Enhancing OntoUML Accessibility through Structured Text Generation and Text-to-Speech

TOM NIEUWLAND, University of Twente, The Netherlands

OntoUML is a modeling language for creating conceptual ontologies, which often presents challenges in comprehension for non-experts. Additionally, its graphical representation poses accessibility challenges for visually impaired individuals. This research addresses these issues by exploring methods to enhance the understanding and accessibility of OntoUML models through structured text generation and text-to-speech conversion. I propose a translation approach that converts OntoUML models into controlled natural language, using Semantics for Business Vocabulary and Business Rules (SBVR) as an intermediate representation, and subsequently into speech. This approach aims to make OntoUML models more accessible and improve their usability, thereby; facilitating collaboration between modelers and other stakeholders, including individuals with visual impairments.

Additional Key Words and Phrases: OntoUML, SBVR, Natural Language Generation, Accessibility, Text-to-Speech

1 INTRODUCTION

OntoUML extends the Unified Modeling Language (UML) and is used as a language for the conceptual modeling of ontologies. The language has its roots in the Unified Foundational Ontology (UFO), which is a foundational ontology that addresses the fundamental notions of conceptual modeling [14]. The commitment to this meta-ontology makes OntoUML ontologies more accurate when compared to ontologies represented in other forms, for example, in the Web Ontology Language (OWL), or in standard UML [12].

OntoUML enables users with the ability to model conceptualizations and abstractions into concrete artifacts that can be communicated and analyzed [13]. These models are useful in various domains like requirement engineering, semantic web, and software engineering [8, 10]. The difference between OntoUML and standard UML lies in the addition of stereotypes to classes and relations [7, 19]. These stereotypes highlight a deeper ontological meaning of the concepts and relationships they are associated with. They can also impose restrictions on how concepts can relate to each other. However, the inherent complexity of OntoUML models makes them hard to understand for people who are not experts in modeling and ontologies [5]. Furthermore, these conceptual models are created and represented graphically, creating a significant limitation to the use of OntoUML by people that suffer from visual impairment.

This research aims to reduce the gap between the complexity of OntoUML and its accessibility. One potential approach to reduce this gap, is to translate OntoUML models, serialized in OntoUML Vocabulary [22], into natural language specifications. OntoUML Vocabulary is a machine processable representation of OntoUML models which makes it a suitable representation for this translation, since the translation will be handled by a computer program. The translation process could use an intermediate representation, such as structured English using SBVR. SBVR is a formal framework developed by the Object Management Group (OMG) to capture business semantics in a precise and unambiguous manner [20]. It uses a form of structured English to define business vocabulary and rules. Due to this approach, SBVR is well-suited for use as an intermediate representation between modelers and non-experts [4]. The SBVR representation of OntoUML models can then be translated into a natural language specification. By employing text-to-speech technology on these natural language specifications, they can be made accessible to people with impaired vision.

This research will explore how to implement this approach to answer the following question: *"How can OntoUML models serialized in OntoUML Vocabulary be converted into controlled English text that explains the models to people unfamiliar with the concepts, making it also accessible to people with visual impairments?"*. To ensure a successful response to the research question, it is important to begin by answering the following four sub-questions.

(1) What methods exist for converting (Onto)UML models into text, and what are their corresponding advantages and limitations?
(2) How can an OntoUML model serialized in OntoUML Vocabulary be converted into English text while balancing its understandability with loss of meaning?
(3) How can English text be converted to speech?
(4) How to implement the conversion into a working prototype?

The rest of this paper is structured as follows: Section 2 presents related work. Section 3 details the development of the tool and its design decisions. Finally, Section 4 discusses the results, answering the sub-research questions and the main research question and concludes the paper.

2 RELATED WORKS

To gather relevant literature for this research, databases such as Google Scholar, ResearchGate, and ScienceDirect were used. Search terms as “OntoUML to natural language”, “Natural Language Generation”, “SBVR to natural language” and “Making models accessible to visually impaired” were employed to find scientific papers. Based on the resulting papers' abstracts, a selection of these papers was chosen for further analysis.

Papers [1, 6, 24] researched the topic of translating ontologies into English text. Specifically, [24] provided a brief overview of various ontology verbalizers; however, none of these focused on OntoUML. Additionally, some approaches required manual input to work [3, 9, 15]. All reviewed papers used some form of intermediate representation, indicating the benefits of using such an approach for translation. An intermediate representation serves as a bridge between the original information (in this work's case, OntoUML

TScIT 41, July 5, 2024, Enschede, The Netherlands

Vocabulary), and the final text, facilitating the removal of unnecessary and repetitive information. However, these papers relied on self-defined intermediate solutions, limiting their applicability in future research. By using a standardized intermediate representation, tools and research developed for the intermediate representation would be directly applicable to this research.

