PM

Generation of Business Rules Code from Natural Language

MASTER'S DEGREE PROJECT

Nuno Miguel Sousa Gonçalves
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

September | 2024

Generation of Business Rules Code from Natural Language

MASTER'S DEGREE PROJECT

Nuno Miguel Sousa Gonçalves
MASTER IN INFORMATICS ENGINEERING

SUPERVISOR
Pedro Dionísio Valente

CO-SUPERVISOR
Eduardo Leopoldo Fermé

Generation of Business Rules Code from Natural Language

Nuno Miguel Sousa Gonçalves

Setembro 2024

Supervisor: Pedro Dionísio Valente

Co-Supervisor: Eduardo Leopoldo Fermé

Abstract

The coding of business rules (BR) is a critical task in information systems (IS) development, particularly within the typical model-view-controller (MVC) architecture. In this architecture, the controller (which includes business logic and BR) handles the flow of data between the view (user interface) and the model (database). Although implementing BR can be complex, time-consuming, and even daunting, current software development tools remain more focused on designing the user interface and the database. This emphasis is understandable, given the challenges of defining complex structures like classes or tables, as well as their relationships, attributes, and fields.

However, an alternative approach lies in the domain of natural language processing (NLP). Natural language (NL) may offer a more suitable means of modeling BR, especially as citizen developers become increasingly involved in software development. With this perspective in mind, this thesis reviews the state of the art in generating BR code from NL. The study examined 604 articles through forward, backward, and lateral snowballing techniques, starting with four cornerstone papers and ultimately narrowing the field to 11 relevant articles.

These selected articles propose solutions that leverage the semantics of business vocabulary and business rules (SBVR) or decision model and notation (DMN) as a "bridge" to produce code in object constraint language (OCL), entity-relationship (ER) diagrams, or extensible markup language (XML). Our findings suggest that while these solutions are valuable, they fall short of generating BR code in diverse programming languages such as C, Java, or Python. To address this gap, we aim to create a prototype that can effectively translate natural language into machine-readable code. This thesis is part of a larger project called "Hydra Code Generation Tool (HydraCGT)," which belongs to the Department of the University of Madeira – Office for the Development of Information Technology Applications (GDAI).

Keywords: Artificial Intelligence; Natural Language Processing; Systems Engineering; Business Logic; Business Rules; SBVR; OCL; DMN; BPMN.

Resumo

A codificação de regras de negócio (RN) é uma tarefa crítica no desenvolvimento de sistemas de informação (SI), especialmente dentro da arquitetura típica modelo-visão-controlador (MVC). Nesta arquitetura, o controlador (que inclui a lógica de negócios e as RN) gerencia o fluxo de dados entre a visão (interface do usuário) e o modelo (banco de dados). Embora a implementação de RN possa ser complexa, demorada e até intimidadora, as ferramentas de desenvolvimento de software atuais continuam mais focadas no design da interface do usuário e do banco de dados. Essa ênfase é compreensível, dada a dificuldade de definir estruturas complexas como classes ou tabelas, além de seus relacionamentos, atributos e campos.

No entanto, uma abordagem alternativa reside no domínio do processamento de linguagem natural (PLN). A linguagem natural (LN) pode oferecer um meio mais adequado para modelar RN, especialmente à medida que desenvolvedores cidadãos se tornam cada vez mais envolvidos no desenvolvimento de software. Com essa perspectiva em mente, esta tese revisa o estado da arte na geração de código de RN a partir de LN. O estudo examinou 604 artigos por meio de técnicas de "snowballing" para frente, para trás e lateral, começando com quatro artigos fundamentais e, por fim, reduzindo o campo para 11 artigos relevantes.

Os artigos selecionados propõem soluções que aproveitam a semântica do vocabulário e das regras de negócio (SBVR) ou do modelo de decisão e notação (DMN) como uma "ponte" para produzir código em linguagem de restrição de objetos (OCL), diagramas de entidade-relacionamento (ER) ou linguagem de marcação extensível (XML). Nossos resultados sugerem que, embora essas soluções sejam valiosas, elas não conseguem gerar código de RN em linguagens de programação diversas, como C, Java ou Python. Para abordar essa lacuna, buscamos criar um protótipo que possa efetivamente traduzir linguagem natural em código legível por máquina. Esta tese faz parte de um projeto maior chamado "Hydra Code Generation Tool (HydraCGT)", pertencente ao Departamento da Universidade da Madeira – Gabinete para o Desenvolvimento de Aplicações Informáticas (GDAI).

Palavras-chave: Inteligência Artificial; Processamento de Linguagem Natural; Engenharia de Sistemas; Lógica de Negócios; Regras de Negócio; SBVR; OCL; DMN; BPMN.

Index

Figure Index

Table Index

Glossary

AI - Artificial Intelligence

BPMN - Business Process Model and Notation

BP - Business Process

BR - Business Rules

CASE - Computer-Aided Software Engineering

CA - Cornerstone Articles

DEMO - Design and Engineering Methodology for Organizations

DMN - Decision Model and Notation

DRD - Decision Requirements Diagram

DSL - Domain Specific Language

EE - Enterprise Engineering

ER - Entity-Relationship

FEEL - Friendly Enough Expression Language

IS - Information Systems

MDE - Model-Driven Engineering

NER - Named Entity Recognition

NL - Natural Language

NLTK - Natural Language Toolkit

NLP - Natural Language Processing

OCL - Object Constraint Language

OMG - Object Management Group

POS - Part-of-Speech

SBVR - Semantics of Business Vocabulary and Business Rules

SE - Software Engineering

SLR - Systematic Literature Review

SQL - Structured Query Language

SRS - Software Requirements Specifications

UML - Unified Modeling Language

XML - eXtensible Markup Language

1 Introduction

In this chapter, we will outline the motivations behind the creation of this project and outline the current challenges in the field of low-code development. Additionally, we will discuss the specific direction we have chosen to pursue in our work. Finally, we will present a brief overview of the structure and organization of the chapters in this thesis.

1.1 Context

The development of information systems (IS) has increasingly emphasized modeling and visual programming [1], largely driven by the rise of low-code development [2][3]. This trend is reshaping the development process and empowering citizen-developers [4]. The paradigm shift suggests a future where traditional coding may become obsolete, encapsulated by the concept of "not coding at all" [5].

Despite mixed evaluations from the scientific community [6][7][8] and some skepticism from traditional software developers [9][10][11][12], enthusiasm for low-code platforms continues to grow [13]. This excitement rivals, or even surpasses, that of previous innovations, such as Computer-Aided Software Engineering (CASE) tools [14], rapid application development [15], end-user development [16], and model-driven engineering [17][7].

1.1.1 Challenges and Current State of Low-Code Development

Although the path toward simplifying IS development seems well-defined, there is still a long journey ahead before tools, whether low-code or traditional, can offer an accessible and intuitive solution for everyday users. The ultimate goal is to enable individuals to create their own IS as easily as they might engage in a self-taught hobby, such as creating art or woodworking. This ambition, often referred to as the search for the "silver bullet" in software engineering, remains elusive [18][19].

A significant example of this challenge lies in the development of business logic, particularly business rules (BR). Preliminary analyses of how low-code tools address this issue reveal a lack of standardized terminology. For instance, business logic is inconsistently referred to as "Business logic specification mechanisms" [20] or "Functional Perspective."[6]. Moreover, although some of these tools incorporate artificial intelligence (AI) features like text recognition or sentiment analysis, they typically do not support the development of BR or the use of natural language (NL) for this purpose.

1.1.2 Work Direction

Given these gaps, we shift our focus to the domain of artificial intelligence (AI), specifically natural language processing (NLP), to explore its state-of-the-art capabilities and potential in generating business rules (BR) code from natural language (NL). To this end, we conducted a systematic literature review (SLR) to examine the current state of NLP, including

its background, techniques, models, and the most advanced approaches for extracting BR from NL.

Our research aims to leverage NLP techniques to enhance the generation of BR from NL. By gaining a deeper understanding of the existing NLP landscape and cutting-edge approaches, we aspire to develop a prototype that addresses the limitations of current low-code tools, bridging the gap between business logic development and natural language comprehension.

Our work proposal is designed to adhere to the best practices and recognize the limitations within the domain of business rules (BR) research. We plan to utilize the principles of Natural Language Processing (NLP), Unified Modeling Language (UML), Object Constraint Language (OCL), Model-Driven Engineering (MDE), and low-code development to create a prototype for processing BR into machine readable code. This prototype will enable the transformation of natural language inputs into structured, executable rules, ensuring alignment between business objectives and software functionalities, in order to be used within the Hydra Code Generation Tool (HydraCGT).

1.1.3 Thesis organizations

This thesis is organized as follows: Chapter 2 presents the state of the art in the area of our research. Chapter 3 provides the literature review. Chapter 4 details the implementation of the prototype. Finally, Chapter 5 outlines our conclusions and suggests directions for future work.

2 State Of The Art

In this chapter, we will explore the key concepts and background that are central to our research, with a particular focus on the intersection of business rules (BR), natural language processing (NLP), and model-driven engineering (MDE). Our objective is to provide a comprehensive understanding of these domains to lay the groundwork for our proposed approach.

2.1 Background

The term "Business Rule" (BR) [21][22] intuitively suggests "a rule within a business," a concept that is easily grasped by most people, including potential citizen-developers. It is important to emphasize that the concept of Business Rules (BR) is significant in both enterprise engineering (EE) and software engineering (SE).

2.1.1 Business Rules in Enterprise Engineering (EE)

In enterprise engineering (EE), the concept of business rules (BR) is central to the Design and Engineering Methodology for Organizations (DEMO) methodology [23]. Within organizations, BRs are structured as "Action Rules." An Action Rule consists of the following components:

- **Conditions:** Circumstances under which the rule can be executed.
- **Actions:** A set of actions to be performed.
- **Constraints:** Limitations that need to be considered.

These Action Rules are triggered during the execution of a business process (BP) or a portion of it, referred to as a transaction. In enterprise engineering (EE), Action Rules primarily focus on coordinating activities within a BP and ensuring the smooth flow of transactions. They govern how tasks are carried out, helping to maintain order and consistency in business operations.

2.1.2 Business Rules in Software Engineering (SE)

In software engineering (SE), the concept of a business rule (BR) is analogous to that of a controller [24]. A controller in SE can be understood as a mechanism that imposes restrictions on existing persistent data, respecting their structural relationships. It ensures that certain conditions are met before validating actions, and it defines subsequent actions, such as approval or denial based on those validations [25][26]. Unlike enterprise engineering (EE), where the focus is on coordinating activities within a business process, SE emphasizes the implementation and enforcement of constraints within software systems to maintain data integrity and ensure system functionality.

The key distinction between the EE and SE interpretations of BR is that in EE, Action Rules typically focus on the coordination and flow within a business process and its transactions. In contrast, in SE, controllers primarily concern themselves with enforcing constraints on data to regulate how the data is used and to prevent incorrect or unauthorized changes.

2.1.3 Bridging EE and SE Perspectives

Initially, research on business rules (BR) aimed to consolidate business directives, including business process (BP) coordination, into a single platform. This effort sought to identify and harmonize similarities between enterprise engineering (EE) and software engineering (SE) perspectives [27][28]. This holistic approach aligns more closely with the EE definition of BR, which emphasizes coordination within business processes.

However, another strand of research has focused exclusively on BRs, excluding BP coordination [29]. This approach aligns more closely with the SE definition, concentrating on the enforcement and implementation of rules as constraints within software systems. This latter approach has yielded findings that are more directly relevant to our research objectives, which involve translating natural language (NL) into actionable business rules. Specifically, this involves targeting structures defined by the Semantics of Business Vocabulary and Business Rules (SBVR) or adapting these rules into database structures. Such a focus on precise rule extraction and implementation from NL is crucial for developing systems that can interpret and apply BRs effectively within software environments [30][31].

2.2 Natural Language Processing Concepts (NLP)

In this section, we will elucidate the most important terms and concepts that are essential for understanding the remainder of this thesis. We will start by defining each concept and then demonstrate how these concepts are interconnected. Our emphasis will be on presenting our work proposal in a manner that adheres to best practices and acknowledges the existing limitations within this field of knowledge. This approach will ensure that the foundational elements of our research are clearly understood, facilitating a deeper comprehension of the subsequent chapters and our overall research contribution.

Natural Language Processing (NLP) is a branch of artificial intelligence (AI) that focuses on converting everyday language, or natural language (NL), into a format that computers can understand. NLP is employed in virtual assistants like Amazon’s Alexa, Google Assistant, and Apple's Siri, which use NLP to interpret voice commands and generate appropriate responses. The process involves analyzing text to grasp its syntax (sentence structure), semantics (meaning), grammar, and overall organization. This analysis encompasses the identification of entities (recognizable concepts), understanding context, sentiments, interactions, and relationships among different text elements. Significant challenges in NLP include deciphering ambiguity, sarcasm, and the cultural nuances of language. NLP serves various functions, including indexing, information retrieval, text categorization, automatic translation, summarization, knowledge acquisition, and the generation of text and dialogues [32].

2.2.1 Key Concepts in NLP

2.2.1.1 Natural Language Processing (NLP):

Natural Language Processing (NLP) translates everyday language into a format that computers can comprehend. This process utilizes various techniques such as tokenization, part-of-speech (POS) tagging, named entity recognition (NER), and parsing. NLP plays a crucial role in transforming natural language inputs into structured forms that computers can effectively process.

2.2.1.2 Tokens:

Tokenization is the process of breaking down text into smaller units known as tokens, which may be words, phrases, or characters. This step is crucial in Natural Language Processing (NLP) as it transforms raw text into a structured format that can be further analyzed and processed. This foundational stage is essential for subsequent NLP tasks, enabling more sophisticated analyses and interpretations of the text.

For example, the sentence "That cat is black." can be tokenized into, see Figure. 1:

```
Tokens: ['That', 'cat', 'is', 'black', '.']
```

Figure 1 - Using SpaCy library

2.2.1.3 Part-of-Speech (POS) Tagging:

Part-of-speech (POS) tagging involves assigning a label to each token in a sentence that corresponds to its part of speech, such as noun, verb, adjective, etc. For example, in the sentence "Madeira has beautiful landscapes," POS tagging would label "Madeira" as PROPN (proper noun), "has" as VERB, "beautiful" as ADJ (adjective), and "landscapes" as NOUN, see Figure. 2. POS tagging is crucial for understanding the syntactic structure of sentences and plays an essential role in tasks such as parsing and entity recognition, helping further analyze the grammatical relationships within the text.

```
Tokens and POS tags: [('Madeira', 'PROPN'), ('has', 'VERB'), ('beautiful', 'ADJ'), ('landscapes', 'NOUN'), ('.', 'PUNCT')]
```

Figure 2 - Using SpaCy library

2.2.1.4 Named Entity Recognition (NER):

Named Entity Recognition (NER) is the process of identifying and classifying named entities in text into predefined categories such as person names, organizations, locations, dates, and other proper nouns. For instance, in the sentence "Cristiano Ronaldo was born in Madeira," NER would identify "Cristiano Ronaldo" as a person and "Madeira" as a location. NER is critical for extracting structured information from unstructured text and is extensively utilized in applications like information retrieval and text mining, helping to enhance the accessibility and analysis of data, see Figure. 3.

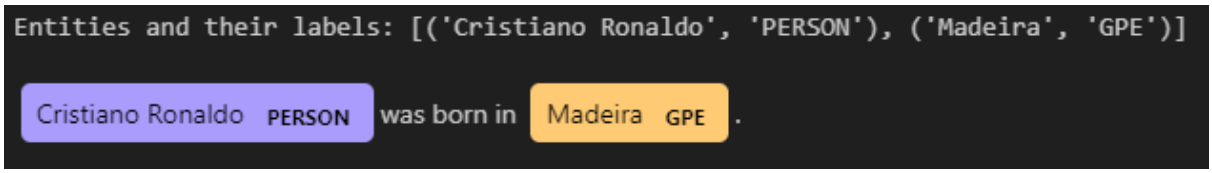


Figure 3 - Using SpaCy library

*GPE - Geo-Political Entity

2.2.1.5 Parsing:

Parsing involves analyzing the grammatical structure of a sentence to understand the relationships between words. It typically produces a parse tree that visually represents these

relationships. For instance, in parsing the sentence "A customer should be 18 years old," the analysis would indicate that "A customer" is the subject, "should be" is the verb phrase, and "18 years old" is a noun phrase that serves as the complement to the verb, see Figure. 4. Parsing is essential for a deep syntactic analysis and for understanding the hierarchical structure of language. It is important not to confuse parsing with POS tagging. While POS tagging assigns part-of-speech labels to individual words, parsing provides a more comprehensive analysis that maps out the relationships among those words, offering a richer and more detailed understanding of sentence structure, see Table 1.

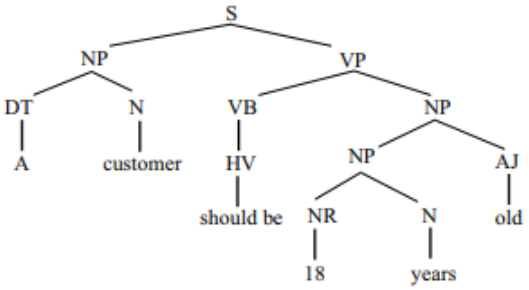


Figure 4 - Structure Tree [33]

Table 1 - Syntactic Meaning [33]

Syntactic Elements	Meanings	Syntactic Elements	Meanings
S	English Sentence	V	Action Verb
NP	Noun Phrase	DT	Determiner
VP	Verb Phrase	AJ	Adjective
VB	Verb	N	Noun
P	Pronoun	NR	Number
HV	Helping verb	PP	Preposition Phrase

2.2.1.6 Dependencies:

Dependency parsing concentrates on identifying the dependencies between words in a sentence, representing these relationships as a directed graph where nodes are words and edges denote grammatical relationships. For example, in the sentence "She enjoys playing soccer," dependency parsing would show "She" as the subject linked to the verb "enjoys," and "playing soccer" as the object of the verb. This method provides a clear representation of the syntactic structure of a sentence and is instrumental in various NLP tasks, including machine translation and information extraction, by helping to clarify how words interact and depend on each other within sentences, see Table 2 and Figure 5.

Table 2 - Model Output (Using SpaCy library)

Token	Head	Dependency	POS
She	enjoys	nsubj	PRON
enjoys	enjoys	ROOT	VERB
playing	enjoys	xcomp	VERB
soccer	playing	dobj	NOUN
.	enjoys	punct	PUNCT

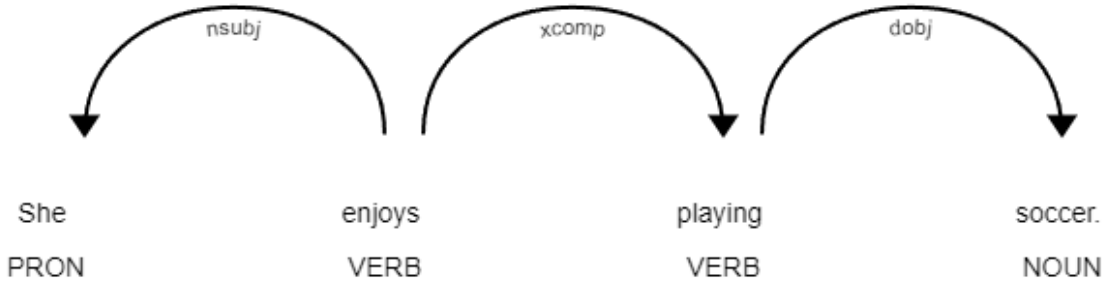


Figure 5 - Graphic of the Model Output (Using SpaCy library)

2.2.1.7 Relation Classification:

Relation classification involves identifying and categorizing semantic relationships between entities within a text. Entities typically include names of people, organizations, locations, dates, and other identifiable concepts. Relationships can vary widely and may encompass types such as cause-effect, spatial, and temporal relationships. For instance, in the sentence "Ana was born in Madeira in 1985," relation classification would identify "Ana" as a person, "Madeira" as a location, and "1985" as a date, see Figure 6. It would categorize these entities around a birth event, establishing a temporal and spatial relationship between them. Accurate classification necessitates a thorough understanding of the sentence's context and the ability to correctly identify these relationships based on linguistic cues. This process is essential for extracting meaningful and structured information from unstructured text, which is crucial for applications such as knowledge graph construction, information retrieval, and data summarization.

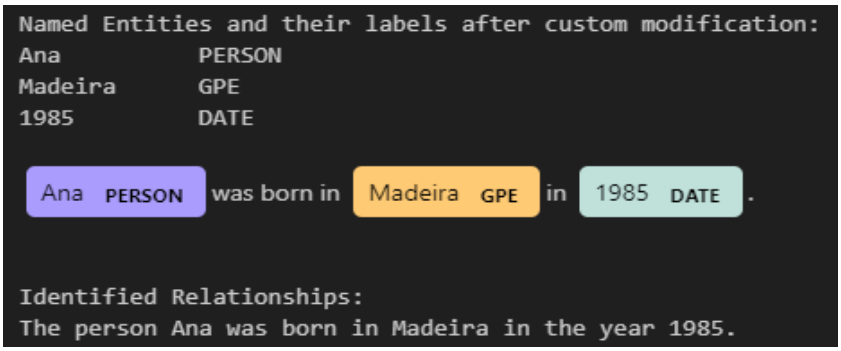


Figure 6 - Output of Entities (Using SpaCy library)

2.2.1.8 Domain Specific Language (DSL):

A Domain Specific Language (DSL) is a programming language specifically designed to be highly effective for a particular application area. Unlike general-purpose programming languages that address a wide range of problems across various domains, DSLs concentrate on specialized issues within a specific context. This specialized focus enhances expressiveness and efficiency within the designated domain, making the language more intuitive for experts in that field. For example, SQL is a DSL tailored for database queries, allowing users to efficiently access and manipulate data without the need for deep knowledge of the database's

internal mechanics. By abstracting and simplifying complex tasks specific to its domain, a DSL enhances productivity, reduces the likelihood of errors, and facilitates more precise and streamlined development processes.

2.2.1.9 Domain Analysis:

Domain analysis involves a thorough understanding and analysis of a specific domain to develop effective and relevant NLP solutions. This process requires a deep comprehension of the terminology, concepts, common practices, and specialized knowledge structures pertinent to that domain. For example, in the legal sector, it is crucial to grasp legal language, types of documents (such as contracts, case law, and statutes), and key legal concepts to create NLP solutions capable of accurately interpreting and processing legal texts. Similarly, in the medical field, domain analysis would concentrate on medical terminology, the various types of medical records, and the structure of clinical notes. This focused approach enables the development of NLP tools that can effectively extract patient information or aid in diagnosing conditions, ensuring that the solutions are both technically sound and contextually appropriate for the domain in question.

2.2.1.10 Grammar Processing:

Grammar processing dissects sentences into their constituent parts, such as nouns, verbs, and adjectives, and examines their grammatical structure. Formal grammars, such as context-free grammars and dependency grammars, model the structure of language. These grammatical frameworks offer rules that elucidate how words combine to form valid sentences. For instance, context-free grammar breaks sentences down into a hierarchical structure of nested phrases. Each phrase is analyzed for its role and structure within the larger sentence. On the other hand, dependency grammar emphasizes the relationships between words in a sentence, representing these relationships as a tree structure. In this tree, words act as nodes linked by edges that denote dependencies, illustrating how each word relates to others within the sentence. This form of analysis is crucial for tasks such as parsing, where understanding the grammatical relationships is key to interpreting the meaning and structure of sentences.

