applied sciences

MDPI

Article

Extracting SBVR Business Vocabularies from UML Use Case Models Using M2M Transformations Based on Drag-and-Drop Actions

Tomas Skersys [1,2,*], Paulius Danenas [1], Rimantas Butleris [1,2], Armantas Ostreika [3] and Jonas Ceponis [4]

[1] Center of Information Systems Design Technologies, Kaunas University of Technology, K. Barsausko Str. 59, 51423 Kaunas, Lithuania; paulius.danenas@ktu.lt (P.D.); rimantas.butleris@ktu.lt (R.B.)
[2] Department of Information Systems, Kaunas University of Technology, Studentu Str. 50, 51368 Kaunas, Lithuania
[3] Department of Multimedia Engineering, Kaunas University of Technology, Studentu Str. 50, 51368 Kaunas, Lithuania; armantas.ostreika@ktu.lt
[4] Department of Computer Sciences, Kaunas University of Technology, Studentu Str. 50, 51368 Kaunas, Lithuania; jonas.ceponis@ktu.lt
* Correspondence: tomas.skersys@ktu.lt

Featured Application: The main application of the developed solution lies in the areas of problem domain analysis and system design where the researched SBVR business vocabularies, UML use case models, and model-to-model transformation technology are utilized to their full extent. The presented solution enables one to use our previously developed model-to-model transformation technology in the course of the specification of business knowledge formally expressed in SBVR business vocabularies based on UML use case models.

Abstract: In the domain of model-driven system engineering, model-to-model (M2M) transformations present a very relevant topic because they may provide much-needed automation capabilities to the whole CASE-supported system development life cycle. Nonetheless, it is observed that throughout the whole development process M2M transformations are spread unevenly; in this respect, the phases of Business Modeling and System Analysis are arguably the most underdeveloped ones. The main novelty and contributions of this paper are the presented set of model-based transformations for extracting well-structured SBVR business vocabularies from visual UML use case models, which utilizes M2M transformation technology based on the so-called drag-and-drop actions. The conducted experiments show that this new development provides the same transformation power while introducing more flexibility to the model development process as compared to our previously developed approach for (semi-)automatic extraction of SBVR business vocabularies from UML use case models.

Keywords: UML use case model; SBVR business vocabulary; model-driven system development; model-to-model transformation; drag-and-drop action

Citation: Skersys, T.; Danenas, P.; Butleris, R.; Ostreika, A.; Ceponis, J. Extracting SBVR Business Vocabularies from UML Use Case Models Using M2M Transformations Based on Drag-and-Drop Actions. *Appl. Sci.* **2021**, *11*, 6464. https://doi.org/10.3390/app11146464

Academic Editors: Eugenio Parra and Juan Llorens

Received: 3 April 2021
Accepted: 8 July 2021
Published: 13 July 2021

1. Introduction

While Unified Modeling Language (UML) [1] is the most popular visual language for the model-driven specification of software systems, the Use Case Diagram stands out as one of the most frequently used diagrams adopted by UML. IT people use UML use case models (UCM) and their graphical representations in the form of use case diagrams (UCD) for the specification and communication of functional requirements of the future system. However, any IT project involving business individuals will most likely face difficulties in communicating these requirements among IT and business individuals. This is because business individuals will more likely prefer communicating their business needs in natural language (NL) format and will be cautious about any formal visual

representations of the business domain knowledge; this includes UML, which is often comprehended as 'too technical'. However, the communication of requirements in NL may potentially come at a cost of increased ambiguity, redundancy, and inconsistency. This increases the risk of miscommunication among the interested parties and the overall poor quality of requirements.

One method to deal with this situation is by using formal visual models synchronized with their representation in well-structured textual specifications based on controlled natural language. Our approach uses UML UCD as visual means of representing functional requirements in contrast to OMG's Semantics of Business Vocabulary and Rules (SBVR) [2], which was adopted to represent parts of the requirements in a well-structured textual form as its primary for of representation. SBVR is a standard aimed at specifying any human knowledge based on natural language expressions. It is comprised of business vocabularies (BV) and business rules (BR) for which visual representation 'looks and feels' similar to natural language. Simultaneously, SBVR is founded upon the formal specification of Meta Object Facility (MOF) [3] and, therefore, its models are fully interpretable by computers. Another favorable argument about using natural language-based formalism for expressing business knowledge is that such an approach enables the reuse of structured knowledge in every phase of the system development while at the same time aligning itself with the basic principles of OMG's Model Driven Architecture (MDA) [4]. Such accumulated knowledge could then be reused elsewhere, e.g., in other IT projects carried out within the given organization.

