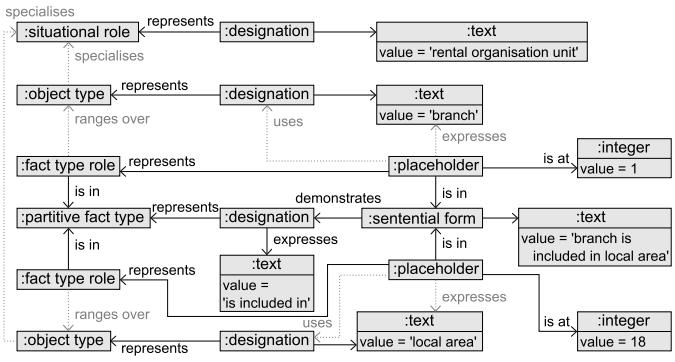

Fig. 4. SBVR model of the example vocabulary

SBVR model elements for each concept. The separate vocabulary entries are also linked appropriately, for example, the roles/placeholders of fact types are connected to the object types that fill them. Fig. 4 shows a portion of the SBVR model corresponding to the vocabulary defined in the SBVR-SE example of Fig. 2.

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.

Differentiating between object types and individual concepts is 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.

The ambiguity can be resolved through structural information, if present, as some are relevant only to object types or are required by individual concepts. Furthermore, the type could be disambiguated through a synonym if one exists. For example, the synonym for 'EU-Rent operating company' is 'operating company', which is more clearly an object type. Finally, the parsing of the definition or rules may identify an inconsistency in the term's usage and update the lexical entry accordingly. For example, the definition of '**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, we identify the placeholders (i.e. terms for object types) included in them. For example, the fact type 'branch *is included in* local area' contains two placeholders: 'branch' and 'local area'. Fact types could include one, two, or more placeholders for *characteristics*, *binary fact types*, and *associative 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 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 a fact type with one less placeholder. In these situations, the structural elements, if present, may clarify the intended type of the vocabulary entry. Otherwise, the processing of the definition and/or rules may identify inconsistencies in its usage and update the lexical entry accordingly.

This approach does not depend on any external lexical resources. The advantage being that the process focuses on the intended, domain specific, meaning of terms rather than the potentially dozens of meanings in a general purpose resource. 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 or verb phrase. This may require additional effort, but is beneficial in the development of a formal vocabulary.

C. Rule Parsing

Once the lexicon has been populated, the definition statements and rules can be parsed. Our parsing process is based on [28], a computational model of Cognitive Grammar. This approach utilises so called *grammatical expectations* as the only lexical information, which are paired with elaboration sites of semantic structures. Using SBVR, object types, fact types, etc. become our semantic structures, fact type roles the elaboration sites, and placeholders the grammatical expectations. In [28], only left and right expectations were introduced, which search to the left or right for another expression to fill it. We also introduced internal expectations, which search within the span of the expression, in order to more easily deal with SBVR fact types with more than two placeholders.

The parsing process follows two iterative steps to incrementally parse sentences: (1) lexical analysis, and (2) semantic accommodation. The first utilises expectations to propose possible parses of a sentence, while the second combines the semantic aspects of the suggested parses into a complete structure using knowledge-based configuration. As such, the parsing process is non-deterministic, an important aspect when dealing with less restricted natural language. The result of the parsing is a progressively more detailed SBVR model containing the concepts, their definitions, and associated rules.

1) Lexical Analysis: The lexical analysis is performed incrementally using the placeholders of lexical entries to identify word combinations in a sentence and propose possible parses. The suggested parses are kept track of in a worklist. The number of suggested parses is kept small by ordering the worklist using a number of heuristics to identify the *best* parse and prune off any suggestions over a maximum limit or that are not considered good enough [28]. The general algorithm for the lexical analysis is as follows:

1) Retrieve the entry for the current word from the lexicon
2) If the current word has any left placeholders, for each suggested parse in the worklist:
   a) catch the closest word (or word combination) to the left of the current word
   b) add the new suggested parse to the worklist
3) Else, for each suggested parse in the worklist, if the previous word or combination has any internal placeholders