As the intermediate representation, this work uses SBVR, which was found to be a suitable format since it provides a form of structured English that allows for precise expression of rules while remaining understandable to domain experts. [4] highlights a tool developed to translate UML models extended with the Object Constraint Language (OCL) to SBVR structured English. Again, this work did not work for OntoUML models, however it provided valuable insight into the translation of UML models to SBVR. Its approach of creating a translation for each type of UML construct was carried over to this research, in the form of a translation for each type of OntoUML construct. [21] shows a tool for creating a translation from OntoUML to SBVR. The tool has been archived since 2021, but shows a working translation. However, the translation is very extensive and precise, this makes not very applicable in this research, since this research aims to make the model more understandable. However, it does show promising use of SBVR Vocabulary entries to represent OntoUML classes.

The performed literature research did not result in any works regarding the translation of SBVR into natural language. This is likely because SBVR is usually created from natural language to embed more precision rather than translating SBVR to natural language, since SBVR is already understandable to most people.

Literature research into text-to-speech tools, yielded papers [16, 17, 23], that all present tools for blind and visually impaired people. All of these papers make use of the open-source Google Text-to-Speech (gTTS) tool [11]. This tool provides the exact functionality that is needed in the last step if the proposed translation method of this research.

3 DEVELOPMENT

After the exploration of various methods that can be used to transform OntoUML to a form of natural language and speech, a three-step process was developed. The first step is the translation of an OntoUML model, serialized in OntoUML Vocabulary, into a form of SBVR structured English. The next step in the process is the translation of this SBVR structured English into a form of natural language. The final step is to transform this natural language into speech.

3.1 Scope

The translation that will be described below is not functional for all OntoUML models. The scope of the translation and tool were narrowed to work for a subset of UML and OntoUML constructs. This tool will make use of UML classes and their names, furthermore it will consider inheritance, bidirectional associations, and unidirectional associations. The coverage of OntoUML stereotypes can be found in Table 1.

Table 1. OntoUML stereotype coverage

Covered	Not covered
All Class stereotypes	Formal
Material	Derivation
Mediation	Containment
Characterization	SubCollectionOf
ComponentOf	SubQuantityOf
MemberOf	

Valid models are limited to models using the OntoUML prefix: `<https://purl.org/ontouml-models/vocabulary/>`. In the natural language specification, the existence of grammatical errors, in the form of wrong articles and improper use of plural forms, will be tolerated.

Although this narrowed scope restricts the valid models that can be translated by the tool, it was necessary to keep the research feasible within the allotted time. Without the restricted scope, it would not be possible to provide well-thought out translation patterns and a working tool. All limitation to the scope can be seen as points for future research.

3.2 OntoUML to SBVR

This work makes use of the Vocabulary entries from the SBVR specification [20]. A vocabulary entry in SBVR is a representation of a single concept that is built up following a specific structure. The primary representation of the entry is the concept which it describes. This concept can then be explained using the following: definitions, sources, a dictionary basis, a general concept, a concept type, necessities, possibilities, reference schemes, notes, and examples. In our translation, we will only make use of general concepts, concept types, necessities and possibilities to explain the concept. The other options for explanation are either difficult to formulate with the limited data available in the OntoUML Vocabulary representation of a model, or would only add repetitive information. Therefore, the use of these would not be beneficial to the understandability of the OntoUML model, making them redundant for our application.

The translation is done by defining a translation pattern for one OntoUML construct at a time. This is a similar approach used in [4], but using OntoUML constructs instead of UML constructs. First, OntoUML classes and their stereotypes are used to establish the Vocabulary entries and their concept type. Then, OntoUML generalizations will be used to determine the general concept, for applicable entries. Following that, OntoUML generalization sets will be used to generate necessity and possibility rules. Finally, OntoUML relations and their stereotypes are used to generate additional necessity and possibility rules.

3.2.1 *Creating Vocabulary entries.*

For every class in the OntoUML model, we create an entry. The primary representation of this entry is the name of the class. This is done since every entry should cover exactly one concept, which aligns with how classes are used in OntoUML.

3.2.2 *Determining concept types.*

For each Vocabulary entry, we then determine the concept type. The concept types of an entry can

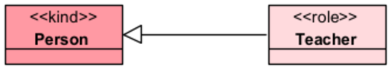


Fig. 1. OntoUML model example 1

be an ‘individual noun concept’, ‘general concept’, ‘verb concept’, ‘characteristic’, ‘binary verb concept’ or a ‘role’. Since each of our entries has a term as the primary representation, we only use the concept types ‘general concept’ and ‘role’. The other concept types are meant for types of primary representations that are not used in this translation. To distinguish between ‘general concept’ and ‘role’, we can look at the stereotype of the OntoUML classes. This tells us how the concept is used in the model and which concept type best fits. The OntoUML class stereotypes are split in the following way: ‘role’, ‘roleMixin’, ‘phase’ and ‘phaseMixin’ are translated into the concept type ‘role’. That is done because these stereotypes are used on concepts that describe a state or role of other concepts. That makes them more fitting to the concept type ‘role’, compared to ‘general concept’. All the other class stereotypes are translated into the concept type ‘general concept’. This covers the two possible resulting rules:

(1) `Concept Type: role`
(2) `Concept Type: general concept`

3.2.3 *Determining general concepts.*

After determining the concept type for each entry, we can determine the general concept. The general concept of an entry can only be present if the entry has some general concept above it in the model. Therefore, we have a look at the generalizations in the model. A generalization is a relation that connects two concepts together. One of these concepts is considered the ‘general’ and the other is considered the ‘specific’. A generalization tells us that the ‘specific’ is a specialization of the more general concept ‘general’. For every class in the model that is the specific part of some generalization, we can define the general of that generalization to be its general concept. However, some generalizations are also part of generalization sets. These sets will provide more information about the generalizations it groups together. Therefore, these sets are handles separately in the next section. To prevent repetition, the following rule is only generated for the specific of a generalization that is not part of some generalization set:

- `General Concept: (general)`

Figure 1 shows an example of two OntoUML classes and a generalization. Their corresponding SBVR entries, according to the rules discussed so far, would be:

Person
Concept type: general concept

Teacher
Concept type: role
General Concept: Person

After determining the general concept, we will determine the necessities and possibilities. This is split into two different stages.

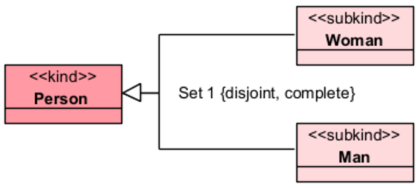


Fig. 2. OntoUML model example 1

First, we will handle all the generalization sets and second, we will have a look at all other relationships.

3.2.4 *Handling generalization sets.*

A generalization set is a group of generalizations. The term ‘specifics’ is used to describe the group of every ‘specific’ of each individual generalization, that is part of the set. The ‘general’ of every generalization in the set is always the same, and is considered to be the ‘general’ of the set as well. Generalization sets will generate a necessity rule, for each of the specifics of the set. This will be of the form: `’Necessity: each (specific) if of type (general)’`. For the general of the set, the necessity/possibility rule is dependent on whether the set is disjoint and complete. ’Disjoint’, means that an instance of the ‘general’ can only be an instance of one of the specifics. When a set is not disjoint, a general can be multiple specifics at the same time. ’Complete’, means that the specifics of the set cover all possibilities that the general can be an instance of. When a set is not complete, it means that the general can be an instance of something that is not present in the model. We have four different combinations of disjoint and complete, each creating a unique sentence to describe the relationship between the general and the specifics. The sentences can be found in Table 2. These sentences are created in a way that they accurately describe the relation without the use of technical terms and keeping to the short format that is desired.

Figure 2 shows an example of an OntoUML class as the general of a generalization set, with two specific classes below it. Their corresponding SBVR entries, according to the rules discussed so far, would be:

Person
Concept type: general concept
Necessity: each Person is exactly one Man or Woman

Man
Concept type: general concept
Necessity: each Man is of type Person

Woman
Concept type: general concept
Necessity: each Woman is of type Person

3.2.5 *Handling relations.*

When talking about a relation between two concepts, the terms ‘source’ and ‘target’ are used. The ‘source’ and ‘target’ are the concepts that the relationship connects. The relation is considered to originate from the ‘source’ and applies to the ‘target’. For example, a relation ‘component of’ between

Table 2. Translation patters for generalization sets

Type of generalization set	Resulting rule
disjoint / complete	`Necessity: each (general) is exactly one (specifics)`
disjoint / not complete	`Necessity: each (general) is at most one (specifics)`
not disjoint / complete	`Necessity: each (general) is at least one (specifics)`
not disjoint / not complete	`Possibility: it is a possibility that a (general) is one or multiple (specifics)`

Table 3. Translation patters for relations

Relational stereotype	Resulting rule part
Component of (TGT)	Is a component of
Component of (SRC)	Is composed of
Characterization (TGT)	Characterizes
Characterization (SRC)	Is characterized by
Member of (TGT)	Is a member of
Member of (SRC)	Has as member
Mediation (TGT)	Is required by
Mediation (SRC)	Requires
Material (TGT)	Is connected to
Material (SRC)	Is connected to
No stereotype (TGT)	Is associated to
No stereotype (SRC)	Is associated to

the source 'Table leg' and the target 'Table', signifies that 'A table leg is a component of a table'. Relations will generate a necessity or possibility rule based on their stereotype and cardinality, for both the target concept and source concept of the relation. The stereotype of the relationship determines the middle part of the rule that describes the relation between the two connected concepts. These sentences are intentionally kept short and precise to fit the style of SBVR structured English. This causes the deeper ontological meaning of stereotypes to be hidden in the SBVR specification. This is required for keeping the it concise, and understandable to non-experts. They are constructed in a way that they still reflect most of the meaning of the stereotype in an understandable way, setting them apart from generic UML relations. Table 3 shows the resulting middle rule part, for each of the covered stereotypes (see Table 1 for the coverage). This is done from both the target's perspective (TGT), and the source's perspective (SRC).