2.2.1.11 Adaptive Information Extraction:

Adaptive information extraction is a technique that focuses on automatically extracting structured information from unstructured text, adapting effectively to different contexts and domains. This approach is especially valuable in scenarios where the types of information that need to be extracted vary significantly or when the data undergo frequent changes. For example, extracting key information from social media posts, which can vary widely in content and format, requires adaptive methods capable of learning from examples, rather than relying solely on predefined rules. Deep learning techniques, such as neural networks, are commonly employed to develop models that can generalize from training data to new, unseen examples. These models learn to recognize patterns and extract relevant information across diverse and dynamic datasets, making them essential for robust and flexible information extraction tasks.

2.2.1.12 Word Embeddings:

Word embeddings are dense vector representations of words that capture their meanings based on context. Techniques like Word2Vec create embeddings that place semantically

similar words close to each other in a high-dimensional space. For example, the words "man" and "woman" would have similar embeddings, see Table 3. Word embeddings are fundamental for various NLP tasks as they enable the model to understand semantic relationships between words.

Some examples:

Table 3 - Word Embeddings example (Using SpaCy and pandas)

	apple	man	orange	queen	woman
apple	1.0	0.107	0.6135	0.0932	0.0872
king	0.1952	0.4166	0.208	0.6109	0.3573
man	0.107	1.0	0.1079	0.3541	0.8273
queen	0.0932	0.3541	0.0656	1.0	0.4757
woman	0.0872	0.8273	0.0947	0.4757	1.0

2.2.1.12.1 Relevant observations

In Table 3, the words "queen" and "woman" exhibit an embedding similarity of 0.4757%, see Table 3, indicating a lower level of relatedness than might be expected. This reduced similarity can sometimes lead to confusion, as word embedding models are trained not solely based on direct semantic similarity, but also on contextual and syntactic differences between words. Although "queen" and "woman" are semantically related, they function in distinct contexts and roles, which contributes to their lower similarity score. This aspect highlights the complexity of word embeddings and underscores the need for careful consideration of both semantic and contextual elements when interpreting these models.

2.2.1.13 Text Summarization:

Text summarization is the process of creating a concise summary of a longer text document. There are two main types: extractive and abstractive summarization. Extractive summarization involves selecting key sentences directly from the original text to compose the summary, effectively capturing the main points without generating new content. Abstractive summarization, on the other hand, involves generating new sentences that encapsulate the core ideas of the text, often rephrasing or using synthesis to convey the essential information in a new form. This technique is particularly useful for quickly understanding the gist of large documents, such as news articles or research papers, allowing readers to grasp the main points without reading the entire text.

2.2.1.14 Coreference Resolution:

Coreference resolution is the task of identifying which words or phrases in a text refer to the same entity. For instance, in the sentences "Patrícia went to the park. She enjoyed her time there," coreference resolution would determine that "She" refers to "Patrícia" and "there" refers to "the park." This process is essential for understanding the coherence and meaning of text, as it helps clarify the relationships and continuities between different parts of the text, enabling more accurate interpretation and analysis.

2.2.1.15 Low-Code Development:

Originating from Model-Driven Engineering (MDE) principles, low-code development seeks to simplify the software development process, allowing users, including those with minimal technical expertise, to develop applications rapidly. By understanding these concepts

and their interrelationships, we can develop robust methodologies for translating natural language business rules into structured formats that are compatible with both Enterprise Engineering (EE) and Software Engineering (SE) frameworks.

2.3 Techniques

The Object Management Group (OMG) is an international, open membership, not-for-profit technology standards consortium. Founded in 1989, OMG is dedicated to creating and maintaining standards that facilitate the interoperability of computer systems, with a particular focus on software engineering. This organization plays a critical role in the development of industry-wide standards that ensure software products can operate seamlessly across various platforms and environments [34].

OMG's Role and Operations

- **Membership**: OMG has hundreds of member organizations, including major technology companies, government agencies, and academic institutions. Members collaborate to develop and approve standards.
- **Process**: OMG's standards are developed through a rigorous process that involves discussion, draft creation, and voting. This ensures that the standards meet the needs of the industry and are robust and reliable.
- **Adoption**: OMG standards are widely adopted across various industries, ensuring interoperability and enabling a wide range of technologies to work together seamlessly.

Importance of OMG

- **Interoperability**: By creating standards, OMG ensures that different systems and technologies can work together, facilitating interoperability.
- **Innovation**: Standards provided by OMG allow developers to focus on innovation rather than reinventing the wheel for common functionalities.
- **Quality and Efficiency**: OMG standards help improve the quality and efficiency of software and system development, reducing costs and time-to-market.

If possible, we will be using the official documentation of the OMG to explain the next subjects.

2.3.1 Semantics of Business Vocabulary and Business Rules (SBVR)

Context:

SBVR (Semantics of Business Vocabulary and Business Rules) is a specification developed by the Object Management Group (OMG), utilized across various business contexts such as business process modeling, information systems (IS) development, and compliance management of corporate systems. SBVR offers a method for articulating business rules (BR) and business vocabulary in a formal and precise manner. This facilitates communication and the effective implementation of policies and procedures within and across organizations. By providing a standardized framework, SBVR ensures that all parties involved have a common

understanding of the rules and terms, enhancing clarity and consistency in business operations [35].

2.3.1.1 Applicability:

The SBVR (Semantics of Business Vocabulary and Business Rules) specification plays a crucial role across a wide range of business activities in diverse organizations. It provides a clear, meaning-focused, multilingual, and semantically detailed framework for defining the meanings of language used by individuals in any industry, profession, discipline, field of study, or organization. This comprehensive framework ensures that business vocabularies and rules are precisely articulated, promoting better understanding and consistency in their application across various contexts [35].

This specification is optimally designed for businesspeople, rather than automated processing, and is intended for business purposes independent of information systems design. It aims to:

- **Defining Terms Clearly:** Ensure all business concepts and rules are defined unambiguously, providing consistent terminology that is easy for both businesspeople and lawyers to understand.
- **Expressing Concepts in Local Languages:** Allow business professionals from various communities to express business concepts and rules in their own languages, with each expression uniquely tied to a specific meaning within its context.
- **Converting Between Human and Tool Formats:** Make it easy to transform business concepts and rules from human-readable forms to formats that tools can process, and back again.
- **Using Logic for Interpretation:** Apply logical methods to interpret business concepts and rules, helping to identify any inconsistencies or gaps within an SBVR Content Model.
- **Applying to Real-World Scenarios:** Use the defined business concepts and rules to make decisions in real-world business situations, ensuring decisions are reproducible and identifying behavior that conforms to or deviates from the rules.
- **Facilitating Information Exchange:** Enable the seamless exchange of business concepts and rules between humans and tools, and between different tools, ensuring no loss of essential information.

2.3.1.2 Usage of an SBVR Content Model:

In SBVR (Semantics of Business Vocabulary and Business Rules) Content Models, the concepts are strictly limited to those that exist in the actual or planned reality of the organization. The extensions of these concepts do not include elements from the SBVR Content Model itself; rather, they are composed of terms and representations that describe these real-world concepts [36].

SBVR Content Models are primarily focused on defining meanings and the expressions that communicate these meanings. They explicitly do not address or engage with assertions about the truth-value of propositions, which fall outside their scope and into the domain of data and rules enforcement. While organizations may publish their business vocabulary in an SBVR Business Vocabulary and their business rules in an SBVR Rulebook to reflect the current

vocabulary and rules, such assertions about their enforcement or application are not covered by the SBVR specification and Content Models. For instance, an organization might list rules in a rulebook that are never actually enforced, highlighting that SBVR Content Models strictly exclude any business data beyond the business vocabulary and business rules content [36].

The primary relationship within SBVR Content Models is between the meanings articulated in the business vocabulary or rulebook and the real-world entities they describe. In contrast, IT systems typically associate classes in the data/reasoning model with various forms of recorded business data, showing a different focus and method of association [36].

2.3.1.3 SBVR application in the real world:

When a business needs to support a contract, law, or regulation, it undertakes several essential steps. First, the business must interpret the legal terms using its own conceptual framework and resolve any ambiguities that arise. For example, if a contract broadly defines the term "order," the business needs to decide whether a "letter of intent" qualifies as an "order." This initial interpretation is crucial to ensure all parties have a shared understanding of the terms [36].

Second, the business must interpret the legal conditions within the context of its operations. For instance, a contract stipulating that "all orders must be paid within 30 days" needs clarification on whether this refers to "within 30 days of order receipt," "within 30 days of order fulfillment," or another specific point in the order lifecycle. Such clarifications are vital for aligning contract terms with business processes and operational capabilities [36].

Legal contracts are typically composed of "terms" and "conditions." The terms define and name the concepts used throughout the contracts, establishing a common language between the parties. The conditions, on the other hand, specify constraints on these terms or on the behavior of the contract participants. Both terms and conditions can vary in their level of precision, affecting how they are implemented and enforced [36].

customer
Definition: one that purchases a commodity or service
Source: Merriam Webster Collegiate Dictionary

customer *submits* order *to* company
Definition: **the** customer transmits **the** order to **the** company

firm order
Definition: order **that** *is final*

letter of intent
Definition: order **that** *is* **not** *final*
Synonym: LOI
Note: A customer may submit an LOI to get a price quote, delivery schedule, or other terms of sale.

order
Definition: customer *request for* items

order *has* item
Necessity: **Each** order *has* **at least** one item.

Figure 7 - Definition of SBVR Rules [36]

SBVR (Semantics of Business Vocabulary and Business Rules) plays a significant role in this context by aiming to describe these details accurately when they are significant to a business. One practical output of SBVR-style modeling is a formal business vocabulary, which may be presented as a dictionary of nouns and relationships. For example, a business dictionary might illustrate that both "firm order" and "letter of intent" are categorized under "order." This distinction allows a customer to submit either type of order, with the stipulation that every order must include at least one item. Such a business dictionary aids in reducing misunderstandings and clarifying expectations, ensuring, for instance, that an "order" consistently includes at least one "item" as depicted in Figure 7. This clarity is essential for smooth business operations and legal compliance [36].

Vocabulary in the context of business rules often include structural rules, which define constraints on the business concepts themselves and thus cannot be violated without impacting the system's integrity. An example of such a structural rule is the necessity rule "each order has at least one item," as illustrated in Figure 7. This rule ensures that the fundamental definition of an "order" in the business context includes at least one item, establishing a basic operational standard [36].

The other major category of SBVR rules is behavioral or deontic rules, which govern or direct business behavior rather than strictly structural aspects. These rules are presented in various forms, including obligations and prohibitions, and unlike structural rules, they can be violated. Figure 8 provides an example of such rules, including both an obligation and a prohibition, along with explanations to aid user understanding. These explanations are crucial for ensuring that all stakeholders comprehend the rules but are not interpreted otherwise [36].

Behavioral rules specify not only the expected behaviors but also the actions that should be taken when these rules are breached. For instance, the "Enforcement Level" in Figure 8 indicates that the customer should be notified if an order is shipped with some items not in inventory. This notification process is part of the protocol for handling violations of the behavioral rule, which stipulates conditions under which the business must operate and communicate, thereby guiding operational responses and customer interactions [36].

A clerk **may** *ship* **an** order **only if all the** items *of* **the** order *are in inventory*.
Synonymous Statement: **No** clerk **may** *ship* **an** order **if an** *item* **of the** order *is* **not** *in inventory*.
Description: All parts of the order must be available in the warehouse before any of the order may be shipped.
Enforcement Level: notify customer

Figure 8 - Example of SBVR in use [36]

This methodology enables considerable flexibility for modeling completeness. For instance, vocabulary entries can have informal definitions, such as for "customer," that use terms not otherwise defined. Noun concepts can also be partially formalized for example, one can specify that a noun is a subtype of another noun while describing the distinguishing characteristics informally. Fully formal definitions, such as for "firm order," enable automated reasoning to determine if a particular order qualifies as a firm order [36].

Disadvantages:

- Despite its flexibility, modeling of any kind is labor-intensive making it costly.
- Another risk is the excessive or unpredictable development time and effort required for information systems.
- Practical impacts can include:
    - Increased business costs
    - Loss of business agility
    - Legal consequences for business

These risks often motivate enterprises to invest in clarifying their terms and explicating their business rules [36]. With that in mind we believe that this methodology is of great importance for our project.

2.3.2 Business Process Model and Notation (BPMN)

Context:

Developed by the OMG, BPMN, is a standard for business process modeling that provides a graphical notation for specifying business processes in a process flow diagram, and is widely used to document, analyze, and optimize business processes. BPMN provides a standardized set of graphic symbols and rules to describe the flow of business processes, including elements such as events, activities, gateways, and flows. The notation is designed to be easy to understand and use by both business professionals and IT developers. This promotes better communication and understanding among different stakeholders. Another great advantage of BPMN is its interoperability with other technologies and modeling standards, allowing for easy integration of business processes with existing IT systems [37].

Applicability:

The Object Management Group (OMG) has established a standard known as Business Process Model and Notation (BPMN). The main objective of BPMN is to offer a notation that is easily comprehensible to all business users, ranging from business analysts who draft the initial process designs to technical developers responsible for implementing the processes, and ultimately, to the business managers who oversee and monitor these processes. BPMN thus acts as a standardized bridge between business process design and process implementation [37].

Another significant objective of BPMN is to ensure that XML languages, such as Web Services Business Process Execution Language, designed for executing business processes, can be visualized using a business-oriented notation. This specification consolidates the best practices from the business modeling community to define the notation and semantics of Collaboration diagrams, Process diagrams, and Choreography diagrams. The aim of BPMN is to standardize business process modeling and notation amid a plethora of different modeling notations and perspectives. By doing so, BPMN facilitates straightforward communication of process information among business users, process implementers, customers, and suppliers [37].

BPMN is specifically designed to support only the concepts relevant to Business Processes. Consequently, other types of modeling conducted by organizations for business purposes fall outside the scope of BPMN, with “Business rules models” being one of the things that is out of the scope of model of BPMN [37]. This as the consequence of having less priority

for our project, due to the fact that BPMN does not have relation with Business Rules, but we need to see if it is applicable for processes of NLP that can be of use for us.

BPMN Example:

As a starting point, a simple BPMN process model can be considered. The model of posting a job, illustrated in Figure 9, is readily understandable to most individuals who have prior experience with any form of process modeling. The modeling approach is like well-known flow charts and activity diagrams, making it accessible and intuitive for those familiar with these types of diagrams [37].

At this stage, there are two possible outcomes: either the job posting is acceptable, or it is not. If the job posting is not acceptable, it is reworked by the human resources department. After reworking, the job posting is reviewed again by the business department. This review can again result in the job posting being deemed acceptable or not. Consequently, the job posting may need to be reviewed multiple times. Once the job posting is deemed acceptable, it is published by the human resources department, marking the end of the process [37].

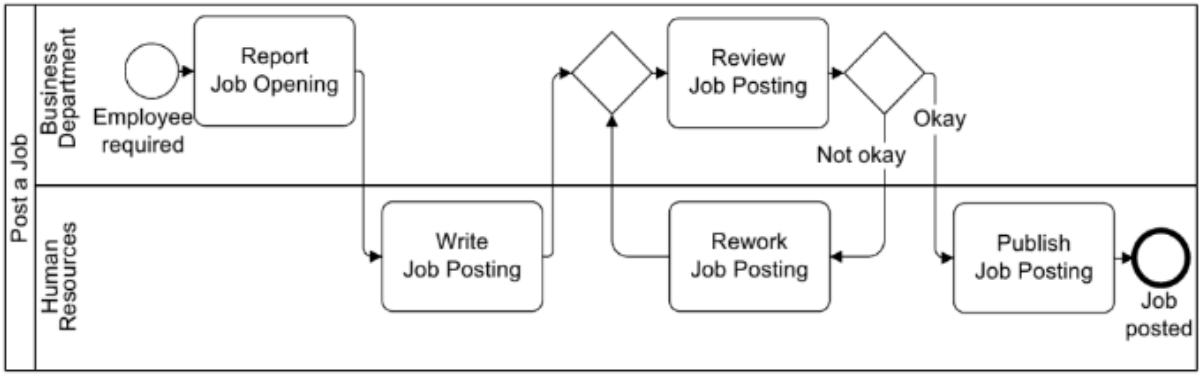


Figure 9 - Job posting process [37]

2.3.3 Decision Model and Notation (DMN)

Context:

DMN is a modeling and execution system for business decisions maintained by the OMG. DMN allows the structured and formal modeling of business decisions. This includes the definition of decision criteria, inputs, processes, and outputs. DMN was designed to be compatible with other business modeling standards such as BPMN. It promotes interoperability and integration between business process modeling and decision-making. Decisions modeled in DMN can be directly executed by decision support systems, reducing the need for complex manual coding [38].

Scope and uses of DMN:

Decision modeling is undertaken by business analysts to understand and define the decisions utilized within a business or organization. These decisions are typically operational, made in day-to-day business processes, rather than strategic decisions, for which fewer rules and representations are available. In this context, three uses of DMN can be identified [38]:

- **Modeling Human Decision-Making:** can be used to represent and analyze the decision-making processes that humans perform within an organization.

- **Modeling the Requirements for Automated Decision-Making**: DMN helps in specifying the requirements and criteria for decisions that are intended to be automated, ensuring clarity and accuracy.
- **Implementing Automated Decision-Making**: DMN can be employed to design and deploy decision logic that will be executed by automated systems, facilitating consistent and efficient decision-making processes.

By applying DMN in these ways, organizations can effectively bridge the gap between human decision processes and automated decision systems, enhancing overall decision accuracy and efficiency [38].

Applicability:

The purpose of DMN is to offer the constructs necessary for modeling decisions, enabling organizational decision-making to be clearly depicted in diagrams, accurately defined by business analysts, and optionally automated [38].

Existing modeling standards address decision-making from two different perspectives:

- **Business process models:** These models describe the coordination of decision-making within business processes by defining specific tasks or activities where decision-making occurs.
- **Decision logic:** These models define the specific logic used to make individual decisions, such as business rules, decision tables, or executable analytic models.

However, decision-making has an internal structure that is not conveniently captured by either of these perspectives. Therefore, DMN aims to provide a third perspective, the Decision Requirements Diagram (DRD), which serves as a bridge between business process models and decision logic models:

- **Business process models** will outline tasks within business processes where decision-making is necessary.
- **Decision Requirements Diagrams** will specify the decisions to be made within those tasks, their interrelationships, and their requirements for decision logic.
- **Decision logic** will detail the required decisions sufficiently to enable validation and/or automation.

Together, Decision Requirements Diagrams and decision logic can offer a comprehensive decision model that complements a business process model by detailing the decision-making involved in process tasks. The relationships between these three modeling aspects are illustrated in Figure 10.

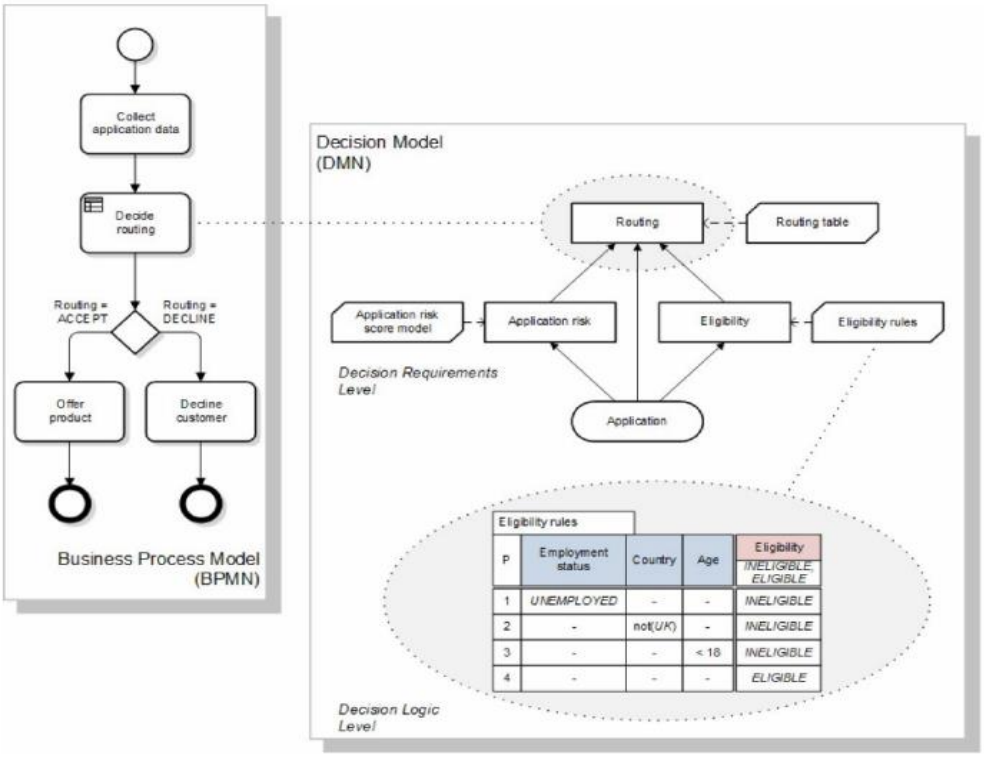


Figure 10 - Example of a DMN Model [38]

The interconnected set of models will enable detailed modeling of business rules and analytic models in business processes, cross-validation of models, top-down process design and automation, and automatic execution of decision-making [38].

Although Figure 10 shows the connection between a business process model and a decision model to explain DMN's relationship with other standards, DMN is not dependent on BPMN. DMN's two levels—decision requirements and decision logic—can be used independently or together to model decision-making without referencing business processes (see Figure. 11) [38].

DMN provides constructs for both decision requirements and decision logic modeling. For decision requirements modeling, it defines the Decision Requirements Graph (DRG) with a corresponding notation: the Decision Requirements Diagram (DRD). For decision logic modeling, DMN uses FEEL (Friendly Enough Expression Language) for defining and assembling decision tables, calculations, if/then/else logic, simple data structures, and externally defined logic from Java and PMML into executable expressions. DMN also offers a graphical notation for decision logic ("boxed expressions") that can be associated with elements of a Decision Requirements Diagram [38].