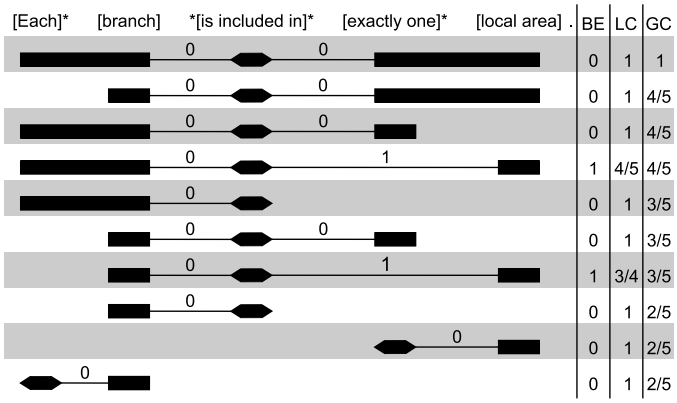

Fig. 5. Example lexical analysis on the rule 'Each branch is included in exactly one local area.' Asterisks denote expectations, hexagons represent the catching word, rectangles represent the caught word. The metrics are: catching distance (shown above each line), binding energy (BE), local coverage (L. Cov.), and global coverage (G. Cov.).

and the current word is within its span:

a) catch the current word with the internal placeholder
b) add the newly suggested parse to the worklist

4) Else, for each suggested parse in the worklist, if the previous word or combination has any right placeholders:
   a) catch the current word with the right placeholder
   b) add the newly suggested parse to the worklist
5) Update the heuristics, distances between words, order the worklist, and cull excess entries
6) Provide newly suggested parses to the semantic accommodation process
7) Remove suggestions that failed accommodation
8) Make the next word the current and return to step 1

An example of the lexical analysis after a complete parse is displayed in Fig. 5 for the rule 'Each branch *is included in* exactly one local area'. The suggestions are ordered top-to-bottom by: firstly, global coverage, as the best parse should cover the entire sentence; then local coverage, as suggestions without holes are preferred; and, lastly binding energy, as words captured at the shortest distance are preferred.

In the top suggestion, the fact type '*is included in*' catches the combined expression 'each branch' to its left and the combined expression 'exactly-one local area' to its right. These combined expressions, which were suggested early on in the lexical analysis and can be seen in the bottom two rows of the figure, were suggested when 'Each' caught 'branch' and 'exactly-one' caught 'local area'. Although these suggestions are at the bottom of the list once the entire sentence has been parsed, at the time they were suggested they would have been good candidates and successfully accommodated.

The catching information constitutes a parse tree, albeit not a traditional one. Fig. 6 shows an example of a parse tree produced by our lexical analysis (b) compared to a more traditional parse tree (a). The parse tree and the relaxed SBVR model of the suggested parse are provided to the semantic accommodation phase.

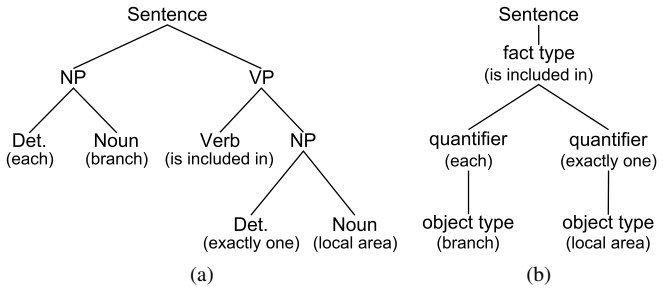

Fig. 6. A traditional parse tree (a) and one created by our analysis (b)

2) Semantic Accommodation: As parses are suggested by the lexical analysis they are sent to the semantic accommodation process to determine whether or not they are admissible in the SBVR semantics. Rather than the numerous processes for accommodation discussed in [28], we utilise knowledge-based configuration to perform the accommodation. Using configuration for parsing has been shown to be an effective and flexible technique by [29] and [7], in which they generate traditional parse trees by the configuration of property grammars. However, we go further by performing configuration on the SBVR model directly. This 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 locations.