The cardinality is responsible for both describing the cardinal relationship between the concepts and determining whether the rule is a necessity or a possibility. The cardinality constraint of the relation is determined based on the cardinalities' lower and upper bounds in the following way:

- If the lower and upper bounds are the same, the cardinality constraint is written as 'exactly (lower bound)'.
- If the lower bound is 0 and upper bound is '*', the cardinality constraint is written as 'some'.
- If the lower bound is 0 and the upper bound is not '*', the cardinality constraint is written as 'at most (upper bound)'.
- If the lower bound is not 0 and the upper bound is '*', the cardinality constraint is written as 'at least (lower bound)'.

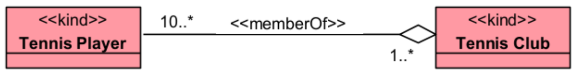


Fig. 3. OntoUML model example 3

- If the lower bound is not 0 and the upper bound is not '*', the cardinality constraint is written as 'at least (lower bound) and at most (upper bound)'.

In the case where the cardinality constraint is written as 'some', the rule is written in the form of a possibility. It cannot be written as a necessity, since it can be zero or more. The rule is written as a necessity in all other cases, creating the following four options for the rule:

From the target's perspective:

(1) `Necessity: each (target) (rule part) (cardinality constraint) (source)`
(2) `Possibility: it is a possibility that a (target) (rule part) (cardinality constraint) (source)`

From the source's perspective:

(1) `Necessity: each (source) (rule part) (cardinality constraint) (target)`
(2) `Possibility: it is a possibility that a (source) (rule part) (cardinality constraint) (target)`

Figure 3 shows an example of two OntoUML classes connected with a 'memberOf' relation. The corresponding SBVR entries, according to the rules discussed so far, would be:

Tennis Player
Concept type: general concept
Necessity: each Tennis Player is a member of at least 1 Tennis Club

Tennis Club
Concept type: general concept
Necessity: each Tennis Club has as members at least 10 Tennis Player

An example of the set of SBVR Vocabulary entries that are generated from a larger OntoUML model can be found in Section 1 of Appendix B.

3.3 SBVR to Natural Language

The SBVR to natural language transformation is done by transforming our previously defined SBVR Vocabulary entries into a form of natural language. This process involves four main steps.

3.3.1 Grouping. The first step in the process is to group concepts together, based on mutual referencing and generalization sets. This is done, to group concepts together that lie close to each other in the original model. This way we can write a paragraph about every group, which will result in a more natural text. If we would not group concepts together, the entire text would be just a list with explanations of concepts. The grouping works by iterating over all rules and making a group per rule that contains all concepts listed in that rule. We then remove any groups that are a subset of another group. This removes unwanted repetition, while making sure all concepts are still part of at least one group. The next step is to group together any triangles, where a triangle is a combination of three groups that only contains three distinct concepts. These groups can be represented by a single group consisting of the three distinct concepts.

3.3.2 Ordering. The next step is to create an ordering of the groups. This is done to create coherence between subsequent paragraphs in the final text. When the subsequent paragraphs contain overlapping elements, the text as a whole seems more natural and coherent. To achieve the ordering of the groups, a greedy algorithm is used. This greedy algorithm puts the first group at the top of a list. Then it tries to find another group with an overlapping element. When such a group is found, this group is put to the top of the list, and the process repeats. This is done until all groups are on the list. When no group can be found with an overlapping concept, a random group is picked, and put on top of the list.

3.3.3 Paragraph writing. After the groups have been formed and ordered, a paragraph is written for each group. The first step in this process is to loop over all the Vocabulary entries of the group. For each entry, the rules must be filtered. The rules are checked for whether they contain concepts that are not part of the group. When that is the case, these rules are removed. The remaining rules are split into three different categories: normal rules, generals of generalization sets, and specifics of generalization sets. Each category of rules is processed differently. The type of rule can be determined by the middle part of the rule, based on the translation patterns used in the OntoUML to SBVR section.

Normal rules get translated from their structured representation in the Vocabulary entry, into a more natural flowing sentence. For every old rule, a new sentence is rewritten to the form: `'A(n) (concept) (identifier) (cardinality + targets).'` The identifier is determined based on the original rule, where '(P)' and '(N)' represent 'Possibility:' and 'Necessity: ' respectively. The translation patterns for this part, can be found in Table 4.

After the rules are translated, they are processed further. This process consists of grouping together rules in two different ways. The first way is to group together rules that connect the same concept in the same way to different targets.
For example, the sentence `'A table leg is a component of exactly 1 table.'`, and the sentence `'A tabletop is a component of exactly 1 table.'`, get combined into the single sentence `'A table leg or tabletop is a component of exactly 1 table.'`. The second way is to group together rules that connect the same target in the same way to different concepts.
For example, the sentence `'A table consists of exactly 4 table leg.'`, and the sentence `'A table consists of exactly 1 tabletop.'`, get combined into the single sentence `'A table consist of exactly 4 table leg, and exactly 1 tabletop.'`. This is done to reduce repetition and once again group together concepts that lie closely together.

Rules that connect a general to its specifics get translated next. These rules are rewritten into a sentence that explains the rule in more detail than before. This is done to better highlight the differences between sets based on their completeness and disjointedness. The translation of these rules can be found in Table 5.