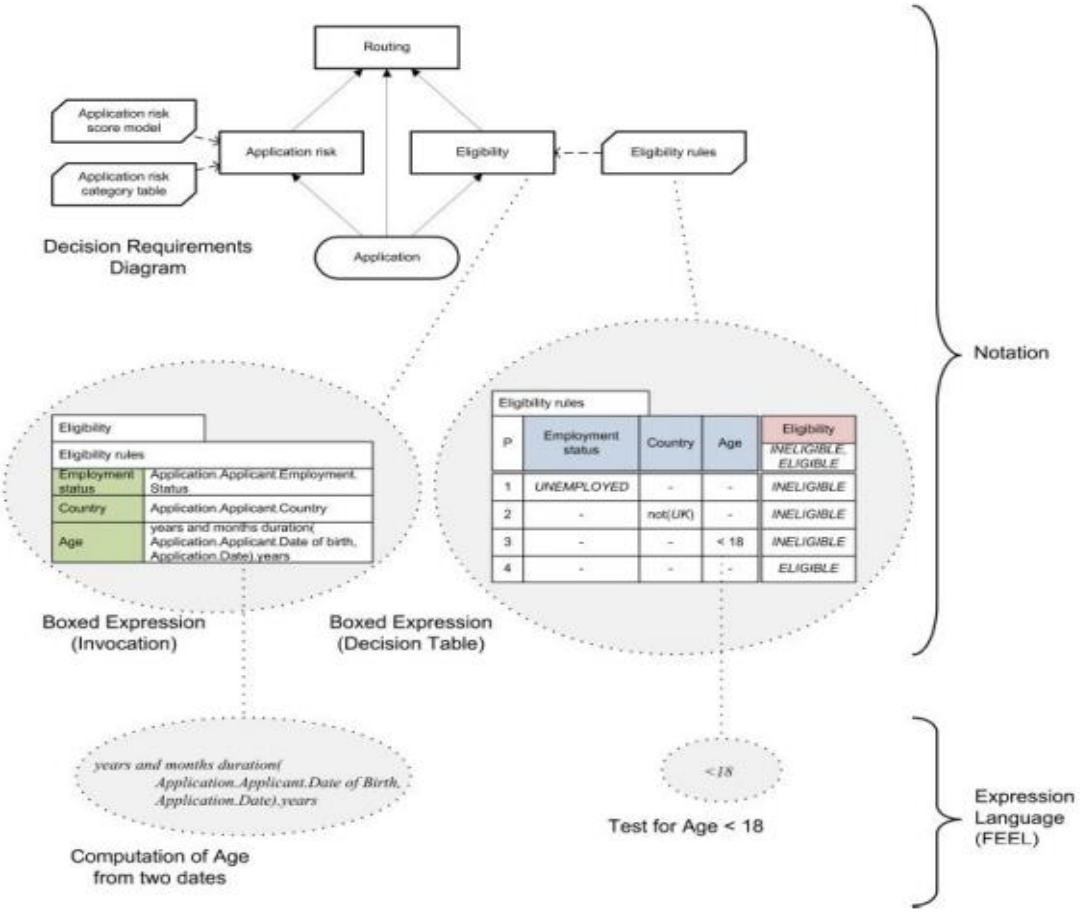


Figure 11 - Adding Expression Language to a DMN Model [38]

2.3.4 Object Constraint Language (OCL)

Context:

The Object Constraint Language (OCL) is a formal language used to describe expressions in Unified Modeling Language (UML) models [39]. OCL enables developers to specify implementation details that cannot be graphically represented, functioning primarily as a constraint language. This means it is used to define rules that objects in a UML model must adhere to, including integrity constraints, conditions, and business rules. Due to its high compatibility with database modeling and development, OCL facilitates a level of specification that can streamline business rule (BR) coding.

Applicability:

A UML diagram, like a class diagram, often lacks the detail needed to capture all relevant aspects of a specification. Natural language descriptions of additional constraints can lead to ambiguities. To address this, formal languages have been developed, but they are often difficult for those without a strong mathematical background [40].

The Object Constraint Language (OCL) bridges this gap by providing a formal, yet readable and writable language. Originating from IBM's Insurance division and the Syntropy

method, OCL is a pure specification language, meaning its expressions have no side effects and do not alter the model's state upon evaluation [40].

OCL is not a programming language, it cannot handle program logic, flow control, or invoke processes. It is a typed language, requiring expressions to conform to type rules, and it excludes implementation details, ensuring instantaneous evaluation without changing object states [40].

Scope and uses of OCL:

OCL (Object Constraint Language) can be used for a variety of purposes [40]:

- **As a query language:** OCL can be utilized to query information from UML models.
- **To specify invariants on classes and types in the class model:** OCL can define conditions that must always hold true for instances of a class or type.
- **To specify type invariants for Stereotypes:** OCL can be used to define conditions that must always be met for instances of a stereotype.
- **To describe pre- and post-conditions on Operations and Methods:** OCL can specify conditions that must be true before an operation or method is executed (pre-conditions) and conditions that must be true after its execution (post-conditions).
- **To describe Guards:** OCL can define conditions that control the flow of execution in state machines or activity diagrams.
- **To specify target (sets) for messages and actions:** OCL can determine the recipients or targets of messages and actions within a model.
- **To specify constraints on operations:** OCL can enforce rules and restrictions on the behavior of operations.
- **To specify derivation rules for attributes:** OCL can define how the values of derived attributes are computed based on other attributes or elements in the UML model.

By supporting these various functions, OCL enhances the precision and clarity of UML models, ensuring that all specified constraints and conditions are explicitly defined and unambiguous [40], see Figure. 12.

Example of OCL:

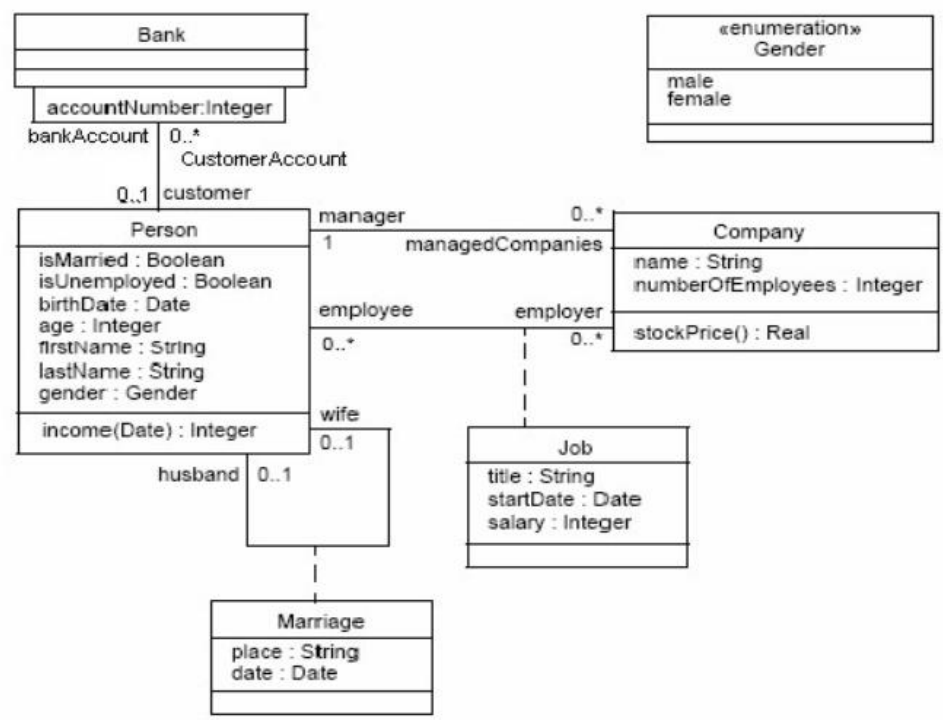


Figure 12 - Example of the use of OCL [40]

2.3.5 Model-Driven Engineering (MDE) and Low-Code

Model-Driven Engineering (MDE) focuses on the creation and use of abstract models, often utilizing the Unified Modeling Language (UML), to represent knowledge and activities within a specific domain. This methodology allows developers to work at a higher level of abstraction compared to traditional programming, offering high-level system abstractions that guide and inform all phases of development. MDE makes intensive use of specialized modeling tools to support the creation, manipulation, and transformation of models, promoting continuous development [41].

On the other hand, low-code development, which has its roots in the principles of MDE, adopts a more straightforward way to increasing software productivity. It seeks to facilitate users' ability to independently develop information systems (IS) by simplifying the development process. This user-centered approach has contributed to its growing popularity in comparison to MDE in recent years [42].

2.4 Conclusion

To conclude this chapter, we can affirm that the topics studied will be of great importance in achieving the objectives of this thesis. Many possibilities are incompatible with each other, for example, "SBVR" and "BPMN" do not yet have an architecture that establishes a strong relationship between them [43]. Considering this, one of our tasks will be to decide the best approach to obtain the best possible path to get the better results.

3 Literature Review

In this chapter, we will present the results of the work we conducted for the literature review stage. We will explain where we started with our sources, the most important criteria used, as well as the results and conclusions we reached. Additionally, we will discuss the options we have for our final work.

With this analysis methodology, we aim to produce a work that serves as a "stepping stone" to demonstrate that the project is feasible and that we have the capability to present it in the form of a prototype.

3.1 Methodology

In our systematic literature review (SLR), we applied a rigorous exclusion criterion to initially identify cornerstone articles (CA) and then, through a process of snowballing, identify a broader set of articles. This larger set was then subjected to an extended criteria to specifically retrieve proposals aimed at transforming natural language (NL) into business rules (BR), and if possible, programmable code. We detail the process as follows:

3.1.1 Exclusion Criteria

To ensure the relevance and quality of the analyzed documents, we established the following exclusion criteria, applied initially for the identification of candidate articles (CA) and subsequently during the snowballing process:

1. **Book:**
    - Can include any chapter, but not the entire book.
2. **Conference Proceedings:**
    - Can include any article, but not the entire conference proceedings.
3. **Survey or Literature Review:**
    - Cannot be a study solely focused on existing related literature. If it is, it is considered only for qualitative analysis.
4. **Project Context Analysis:**
    - Excludes documents that primarily analyze the context of specific projects.
5. **Keywords:**
    - If keywords are provided, they should include at least two of the following:
        - **Natural Language, Natural Language Processing, or equivalent.**
        - **Transformation, or equivalent.**
        - **Business Rule, or equivalent (including SBVR).**
6. **Abstract:**
    - If keywords are not provided, verify if the paper is a proposal for the transformation of natural language (NL) into business rules (BR).
7. **Language:**
    - Must be written in English preferably, or alternatively in Portuguese.
8. **Availability:**
    - The article must be available for download.

These exclusion criteria were designed to filter out documents that do not meet the specific needs of our research focus on the transformation of natural language into business rules. This structured approach helps ensure that the selected documents are directly relevant and accessible, facilitating a more effective and comprehensive analysis.

3.1.2 Systematic Literature Review (SLR) Methodology

The SLR was conducted using the search terms “business rules” and “natural language” to identify candidate articles (CA) with 10 or more citations per year, excluding the publication and search years. Three primary sources were utilized for this search:

1. **IEEE Xplore Digital Library:** 443 results.
2. **ACM Digital Library:** 376 results.
3. **Google Scholar:** Approximately 11,800 results.

We analyzed all records from IEEE Xplore and ACM Digital Library, finding no results that matched both the CA and exclusion criteria. For Google Scholar, we relied on the relevance sorting feature and analyzed the first 500 results.

From this process, four candidate articles were identified:

1. Arco et al., 2021
2. Bajwa et al., 2011
3. Danenas et al., 2020
4. zur Muehlen & Indulska, 2010

Forward, backward, and lateral snowballing techniques were then applied to these articles, resulting in a total of 604 articles. The exclusion criteria were reapplied to this set, narrowing it down to 56 articles.

Based on these criteria, we extracted and analyzed the data, culminating in the characterization presented in Figure 13.

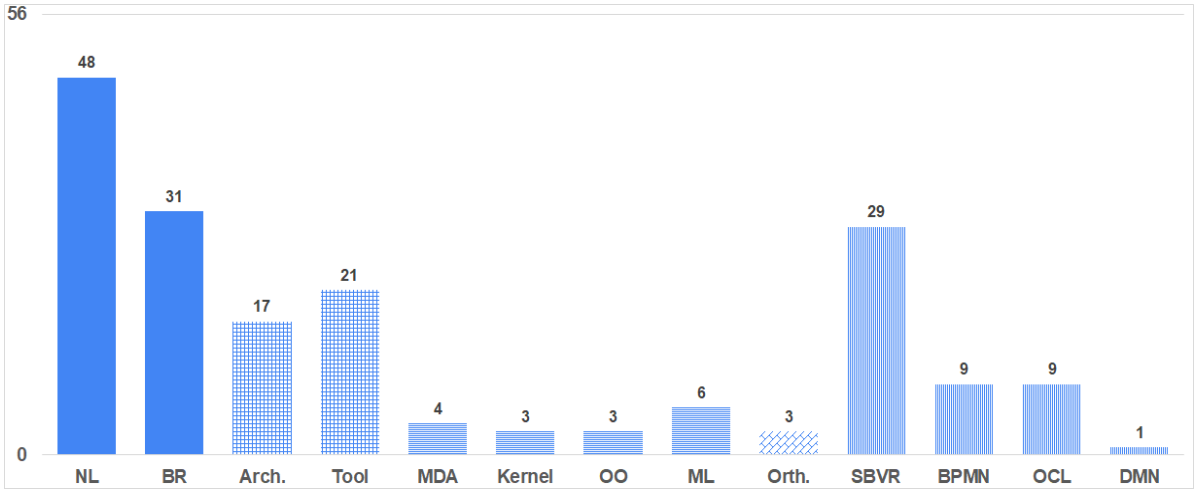


Figure 13 - Characterization of the set of 56 accepted articles

An important point is that the first 4 columns (NL, BR, Arch., Tool) are the most important. To build our prototype, we are looking for papers that use NL with a focus on BR. The tags (Arch.; Tool) help us provide a better overview of the process and help avoid erroneous architectures, which will help reduce the time needed to build the prototype.

We decided to ignore the tags (Kernel, OO, Orth.) because none of the papers we have with these tags include Arch. (Architecture), and since we have little time to create the prototype, we are leaving these papers out of our scope.

3.1.3 Transformation of NL into BR Proposals

An extended verification was applied to the 56 accepted articles based on the following criteria:

Main Criteria (Mandatory for All Articles)

1. **Natural Language Processing (NL):** The article addresses the processing of natural language.
2. **Business Rule (BR):** The article targets the Business Rule.

Completeness Criteria (At Least One Must Be Met)

1. **Architecture:** The article provides a software architecture of the solution.
2. **Tool:** The article verifies the use of a tool in the solution.

Basis Criteria (At Least One Must Be Met)

1. **Model-Driven Architecture (MDA):** The solution is based on Model-Driven Architecture.
2. **Kernel-Based:** The solution is based on a kernel.
3. **Object-Orientation (OO):** The solution is based on object-orientation.
4. **Machine Learning (ML):** The solution incorporates machine learning.

Optional Criteria

1. **Orthography:** The solution considers orthographic correction.

Output Criteria (At Least One Must Be Met)

1. **SBVR:** The solution targets the Semantics of Business Vocabulary and Rules (SBVR) as an output.
2. **BPMN:** The solution targets the Business Process Model and Notation (BPMN) as an output.
3. **OCL:** The solution targets the Object Constraint Language (OCL) as an output.
4. **DMN:** The solution targets the Decision Model and Notation (DMN) as an output.
5. **Structured Data:** The solution targets defined data structures.

3.1.3.1 Selection Results

From the extended verification, we identified 11 papers [33, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53] that consider natural language as an input and transform it to generate one or more

formally structured models such as SBVR, UML, DMN, BPMN, OCL, or XML. The following 45 articles were rejected: [54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98].

3.2 Proposals analysis

Our SLR resulted in a total of 11 proposals for the transformation of NL intro BR. We now proceed with a quantitative characterization of the proposals, and afterwards, we analyze their contents in detail, see Table 4.

Results of the systematic review:

Table 4 - Results of the analysis

Ref.	NL	BR	Arch.	Tool	SBVR	BPMN	OCL	DMN
[45]	✓	✓	✓	✓	✓			
[46]	✓	✓	✓	✓	✓			
[47]	✓		✓	✓				✓
[33]	✓	✓	✓	✓	✓		✓	
[48]	✓		✓	✓	✓			
[49]	✓	✓	✓	✓	✓		✓	
[50]	✓	✓	✓		✓			
[51]	✓		✓		✓			
[52]	✓		✓	✓	✓			
[53]	✓			✓	✓			
[44]	✓		✓		✓			

Since there is no way to convert SBVR to BPMN or vice versa, and as previously discussed in section 2.3.2, we decided to leave BPMN out of the scope of our problem. Therefore, we don't have any papers with the BPMN tag.

3.3 Brief Introduction of the Papers

In this chapter, we will summarize the papers that had the greatest impact on the construction of our prototype. For each of the papers, we will provide a summary and use bullet points to highlight the most important content from each of them. Additionally, we will present the architecture to facilitate the understanding of the paper.

3.3.1 ID2SBVR: A Method for Extracting Business Vocabulary and Rules from an Informal Document

This paper introduces an approach called ID2SBVR, which uses informal documents, typically derived from open interviews, to extract operational rules for SBVR (Semantics of Business Vocabulary and Rules). This method addresses the lack of formal documents in many organizations by extracting types of facts, such as compound, complex, and compound-complex sentences from unstructured natural language and translating them into structured operational rules in SBVR [44].

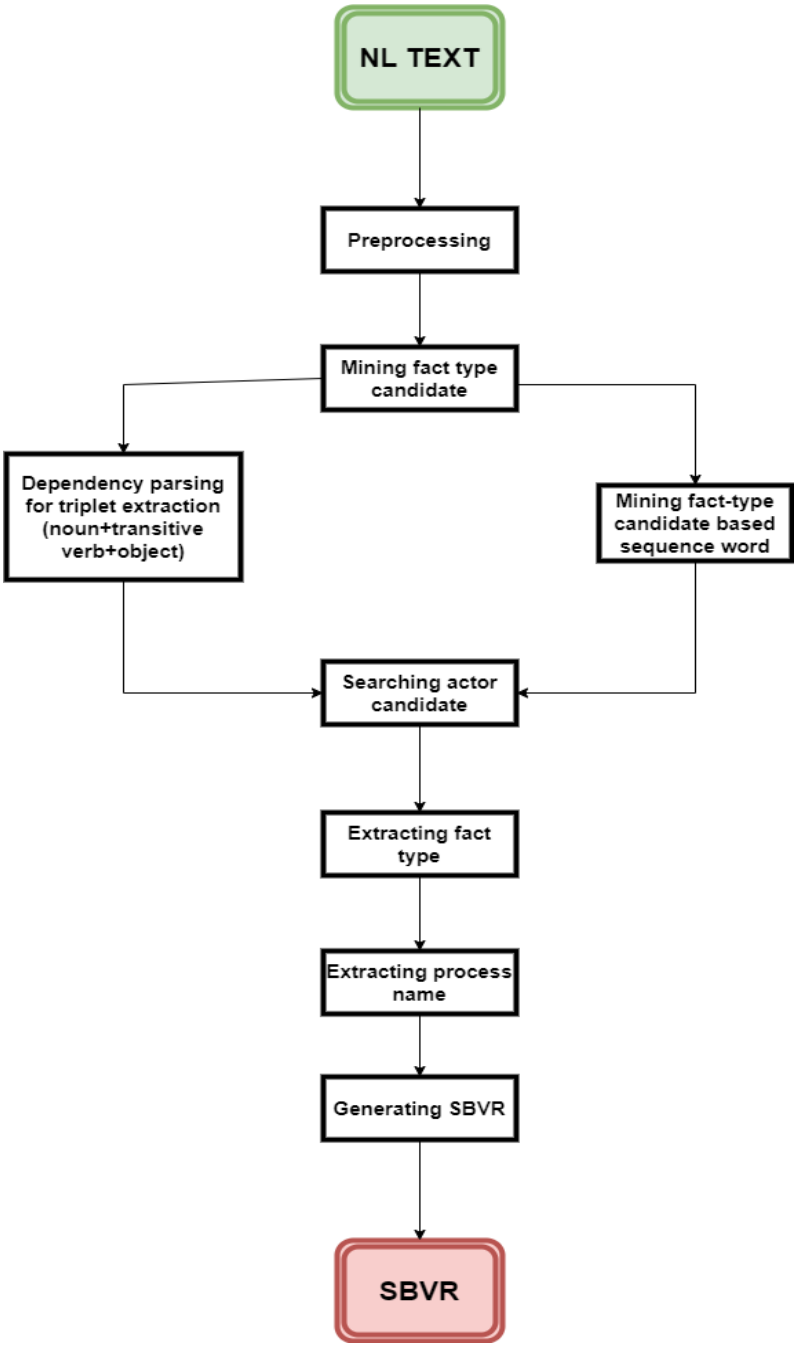


Figure 14 - Architecture of 3.3.1

The approach transforms these informal documents into semi-structured data suitable for further processing and, ultimately, for the generation of business process models [44].

Key Points:

- The method leverages informal documents to create structured operational rules in SBVR.
- Informal documents from open interviews.
- POS Tagging (Part of Speech Tagging) is used in a non-flexible manner, utilizing a Matcher tool.
- Compound, complex, and compound-complex sentences.

Practical Benefits:

- Enhances understanding and improvement of organizational processes.
- Facilitates business processes.
- Serves as a foundation for analyzing the connection between formal business processes and their actual implementation.

Limitations:

- There is a noted loss of precision in compound sentences.
- The authors don't provide access to the tool.

Architecture, see Figure. 14:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn't use external tools.

3.3.2 Natural Language Ambiguity Resolution by Intelligent Semantic Annotation of Software Requirements

This paper explores the ambiguity in Software Requirements Specifications (SRS) documents due to the use of natural language, which often lacks clarity and consistency for all stakeholders involved. To address this issue, the study proposes the adoption of controlled natural languages, such as SBVR (Semantics of Business Vocabulary and Rules), as a viable alternative to traditional natural language, which can be complex and difficult for stakeholders to understand [45].

SBVR provides a syntactically unambiguous and semantically consistent explanation that serves as a bridge between natural language and formal representations, enabling clearer and more precise documentation in the software development process [45].

Key Points:

- To resolve ambiguity in SRS documents by using controlled natural languages like SBVR.
- Natural language (NL) requirements from SRS documents.
- Flexible POS Tagging (Part of Speech Tagging) to identify individual components of SBVR in the sentence.

Practical Benefits:

- Provides clear, unambiguous, and semantically consistent representations.
- The study introduces an automated tool that processes natural language to generate SBVR-based controlled representations of software requirements.
- The tool converts English-written software requirements into clear and unambiguous rules stored in XML format, making them easily portable and machine-processable.

Limitations:

- The authors don't provide access to the tool.