In our approach to semantic accommodation, the SBVR meta-model acts as the configuration model and the configuration goal is provided by the relaxed SBVR models and parse trees produced by the lexical analysis. The configurator can then search for and generate new SBVR model elements and relationships as necessary to complete the transformation from the relaxed meta-model to the constrained meta-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 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 SBVR model (including the vocabulary and successfully accommodated rules up to this point), the new elements created through evocation of the lexicon, and the parse tree of Fig. 6b are provided to the configurator. The parse tree provides additional constraints to the configuration, including:

1) the universal quantification (through its variable) must

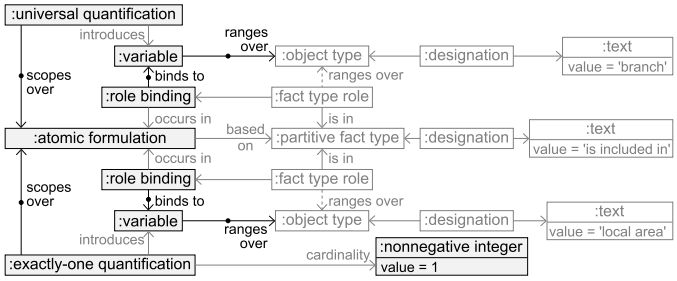

Fig. 7. Configuration result of the example rule. Existing vocabulary elements are empty, grey boxes; elements created through evocation are filled, black boxes; and relationships created by the generative constraint solver are highlighted with a dot.

be connected to the object type 'branch',
2) the exactly-one quantification (through its variable) must be connected to the object type 'local area',
3) the fact type (through the atomic formulation) must be connected to the universal quantification,
4) the fact type (through a role binding) must be connected to the variable of universal quantification,
5) the fact type (through the atomic formulation) must be connected to the exactly-one quantification, and
6) the fact type (through a role binding) must be connected to the variable of the exactly-one quantification

The resulting configuration is displayed in Fig. 7.

In this way, the configuration process incrementally builds a complete and (internally) consistent semantic representation of a business specification as each rule is successfully processed. However, if the accommodation fails the suggested parse is considered to be *ungrammatical* in terms of SBVR's semantics and the current model. Furthermore, if all of the suggested parses fail then the business specification is erroneous or inconsistent. 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 placeholder of a fact type; (2) there is an error in the lexicon created from the formal vocabulary, for example, a fact type learnt as an object type; or (3) a definition or rule has been specified informally, i.e. using terms not formally included in the vocabulary.

In the case that there is an error in the rule or definition, alternative lexical entries are tested to see if they fit "better". For example, for the erroneous rule 'Each rental organisation unit *is included in* exactly one local area', since no corresponding fact type exists, a suggestion would be that 'rental organisation unit' be changed to 'branch' instead. If only a single candidate exists, it will be used and the change marked for the user to review; otherwise, potential corrections are provided to the user, allowing them to correct the specification appropriately.

Alternatively, an error in the learnt lexicon, is identified by checking if the error is removed by changing the type of the lexical entry, for example, from an individual concept to an object type. Changes from an object type/individual concept to a fact type are more difficult as we must identify the placeholders, which may not always be possible. For example, in the rule 'Each local area *is included in* exactly one operating company' the keywords quantify the object types that are the placeholders for the fact type and clearly delineate the second placeholder. However, if the object type 'local area' has been omitted from the vocabulary, it is uncertain as to whether the placeholder between 'Each' and 'exactly one' is 'local' or 'local area'. Changes to the lexicon are marked to allow the user to review them and ensure their correctness.

Finally, a definition or rule may be specified informally either intentionally or unintentionally. If it is unintentional it is likely due to accidentally omitting vocabulary from the business specification. Therefore, to aid in correcting the specification, definitions or rules that are considered informal are marked as such, allowing the user to review them and update the specification appropriately so that they are no longer regarded as informal statements.

V. Comparison of Related Work

We compare our approach to transforming natural language business specifications into formal models to others based on a number of criteria, including: controlled natural language (CNL) style, completeness of parsing, analysis type, tool support, required additional inputs, level of automation, SBVR limitations, ability to be customised. The different approaches surveyed fall into three main categories: (1) those that perform a complete parse of the specification using a formalist approach to language processing, (2) those that take a naturalist approach to parsing natural language using Information Extraction (IE) techniques, and (3) those that perform a complete parse of the sentence while taking a (non-IE) naturalist approach to language processing.