Rules that connect a specific of a generalization set to a general are not used for the translation. This is because the link between these concepts is already covered in the previous section. Adding another line, to state the connection, would be repetitive. The reason the rules from the perspective of the specifics is left out as opposed to the rule from the general's perspective is due to the extra information carried in the rule from the general's perspective. That rule offers insight into the completeness of the set, and whether the generalization set is disjoint.

After all sentences have been generated and processed, they are combined to form a single paragraph of the natural language specification.

3.3.4 Combining everything. When all groups have been processed, and all paragraphs have been written, the paragraphs get written into a single text file. This text file resembles the natural language specification of the model.

Table 4. Translation patters for identifiers

SBVR notation	**Natural Language notation**
General concept: ..	.. is a(n) ..
(N) .. requires ..	.. needs ..
(N .. is a component of ..	.. is a component of ..
(P) .. is a component of ..	.. can be a component of ..
(N) .. consists of ..	.. consist of ..
(P) .. consists of ..	.. can consist of ..
(N) .. characterizes ..	.. characterizes ..
(P) .. characterizes ..	.. can characterize ..
(N) .. is characterized by ..	.. is characterized by ..
(P) .. is characterized by ..	.. can be characterized by ..
(N) .. is a member of ..	.. is a member of ..
(P) .. is a member of ..	.. can be a member of ..
(N) .. is required by ..	.. is required by ..
(P) .. is required by ..	.. can be required by ..
(N) .. is associated to ..	.. is associated to ..
(P) .. is associated to ..	.. can be associated to ..
(N) .. is connected to ..	.. is connected to ..
(P) .. is connected to ..	.. can be connected to ..

Table 5. Translation patters for generalization sets from SBVR to NL

Rule in SBVR specification	Resulting sentence
.. exactly one ..	Every (general) is either a(n) (specifics).
.. at most one ..	A(n) (general) can be a(n) (specifics), or another possibility, but only 1 at the same time.
.. at least one ..	Every (general) is at least one (specifics), but can also be multiple at the same time.
.. is one or multiple ..	A(n) (general) can be a(n) (specifics), or another possibility, it can also be multiple at the same time.

An example of the natural language specification generated from a set of SBVR Vocabulary entries can be found in Section 2 of Appendix B.

3.4 Natural Language to Speech

The natural language processing starts by slightly altering the text to increase the pauses between paragraphs and at the end of sentences. Using the standard pause length, the speech was quite fast, which made it hard to follow. Increasing the pauses between sentences and paragraphs improved the understandability of the speech.

After the text is altered, it is passed to the gTTS tool. This tool is generates a speech version from the inputted text. This speech version of the text is than saved in MP3 format.

4 CONCLUSIONS

This research focused on finding a way in which OntoUML could be made more accessible by answering the main research question. This question is answered by the cumulative answer to the sub-questions. In the search for finding existing methods of converting (Onto)UML models to text, many different approaches were found in the literature. Unfortunately, none of these approaches were based on OntoUML. However, taking inspiration from them, this research provides a methodology of how to approach the translation of OntoUML models into a form of natural language and speech using a three-step process. This three-step approach shows a way how OntoUML models can be translated into understandable English without sacrificing a lot of the meaning embedded in the models. Using gTTS to convert the natural language specification to speech proved a sufficient way for the conversion of English text into speech. The whole three-step process is implemented in a single Python based prototype tool, which is available in [18]. This tool performs the transformation that bridges the gap between the sophistication and complexity of OntoUML and the understandability of the models that can be created using it, broadening the use of OntoUML to non-experts and people that suffer from visual impairment.

Future work is needed to take the methodology and the tool to the next level. This work can be done in a few different directions. The first direction would be the extension of the current coverage. This would involve research into the uncovered UML and OntoUML constructs and how these constructs can be translated into SBVR Vocabulary entries. A consequent step would be to define new translation patterns for the SBVR to natural language transformation, in case new types of SBVR rules are generated. The second direction for future research could involve improvement of the current tool. With the current tool there are at least three things that could be improved:

(1) Improved grouping algorithm
(2) Improved group ordering algorithm
(3) Grammatical correctness with regards to articles and plurals

Further improvements could show themselves when extensive testing of the tool is performed. The third direction for future research could involve this extensive testing of the current tool. This should at least include the following tests:

(1) The natural language specifications should be evaluated by modeling experts, to ensure correctness and accuracy
(2) The natural language specifications should be evaluated by non-experts, to ensure their understandability
(3) The speech translation of the models should be evaluated by people with visual impairment, to ensure their understandability
(4) The tool should be evaluated across a diverse range of models, to ensure its robustness

As a final direction for future research, the use of Large Language Models (LLMs) and Artificial Intelligence (AI) could be explored to improve the natural language specification. The use of these tools could make the text more natural, since you would not have to rely on strictly defined translation patters for the translation. However, this could also impose problems with respect to the accuracy of the natural language specification.