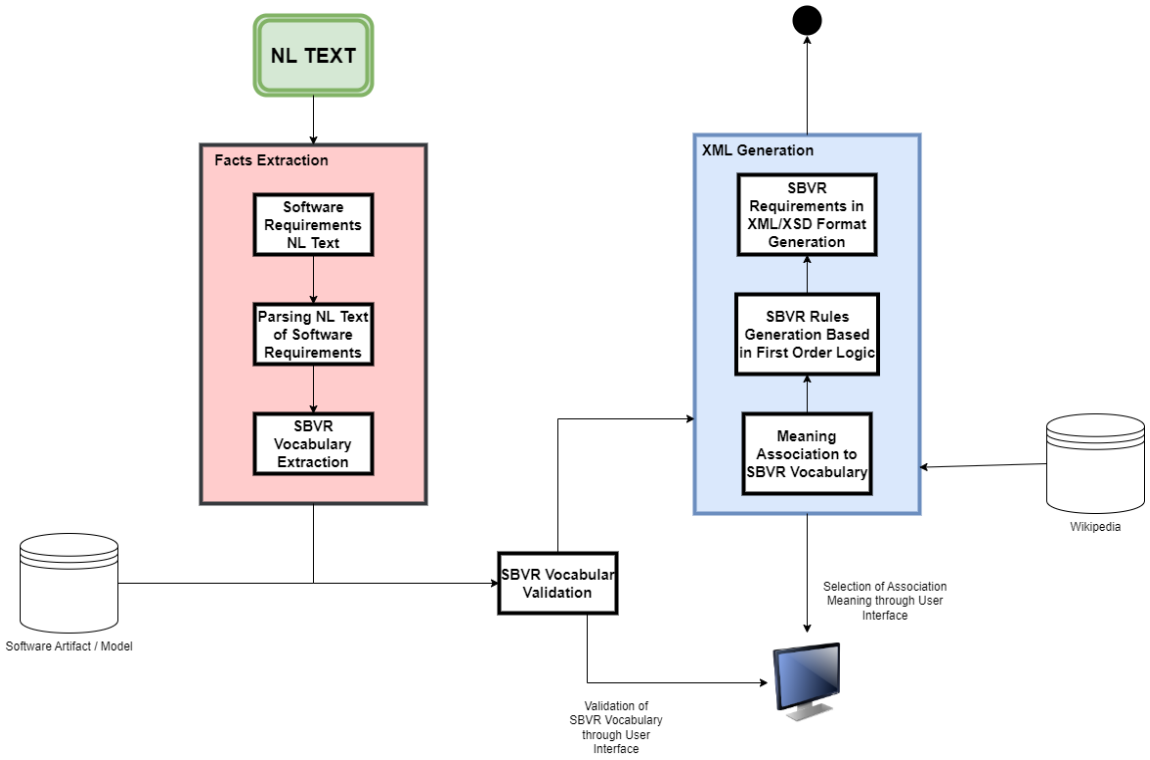


Figure 15 - Architecture of 3.3.2

Architecture, see Figure 15:

- Uses external tools; Wikipedia and Software Artifact / Model
- Needs help of a professional to confirm the results.
- Accepts a bigger range of parameters.

3.3.3 OCL Constraints Generation from Natural Language Specification

This paper presents a method to enhance the adoption of OCL (Object Constraint Language) in UML (Unified Modeling Language) creation by addressing its complex and unfamiliar syntax, which often discourages software engineers. The proposed method leverages natural language (NL) and Model Transformation technology to enable users to write constraints and pre/post conditions in simple English. These are automatically transformed into equivalent OCL statements through a framework designed to simplify the generation process and improve the usability of UML tools that support OCL [33].

The approach utilizes SBVR (Semantics of Business Vocabulary and Rules) to facilitate the formulation of natural language expressions in their transformation to OCL.

Key Points:

- To increase the adoption of OCL by simplifying its syntax using natural language.
- Natural language specifications provided by the user.
- Flexible POS Tagging (Part of Speech Tagging) to identify individual components of SBVR in the sentence.
- The prototype allows dynamic generation of OCL constraints from user-provided natural language specifications.

- The system identifies key linguistic elements such as nouns, verbs, and adjectives in the input text.

Practical Benefits:

- Users write constraints and conditions in simple English.
- SBVR is used to transform natural language expressions into structured rules, which are then converted into OCL expressions.
- Simplifies the creation of OCL constraints.
- Improves usability of UML tools supporting OCL.
- Facilitates better communication and understanding of constraints among stakeholders.

Limitations:

- The authors don't provide access to the tool.

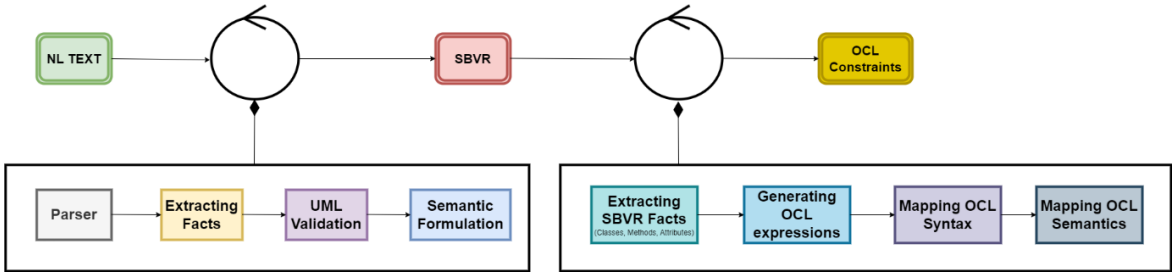


Figure 16 - Architecture of 3.3.3

Architecture, see Figure 16:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn't use external tools.

3.3.4 Generating SBVR-XML Representation of a Controlled Natural Language

This paper introduces an approach to convert SBVR (Semantics of Business Vocabulary and Rules) software requirements into an XML schema using the tool "VeTIS" as the foundation for "SBVR2XML". The process starts with generating SBVR rules as the initial output. These rules are then provided to an editor that extracts key elements from the SBVR vocabulary, such as noun concepts and fact types [46].

Key Points:

- To convert SBVR software requirements into an XML schema.

Practical Benefits:

- Produces clear and precise XML representations of business requirements.
- Facilitates better communication and understanding among stakeholders.
- Enhances the accuracy and usability of software requirement documentation.

Limitations:

- Translates SBVR into XML, without using natural language due to complications such as ambiguities, coherence issues, and references.
- The tool is not being updated.
- The tool is not open source.

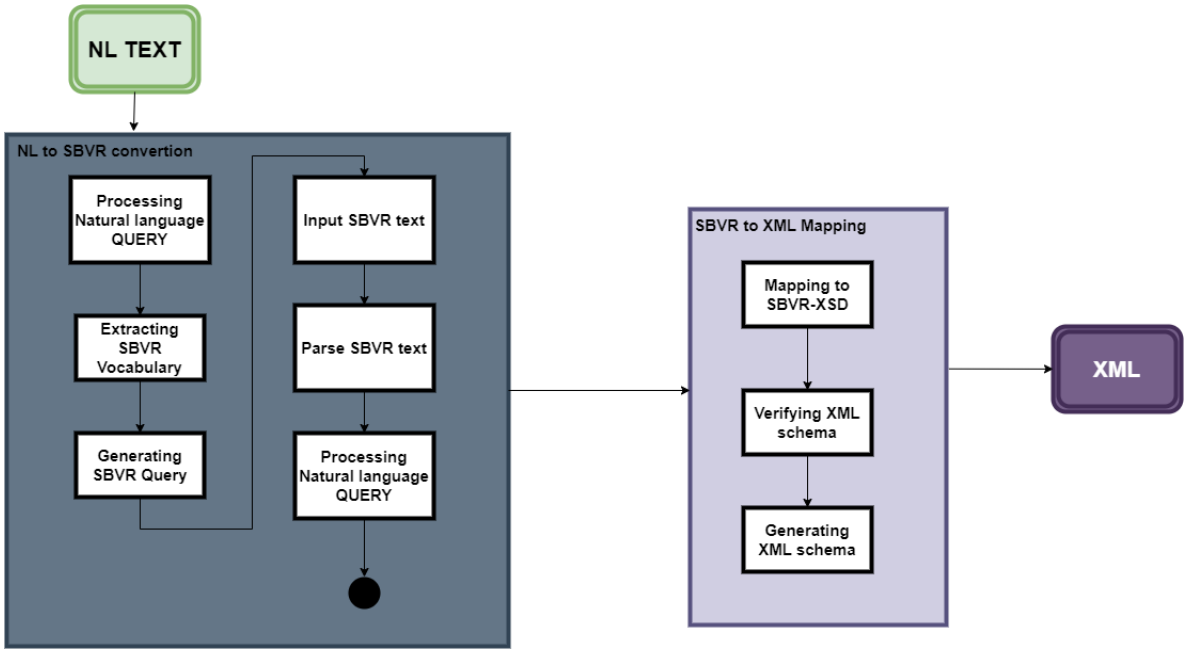


Figure 17- Architecture of 3.3.4

Architecture, see Figure 17:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn't use external tools.

3.3.5 Natural Language Techniques Supporting Decision Modelers

This paper presents an implementation to transform natural language (NL) into Decision Model and Notation (DMN). The implementation facilitates the automatic extraction and generation of rules and decision tables directly from natural language descriptions. The process operates through three phases: discourse and semantic analysis, syntactic analysis, and the construction of decision tables [47].

Key Points:

- To transform natural language descriptions into DMN, aiding decision modelers by simplifying the creation of decision models.
- Natural language descriptions in English.
- Flexible POS Tagging (Part of Speech Tagging) to identify individual components of SBVR in the sentence.

Practical Benefits:

- The process simplifies the task for analysts, reducing the effort needed to create precise decision models.
- Single-sentence analyses are conducted to extract these components efficiently.
- Simplifies the creation of decision models for analysts.
- Improves the accuracy and usability of decision tables.

Limitations:

- The authors don't provide access to the tool.
- Although the output (decision tables) is not our primary focus, the algorithms used in this paper are highly relevant for the final work, providing valuable insights and techniques for transforming natural language into structured decision models.

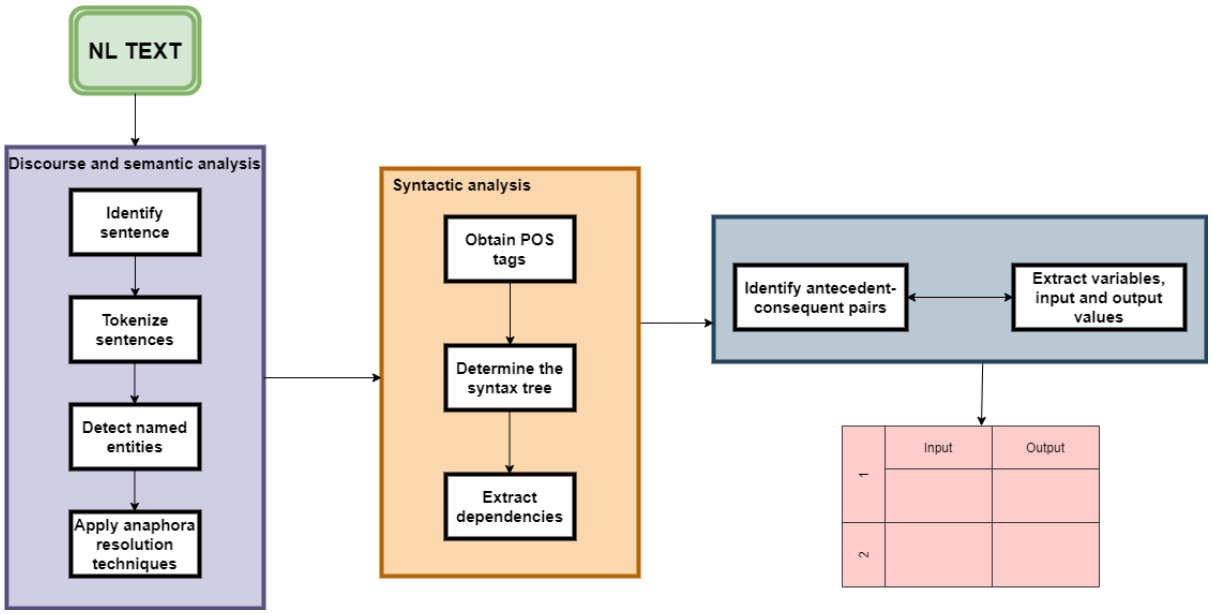


Figure 18 - Architecture of 3.3.5

Architecture, see Figure 18:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn't use external tools.
- Unconventional output.

3.3.6 An Approach to Mine Business Rule Intents from Domain-specific Documents

This paper describes a method for automatically extracting essential atomic facts, referred to as "business rule intents (Facts)," from each sentence. The method employs a dependency tree to analyze and extract these rule intents from business documents, primarily operational procedures, terms, and conditions written in natural language [48].

Key Points:

- To extract formal business rules and processes from domain-specific documents by systematically analyzing requirements documents.
- Natural language documents, including operational procedures, terms, and conditions.
- Flexible POS Tagging (Part of Speech Tagging) to identify individual components of SBVR (Semantics of Business Vocabulary and Rules) in the sentence.

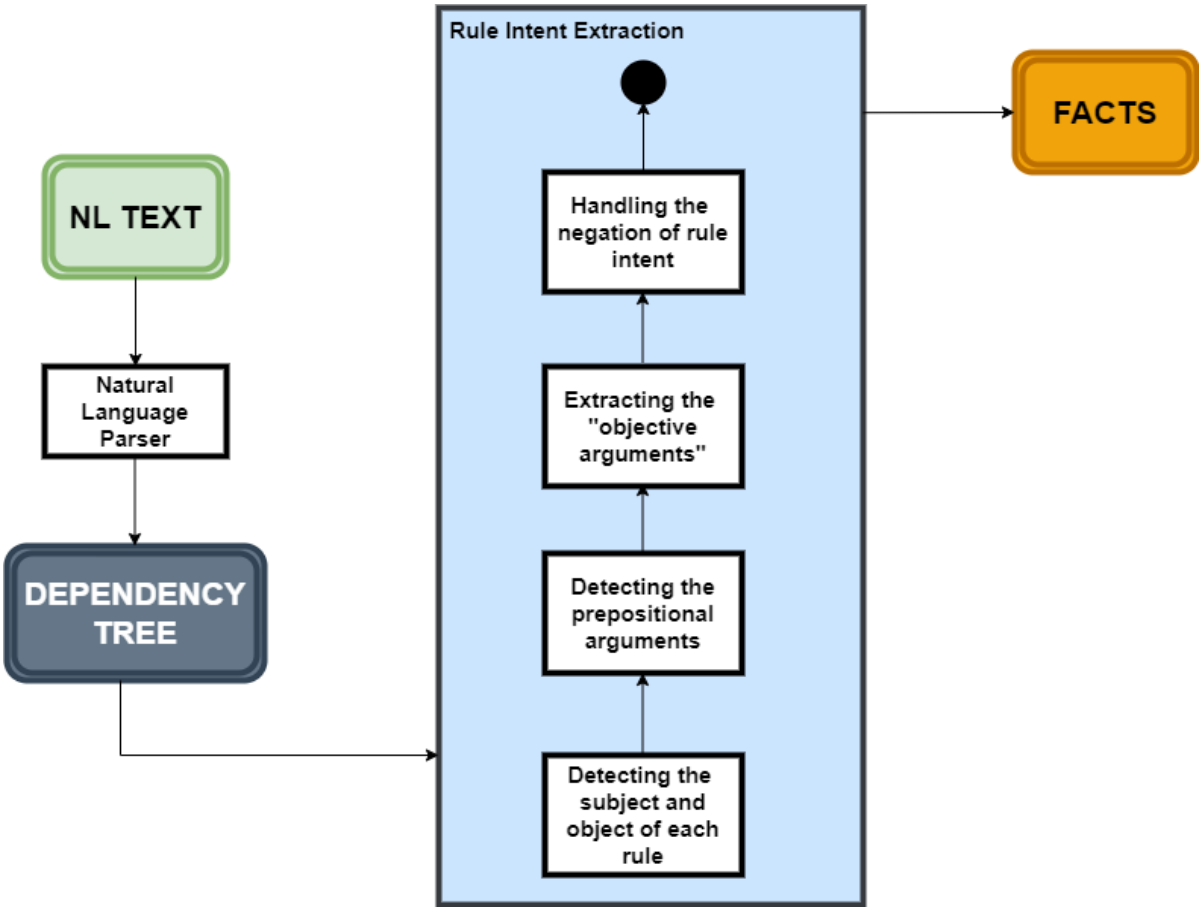


Figure 19 - Architecture of 3.3.6

Practical Benefits:

- Enhances the automation of extracting business rules and processes.
- Improves the clarity and formalization of business requirements.
- Facilitates better communication among stakeholders by providing a formal representation of business rules.

Limitations:

- The technique encounters difficulties with compound sentences, which may complicate the extraction process.
- The authors don't provide access to the tool.

Architecture, see Figure 19:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.

- Limited range of parameters.
- Doesn’t use external tools.

3.3.7 SBVR Business Rules Generation from Natural Language Specification

This paper presents an automated approach for translating natural language business rule specifications into SBVR (Semantics of Business Vocabulary and Rules), addressing the challenge of manual translation, which is typically laborious, time-consuming, and error prone. The approach uses a rule-based algorithm for robust semantic analysis of the English language to generate SBVR rules [49].

Key Points:

- To automate the translation of natural language business rule specifications into SBVR, reducing the effort, time, and errors associated with manual translation.
- Natural language (NL) specifications and XML (UML) inputs.
- Flexible POS Tagging (Part of Speech Tagging) to identify individual components of SBVR in the sentence.

Practical Benefits:

- Reduces the effort and time required for translating business rules.
- Minimizes errors in the translation process.
- Facilitates clearer and more precise business rule documentation.

Limitations:

- The paper provides limited information for forming a comprehensive opinion, needs the help of other papers in the area to show the bigger picture.
- The authors don’t provide access to the tool.

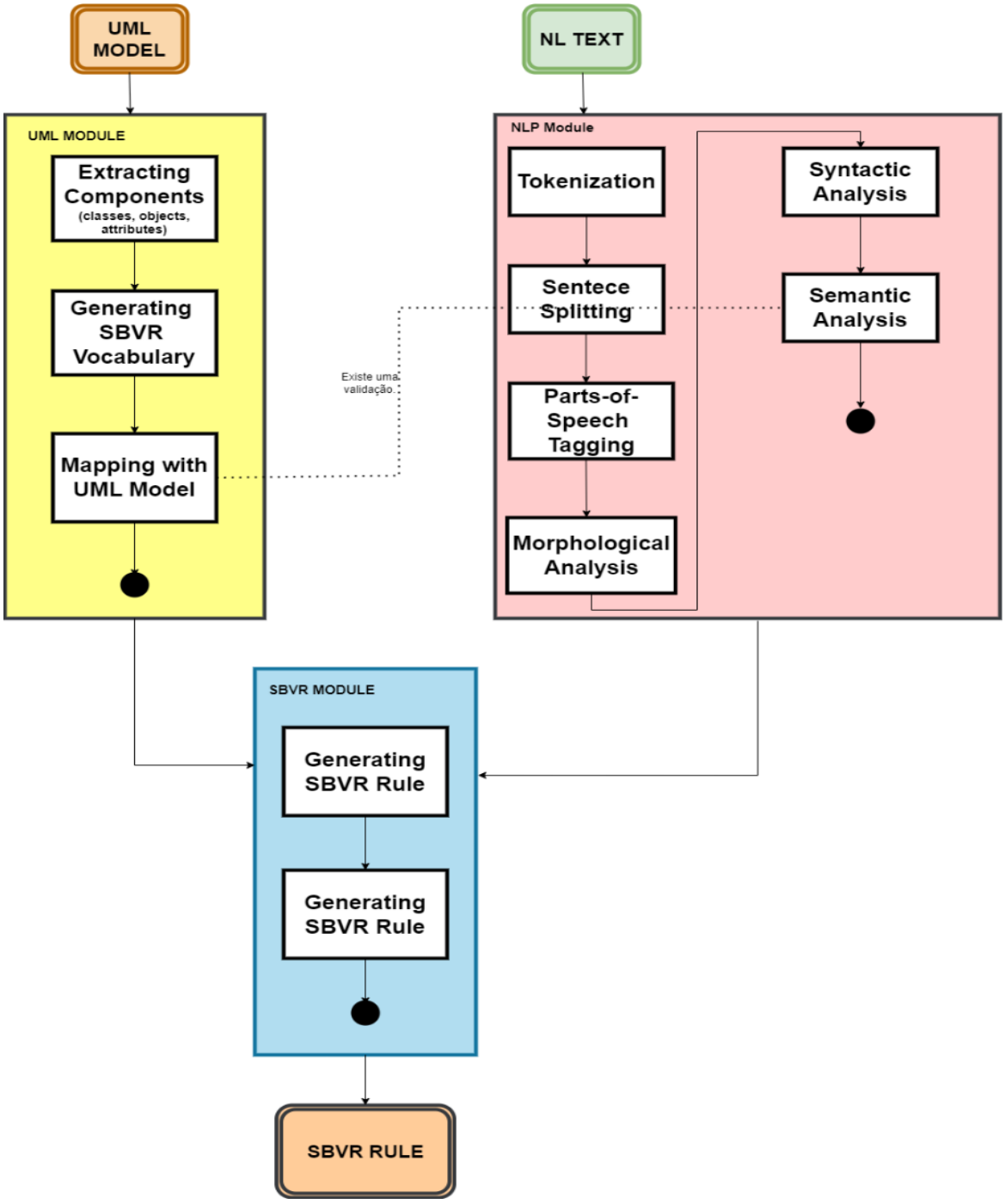


Figure 20 - Architecture of 3.3.7

Architecture, see Figure 20:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Use external tools, UML Model and the respective tool to process it.

3.3.8 An Approach to Mine SBVR Vocabularies and Rules from Business Documents

This paper presents an unsupervised method for extracting SBVR (Semantics of Business Vocabulary and Rules) vocabulary and rules from domain-specific business documents, addressing communication issues between IT professionals and business analysts caused by differing terminologies in their respective fields. The approach aims to minimize errors such as discrepancies and ambiguities in software systems, which often result from the manual extraction of rules by business analysts [50].

This approach provides a valuable solution for automating the extraction of SBVR vocabularies and rules from business documents, potentially improving efficiency and accuracy in business rule documentation and reducing the gap between IT and business domains [50].

Key Points:

- To automatically extract SBVR vocabulary and rules from business documents, improving communication between IT professionals and business analysts.
- Domain-specific business documents.

Practical Benefits:

- Does not require categorized training data.
- Is completely domain-independent, although this may result in a reduced accuracy rate.
- Reduces errors and ambiguities in software systems.
- Enhances clarity and consistency in business rule documentation.

Limitations:

- The current system does not address co-referencing issues due to limitations in the SpaCy library.
- Difficulties in defining and extracting complex sentences.
- The authors don’t provide access to the tool.

Architecture, see Figure 21:

- Standard architecture.
- Doesn’t need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn’t use external tools.
- Uses statistics to get a better understanding of solution.

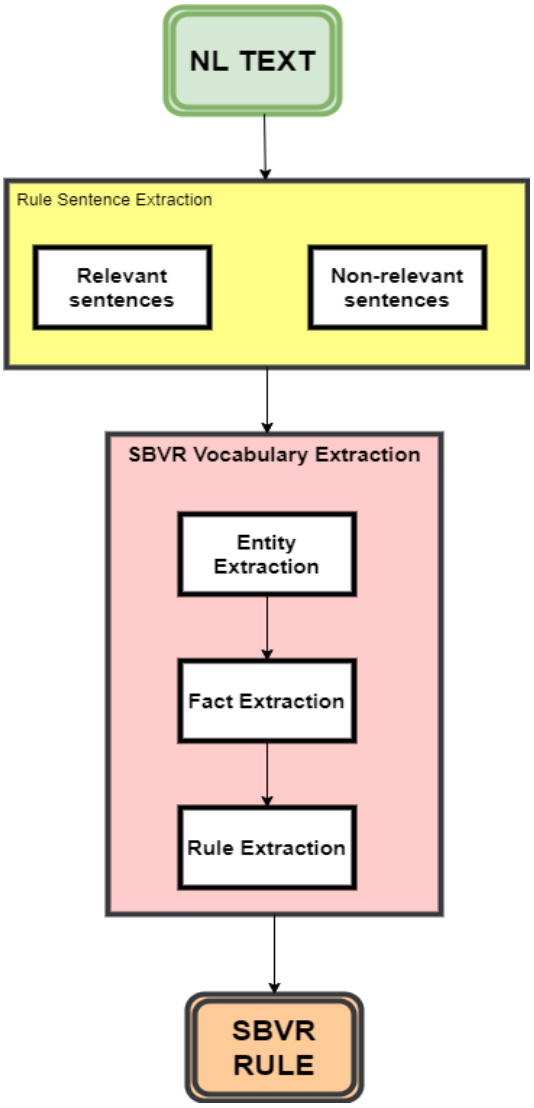


Figure 21 - Architecture of 3.3.8

3.3.9 Automated Generation of Terminological Dictionary from Textual Business Rules

This paper presents an automated approach for generating a business vocabulary from textual business rule statements, using the standards established by SBVR (Semantics of Business Vocabulary and Rules). The method employs a terminological dictionary that provides comprehensive meanings for each concept, utilizing advanced natural language processing (NLP) techniques. These techniques are not only used to compile a simple list of terms and relationships but also to derive extra specifications and implicit knowledge from the business rules [51].