The majority of surveyed approaches fall into the first category, while only a couple fall into the third, including our own. The typical relation between the criteria and the approaches of each category is summarised in Table I, however, individual approaches may differ in one or more criteria. These cases are discussed in the following sections. Although there is no overall best approach, we believe our approach satisfies the appropriate criteria to fulfil our goals while the others may lack in one or more areas.

Although the comparison focuses on SBVR-based approaches, a closely related approach worth mentioning is that of [30]. SBVR and Object-Role Modelling (ORM) ([30]) are both fact-oriented approaches to modelling and utilise very similar textual notations. However, ORM is intended to be used by technical experts who develop the models using a graphical notation and then provide verbalisations using the textual notation to business people for validation purposes. The ORM approach would fall into the first category.

A. Formalist Approaches

The first category is characterised by parsing controlled natural language using a formalist approach, in which a formal language is simplified in order to make it more accessible to non-experts [31]. These approaches provide complete parsing of a specification with respect to their formal grammar, but

TABLE I
SUMMARISED COMPARISON OF RELATED WORK

	Category 1	Category 2	Category 3
Members	[2]–[4], [6], [9], [14], [15]	[5], [8], [10], [11], [13]	[7], This Work
Style	formalist	naturalist (IE)	naturalist
Parsing	complete	incomplete	complete
Analysis Type	syntactic[a]	—	semantic
Tools	yes[a]	yes[a]	in development
Extra Input	—[a]	domain model[a]	lexicon/glossary
LOA	low	moderate[a]	moderate/high
Customisation	both with difficulty[a]	unrestricted structure	structure & keywords
SBVR Limitations	grammar, unary/binary relations only[a]	structure, unary/binary relations only[a]	—

[a] Some approaches differ within the category

require more manual effort to translate specifications into the formal language. In addition, approaches in this category tend to have less natural languages that may require the user to have more technical knowledge of the underlying formalism and perform menial tasks, e.g. converting spaces to underscores.

The use of a formal grammar simplifies the development of tools by allowing parsers to be generated for the language using existing tools. This, in turn, more easily supports the *syntactic* analysis of natural language business specifications. However, syntactic analysis is limited to checking the conformance of the vocabulary and rules to the defined grammar, and of the generated SBVR model to the SBVR meta-model. In the case of [6], [14], which use wizards to guide the user through the creation of rules, the analysis is even more limited.

In contrast, *semantic* analysis aims to ensure the correctness of a natural language specification's *meaning*. This is important for providing high-quality feedback on errors and inconsistencies in the specification to business people. With the exception of [15], which supports semantic analysis through an OWL (Web Ontology Language) reasoner, the members of this category are restricted to syntactic analysis.

Other prototypical characteristics of this category are:

1) no inputs other than the vocabulary and/or rules are required; with the exception of [6] and [14] that require a domain model and rule templates, respectively
2) the document structure and keywords used to define business specifications can be customised, albeit with difficulty and not by the end user as the formal grammar needs to be redefined and the parsers regenerated (or the wizards redeveloped for the wizard-based approaches).
3) formal grammars do not support all SBVR-SE features, in particular n-ary fact types are mostly not supported, which may simplify transformations to other formalisms but requires more effort on the part of the user to split up n-ary relations into sets of binary fact types

B. Naturalist IE Approaches

The second category includes approaches that take a naturalist approach to natural language processing using Information Extraction (IE) techniques. In contrast to formalist approaches, naturalist approaches aim to reduce the complexity of natural language to make it more easily processable by computers, resulting in a more natural and fluent language for users.

In IE-based approaches, natural language is simplified by scanning sentences for instances of *relevant* relations and extracting them. This allows for the processing of unrestricted specifications and minimal or no manual translation of the natural language text. However, the processing is incomplete and the approaches are dependent on language patterns and domain models (specified using UML class diagrams, for example) in order to extract information. Since unrecognised language patterns and relations that do not exist in the domain model (either intentionally or unintentionally) are ignored, important relations could be missed and it is difficult to report erroneous statements to business people.