REFERENCES

[1] Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. 2014. Generating Natural Language Descriptions from OWL Ontologies: the NaturalOWL System. *The Journal of Artificial Intelligence Research (JAIR)* 48 (04 2014), 671–715. https://doi.org/10.1613/jair.4017
[2] Pedro Paulo F. Barcelos, Tiago Prince Sales, Mattia Fummagali, and Claudenir M. Fonseca. 2023. OntoUML/UFO Catalog. https://doi.org/10.5281/zenodo.8188545
[3] Kalina Bontcheva and Yorick Wilks. 2004. Automatic Report Generation from Ontologies: The MIAKT Approach. *In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems* 3136, 324–335. https://doi.org/10.1007/978-3-540-27779-8_28
[4] Jordi Cabot, Raquel Pau, and Ruth Raventós. 2010. From UML/OCL to SBVR specifications: A challenging transformation. *Information Systems* 35, 4 (2010), 417–440. https://doi.org/10.1016/j.is.2008.12.002
[5] Guadalupe Cea, Elena Montiel-Ponsoda, and Mari Carmen Suárez-Figueroa. 2009. Approaches to ontology development by non ontology experts. (2009).
[6] Dragos Alexandru Cojocaru and Stefan Trausan-Matu. 2015. Text Generation Starting from an Ontology. In *Romanian Conference on Human-Computer Interaction*. 55–60. https://api.semanticscholar.org/CorpusID:5687270
[7] OntoUML Contributors. 2024. OntoUML Documentation. https://ontouml.readthedocs.io/en/latest/. Accessed: 2024-06-29.
[8] Maria Ferreira, Joao Moreira, Maria Campos, Bernardo Braga, Tiago Prince Sales, Kelli Cordeiro, and Marcos Borges. 2015. OntoEmergePlan: variability of emergency plans supported by a domain ontology. In *12th International Conference on Information Systems for Crisis Response and Management, ISCRAM 2015*. University of Agder, 1–9.

[9] Dimitrios Galanis and Ion Androutsopoulos. 2007. Generating Multilingual Descriptions from Linguistically Annotated OWL Ontologies: the NaturalOWL System. *Proceedings of the 11th European Workshop on Natural Language Generation, ENLG 07* (01 2007), 143–146. https://doi.org/10.3115/1610163.1610188

[10] Bernardo Gonçalves, Veruska Zamborlini, Giancarlo Guizzardi, and José Pereira Filho. 2009. An ontology-based application in heart electrophysiology: Representation, reasoning and visualization on the web. *Proceedings of the ACM Symposium on Applied Computing*, 816–820. https://doi.org/10.1145/1529282.1529456

[11] Google. 2024. GTTS: Google Text-to-Speech Documentation. https://gtts.readthedocs.io/en/latest/. Accessed: 2024-06-29.

[12] Giancarlo Guizzardi. 2005. *Ontological Foundations for Structural Conceptual Models*. Ph. D. Dissertation.

[13] Giancarlo Guizzardi. 2007. On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. *Frontiers in artificial intelligence and applications* 155, 18–39.

[14] Giancarlo Guizzardi, Alessander Benevides, Claudenir Fonseca, Daniele Porello, João Almeida, and Tiago Prince Sales. 2022. UFO: Unified Foundational Ontology. *Applied Ontology* 17, 1 (01 2022), 167–210. https://doi.org/10.3233/AO-210256

[15] Feikje Hielkema. 2009. *Using natural language generation to provide access to semantic metadata.* Ph. D. Dissertation. University of Aberdeen.

[16] Tushar Khete and Aditya Bakshi. 2022. Autonomous Assistance System for Visually Impaired using Tesseract OCR gTTS. *Journal of Physics: Conference Series* 2327 (08 2022), 012065. https://doi.org/10.1088/1742-6596/2327/1/012065

[17] Supriya Kurlekar, O Deshpande, A Kamble, A Omanna, and D Patil. 2020. Reading Device for Blind People using Python OCR and GTTS. *International Journal of Science and Engineering Applications* 9, 4 (2020), 49–52.

[18] Tom Nieuwland. 2024. OntoUML Translator. https://gitlab.utwente.nl/s2812304/ontouml2sbvr. Accessed: 2024-06-20.

[19] Object Management Group. 2017. About the Unified Modeling Language® - Version 2.5.1. https://www.omg.org/spec/UML/2.5.1/About-UML. Accessed: 2024-06-29.

[20] Object Management Group. 2017. Semantics of Business Vocabulary and Business Rules (SBVR). http://www.omg.org/spec/SBVR/. Accessed: 2024-06-29.

[21] Tiago Prince Sales, John Guerson, Freddy Brasileiro, and Bernardo . 2021. menthor-editor ontouml2sbvr. https://github.com/MenthorTools/menthor-editor/tree/master/net.menthor.ontouml2sbvr. Accessed: 2024-05-28.

[22] Tiago Prince Sales, Claudenir M. Fonseca, and Pedro Paulo Favato Barcelos. 2023. OntoUML Vocabulary. https://w3id.org/ontouml/vocabulary. Accessed: 2024-06-20.