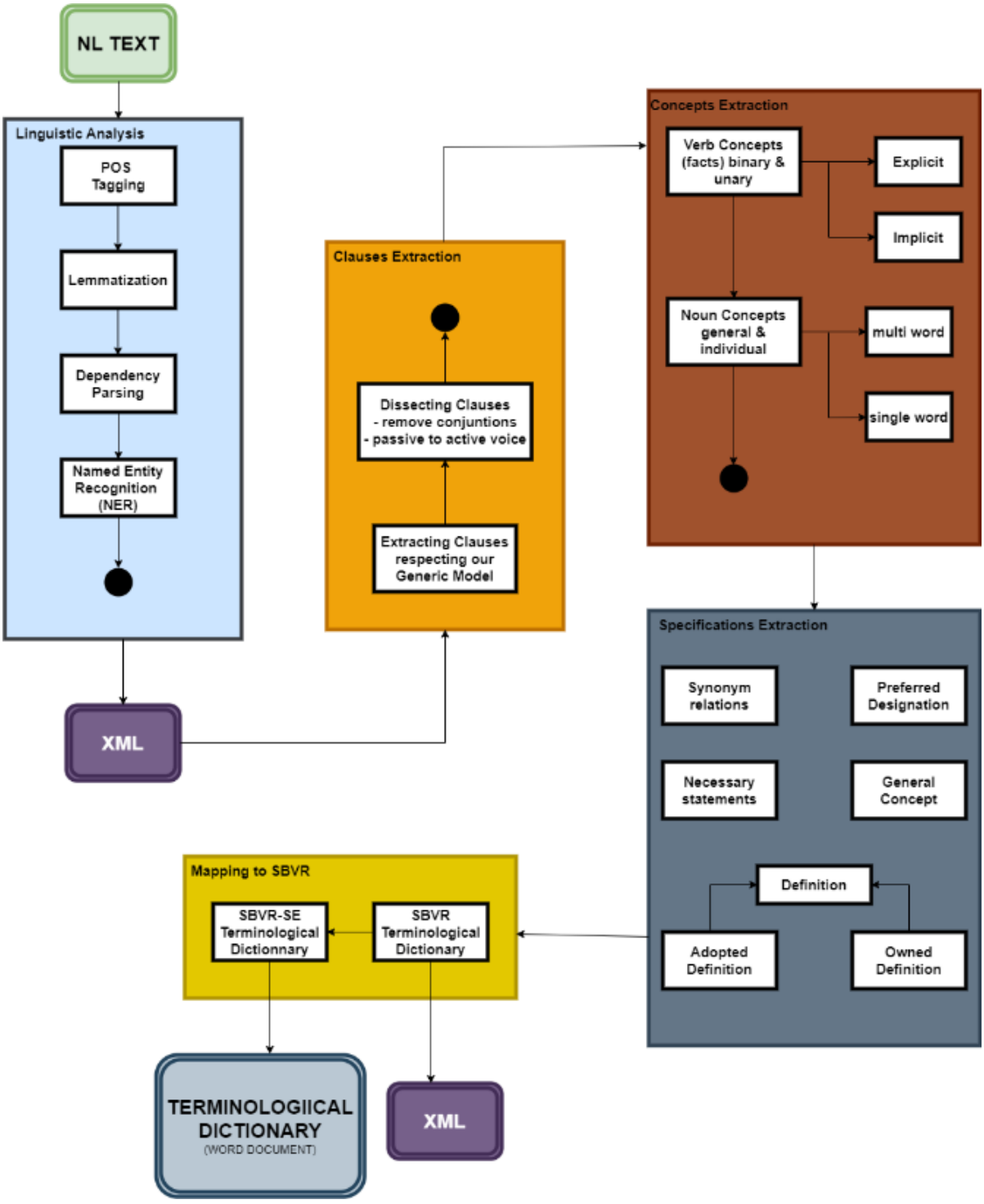


Figure 22 - Architecture of 3.3.9

Key Points:

- To automate the generation of a business vocabulary from textual business rules, enhancing clarity and understanding.
- Textual business rule statements.

Practical Benefits:

- Text2Model: Extracts business vocabulary directly from business rule statements.

- Generates a dictionary that includes explicit knowledge and implicit information.
- Enriched with definitions, synonyms, and generalization relationships.
- Enhances the usability of business vocabularies in various applications.
- Improves the clarity and precision of business rule documentation.

Limitations:

- The authors do not provide access to the tool.

Architecture, see Figure 22:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn't use external tools.

3.3.10 NLP for Automated Conceptual Data Modeling

This paper presents an automated approach for generating Entity-Relationship (ER) diagrams from natural language using syntactic and semantic heuristics. The process initially receives English paragraphs as input and creates summaries based on the principle of code quantity. These summaries are then used by the system to automatically form a database schema, identifying critical ER components such as entities, attributes, and relationships. The system effectively utilizes a combination of syntactic and semantic learning to facilitate the automatic generation of ER diagrams, culminating in the creation of an XMI file that can be visualized in any UML modeling tool [52].

Key Points:

- To automate the generation of ER diagrams from natural language descriptions, improving efficiency and accuracy in conceptual data modeling.
- English paragraphs describing the system or business domain.
- Uses word frequency analysis and POS Tagging to identify key components.

Practical Benefits:

- Reduces the manual effort required to create ER diagrams.
- Improves the accuracy and consistency of database schema generation.
- Facilitates better communication and understanding of data models among stakeholders.

Limitations:

- Uses word frequency analysis and POS Tagging. Note that this approach is more suitable for larger texts and may not perform well with shorter sentences.

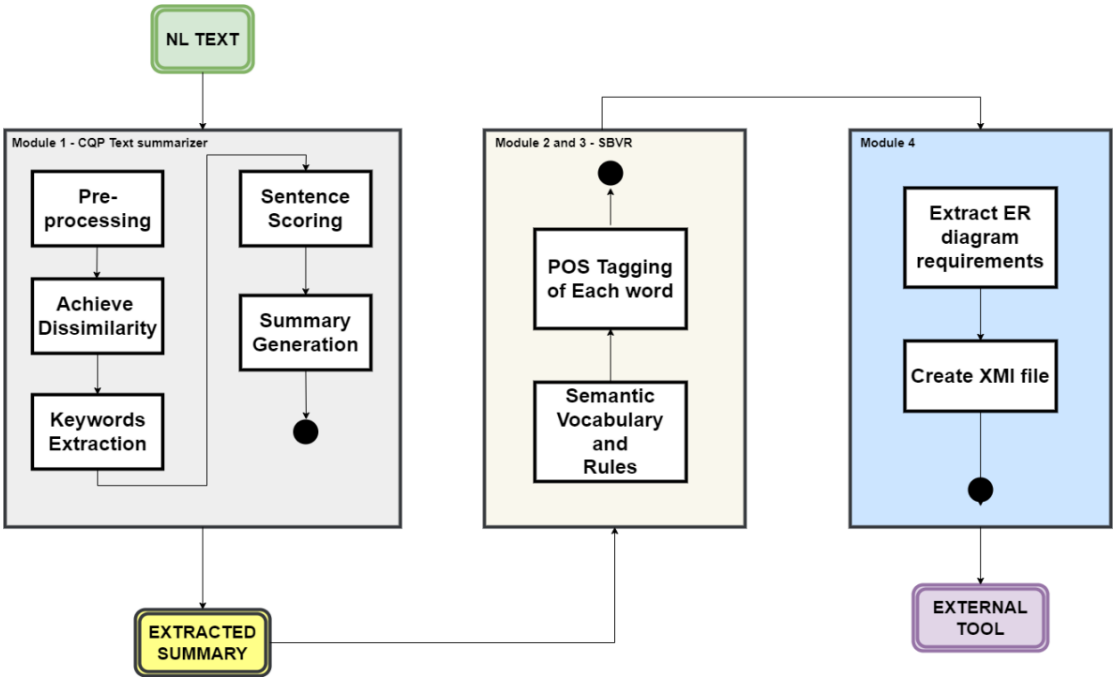


Figure 23 - Architecture of 3.3.10

Architecture, see Figure 23:

- Standard architecture.
- Doesn't need supervision of professional in the selected area of the input documents.
- Limited range of parameters.
- Doesn't use external tools.

3.3.11 VETIS Tool for Editing and Transforming SBVR Business Vocabularies and Business Rules into UML & OCL Models

This paper discusses the implementation and capabilities of the VeTIS tool, designed to edit Business Vocabularies and Business Rules and transform them into UML class models with integrated OCL (Object Constraint Language) constraints. VeTIS is integrated with the MagicDraw UML CASE tool and aims to simplify the development process by defining requirements through use cases and modeling business processes through activities. The tool addresses the complexity of the SBVR metamodel, which has hindered its widespread adoption despite growing interest in its application across various business communities [53].

Key Points:

- To facilitate the editing and transformation of SBVR business vocabularies and rules into UML class models with OCL constraints.
- SBVR Business Vocabulary and Business Rules in Natural Language (NL).
- VeTIS is embedded within MagicDraw to enhance its modeling capabilities.
- Integrates OCL constraints into the UML models.
- Allows users to edit SBVR business vocabularies and rules.
- Transforms these vocabularies and rules into UML class models.

Practical Benefits:

- Simplifies the process of translating business rules into formal models.
- Enhances the accuracy and consistency of business process modeling.
- Facilitates better communication and understanding of business rules among stakeholders.

Limitations:

- Does not show the implementations.
- The tool is not open source.

3.4 Result Analysis

The following table (Table 5) presents the main characteristics identified in each paper providing an overview of the contents.

Table 5 - Main characteristics of each paper

Ref.	**Input**	**Output**	**POS-Tagging**	**Matcher**	**Word frequency**	**Access to Code**
[44]	NL	SBVR	Non-flexible	✓	×	×
[45]	NL	XML	Flexible	×	×	×
[33]	NL	OCL Constraints	Flexible	×	×	×
[46]	SBVR	XML	Non-flexible	✓	×	×
[47]	NL	Decisions Table	Flexible	×	×	×
[48]	NL	SBVR	Flexible	×	×	×
[49]	XML(UML), NL	SBVR	Flexible	×	×	×
[50]	NL	SBVR	Flexible	×	×	×
[51]	NL	XML	Flexible	×	×	×
[52]	NL	XML	Flexible	×	✓	×
[53]	SBVR, NL	UML class model w/ OCL	undefined	undefined	undefined	×

Flexible - Uses a greater number of functionalities of the POS Tagging is a better implementation to minimize code but increases complexity of the solution.

Non-Flexible - Uses the minimum functionality of the POS Tagging, instead uses a Matcher to make the bulk of the work, decreases complexity of the solution but increases the quantity of code required

3.4.1 Observations

Code Availability –

One of the biggest problems that we encountered was the lack of access to the code. The only opportunity we had to observe the code written by the authors of the different papers was through snippets, which does not allow for a clear or complete exploration to fully understand the tool. We believe the reason for this absence of code is that, out of the 11 selected papers, only one represents a fully constructed tool, while the rest are still in an early stage of the project, with many of the requirements yet to be defined and built.

Outputs-

The methodologies present in the papers are SBVR, OCL, and DMN. Out of these, we decided to choose SBVR because it is the methodology that is the closest to what we want to achieve. Our goal is to analyze text to extract information, which aligns well with the SBVR methodology.

The issue with the other methodologies is that they are more specific to problems outside the scope of our work. For example, BPMN, OCL, and DMN aim to create diagrams for business use, whereas we want to extract information from text that can later be processed into machine language. We are aiming for a more generic process, which is why we chose SBVR. Additionally, since there are no good techniques yet for converting SBVR into BPMN or DMN, we decided to exclude papers that did not use SBVR in their development.

3.5 Conclusion

With this literature review, we were able to understand how to break the problem into smaller parts, allowing for a greater comprehension of the issue. Additionally, the interpretation of each author's solution gives us the opportunity to utilize the best aspects of each while minimizing their disadvantages. With this approach in mind, we aim to produce a superior result compared to the previously described papers.

To conclude the literature review, we identified several important points:

1. **Lack of Source Code and Architectures**: Many papers did not provide source code or detailed architectures, making it challenging to verify their results and compare them effectively. As a result, only 11 out of 56 papers were accepted for detailed analysis.
2. **Popular Tools Highlighted**: We focused on the most popular tools. Although several papers mentioned a variety of tools, our research showed that these were no longer receiving updates. Therefore, we did not consider them important to mention. We will be explaining the best tools in the next section 4.4 Tools.
3. **Quality Papers Identified**: Despite the challenges, we found some high-quality papers that we deemed valuable. These papers provided detailed explanations of best practices and tested architectures in the field of NLP.

Despite the issue mentioned in the "Code Availability" section, which delayed the development of our work, the literature review generally provided us with a good foundation to build on. We were able to understand which tools and methodologies are the most popular in the field and in what situations they are appropriate to use, considering the problems they address.

4 Development

In this section of the dissertation, we will discuss the development of our tool, focusing on the technologies adopted, the problems we need to solve, the chosen architecture, and the limitations of the tool.

It is important to keep in mind that, as this work is a proof of concept, the tool we are creating should not be viewed as a final version. Considering this, and due to time constraints, we were forced to reduce the level of rigor in certain phases of the software development process.

4.1 Objectives

To explain the functionality of the tool and the underlying concepts, we will utilize Jupyter Notebooks, which provides a robust visualization capabilities. These visualizations will be instrumental in demonstrating the functionality of the SpaCy library and the various NLP concepts integrated into our tool.

The primary objective of our tool is to extract crucial information from text. Specifically, we aim to identify key points of business rules so that this information can be translated into other computer languages, such as SQL, JavaScript, XML, etc. However, due to time constraints, the current iteration of the tool will focus solely on converting business rules expressed in natural language into SQL statements. This focused approach will allow us to refine the tool’s functionality in handling this specific task efficiently.

4.2 Development Process

In order to create a better relation between all the produced work, the present section was created to provide an overview of the different steps of the chosen development process.

4.2.1 Requirements

In this phase, the primary goal is to understand and define the specific needs and objectives for the thesis. We gathered the necessary functional and non-functional requirements, ensuring that they align with the project's goals. This includes identifying key features, user interactions, performance expectations, and any technical constraints that must be considered during development. Chapters “1 Introduction”, “2 State Of The Art”, and “4.3 Requirements” also relate to this step of the process.

4.2.2 Analysis

The analysis phase focuses on breaking down the problem into smaller, manageable components that we can use to create string for research purposes. Where, we examined the papers in detail to understand the underlying challenges, evaluate potential risks, and determine the feasibility of proposed solutions. This step provides a clearer picture of how to approach

the development and ensures that each aspect of the problem is addressed. This step more closely relates to Chapter “3 Literature Review” and Section “4.4 Tool’s”.

4.2.3 Design

During the design phase, we created the system's architecture and structure based on the analysis results. This includes defining the components, their interactions, and how they will function together. This step will be presented in the Section “4.5 Architecture”.

4.2.4 Implementation

In this phase, the actual coding and construction of the system was explained. Based on the design specifications ensuring that all requirements where met. The implementation was done iteratively, with a continuous testing and adjustments to ensure that the system operates as required. The section explaining the implementation is Section “4.6 Code Implementation”

4.2.5 Testing

Regarding the testing phase, we were only able to verify that it meets the proposed minimum requirements (see 4.3.1 The minimum complexity that the tool should be able to process), as time was limited in order to create and conduct more complex tests. Example of test results are presented in Sections “4.6.1 NLP Implementation” e “4.6.2 SQL Implementation”.

4.3 Requirements

For creating the input for the module of NLP, we need to use a tool called HydraCGT, where there will be several graphs representing the system's functionalities. These modules will have the logic represented in the form shown in Figure 24.

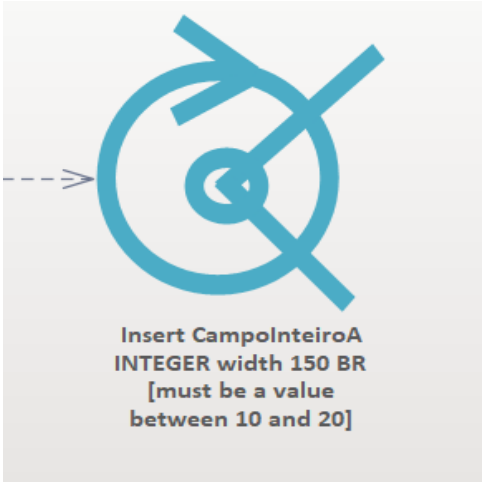


Figure 24 - User Interaction where the BR is defined

To create a BR, the user only needs to modify the field, which in this case, according to Figure 24, is "[must be a value between 10 and 20]." The advantage of this approach is that the user does not need to consider the system's different variables, such as "CampoInteiroA." They

need to use Hydra, which presents the system in an organized and clear way, minimizing human error.

Since we know the basic structure (as it must follow the rules defined in Hydra), we can extract the minimum requirements that our prototype must be able to address to ensure the work is considered acceptable.

4.3.1 The minimum complexity that the tool should be able to process.

What we will discuss next considers that the output will be in SQL statements. Considering the types of rules present in the Hydra, we can conclude that at a minimum the tool must detect the names of tables and their attributes and be able to distinguish between them. This will be true for all of Chapter 4.

 # >=10 && <=20
The CampoInteiroA of TTabelaRegistos must be a value between 10 and 20.

 # >10
The CampoInteiroA of TTabelaRegistos must be a value bigger than 10.
The CampoInteiroA of TTabelaRegistos must be bigger than 10.

 # >=10
The CampoInteiroA of TTabelaRegistos must be a value bigger or equal to 10.

 # <20
The CampoInteiroA of TTabelaRegistos must be a value lower than 20.
The CampoInteiroA of TTabelaRegistos must be a value smaller than 20.

 # <=20
The CampoInteiroA of TTabelaRegistos must be lower or equal to 20.
The CampoInteiroA of TTabelaRegistos must be smaller or equal to 20.

The CampoTextoA of TTabelaRegistos must not exceed 200 characters.
The CampoTextoA of TTabelaRegistos must not exceed 200 chars.

 # >=5
The CampoTextoA of TTabelaRegistos must have at least 5 characters.
The CampoTextoA of TTabelaRegistos must exceed 5 characters.
Each TTabelaRegistos must have no more than 2 TTabelaSubRegistos

4.4 Tool's

In this chapter we will be discussing the various tools used by the authors of the papers in our literature review.

4.4.1 NLTK

The information presented for the NLTK library was retrieved from the official page of the developers.

The Natural Language Toolkit (NLTK) was established in 2001 as a component of a computational linguistics course within the Department of Computer and Information Science at the University of Pennsylvania. Since its creation, NLTK has undergone continuous development and expansion due to the contributions of numerous individuals. Presently, it is extensively utilized in university courses worldwide and serves as a fundamental instrument for various research initiatives [99].

The field of Natural Language Processing (NLP) possesses extensive applicability across scientific, economic, social, and cultural domains and is experiencing rapid growth as its theories and methods are progressively integrated into emerging technologies. In the academic realm, NLP is indispensable for scholars in areas such as humanities computing, corpus linguistics, computer science, and artificial intelligence. Within the scholarly community, NLP is frequently referred to as "Computational Linguistics." [99].

The widespread adoption and ongoing development of NLTK underscore its significance in advancing the field of NLP. As NLP continues to develop, the toolkit remains an essential resource for educational and research innovations, particularly in the educational sector [99], see Table 6.

4.3.1.1 Language processing tasks and corresponding NLTK modules with examples of functionality:

Table 6 - NLTK functionality [99]

Language processing task	NLTK modules	Functionality
Accessing corpora	corpus	standardized interfaces to corpora and lexicons
String processing	tokenize, stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	tag	n-gram, backoff, Brill, HMM, TnT
Machine learning	classify, cluster, tbl	decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	chunk	regular expression, n-gram, named-entity
Parsing	parse, ccg	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	sem, inference	lambda calculus, first-order logic, model checking
Evaluation metrics	metrics	precision, recall, agreement coefficients
Probability and estimation	probability	frequency distributions, smoothed probability distributions
Applications	app, chat	graphical concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	toolbox	manipulate data in SIL Toolbox format

4.4.1.2 The design of the Natural Language Toolkit (NLTK) was guided by four primary objectives:

1.Simplicity: NLTK aims to provide an intuitive framework that includes substantial building blocks, allowing users to gain practical knowledge of Natural Language Processing (NLP) without becoming overwhelmed by the tedious housekeeping tasks typically associated with processing annotated language data. The focus on simplicity ensures that users can quickly grasp and apply NLP concepts and techniques [99].

2.Consistency: NLTK offers a uniform framework with consistent interfaces and data structures, alongside easily guessable method names. This consistency facilitates ease of use and reduces the learning curve for users, enabling them to focus on developing and applying NLP solutions [99].

3.Extensibility: NLTK is designed to accommodate new software modules seamlessly, including alternative implementations and competing approaches to the same task. This

extensibility ensures that the toolkit remains relevant and adaptable to the evolving needs and advancements in the field of NLP [99].

4.Modularity: NLTK provides components that can be used independently, without requiring a comprehensive understanding of the entire toolkit. This modularity allows users to utilize specific parts of NLTK for their projects, enhancing flexibility and usability [99].

4.4.1.3 Limitations of NLTK:

While NLTK is efficient enough to support meaningful NLP tasks, it is not highly optimized for runtime performance. Achieving such optimizations often involves more complex algorithms or implementations in lower-level programming languages such as C or C++. Consequently, users may encounter performance limitations when working with large datasets or computationally intensive tasks. Despite this, NLTK remains a valuable resource for educational purposes and for developing a foundational understanding of NLP techniques [99].

4.4.2 SpaCy

The information presented for the SpaCy library was retrieved from the official page of the developers [100].

SpaCy is a free, open-source library designed for advanced Natural Language Processing (NLP) in Python, it is specifically intended for production use. The tool can support the development of applications that handle large volumes of text and is particularly well-suited for constructing information extraction or natural language understanding systems [100].