Although manual translation to a (more) formal language is not required for these approaches, other areas of the process require manual effort. For example, the development of the domain model is a manual process that would likely need a thorough understanding of the business specification and possibly technical knowledge of the formalism used to specify the domain model. Furthermore, validation of both the domain model and the model extracted from the business specification must be performed; a laborious task for large specifications.

Since the approaches in this category can process unrestricted texts, there is no need for them to be tailored to different document structures used in different business specifications. Although they could, in principle, support the customisation of keywords by a business person, the surveyed approaches do not appear to support this, requiring instead that the parser be modified to allow alternative keyword expressions. Finally, these IE approaches tend to support only structural rules and are often restricted to unary and binary fact types, similarly to the approaches of the first category.

C. Naturalist (non-IE) Approaches

The third category contains only two approaches, namely [7] and our own, that are characterised by a naturalist approach to processing controlled natural language while maintaining parsing completeness. Furthermore, the approaches of this category aim for the semantic analysis of business specifications, rather than only syntactic, allowing for more detailed analysis of errors and inconsistencies in the provided specifications.

The approaches of this category combine the advantages of the previous two, and more, in that they support:

1) more natural and fluent languages for users, requiring less technical knowledge and allowing less manual translation than the formalist approaches
2) more robust feedback on errors and inconsistencies, better supporting improvement of the specification and reducing manual validation effort of the output compared to the IE-based approaches

3) wider range of SBVR features, such as n-ary relations, than either of the other categories
4) customisation of the document structure and keywords by business people, rather than technical experts

Our approach differs from [7] in keys areas:

- [7] requires a domain specific lexicon containing detailed linguistic information, whereas our approach requires a relatively simple glossary of terms
- due to the effort required in identifying and including the detailed linguistic information, we believe our approach provides a slightly higher level of automation
- while both approaches utilise configuration, [7] uses it as a more flexible means of generating traditional parse trees, which are syntactic in nature, whereas our approach applies configuration to the domain knowledge, i.e. the semantics, directly

VI. 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 control and maintain their business vocabulary and rules using a flexible natural language and document structure, (2) 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, etc.

In addition, we have presented a survey of state-of-the-art approaches to transforming natural language business specifications into formal models based on SBVR. While there is no best approach, we argue that our approach meets the criteria required to achieve the stated goals.

Future work will look at improving the proof-of-concept in the following ways: developing user interfaces to enable the customisation of keywords and document structure by non-technical experts; expanding the natural language to allow, for example, the use of adjectival forms; and, suggesting a candidate vocabulary for business specifications that do not already provide a vocabulary/glossary in order to reduce the amount of manual effort required in documenting the vocabulary.

References

[1] OMG, *Semantics of Business Vocabulary and Business Rules (SBVR), v1.0*. Object Management Group, 2008.

[2] M. De Tommasi and A. Corallo, "SBEAVER: A tool for modeling business vocabularies and business rules," in *Proc. KES 2006*, ser. LNCS, vol. 4253, 2006, pp. 1083–1091.

[3] A. Raj, T. V. Prabhakar, and S. Hendryx, "Transformation of SBVR business design to UML models," in *Proc. ISEC'08*. ACM, 2008, pp. 29–38.

[4] L. Nemuraite, T. Skersys, A. Sukys, E. Sinkevicius, and 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.

[5] G. Anandha Mala and G. Uma, "Automatic construction of object oriented design models [UML diagrams] from natural language requirements specification," in *Proc. PRICAI 2006*, ser. LNCS, 2006, vol. 4099, pp. 1155–1159.

[6] M. H. Linehan, "Semantics in model-driven business design," in *Proc. 2nd International Semantic Web Policy Workshop*, 2006, pp. 86–93.

[7] M. Kleiner, P. Albert, and J. Bézivin, "Parsing SBVR-based controlled languages," in *Proc. MODELS 2009*, ser. LNCS, vol. 5795, 2009, pp. 122–136.

[8] I. Bajwa, B. Bordbar, and M. Lee, "OCL constraints generation from natural language specification," in *Proc. EDOC 2010*, Oct. 2010, pp. 204–213.

[9] B. Steen, L. Pires, and M.-E. Iacob, "Automatic generation of optimal business processes from business rules," in *Proc. EDOCW 2010*, Oct. 2010, pp. 117–126.