[23] C Venkata Sai, D Thrinethra, and SVS Devi. 2023. Voice Based Email System for Blind People. *Journal of Electronics and Informatics* 5, 2 (2023), 226–234.

[24] Robert Stevens, James Malone, Sandra Williams, Richard Power, and Allan Third. 2011. Automating Generation of Textual Class Definitions from OWL to English. *Journal of biomedical semantics* 2 (05 2011), 1–20. https://doi.org/10.1186/2041-1480-2-S2-S5

A USE OF ARTIFICIAL INTELLIGENCE

During the preparation of this work the author used ChatGPT-3.5, in order to generate basic documentation for the implemented tool. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

B ONTOUML TO NATURAL LANGUAGE SPECIFICATION EXAMPLE

In this Appendix, an example will be shown of the developed translation patterns and tool. Figure 4, shows an OntoUML model in its graphical state. This model is taken from the OntoUML/UFO catalog [2]. The model describes various concepts from Formula One and how they relate to one another. Section 1 will show the set of SBVR Vocabulary entries that are generated by the tool, based on the Formula One OntoUML model shown in Figure 4. Finally, Section 2 will show the Natural Language specification that is generated based in the set of SBVR Vocabulary entries from Section 1.

Note that there is an issue related to the concepts 'Driver' and 'Race', in both the SBVR specification and natural language specification. These rules are reversed from how they are represented in Figure 4. This is due to a mistake in the serialized version of the OntoUML model, where the relation is represented in the wrong direction. It is not a problem with the tool.

B.1 Formula One represented by SBVR Vocabulary entries

F1 Season
Concept Type: general concept
Necessity: each F1 Season is composed of at least 8 Race

Race
Concept Type: general concept
Necessity: each Race is exactly one Regular Race or Sprint Race
Necessity: each Race is a component of exactly 1 F1 Season
Necessity: each Race requires exactly 1 Circuit
Possibility: it is possible that a Race is required by some Driver
Necessity: each Race requires exactly 1 Grand Prix
Necessity: each Race requires exactly 1 F1 Car

Grand Prix
Concept Type: general concept
Necessity: each Grand Prix is required by at least 1 Race
Necessity: each Grand Prix is connected to at least 1 Circuit

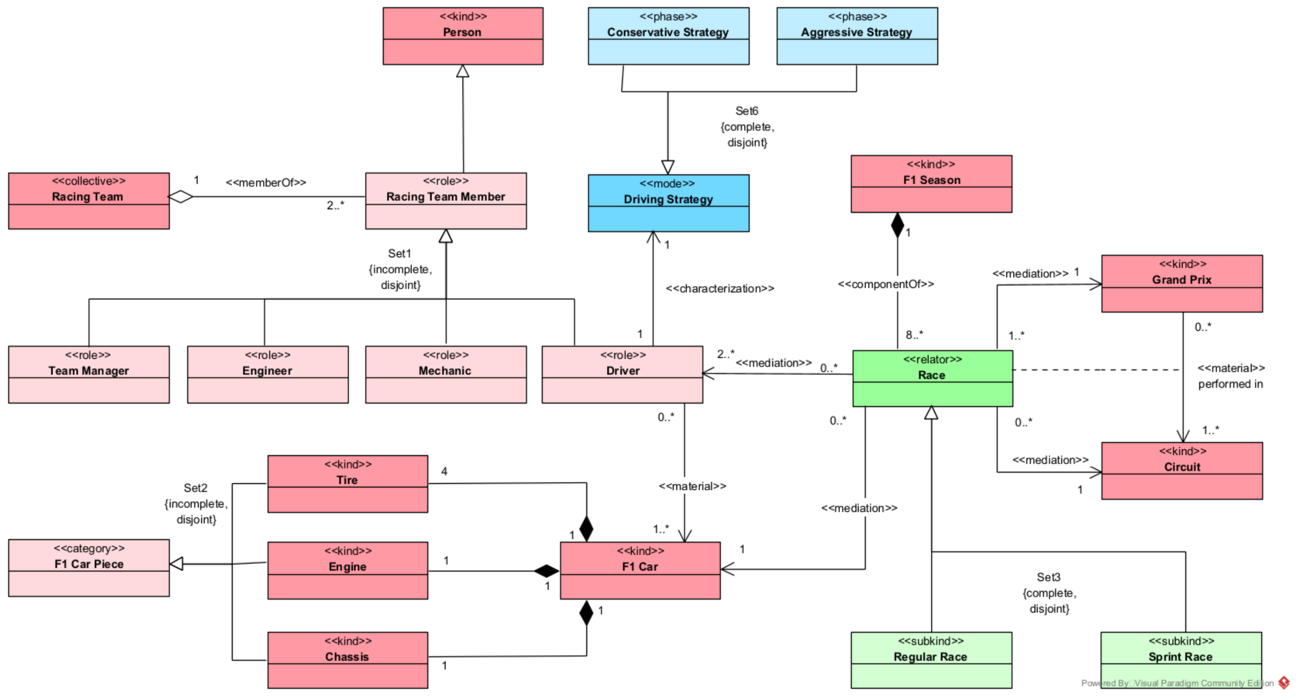
Driver
Concept Type: role
Necessity: each Driver is of type Racing Team Member
Necessity: each Driver requires at least 2 Race
Necessity: each Driver is connected to at least 1 F1 Car
Necessity: each Driver is characterized by exactly 1 Driving Strategy