4.4.2.1 SpaCy Overview [100]:

- **Language Support**
  - Supports over 75 languages
  - 84 trained pipelines for 25 languages
- **Modeling and Learning**
  - Multi-task learning with pretrained transformers like BERT
  - Access to pretrained word vectors
- **Performance**
  - State-of-the-art processing speed
  - Production-ready training system
- **Linguistic Capabilities**
  - Linguistically motivated tokenization
  - Components for:
    - Named Entity Recognition (NER)
    - Part-of-Speech (POS) tagging
    - Dependency parsing
    - Sentence segmentation
    - Text classification
    - Lemmatization
    - Morphological analysis
    - Entity linking
- **Extensibility**

    - Easily extensible with custom components and attributes
    - Supports custom models in frameworks like PyTorch and TensorFlow
- **Visualization and Deployment**
    - Built-in visualizers for syntax and NER
    - Simplified model packaging, deployment, and workflow management
- **Accuracy and Reliability**
    - Robust, rigorously evaluated accuracy

SpaCy is designed to streamline the development of production-ready NLP applications, offering a comprehensive suite of tools and features for advanced text processing and analysis [100].

4.4.2.2 Architecture:

The core data structures in SpaCy comprises the Language class, the Vocab, and the Doc object. The Language class processes text, converting it into a Doc object, which is typically stored in a variable named 'nlp'. The Doc object manages the sequence of tokens and all associated annotations. By centralizing strings, word vectors, and lexical attributes in the Vocab, SpaCy efficiently conserves memory by avoiding redundant data copies, thereby maintaining a single source of truth [100].

Furthermore, text annotations in SpaCy are structured to uphold this principle of a single source of truth: the Doc object retains ownership of the data, while Span and Token act as views into this data, see Figure 25. The Doc object, initially created by the Tokenizer, is subsequently modified in place by various pipeline components. The Language object harmonizes these components, managing the processing of raw text through the pipeline and returning an annotated document. It also supervises the training and serialization processes [100].

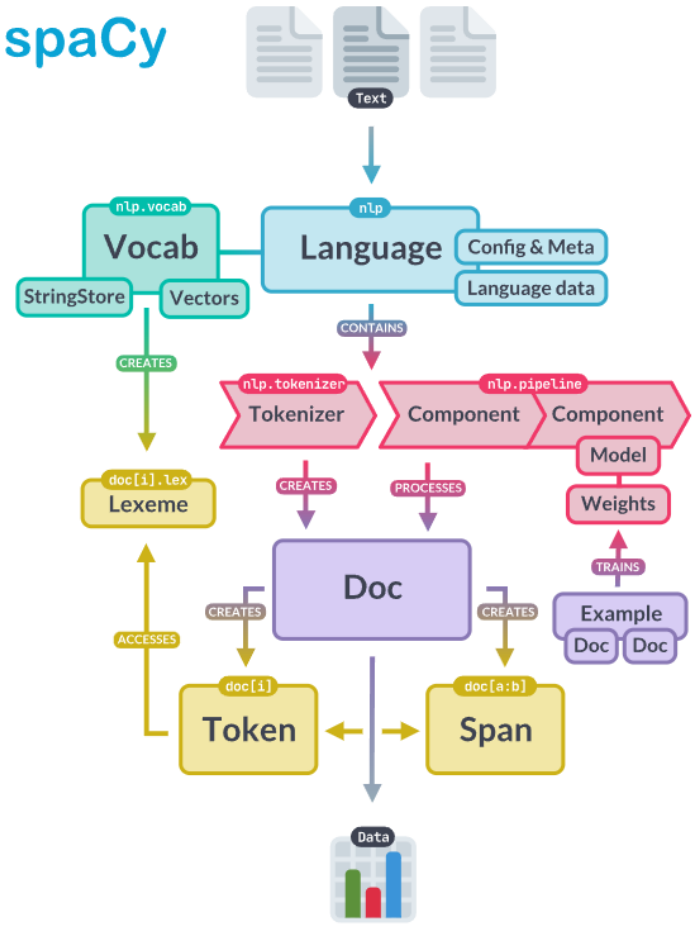


Figure 25 - SpaCy Model [100]

4.4.2.3 Pipelines:

While some of SpaCy's features operate independently, others necessitate the loading of trained pipelines to predict linguistic annotations, such as identifying whether a word functions as a verb or a noun. A trained pipeline may consist of multiple components that utilize a statistical model trained on labeled data. SpaCy currently provides trained pipelines for various languages, available as individual Python modules. The characteristics of these pipeline packages, including size, speed, memory usage, accuracy, and the included data, vary significantly. The selection of a specific package should be tailored to the user's application needs and the characteristics of the texts being processed. For general-purpose applications, the smaller, default packages typically serve as an effective starting point [100].

They typically include the following components:

- **Binary weights for the part-of-speech tagger**, dependency parser, and named entity recognizer to predict those annotations in context [100].
- **Lexical entries in the vocabulary**, such as words and their context-independent attributes like shape or spelling [100].
- **Data files** like lemmatization rules and lookup tables [100].
- **Word vectors**, which are multi-dimensional meaning representations of words that help determine their similarity to each other [100].

- **Configuration options**, including language and processing pipeline settings and model implementations, to set SpaCy to the correct state when you load the pipeline [100].

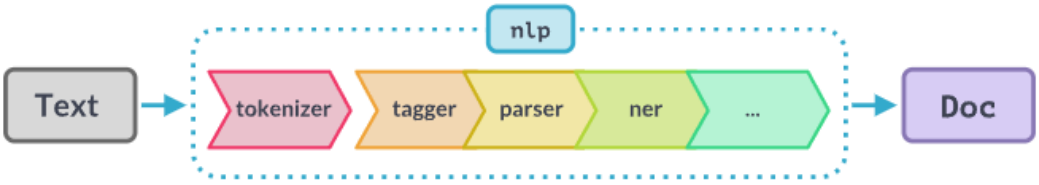


Figure 26 - Model Pipeline [100]

When you call 'nlp' on a text, SpaCy first tokenizes the text to produce a Doc object. The Doc is then processed in several steps, referred to as the processing pipeline. The trained pipelines typically include a tagger, a lemmatizer, a parser, and an entity recognizer. Each pipeline component processes the Doc and passes it on to the next component [100], see Figure. 26 and Figure. 27.

```
============================= Pipeline Overview =============================

#   Component        Assigns              Requires   Scores             Retokenizes
-   --------------   ------------------   --------   ----------------   -----------
0   tok2vec          doc.tensor                                         False

1   tagger           token.tag                       tag_acc            False

2   parser           token.dep                       dep_uas            False
                     token.head                      dep_las
                     token.is_sent_start             dep_las_per_type
                     doc.sents                       sents_p
                                                     sents_r
                                                     sents_f

3   attribute_ruler                                                     False

4   lemmatizer       token.lemma                     lemma_acc          False

5   ner              doc.ents                        ents_f             False
                     token.ent_iob                   ents_p
                     token.ent_type                  ents_r
                                                     ents_per_type
```

Figure 27 - Using SpaCy "Model Pipeline"

4.4.2.4 Limitations:

SpaCy focuses on natural language processing and extracting information from large volumes of text. While it can assist in rewriting existing text, it doesn't include specific functionality for language generation tasks.

SpaCy is built on the latest research, but it is not a research library. If your goal is to write papers and run benchmarks, SpaCy might not be the best choice. However, you can use it to make your research results easily accessible to others, for example, by creating a custom SpaCy component. This leads to different design decisions compared to platforms like **NLTK** or **CoreNLP**, which are geared toward teaching and research. The main difference is that SpaCy is integrated and opinionated, aiming to avoid presenting users with multiple algorithms that

serve the same function. By keeping choices limited, SpaCy aims to deliver better performance and a smoother developer experience.

4.4.3 Stanford CoreNLP

The information presented for the Stanford CoreNLP library was retired from the official page of the developers.

CoreNLP is a comprehensive solution for natural language processing in Java! It allows users to obtain various linguistic annotations for text, such as token and sentence boundaries, parts of speech, named entities, numeric and time values, dependency and constituency parses, coreference resolution, sentiment analysis, quote attributions, and relations. CoreNLP currently supports eight languages: Arabic, Chinese, English, French, German, Hungarian, Italian, and Spanish [101].

4.4.3.1 Overview:

I won't discuss the Stanford CoreNLP library in detail because we consider the NLTK and SpaCy libraries to be better options. The Stanford CoreNLP library does not offer better features than those already available in the other two libraries. On the contrary, it offers fewer possibilities in terms of language support and architectures.

The reason for mentioning the Stanford CoreNLP library in the thesis is for the sake of completeness. Several papers have utilized this library, and thus, it is important to acknowledge its use in the literature.

4.4.4 WordNet

The information presented for the WordNet library, was retired from the official page of the developers.

WordNet is a lexical database for the English language. It organizes nouns, verbs, adjectives, and adverbs into sets of cognitive synonyms, known as synsets, each representing a distinct concept. These synsets are interconnected through conceptual-semantic and lexical relations, forming a network of words and concepts that are meaningfully related and can be explored using a browser or by downloading the database [102].

Although WordNet is like a **thesaurus*** in that it groups words based on their meanings, there are important differences. WordNet labels the semantic relations between words, unlike a thesaurus, which groups words solely based on similarity of meaning without following any explicit pattern [102].

***Thesaurus Definition:**

A thesaurus is a reference tool that lists words grouped according to similarity of meaning (synonyms and sometimes antonyms). Unlike a dictionary, which provides definitions and usage examples, a thesaurus provides lists of words that can be used interchangeably in specific

contexts. Thesauruses can be useful for finding the most appropriate word for a particular context, avoiding repetition, and expanding vocabulary.

For example, in a thesaurus, the word "happy" might be listed with synonyms such as "joyful," "content," "pleased," and "cheerful." Some thesauruses also include antonyms, which are words with opposite meanings, such as "sad" for the word "happy."

4.4.4.1 Overview:

WordNet is not a standalone tool used in NLP projects. It works as a complement to other tools like NLTK and SpaCy.

WordNet provides a rich lexical database that enhances the capabilities of these tools by offering a detailed network of synonyms, antonyms, hyponyms, and hypernyms. When integrated with NLTK (Natural Language Toolkit), WordNet can be used for tasks such as word sense disambiguation, semantic similarity analysis, and text classification. Similarly, in SpaCy, WordNet can enhance the functionality of natural language understanding tasks by providing semantic relationships and aiding in entity recognition and sentiment analysis. This integration helps in creating more accurate and context-aware NLP applications.

4.4.5 Conclusion

For the development of our natural language processing (NLP) tool, we decided to use the SpaCy library. SpaCy is widely recognized as one of the best options in the market for NLP due to its robustness, efficiency, and ease of use. This library is optimized for real-world applications, offering a wide range of components and pre-trained models that simplify the implementation of complex NLP tasks.

We also considered using the Natural Language Toolkit (NLTK) library. While NLTK is a powerful library extensively used in scientific research, it lacks pre-trained models, requiring users to create and train their own models from scratch. This can be advantageous for research and experimentation, where flexibility and control over model training are essential. However, for practical and commercial applications where efficiency and quick implementation are crucial, SpaCy stands out.

Additionally, SpaCy allows the training of custom models with user-specific data, which is extremely useful for tailoring the tool to the needs of a project or domain.

Another significant advantage of SpaCy is its integration with other tools and libraries, such as library WordNet, facilitating the construction of sophisticated NLP processing pipelines.

An additional factor that makes SpaCy particularly attractive is its excellent compatibility with Jupyter Notebook. Using Jupyter, we can leverage the interactive capabilities of the *displaCy* visualizer, which is an integrated SpaCy tool for understanding and visualizing how different NLP techniques work. This not only helps in comprehending the processes involved but also significantly aids in the development and debugging of the tool, allowing for faster and more visual experimentation and iteration.

In conclusion we chose SpaCy for its combination of high-quality pre-trained models, ease of use, custom training capability, and excellent compatibility with interactive

visualization tools like Jupyter Notebook and displaCy. This makes SpaCy the ideal choice for developing an efficient and adaptable NLP tool for real-world applications.

4.5 Architecture

This initial architecture represented our first design tested in production, see Figure 28. Despite achieving positive outcomes, we identified several issues substantial enough to merit the exploration of an alternative implementation. The primary challenge arose from utilizing the matcher throughout all stages of the process. Although functional, the extensive amount of code necessary to accommodate all desired options would have complicated the maintenance and iteration processes excessively. This realization prompted us to devise an alternative strategy.

4.5.1 Old architecture:

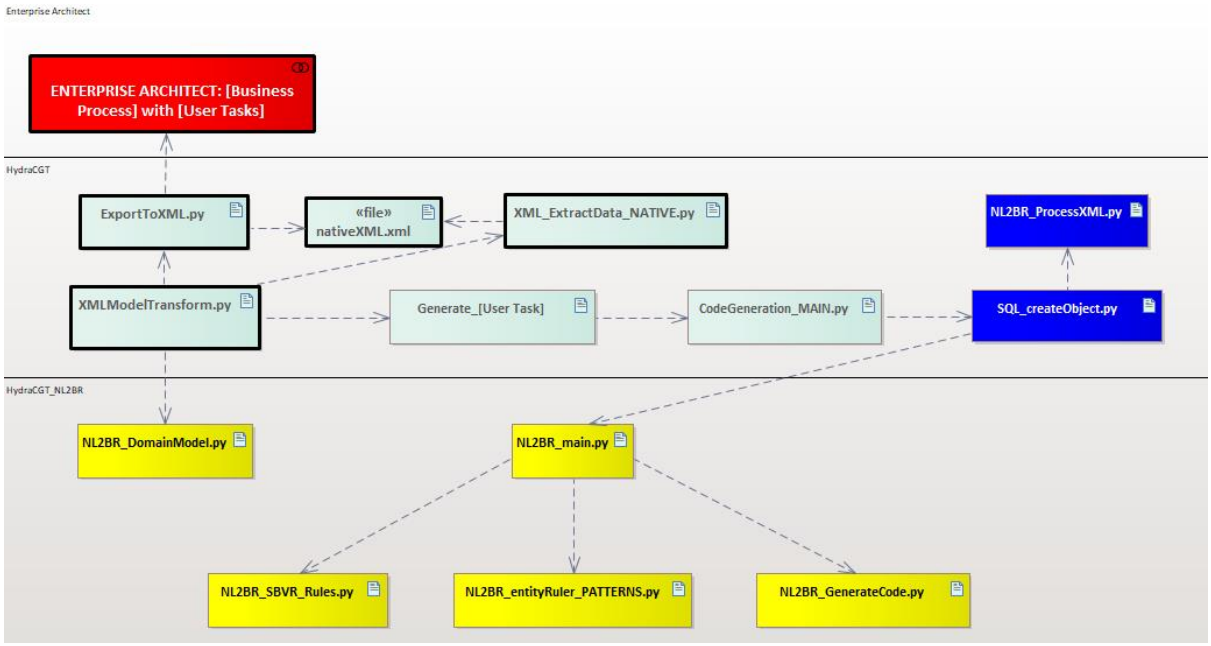


Figure 28 - Old Architecture

Due to time constraints and the fact that we already had a prototype in production, we decided to modify only the NLP transformation module, specifically the modules "NL2BR_main.py", "NL2BR_SBVR_Rules.py", and "NL2BR_entetyRuler_PATTERNS.py", while leaving the SQL creation module, "NL2BR_GenerateCode.py", untouched.

We chose to make greater use of the POS-Tagging functionality, which required us to address the issue of complex sentences [44]. This increased the number of sentences we could process. However, as we did not modify the "NL2BR_GenerateCode.py" module, we had to adapt the output of the new architecture's "NL2BR_main.py" module to be compatible with the existing modules.

4.5.2 New architecture:

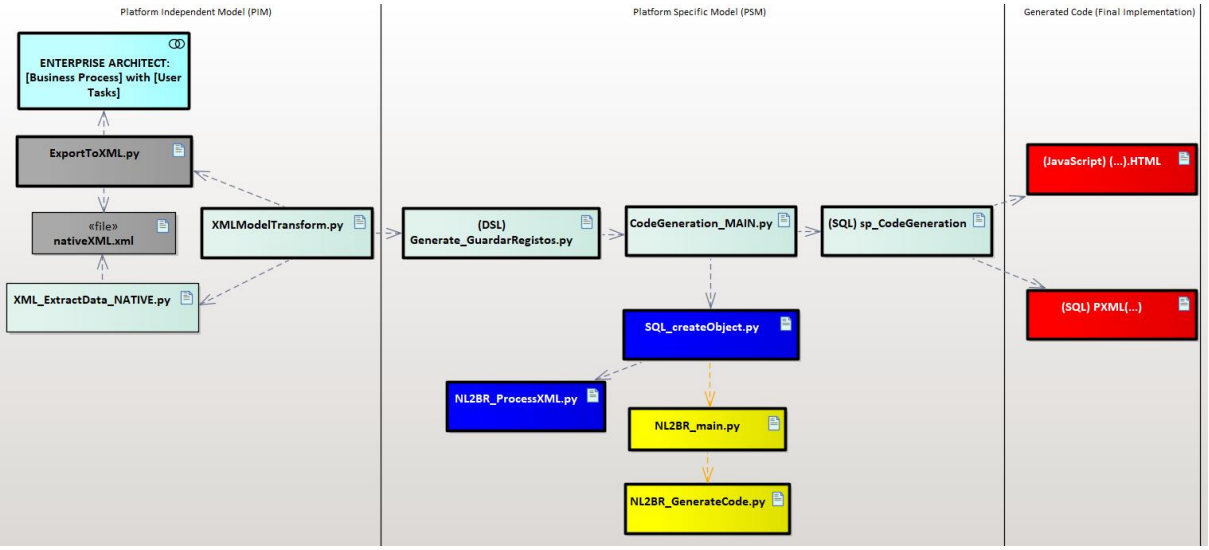


Figure 29 - New Architecture

Glossary of Figure 29 –

- **ENTERPRISE ARCHITECT: [Business Process] with [User Tasks]** - SPARX Systems [103].
- **ExportToXML.py**
  - Export of EA modeling to XML.
- **nativeXML.xml**
  - Support structure in XML.
- **XML_ExtractData_NATIVE.py**
  - Pre-transformation of XML.
- **XMLModelTransform.py**
  - Obtaining XML for DSL Generation.
- **(DSL)Generate_GuardarRegistos.py**
  - Python-based language from which the code is generated.
- **CodeGeneration_MAIN.py**
  - Invocation of code generation of the various components.
- **SQL_createObject.py**
  - Module that generates business logic object code.
- **NL2BR_ProcessXML.py**
  - Obtaining the relevant XML for transforming NL into code.
- **NL2BR_main.py**
  - Natural language processes, ***see 4.6.1 NLP Implementation***
- **NL2BR_GenerateCode.py**
  - Generation of specific business rules code from natural language, to later be included in the final code, ***see 4.6.2 SQL Implementation***
- **(SQL)sp_CodeGeneration**
  - Generation of the code for the various components.
- **(JavaScript)(…).HTML**
  - Code generated responsible for the presentation (presentation template)

- **(SQL)PXML(...)**
  - Generated code responsible for recording data (from business logic).

4.6 Code Implementation

In this chapter, we will explain the process used for building the prototype, as well as the main problems encountered and how we resolved them. Finally, we will explain the different conclusions drawn from the various stages of the prototype.

4.6.1 NLP Implementation

4.6.1.1 Selecting the Model:

One of our constraints is that all the tools used in this work must be available without monetary cost, meaning they must be open source or have a license that allows their use without payment.

Although SpaCy offers many algorithms and models with refined precision and processing capabilities, using it would involve costs that we currently cannot afford.

That said, our options will be among three of its free pre-trained models [100].

- en_core_web_sm

- **Features**: It has context-independent token vectors but lacks word vectors for similarity calculations.
- **Usage**: Ideal for lightweight applications where speed and memory efficiency are crucial. It's suitable for basic text processing tasks like extracting parts of speech, named entities, and syntactic dependencies but may not perform well in tasks requiring semantic similarity or nuanced language understanding.
- **Performance**: Fast and lightweight with lower accuracy compared to larger models. Suitable for rapid prototyping or applications with limited computational resources.

- en_core_web_md

- **Features**: Contains medium-sized word vectors (context-independent token vectors) that enable semantic similarity calculations and clustering of words.
- **Usage**: This model strikes a balance between performance and resource usage. It's suitable for applications that need more detailed language processing and improved accuracy, such as text classification, clustering, and tasks that benefit from better semantic understanding.
- **Performance**: Offers a good trade-off between speed and accuracy. It provides better word vector representations than the small model, enhancing tasks like similarity calculations and word clustering.

- en_core_web_lg

- **Features**: Equipped with large-sized word vectors (context-independent token vectors), providing rich and detailed word representations.

- **Usage**: Best suited for applications that require high accuracy and detailed language understanding, such as sophisticated natural language understanding tasks, text generation, and applications where nuanced interpretation of language is crucial.
- **Performance**: Higher accuracy and richer word vector representations than the small and medium models. This model is more computationally intensive and requires more memory, making it suitable for applications where computational resources are not a constraint and the highest accuracy is needed.

Currently, our primary focus is not on the performance of the code but rather on exploring the capabilities of the available tools. With this perspective, we have selected the 'en_core_web_lg' model Figure 30. Although it is the largest and slowest of the models, it provides the highest precision, aligning with our current priorities.

Looking ahead, as we gain a more comprehensive understanding of the potential applications of SpaCy's tools, we may consider switching to a different model. For instance, the 'en_core_web_md' model could be a viable alternative, offering a favorable balance between speed and precision. However, future research will be necessary to ascertain whether optimizing for speed and reducing the model's size will deliver more advantageous long-term outcomes, even at the expense of some precision.

```
import spacy

# Load a spaCy model
nlp = spacy.load('en_core_web_lg')

# Analyze the pipeline
nlp.analyze_pipes(pretty=True)
```

✓ 1.2s

```
============================ Pipeline Overview ============================

#   Component         Assigns               Requires   Scores             Retokenizes
-   ---------------   -------------------   --------   ----------------   -----------
0   tok2vec           doc.tensor                                          False

1   tagger            token.tag                        tag_acc            False

2   parser            token.dep                        dep_uas            False
                      token.head                       dep_las
                      token.is_sent_start              dep_las_per_type
                      doc.sents                        sents_p
                                                       sents_r
                                                       sents_f

3   attribute_ruler                                                       False

4   lemmatizer        token.lemma                      lemma_acc          False

5   ner               doc.ents                         ents_f             False
                      token.ent_iob                    ents_p
                      token.ent_type                   ents_r
                                                       ents_per_type
```

Figure 30 - Model pipeline used

From the model's pipeline Figure 30, the only mandatory part is the “tok2vec”; all other modules can be modified, or even new ones added to include more information or new functionalities to the model.

A problem we have is that we use words that do not exist in the English dictionary. For example, the sentence:

- "The CampoInteiroA of TTabelaRegistos must be a value between 10 and 20."

In the English dictionary, there are no equivalents for the words “CampoInteiroA” and “TTabelaRegistos.” The consequences of this are that it can cause greater variability when the model adds dependencies between words and in POS-Tagging.

The best solution to this problem in our case is to train new components for the model. For this, we will need a large number of examples of possible business rules. The larger the number of examples, the better the quality of the examples, the better the result will be.

Unfortunately, we do not have this data sample, and creating one with an acceptable quality of data would take more time than we have. To mitigate this problem, we must carefully introduce words that do not exist in the English language. For example, “Campo InteiroA” becomes “CampoInteiroA” or “Campo_InteiroA.” We found that if the word is not used in the sentence as a verb, the model can interpret the inputs correctly. And to ensure the proper functioning of the system, we introduced a "domain_model", which is a component that checks whether the information captured by the model is valid for solving the problem, see Figure 39 in 4.6.2 SQL Implementation.