[10] H. Afreen, I. Bajwa, and B. Bordbar, "SBVR2UML: A challenging transformation," in *Proc. FIT 2011*, Dec. 2011, pp. 33–38.

[11] I. Bajwa, M. Lee, and B. Bordbar, "SBVR business rules generation from natural language specification," in *Proc. 2011 AAAI Spring Symposium Series*, 2011.

[12] M. Bonais, W. Rahayu, and E. Pardede, "Integrating information systems business rules into a design model," in *Proc. NBiS 2012*, Sep. 2012, pp. 104–111.

[13] P. B. F. Njonko and W. El Abed, "From natural language business requirements to executable models via SBVR," in *Proc. ICSAI 2012*, May 2012, pp. 2453–2457.

[14] W. Roover, F. Caron, and J. Vanthienen, "A prototype tool for the event-driven enforcement of SBVR business rules," in *Proc. Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, vol. 99, 2012, pp. 446–457.

[15] A. Sukys, L. Nemuraite, B. Paradauskas, and E. Sinkevicius, "Transformation framework for SBVR based semantic queries in business information systems," in *Proc. BUSTECH 2012*, Jul. 22-27 2012, pp. 19–24.

[16] D. Hay and K. A. Healy, "Defining business rules: What are they really?" Business Rules Group, Tech. Rep., Jul. 2000.

[17] M. Kleiner, M. D. Del Fabro, and P. Albert, "Model search: Formalizing and automating constraint solving in MDE platforms," in *Proc. ECMFA 2010*, ser. LNCS, 2010, vol. 6138, pp. 173–188.

[18] J. Bézivin, "On the unification power of models," *Software & Systems Modeling*, vol. 4, no. 2, pp. 171–188, 2005.

[19] OMG, *Technical Guide to Model Driven Architecture: MDA Guide Version 1.0.1*, J. Miller and J. Mukerji, Eds. Object Management Group, 2003.

[20] A. Gerber, M. Lawley, K. Raymond, J. Steel, and A. Wood, "Transformation: The missing link of MDA," in *Proc. ICGT*, ser. LNCS, vol. 2505, 2002, pp. 90–105.

[21] U. Junker, "Configuration," in *Handbook of Constraint Programming*, F. Rossi, P. van Beek, and T. Walsh, Eds. Elsevier, 2006, ch. 24, pp. 837–873.

[22] D. Geeraerts, "Introduction: A rough guide to cognitive linguistics," in *Cognitive Linguistics: Basic Readings*, ser. Cognitive Linguistics Research. Berlin/New York: Mouton de Gruyter, 2006, vol. 34.

[23] R. W. Langacker, *Cognitive grammar : a basic introduction*. Oxford, New York: Oxford University Press, 2008.

[24] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: a framework and graphical development environment for robust NLP tools and applications," in *Proc. ACL'02*, 2002, pp. 168–175.

[25] The Eclipse Foundation. Eclipse modeling framework project. [Online]. Available: http://www.eclipse.org/modeling/emf/

[26] ——. Epsilon. [Online]. Available: http://www.eclipse.org/epsilon/

[27] M. Stumptner, G. E. Friedrich, and A. Haselböck, "Generative constraint-based configuration of large technical systems," *AI EDAM*, vol. 12, no. 04, pp. 307–320, 1998.

[28] K. B. I. Holmqvist, "Implementing cognitive semantics: image schemata, valence accommodation and valence suggestion for AI and computational linguistics," Ph.D. dissertation, Dept. of Cognitive Science Lund University, Lund, Sweden, 1993.

[29] M. Estratat and L. Henocque, "Parsing languages with a configurator," in *Proc. ECAI'2004*, vol. 16, 2004, pp. 591–595.

[30] T. Halpin, "Fact-orientation and conceptual logic," in *Proc. EDOC 2011*, Aug.–Sep. 2011, pp. 14–19.

[31] P. Clark, W. R. Murray, P. Harrison, and J. Thompson, "Naturalness vs. predictability: A key debate in controlled languages," in *Proc. Workshop on CNL 2009*, ser. LNCS, Jun. 8-10 2010, vol. 5972, pp. 65–81.