Racing Team
Concept Type: general concept
Necessity: each Racing Team has as members at least 2 Racing Team Member

Circuit
Concept Type: general concept
Possibility: it is possible that a Circuit is required by some Race
Possibility: it is possible that a Circuit is connected to some Grand Prix

Engine
Concept Type: general concept
Necessity: each Engine is of type F1 Car Piece
Necessity: each Engine is a component of exactly 1 F1 Car

Tire
Concept Type: general concept
Necessity: each Tire is of type F1 Car Piece
Necessity: each Tire is a component of exactly 1 F1 Car



<<kind>> Person
<<phase>> Conservative Strategy
<<phase>> Aggressive Strategy
Set6 {complete, disjoint}
<<collective>> Racing Team
<<memberOf>>
<<role>> Racing Team Member
<<mode>> Driving Strategy
<<kind>> F1 Season
Set1 {incomplete, disjoint}
<<characterization>>
<<componentOf>>
<<mediation>>
<<kind>> Grand Prix
<<role>> Team Manager
<<role>> Engineer
<<role>> Mechanic
<<role>> Driver
<<relator>> Race
<<material>> performed in
<<kind>> Circuit
<<kind>> Tire
<<material>>
Set2 {incomplete, disjoint}
<<category>> F1 Car Piece
<<kind>> Engine
<<kind>> F1 Car
<<kind>> Chassis
Set3 {complete, disjoint}
<<subkind>> Regular Race
<<subkind>> Sprint Race
Powered By: Visual Paradigm Community Edition

Fig. 4. OntoUML model describing Formula One

Chassis
Concept Type: general concept
Necessity: each Chassis is of type F1 Car Piece
Necessity: each Chassis is a component of exactly 1 F1 Car

F1 Car
Concept Type: general concept
Necessity: each F1 Car is composed of exactly 4 Tire
Necessity: each F1 Car is composed of exactly 1 Engine
Necessity: each F1 Car is composed of exactly 1 Chassis
Possibility: it is possible that a F1 Car is required by some Race
Possibility: it is possible that a F1 Car is connected to some Driver

Person
Concept Type: general concept

F1 Car Piece
Concept Type: general concept
Necessity: each F1 Car Piece is at most one Tire, Chassis or Engine

Engineer
Concept Type: role
Necessity: each Engineer is of type Racing Team Member

Mechanic
Concept Type: role
Necessity: each Mechanic is of type Racing Team Member

Racing Team Member
Concept Type: role
General Concept: Person
Necessity: each Racing Team Member is at most one Driver, Team Manager, Engineer or Mechanic
Necessity: each Racing Team Member is a member of exactly 1 Racing Team

Team Manager
Concept Type: role
Necessity: each Team Manager is of type Racing Team Member

Regular Race
Concept Type: general concept
Necessity: each Regular Race is of type Race

Sprint Race
Concept Type: general concept
Necessity: each Sprint Race is of type Race

Driving Strategy
Concept Type: general concept
Necessity: each Driving Strategy is exactly one Aggressive Strategy or Conservative Strategy
Necessity: each Driving Strategy characterizes exactly 1 Driver

Aggressive Strategy
Concept Type: role

Necessity: each Aggressive Strategy is of type Driving Strategy

Conservative Strategy
Concept Type: role
Necessity: each Conservative Strategy is of type Driving Strategy

B.2 Formula One represented in a Natural Language specification

A Driver needs at least 2 Race. A Driver is connected to at least 1 F1 Car. A F1 Car can be required by some Race. A F1 Car can be connected to some Driver. A Race can be required by some Driver. A Race needs exactly 1 F1 Car.

A Circuit can be required by some Race. A Circuit can be connected to some Grand Prix. A Grand Prix is required by at least 1 Race. A Grand Prix is connected to at least 1 Circuit. A Race needs exactly 1 Circuit, and exactly 1 Grand Prix.

Every Race is either a Regular Race or Sprint Race.

A F1 Season is composed of at least 8 Race. A Race is a component of exactly 1 F1 Season.

A F1 Car Piece can be a Tire, Chassis, Engine, or another possibility, but only 1 at the same time. Every Chassis, Engine, or Tire is a component of exactly 1 F1 Car. A F1 Car is composed of exactly 4 Tire, exactly 1 Engine, and exactly 1 Chassis.

A Racing Team Member is a Person.

A Racing Team Member can be a Driver, Team Manager, Engineer, Mechanic, or another possibility, but only 1 at the same time. A Racing Team has as members at least 2 Racing Team Member. A Racing Team Member is a member of exactly 1 Racing Team.

A Driver is characterized by exactly 1 Driving Strategy. A Driving Strategy characterizes exactly 1 Driver.

Every Driving Strategy is either an Aggressive Strategy or Conservative Strategy.