4.6.1.2 Entity Ruler:

The "ner" stands for Named Entity Recognition. This component is responsible for identifying and categorizing named entities in a text. Named entities are real-world objects that can be associated with a proper name, such as people, organizations, locations, dates, and more. The ner component processes the text and assigns labels to the entities it recognizes, allowing the extraction of structured information from unstructured text, see Figure 31.

Example:

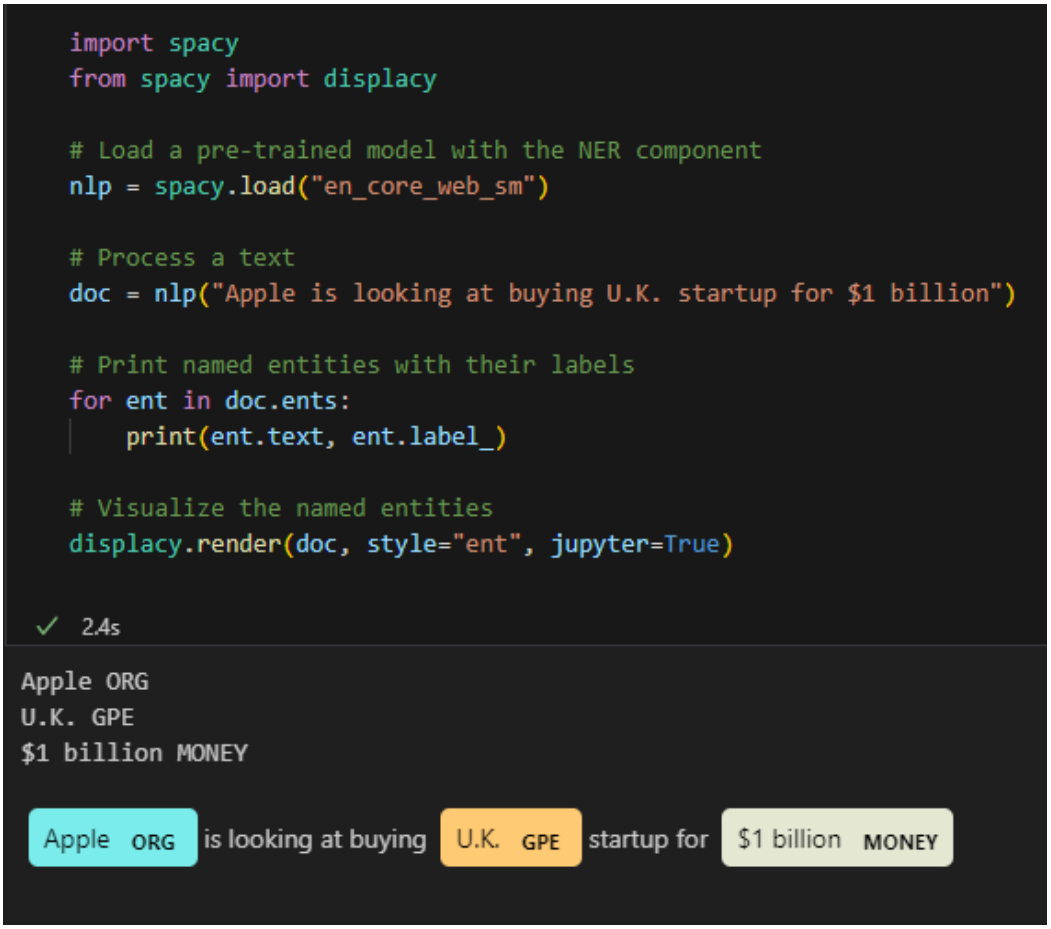
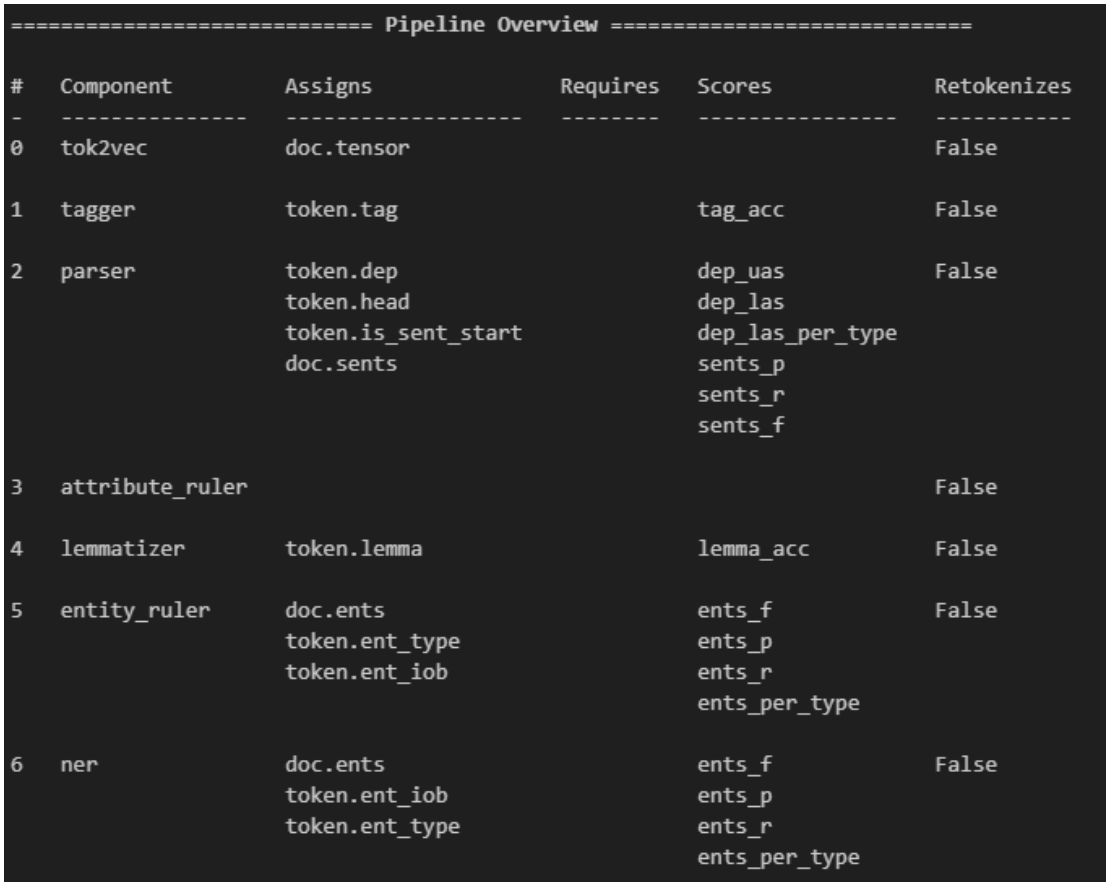


Figure 31 - Jupyter Notebook and SpaCy

Since the component from the "en_core_web_lg" model is already pre-trained, we need to create another one to meet our objectives. This new component will be called "entity_ruler," and we will place it before the "ner" component, see Figure 32. By doing this, whenever the "entity_ruler" makes a modification to the text, the "ner" component will know not to alter it.

Example:



```
============================ Pipeline Overview ============================

#   Component         Assigns               Requires   Scores             Retokenizes
-   ---------------   -------------------   --------   ----------------   -----------
0   tok2vec           doc.tensor                                          False

1   tagger            token.tag                        tag_acc            False

2   parser            token.dep                        dep_uas            False
                      token.head                       dep_las
                      token.is_sent_start              dep_las_per_type
                      doc.sents                        sents_p
                                                       sents_r
                                                       sents_f

3   attribute_ruler                                                       False

4   lemmatizer        token.lemma                      lemma_acc          False

5   entity_ruler      doc.ents                         ents_f             False
                      token.ent_type                   ents_p
                      token.ent_iob                    ents_r
                                                       ents_per_type

6   ner               doc.ents                         ents_f             False
                      token.ent_iob                    ents_p
                      token.ent_type                   ents_r
                                                       ents_per_type
```

Figure 32 - SpaCy pipeline

In the Figure 32 we can see the "entity_ruler" component in the model's pipeline at position 5, and the "ner" component at position 6. This image represents the components of the model used for our project.

4.6.1.3 Compound Sentences-

We will use this component to divide compound sentences into their independent and dependent clauses.

- Compound Sentences: A compound sentence is a type of sentence that contains at least two independent clauses.

- Independent Clause: An independent clause is a group of words that can stand alone as a complete sentence because it has a subject and a predicate. In a compound sentence, these independent clauses are usually joined by a coordinating conjunction (such as "and," "but," "or," "nor," "for," "so," or "yet") or a semicolon.

- Dependent Clause: A dependent clause, also known as a subordinate clause, is a clause that cannot stand alone as a complete sentence because it does not express a complete thought. It depends on another clause (the main clause) to complete its meaning. Dependent clauses are usually introduced by subordinating conjunctions, relative pronouns, or subordinating adverbs.

4.6.1.4 Example of how we will decompose a compound sentence into its different independent and dependent clauses:

"The **CampoInteiroA** of **TTabelaRegistos** must be a ***value lower than 20*** and the **CampoInteiroA** of **TTabelaRegistos** must be a ***value between 10 and 20*** if each **TTabelaRegistos** must have no ***more than 2*** **TtabelaSubRegistos**."

Subject: The subject of a sentence is the person, place, thing, or idea that is performing the action or being described. It answers the question "who" or "what" before the verb, is the entity that performs or is described by the action

Example:

- In the sentence "The cat sleeps on the mat," "The cat" is the subject because it is performing the action of sleeping.

Predicate: The predicate of a sentence tells something about the subject. It includes the verb and everything that comes after it, often describing the action that the subject is doing or the state of the subject.

Example:

- In the sentence "The cat sleeps on the mat," "sleeps on the mat" is the predicate because it tells what the cat (the subject) is doing.

Object: The object of a sentence is the person, place, thing, or idea that receives the action of the verb. It answers the question "whom" or "what" after the verb.

Example:

- In the sentence "The cat eats the fish," "the fish" is the object because it is receiving the action of being eaten by the cat.

We introduced rules in the "entity_ruler",see Figure 33, that identify independent and dependent clauses.

```
{"label": "IF_CLOUSE", "pattern": [{"LEMMA": "if"}]},
{"label": "THEN_CLOUSE", "pattern": [{"LEMMA": "then"}]},
{"label": "AND_CLOUSE_NUM", "pattern": [{"LIKE_NUM": True}, {"LOWER": "and"}, {"LIKE_NUM": True}]},
{"label": "AND_CLOUSE", "pattern": [{"LOWER": "and"}]},
{"label": "OR_CLOUSE_NUM", "pattern": [{"LIKE_NUM": True}, {"LOWER": "or"}, {"LIKE_NUM": True}]},
{"label": "OR_CLOUSE", "pattern": [{"LOWER": "or"}]},
```

Figure 33 - Matcher

And by knowing what each part of the sentence is, we just need to split each component, and we obtain:

Independent Clause:

1. The **CampoInteiroA** of **TTabelaRegistos** must be a ***value lower than 20***
2. the **CampoInteiroA** of **TTabelaRegistos** must be a ***value between 10 and 20***

Dependent Clause:

1.each **TTabelaRegistos** must have no ***more than 2*** **TtabelaSubRegistos**.

An observation is that by differentiating the different clauses of the sentence, we transform a sentence that is complicated to process with the SpaCy library into sentences that become simple with the help of the tools available in SpaCy.

Using the dependency tree provided by the "en_core_web_lg" model, we can extract the information we need.

1. The **CampoInteiroA** of **TTabelaRegistos** must be a ***value lower than 20.***

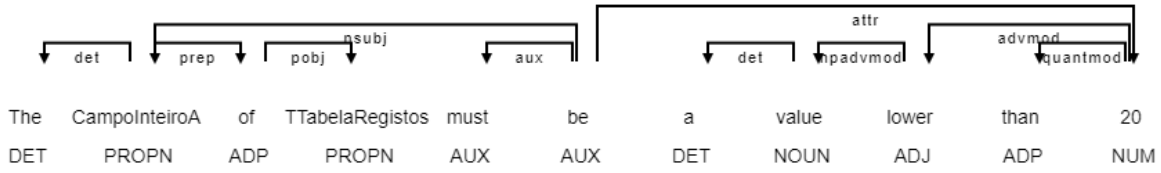


Figure 34 - Example 1

2. the **CampoInteiroA** of **TTabelaRegistos** must be a ***value between 10 and 20***

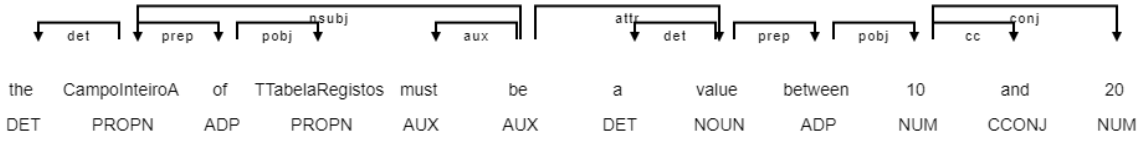


Figure 35 - Example 2

3. each **TTabelaRegistos** must have no ***more than 2*** **TtabelaSubRegistos**.

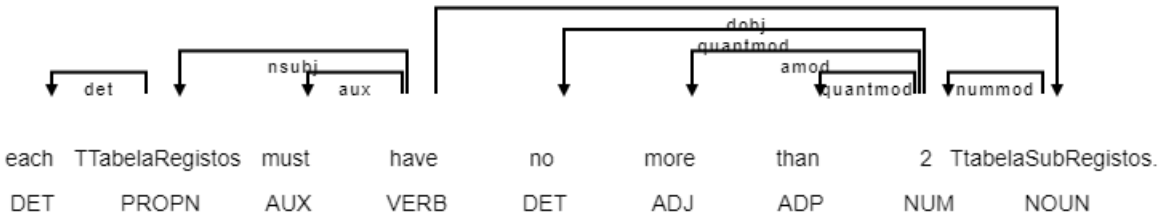


Figure 36 - Example 3

As it is possible to observe in the three examples (Figures; 34, 35, 36), the model classified each word in the sentence with its respective POS-Tagging, as well as the dependencies of each word in the sentence (det, nsubj, aux, etc.).

With respect to the given examples, the information we want to extract includes the Subject (nsubj), Predicate (pobj), Object (dobj), numbers (for which we need to determine if they relate to the subject, predicate, or object), and quantifiers (greater than, less than, equal, etc.).

For our purposes, it is not necessary to retain the verb in the sentence, as our goal is to create SQL statements or equivalents. Since this is not necessary now, we decided to leave it out. If in the future there is a need for it, the logic is already in place, and we can simply build on what we have.

NLP module Output:

By processing the data, our program provides us with two dictionaries containing the useful information present in the sentence, see Figure 37.

Example:

"The CampoInteiroA of TTabelaRegistos must be a *value lower than 20* and the CampoInteiroA of TTabelaRegistos must be a *value between 10 and 20* if each TTabelaRegistos must have no *more than 2* TtabelaSubRegistos."

```
dic_main_CLOUSE
{'AND': [{'1': [{'Subject': 'CampoInteiroA'},
                {'Object': None},
                {'Preposision': 'TTabelaRegistos'},
                {'NUM': {'BETWEEN': ['10 and 20']}},
                {'Relations': {'CampoInteiroA': '10 and 20'}}]}],
 'MAIN': [{'1': [{'Subject': 'CampoInteiroA'},
                 {'Object': None},
                 {'Preposision': 'TTabelaRegistos'},
                 {'NUM': {'EQUAL_TO': ['20']}},
                 {'Relations': {'CampoInteiroA': '20'}}]}],
 'OR': []}

dic_if_CLOUSE
{'AND': [],
 'MAIN': [{'1': [{'Subject': 'TTabelaRegistos'},
                 {'Object': 'TtabelaSubRegistos'},
                 {'Preposision': None},
                 {'NUM': {'LESS_EQUAL': ['2']}},
                 {'Relations': {'TtabelaSubRegistos': '2'}}]}],
 'OR': [],
 'THEN': []}
```

Figure 37 - Output of the NLP module

The dictionary "dic_main_CLOUSE" will always be our primary focus, as "dic_if_CLOUSE" will only contain information in cases where the sentence has a dependent clause. Therefore, we will only explain "dic_main_CLOUSE" since the process is identical for "dic_if_CLOUSE."

- **dic_main_CLOUSE**:
  - '**MAIN**': Captures information from the main sentence, representing the first independent clause of the sentence.

- **'AND'**: Captures information from the remaining dependent clauses that start with the conjunction "and."
- **'OR'**: Captures information from the remaining dependent clauses that start with the conjunction "or."
- **'Subject'**, **'Object'**, **'Preposition'**: These will be identified with the help of the "en_core_web_lg" model.
- **'NUM'**: A dictionary where the key can have the following values: {LESS_THAN (<), LESS_EQUAL (<=), GREATER_THAN (>), GREATER_EQUAL (>=), EQUAL_TO (=), BETWEEN, NOTBETWEEN}.
- **'Relations'**: Used to add extra information, currently indicating which (Subject, Object, Preposition) has a relationship with which numbers.

The advantage of this structure is that regardless of the number of clauses, we will have a structured output to build SQL statements.

4.6.1.5 Modified Output –

To minimize the effort of creating an entirely new architecture, we decided to modify the output of the NLP processing, see Figure 37, to make it compatible with the existing modules, see Figure 38. To achieve this, it was necessary to ignore the functionality that we are not currently using.

4.6.1.6 Output expected:

```
[{'MODAL_OPERATOR': 'MODAL_OPERATOR_PROHIBITION',
  'NL': 'The CampoInteiroA of TTabelaRegistos must not be a value between 5 '
        'and 10.',
  'Number_1': '5',
  'Number_2': '10',
  'Numerical_Relationships': 'NOTBETWEEN',
  'Object': None,
  'Predicate': 'TTabelaRegistos',
  'Relations': {'CampoInteiroA': '5 and 10'},
  'SBVR': 'Each CampoInteiroA that is part of TTabelaRegistos must not not '
          'have a value between 5 and 10.',
  'Subject': 'CampoInteiroA'}]
```

Figure 38 - Output modified

List of modified functionalities:

- We have discarded dependent conjunctions.
- We will only consider one independent conjunction.
- A transformation was created that takes text and transforms it into SBVR.
- A matcher was created to verify "Subjects," "Predictions," and "Objects."

- SBVR phrase types: "MODAL_OPERATOR_PROHIBITION," "MODAL_OPERATOR_PERMISSION," and "MODAL_OPERATOR_OBLIGATION"

4.6.2SQL Implementation

4.6.2.1 Inputs

For creating the SQL statements, we need to understand a few things to facilitate their construction, see Figure. 40 to see the inputs, for the thesis only the “NL” and “Domain_model” are important.

Due to the” domain_model” in the Figure. 39, we already know what the table name is and what the attribute names are.

```
domain_model = [
{'Attribute':'User_GDAI'},
{'Table':'TTabelaRegistos'},
{'Attribute':'CampoTextoA'},
{'Attribute':'CampoInteiroA'}
]
```

Figure 39 - Domain Model

It will modify the keys ('Subject', 'Predicate', and 'Object') in Figure 38 to the 'domain_model' key if the values in both are equal. And we need to pay attention that this "domain_model" is not strictly necessary to use the NLP module, it is more to facilitate the construction of SQL and to ensure that the system does not crash, as this program is a prototype that will undergo several iterations in the future, we need to build some protections to ensure consistency. But considering that we will only be using one independent clause, we can draw some conclusions if we do not want to use the "domain_model".

4.6.2.2 NLP Construction Inputs –

```
#Processar o XML para extrair BR.
if SP_XML == 1:
    NL_list = NL2BR_ProcessXML.NL2BR_ProcessXML ( xml, ei_param )
    if Debug == 1: print("NL_list: ", NL_list)
    for NL in NL_list:
        if NL != '':
            if Debug == 1: print("NL: ", NL)
            NL2BR_Code, updated_main_info = NL2BR_main.NL2BR_main ( NL, domain_model, 'SQL SERVER', Debug )
            NL2BR_CodeTOTAL = NL2BR_CodeTOTAL + '\n' + NL2BR_Code
            #if Debug == 1: print("updated_main_info (SBVR):")
            #if Debug == 1: pprint.pprint(updated_main_info)
            #if Debug == 1: print("\n")
            if Debug == 1: print(f"NL2BR_Code: {NL2BR_Code}")
            if Debug == 1: print(f"NL2BR_CodeTOTAL: {NL2BR_CodeTOTAL}")
```

Figure 40 - Main Inputs of the prototype

*NL – Business Rule (Input Text)

4.6.2.3 SQL Construction Input –

After analyzing the "BR" together with the "domain_model," the NLP component will give us the output shown in the image, Figure 38 – Output modified.

```
#PROHIBITION
"MODAL_OPERATOR_PROHIBITION_LESS_EQUAL" :
    "select * from @{TABLE} where {FIELD} > {NUMBER_1} and Evento in (''I'', ''U'')",
"MODAL_OPERATOR_PROHIBITION_LESS_THAN" :
    "select * from @{TABLE} where {FIELD} >= {NUMBER_1} and Evento in (''I'', ''U'')",
"MODAL_OPERATOR_PROHIBITION_BIGGER_THAN" :
    "select * from @{TABLE} where {FIELD} <= {NUMBER_1} and Evento in (''I'', ''U'')",
#OBLIGATION
"MODAL_OPERATOR_OBLIGATION_LESS_EQUAL" :
    "select * from @{TABLE} where not ({FIELD} > {NUMBER_1}) and Evento in (''I'', ''U'')",
"MODAL_OPERATOR_OBLIGATION_LESS_THAN" :
    "select * from @{TABLE} where not ({FIELD} >= {NUMBER_1}) and Evento in (''I'', ''U'')",
"MODAL_OPERATOR_OBLIGATION_BIGGER_THAN" :
    "select * from @{TABLE} where not ({FIELD} <= {NUMBER_1}) and Evento in (''I'', ''U'')",
#PERMISSION
"MODAL_OPERATOR_PERMISSION_LESS_EQUAL" :
    "select * from @{TABLE} where not ({FIELD} > {NUMBER_1}) and Evento in (''I'', ''U'')",
"MODAL_OPERATOR_PERMISSION_LESS_THAN" :
    "select * from @{TABLE} where not ({FIELD} >= {NUMBER_1}) and Evento in (''I'', ''U'')",
"MODAL_OPERATOR_PERMISSION_BIGGER_THAN" :
    "select * from @{TABLE} where not ({FIELD} <= {NUMBER_1}) and Evento in (''I'', ''U'')",
#PROHIBITION_CHARS
"MODAL_OPERATOR_PROHIBITION_LESS_EQUAL_characters" :
    "select * from @{TABLE} where len(convert(varchar(max), {FIELD})) > {NUMBER_1} and Evento in (''I'', ''U'')",
"BODY" :
    "\t if exists ( {BR} ) \n \
    \t begin \n \
    \t\t select ''{NL}'' as Msg, ''err'' as NivMsg, -1 as RetVal \n \
    \t\t return -1 \n \
    \t end \n"
},
```

Figure 41 - SQL Construction

Once we have this data, it's simply a matter of selecting the corresponding SQL for the "Numerical_Relationships" and "MODAL_OPERATOR" variables present in the image, Figure 39 – Output modified, and modifying the remaining values (TABLE, FIELD, Number_X) in the Figure 41 – SQL Construction.

As a starting point, we chose the simplest SQL implementation, as it will be easier and faster to make future iterations based on user needs.

4.6.2.4 Important points:

For our case, we only implemented the 'object' for sentences like:

1. "The CampoTextoA of TTabelaRegistos must not exceed 200 characters."
2. "The CampoTextoA of TTabelaRegistos must not exceed 200 chars."

In these two cases, the object of the sentence is "characters" and "chars," and for this implementation, these names means LEN(attribute), see Figure 42. Depending on the value of the “object”, it will be necessary to add additional logic.

SQL - Output

```
/***** VERIFICAÇÕES *****/


if exists ( select * from @TTabelaRegistos where len(convert(varchar(max), CampoTextoA)) > 200 and Evento in ('I', 'U') )
                            begin
                                select 'The CampoTextoA of TTabelaRegistos must not exceed 200 characters' as Msg, 'err' as NivMsg, -1 as RetVal
                                return -1
                            end

if exists ( select * from @TTabelaRegistos where not (CampoInteiroA <= 10) and Evento in ('I', 'U') )
                            begin
                                select 'The CampoInteiroA of TTabelaRegistos can not be a value bigger than 10' as Msg, 'err' as NivMsg, -1 as RetVal
                                return -1
                            end
```

Figure 42 - SQL Output

4.7 Conclusion

Upon concluding the implementation of the tool, several insights can be made.

Utilizing SpaCy for transforming natural language into machine language proved to be a sound choice, given its stability and the breadth of options it offers for problem-solving.

Moreover, the availability of professional support from experts is an added benefit, although due to budget constraints, we could not evaluate the value of SpaCy's paid services.

Concerning the performance of the NLP model with compound sentences, as illustrated in Figure 37 – Output of the NLP module, the structure is notably better than that seen in Figure 39 – Output modified. This output is not confined to sentences containing only one independent clause. However, the time required to develop SQL from this output would have extended beyond the permissible timeframe for completing the thesis, prompting the use of the modified output instead. Given that this project serves as a proof of concept and compound sentences were not a critical requirement, this last-minute modification was deemed beneficial for the project.

5 Conclusion

In this chapter, we will explain the obtained results, as well as the strengths and weaknesses of the tool.

5.1 Results

We are satisfied with the result of this work. As we have managed to develop a functional prototype that meets the minimum requirements established in section "4.2 Requirements". This prototype presents a solid proof of concept, demonstrating that the main objective of the project, transforming natural language into code, is viable and achievable.

The current implementation includes two types of validation: checking the value of an integer and the length of a text field. Although the prototype was initially developed for SQL, it already has a flexible structure, ready for future expansions to other programming languages.

This result is a significant step, as it shows that the solution can be improved and adapted to meet the user's needs. The foundation established with this prototype allows for future iterations, enabling the addition of new functionalities and validating the effectiveness of the developed concept.

5.2 Discussion and Future Work

It was important to achieve the objectives set in this project, as the developed solution opens doors for future improvements and collaborations, both by other professionals and the programming community itself. After proper review, the availability of the code [104] will allow the knowledge generated here to be shared and collectively improved, promoting continuous innovation.

Furthermore, a plan for the evolution of the code has already been defined, with significant improvements in mind, namely by means of the recurrent use within the ongoing HydraCGT project, as it is now an inherent part of it. First, it will be essential to include support for other languages, such as Portuguese, French, Chinese, and others, to expand the accessibility and usability of the solution in global contexts. Simultaneously, it will be necessary to broaden the scope of validations, incorporating new types of checks that address different requirements, and adding validations that involve multiple fields, enhancing the capabilities of the analyses and code produced by the system. Lastly, the expansion of the produced code to JavaScript is considered as strategic to the HydraCGT, in order to be able to validate the user's data before it is recorded to the database, and will imply the further decomposition of the code generation in parts common to any target language.

It is important to note that some of these changes will require the creation of a distinct NLP module from the one we already have. For example, in the case of different input natural languages, we will need to program new grammatical rules as well as use a SpaCy model compatible with the language in question. However, we can consider that the construction process will be identical for natural languages that share the same alphabet.

These future developments will ensure that the project continues to grow in relevance and utility, staying aligned with market needs and the expectations of the programming community.

References

[1] Sahay, A., Di Ruscio, D., Iovino, L., & Pierantonio, A. (2023). Analyzing business process management capabilities of low-code development platforms. Software - Practice and Experience, 53(4), 1036–1060. https://doi.org/10.1002/spe.3177

[2] Luo et al., 2021 - Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and challenges of low-code development: The practitioners perspective. International Symposium on Empirical Software Engineering and Measurement. https://doi.org/10.1145/3475716.3475782

[3] Prinz, N., Rentrop, C., & Huber, M. (2021). Low-Code Development Platforms-A Literature Review. AMCIS.

[4] Lebens, M., Finnegan, R. J., Sorsen, S. C., & Shah, J. (2021). Rise of the citizen developer. Muma Business Review, 5(12), 101–111.

[5] Sufi, F. (2023). Algorithms in Low-Code-No-Code for Research Applications: A Practical Review. Algorithms, 16(2). https://doi.org/10.3390/a16020108

[6] Bock, A. C., & Frank, U. (2021). In Search of the Essence of Low-Code: An Exploratory Study of Seven Development Platforms. Companion Proceedings - 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS-C 2021, 57–66. https://doi.org/10.1109/MODELS-C53483.2021.00016

[7] Ihirwe, F., Di Ruscio, D., Gianfranceschi, S., & Pierantonio, A. (2022). Assessing the Quality of Low-Code and MDE Platforms for Engineering IoT Systems. SSRN Electronic Journal, 583–594. https://doi.org/10.2139/ssrn.4267269

[8] Khorram, F., Mottu, J. M., & Sunyé, G. (2020). Challenges & opportunities in low-code testing. Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings, 490–499. https://doi.org/10.1145/3417990.3420204

[9] Capsicum. R. (2021). *Why don't developers give much cred to low-code/no-code tools?* https://rebeccabilbro.github.io/low-code-no-code/

[10] Demeshko, A. (2024). *Why Developers Hate Low Code. And why they shouldn't.* Uibakery.Io.https://uibakery.io/post/why-developers-hate-low-code-and-why-they-shouldnt

[11] Rodenacker, F. (2022). *8 Things Developers Don't Like About Low-Code and No-Code*. Hackernoon.Com. https://hackernoon.com/8-things-developers-dont-like-about-low-code-and-no-code

[12] Wayner, P. (2019). *Why developers hate low-code?* InfoWorld. https://www.infoworld.com/article/3438819/why-developers-hate-low-code.html

[13] Smith, G., Papadopoulos, M., Sanz, J., Grech, M., & Norris, H. (2020). Unleashing innovation using low code/no code--The age of the citizen developer. Ed: Arthur D. Little Prism.

[14] Iivari, J. (1996). Why are CASE tools not used? Communications of the ACM, 39(10), 94–103.

[15] Agarwal, R., Prasad, J., Tanniru, M., & Lynch, J. (2000). Risks of rapid application development. Communications of the ACM, 43(11es), 1--es.

[16] Barricelli, B. R., Cassano, F., Fogli, D., & Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. Journal of Systems and Software, 149, 101–137.

[17] Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? Software and Systems Modeling, 21(2), 437–446.
[18] Brooks, F. (1987). No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, *20*(4), 10–19.
[19] Grier, D. A. (2021). There Is Still No Silver Bullet. *Computer*, *54*(2), 60–62.
[20] Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036
[21] Bajec, M., & Krisper, M. (2005). A methodology and tool support for managing business rules in organisations. Information Systems, 30(6), 423–443. https://doi.org/https://doi.org/10.1016/j.is.2004.05.003
[22] Rouvellou, I., Degenaro, L., Rasmus, K., Ehnebuske, D., & McKee, B. (2000). Extending business objects with business rules. Proceedings 33rd International Conference on Technology of Object-Oriented Languages and Systems TOOLS 33, 238–249. https://doi.org/10.1109/TOOLS.2000.848765
[23] Dietz, J. L. G. (2006). Enterprise ontology. Springer. https://doi.org/10.1007/3-540-33149-2
[24] Deacon, J. (2009). Model-view-controller (mvc) architecture. Online][Citado Em: 10 de Março de 2006.]
[25] Nalepa, G. J., & Kluza, K. (2012). UML representation for rule-based application models with XTT2-based business rules. International Journal of Software Engineering and Knowledge Engineering, 22(04), 485–524.
[26] Valente, P., Silva, T., Winckler, M., & Nunes, N. (2016). Bridging Enterprise and Software Engineering Through an User-Centered Design Perspective. Web Information Systems Engineering (WISE), 2(October), 463–477. https://doi.org/10.1007/978-3-319-26190-4
[27] Zur Muehlen, M., Indulska, M., & Kittel, K. (2008). Towards integrated modeling of business processes and business rules. ACIS 2008 PROCEEDINGS. https://aisel.aisnet.org/acis2008/108
[28] zur Muehlen, M., & Indulska, M. (2010). Modeling languages for business processes and business rules: A representational analysis. Information Systems, 35(4), 379–390. https://doi.org/https://doi.org/10.1016/j.is.2009.02.006
[29] Bajwa, I. S., Lee, M. G., & Bordbar, B. (2011). SBVR Business Rules Generation from Natural Language Specification. AAAI Spring Symposium: AI for Business Agility, SS-11-03, 2–8. https://api.semanticscholar.org/CorpusID:17731244
[30] Haj, A., Balouki, Y., & Gadi, T. (2021). Automated generation of terminological dictionary from textual business rules. Journal of Software: Evolution and Process, 33(5), e2339. https://doi.org/https://doi.org/10.1002/smr.2339
[31] Kopp, A., Orlovskyi, D., & Orekhov, S. (2021). An Approach and Software Prototype for Translation of Natural Language Business Rules into Database Structure. COLINS, 2870, 1274–1291.
[32] Chowdhary, K. R. (2020). Natural Language Processing. In Fundamentals of Artificial ~ Intelligence (pp. 603–649). Springer India. https://doi.org/10.1007/978-81-322-3972-7_19
[33] I. S. Bajwa, B. Bordbar, and M. G. Lee, "OCL Constraints Generation from Natural

Language Specification," in *2010 14th IEEE International Enterprise Distributed Object Computing Conference*, Oct. 2010, pp. 204–213. doi: 10.1109/EDOC.2010.33.
[34] https://www.omg.org/
[35] "Semantics of Business Vocabulary and Business Rules".
[36] M. H. Linehan, "SBVR Use Cases," in *Rule Representation, Interchange and Reasoning on the Web*, vol. 5321, N. Bassiliades, G. Governatori, and A. Paschke, Eds., in Lecture Notes in Computer Science, vol. 5321. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 182–196. doi: 10.1007/978-3-540-88808-6_20.
[37] T. Allweyer, *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. BoD – Books on Demand, 2016.
[38] "DMN.pdf." Accessed: Jun. 13, 2024. [Online]. Available: https://www.omg.org/spec/DMN/1.5/Beta1/PDF
[39] G. Booch, J. Rumbaugh, and I. Jacobson, "for Object-Oriented Development," 1996.
[40] "OCL.pdf." Accessed: Jun. 13, 2024. [Online]. Available: https://www.omg.org/spec/OCL/2.4/PDF
[41] A. Bucchiarone, J. Cabot, R. F. Paige, and A. Pierantonio, "Grand challenges in model-driven engineering: an analysis of the state of the research," *Softw. Syst. Model.*, vol. 19, no. 1, pp. 5–13, Jan. 2020, doi: 10.1007/s10270-019-00773-6.
[42] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?," *Softw. Syst. Model.*, vol. 21, no. 2, pp. 437–446, Apr. 2022, doi: 10.1007/s10270-021-00970-2.
[43] T. Skersys, P. Danenas, E. Mickeviciute, and R. Butleris, "Transforming BPMN Processes to SBVR Process Rules with Deontic Modalities," *Appl. Sci.*, vol. 12, no. 18, Art. no. 18, Jan. 2022, doi: 10.3390/app12188976.
[44] F. Ashfaq and I. S. Bajwa, "Natural language ambiguity resolution by intelligent semantic annotation of software requirements," *Autom. Softw. Eng.*, vol. 28, no. 2, p. 13, Jul. 2021, doi: 10.1007/s10515-021-00291-0.
[45] S. Arshad, I. S. Bajwa, and R. Kazmi, "Generating SBVR-XML Representation of a Controlled Natural Language," in *Intelligent Technologies and Applications*, I. S. Bajwa, F. Kamareddine, and A. Costa, Eds., Singapore: Springer, 2019, pp. 379–390. doi: 10.1007/978-981-13-6052-7_33.
[46] L. Arco, G. Nápoles, F. Vanhoenshoven, A. L. Lara, G. Casas, and K. Vanhoof, "Natural language techniques supporting decision modelers," *Data Min. Knowl. Discov.*, vol. 35, no. 1, pp. 290–320, Jan. 2021, doi: 10.1007/s10618-020-00718-4.
[47] A. Bhattacharyya, P. K. Chittimalli, and R. Naik, "An Approach to Mine Business Rule Intents from Domain-specific Documents," in *Proceedings of the 10th Innovations in Software Engineering Conference*, in ISEC '17. New York, NY, USA: Association for Computing Machinery, Feb. 2017, pp. 96–106. doi: 10.1145/3021460.3021470.
[48] I. S. Bajwa, M. G. Lee, and B. Bordbar, "SBVR Business Rules Generation from Natural Language Specification".
[49] P. K. Chittimalli, C. Prakash, R. Naik, and A. Bhattacharyya, "An Approach to Mine SBVR Vocabularies and Rules from Business Documents," in *Proceedings of the 13th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, in ISEC '20. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 1–11. doi: 10.1145/3385032.3385046.
[50] A. Haj, Y. Balouki, and T. Gadi, "Automated generation of terminological dictionary from textual business rules," *J. Softw. Evol. Process*, vol. 33, no. 5, p. e2339, 2021, doi: 10.1002/smr.2339.

[51] S. A. Jogdand, “NLP for Automated Conceptual Data Modeling”.
[52] L. Nemuraite, T. Skersys, A. Šukys, E. Sinkevicius, and L. Ablonskis, “VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models,” *Int. Conf. Inf. Softw. Technol. IT2010,* Apr. 2010.
[53] “ID2SBVR: A Method for Extracting Business Vocabulary and Rules from an Informal Document.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.mdpi.com/2504-2289/6/4/119
[54] Bonais, Mohammed, Wenny Rahayu, and Eric Pardede. "Integrating information systems business rules into a design model." *2012 15th International Conference on Network-Based Information Systems*. IEEE, 2012.
[55] Bajwa, Imran Sarwar, Ali Samad, and Shahzad Mumtaz. "Object oriented software modeling using NLP based knowledge extraction." *European Journal of Scientific Research* 35.01 (2009): 22-33.
[56] Ciravegna, Dr Fabio. "Adaptive information extraction from text by rule induction and generalisation." (2001).
[57] Mooney, R. "Relational learning of pattern-match rules for information extraction." *Proceedings of the sixteenth national conference on artificial intelligence*. Vol. 328. 1999.
[58] Cheng, Ran, Shazia Sadiq, and Marta Indulska. "Framework for business process and rule integration: a case of BPMN and SBVR." *Business Information Systems: 14th International Conference, BIS 2011, Poznań, Poland, June 15-17, 2011. Proceedings 14*. Springer Berlin Heidelberg, 2011.
[59] Danenas, Paulius, Tomas Skersys, and Rimantas Butleris. "Natural language processing -enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams." *Data & Knowledge Engineering* 128 (2020): 101822.
[60] Finkel, Jenny Rose, Trond Grenager, and Christopher D. Manning. "Incorporating non -local information into information extraction systems by gibbs sampling." *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL’05)*. 2005.
[61] Friedrich, Fabian, Jan Mendling, and Frank Puhlmann. "Process model generation from natural language text." *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings 23*. Springer Berlin Heidelberg, 2011.
[62] Zhou, GuoDong, et al. "Tree kernel-based semantic role labeling with enriched parse tree structure." *Information Processing & Management* 47.3 (2011): 349-362.
[63] Zhou, Guodong, and Min Zhang. "Extracting relation information from text documents by exploring various types of knowledge." *Information Processing & Management* 43.4 (2007): 969-982.
[64] Gatt, Albert, and Ehud Reiter. "SimpleNLG: A realisation engine for practical applications." *Proceedings of the 12th European workshop on natural language generation (ENLG 2009)*. 2009.
[65] Ge, Tao, Furu Wei, and Ming Zhou. "Fluency boost learning and inference for neural grammatical error correction." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.
[66] Huffman, Scott B. "Learning information extraction patterns from examples." *International Joint Conference on Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995.
[67] Haj, Abdellatif, Youssef Balouki, and Taoufiq Gadi. "Automatic extraction of SBVR

based business vocabulary from natural language business rules." *Smart Data and Computational Intelligence: Proceedings of the International Conference on Advanced Information Technology, Services and Systems (AIT2S-18) Held on October 17–18, 2018 in Mohammedia 3*. Springer International Publishing, 2019.
[68] Haj, Abdellatif, Youssef Balouki, and Taoufiq Gadi. "Automated Identification of Semantic Similarity between Concepts of Textual Business Rules." *International Journal of Intelligent Engineering & Systems* 14.1 (2021).
[69] Harmain, H. M., and Robert Gaizauskas. "Cm-builder: A natural language-based case tool for object-oriented analysis." *Automated Software Engineering* 10 (2003): 157-181.
[70] Haj, Abdellatif, et al. "The semantic of business vocabulary and business rules: An automatic generation from textual statements." *IEEE Access* 9 (2021): 56506-56522.
[71] Kar, Saurav Kumar. *Generation of UML class diagram from software requirement specification using natural language processing*. Diss. 2014.
[72] Kleiner, Mathias, Patrick Albert, and Jean Bézivin. "Parsing SBVR-based controlled languages." *Model Driven Engineering Languages and Systems: 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings 12*. Springer Berlin Heidelberg, 2009.
[73] Linehan, Mark H. "Ontologies and rules in business models." *2007 Eleventh International IEEE EDOC Conference Workshop*. IEEE, 2007.
[74] Linehan, Mark H. "SBVR use cases." *Rule Representation, Interchange and Reasoning on the Web: International Symposium, RuleML 2008, Orlando, FL, USA, October 30-31, 2008. Proceedings 2*. Springer Berlin Heidelberg, 2008.
[75] Mickeviciute, Egle, and Rimantas Butleris. "The Comprehensive Modelling of BPMN Business Processes and Business Rules using SBVR Profile." *Doctoral Consortium on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Vol. 2. SCITEPRESS, 2014.
[76] Mickevičiūtė, Eglė, and Rimantas Butleris. "Towards the combination of bpmn process models with sbvr business vocabularies and rules." *Information and Software Technologies: 19th International Conference, ICIST 2013, Kaunas, Lithuania, October 2013. Proceedings 19*. Springer Berlin Heidelberg, 2013.
[77] Zhang, Min, GuoDong Zhou, and Aiti Aw. "Exploring syntactic structured features over parse trees for relation extraction using kernel methods." *Information processing & management* 44.2 (2008): 687-701.
[78] Zur Muehlen, Michael, and Marta Indulska. "Modeling languages for business processes and business rules: A representational analysis." *Information systems* 35.4 (2010): 379 -390.
[79] Mickeviciute, Egle, Lina Nemuraite, and Rimantas Butleris. "Applying SBVR business vocabulary and business rules for creating BPMN process models." *Business Information Systems Workshops: BIS 2014 International Workshops, Larnaca, Cyprus, May 22-23, 2014, Revised Papers 17*. Springer International Publishing, 2014.
[80] Alt, Christoph, Marc Hübner, and Leonhard Hennig. "Improving relation extraction by pre-trained language representations." *arXiv preprint arXiv:1906.03088* (2019).
[81] Manning, Christopher D., et al. "The Stanford CoreNLP natural language processing toolkit." *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014.
[82] Mikolov, T. "Distributed Representations of Words and Phrases and Their

Compositionality." *arXiv preprint arXiv:1310.4546* (2013).
[83] Omar, Nazlia, P. Hanna, and P. Mc Kevitt. "Semantic analysis in the automation of ER modelling through natural language processing." *2006 International Conference on Computing & Informatics*. IEEE, 2006.
[84] Overmyer, Scott P., L. Benoit, and R. Owen. "Conceptual modeling through linguistic analysis using LIDA." *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*. IEEE, 2001.
[85] Riloff, Ellen. "Automatically constructing a dictionary for information extraction tasks." *AAAI*. Vol. 1. No. 1. 1993.
[86] Ramzan, Shabana, et al. "A model transformation from NL to SBVR." *Ninth International Conference on Digital Information Management (ICDIM 2014)*. IEEE, 2014.
[87] Rozovskaya, Alla, et al. "The University of Illinois system in the CoNLL-2013 shared task." *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. 2013.
[88] Arco, Leticia, et al. "Natural language techniques supporting decision modelers." *Data Mining and Knowledge Discovery* 35 (2021): 290-320.
[89] Skersys, Tomas, et al. "Transforming BPMN Processes to SBVR Process Rules with Deontic Modalities." *Applied Sciences* 12.18 (2022): 8976.
[90] Selway, Matt, et al. "Formalising natural language specifications using a cognitive linguistic/configuration based approach." *Information Systems* 54 (2015): 191-208.
[91] Shi, Lei, and Rada Mihalcea. "Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing." *Computational Linguistics and Intelligent Text Processing: 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005. Proceedings 6*. Springer Berlin Heidelberg, 2005.
[92] Skersys, Tomas, Lina Tutkute, and Rimantas Butleris. "The enrichment of BPMN business process model with SBVR business vocabulary and rules." *Journal of computing and information technology* 20.3 (2012): 143-150.
[93] Skersys, Tomas, et al. "Extending BPMN business process model with SBVR business vocabulary and rules." *Information technology and control* 41.4 (2012): 356-367.
[94] Tissier, Julien, Christophe Gravier, and Amaury Habrard. "Dict2vec: Learning word embeddings using lexical dictionaries." *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*. 2017.
[95] Wang, Linlin, et al. "Relation classification via multi-level attention cnns." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016.
[96] Xiao, Minguang, and Cong Liu. "Semantic relation classification via hierarchical recurrent neural network with attention." *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016.
[97] Zhang, Yuhao, Peng Qi, and Christopher D. Manning. "Graph convolution over pruned dependency trees improves relation extraction." *arXiv preprint arXiv:1809.10185* (2018).
[98] Zheng, Suncong, et al. "A neural network framework for relation extraction: Learning entity semantic and relation pattern." *Knowledge-Based Systems* 114 (2016): 12-23.
[99] "NLTK Book." Accessed: Jun. 15, 2024. [Online]. Available: https://www.nltk.org/book/
[100] "spaCy · Industrial-strength Natural Language Processing in Python." Accessed: Sep. 02, 2024. [Online]. Available: https://spacy.io/
[101] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," in *Proceedings of 52nd Annual*

Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 55–60. doi: 10.3115/v1/P14-5010.

[102] Princeton University "About WordNet." WordNet. Princeton University. 2010.Available: https://wordnet.princeton.edu/
[103]“Sparx Systems.” Accessed: Jul. 31, 2024. [Online]. Available: https://sparxsystems.com/
[104] https://github.com/nuno8887/NLP-Module