Indeed, the advantages of using formal visual models combined with natural language-based specifications seem to be quite evident. However, actual system development projects will likely lack well-structured vocabularies comprised of traceable and manageable business concepts, even though the overall relevance of such vocabularies is quite well perceived [5]. We argue that such a situation is conditioned by the lack of proper tools supporting the development and management of such vocabularies in IT projects and beyond. Without proper tools, the development and management of BV will always require significant time and human effort, which may negatively impact the decision on whether to use such vocabularies in a project at all.

To ease these efforts, we first developed a solution for extracting SBVR BV and BR from UML UCM (hereafter, UML UCM → SBVR BV&BR approach) by utilizing a (semi-) automatic M2M transformation technology [6]. However, while experimenting with that solution, we observed there was an evident need for more control over the M2M transformation actions that a user working with those transformations could utilize, particularly, selecting specific source model concepts and applying different localized transformations to transform those source model concepts into different target model concepts based on his expert judgment of the specific situations in the source model. To overcome this shortcoming, we decided to take advantage of the special kind of M2M transformation called *M2M transformation based on drag-and-drop actions* (hereafter, drag-and-drop transformation or, simply, DnD transformation) [7] by developing a model-based executable library of DnD transformation rules for the extraction of SBVR BV from UML UCM. The distinctive feature of such DnD transformation is that it is selectively triggered by a drag-and-drop action. Here, a drag-and-drop action represents a situation when a user drags a certain model element from the browser of a *source* model and drops it onto either a *target* diagram or another element's representation in that diagram to generate a subset of target model elements (such a transformation is called a *partial M2M transformation*).

Main contribution and novelty of the results. This paper presents the basic aspects of the approach for the extraction of SBVR BV from UML UCM (hereafter, UML UCM → SBVR BV) using model transformations based on drag-and-drop actions. Compared to our previously developed UML UCM → SBVR BV approach, the presented solution provides the same model transformation capabilities while simultaneously delivering more flexibility and control over the M2M transformations, starting with the model-based

development of transformations themselves and finishing with the selective execution of the modeled transformations.

The main conceptual novelty and contribution of this research is the proposed *library* of model-based transformation specifications for the UML UCM → SBVR BV approach. It contains a set of transformation specifications based on transformation rules presented in [6]. These specifications are based on the metamodel, which was first introduced in [7] and briefly discussed in this paper as well (Section 3). The developed set of transformations can be executed in the DnD transformation tool implementation [7]. The developed set of the UML UCM → SBVR BV transformation specifications provides a transparent ('white box') view of the actual mappings between UML and SBVR metamodels (the latter being implemented as a UML profile). Moreover, due to the flexibility and universality of the DnD approach itself, the developed set of model transformations could serve as a reference point for the development of corresponding transformations between the models expressed in other modeling notations.

The paper is structured as follows: an overview of the latest relevant research is presented in Section 2; Section 3 presents an overview of the concepts and definitions relevant to the research, together with a brief description of the implementation architecture of the developed approach; in Section 4, the specification of model-based transformation rules for extracting SBVR BV from UML UCM is presented, which is then followed by the experiment results in Section 5; the paper concludes with Section 6, where conclusions and insights for future improvements are drawn.

2. Related Work

It is observed that the public presentation of the research dealing with the integration and transformation of both UML and SBVR models remains very poor. At this moment, no published research directly dealing with UML UCM → SBVR BV&BR transformation was found, except for our previous publications [6,8]. One more study presented results of the opposite transformation [9]; however, it only covered the basic UML UCM elements, such as *Actor*, *UseCase*, and *Association*. Despite this gap, there are multiple relevant publications from the past decade dealing with either visual UML use case models or text-based SBVR models in the presented M2M transformation approaches.

Exploring the possibilities to extract SBVR business vocabularies and business rules from other kinds of visual models started soon after the SBVR 1.0 specification was published in 2008. The author of [10] was the first to present results on the transformation of UML/OCL conceptual data models to SBVR specifications. At about the same time, certain conceptual and engineering results presenting the extraction of SBVR BV&BR from UML/OCL class models were published in [11]. Both [10,11] proved that M2M transformation technology could be applied to SBVR models in model-driven systems development providing practical results. Both approaches represented fully automatic M2M transformations, which utilized formal UML and SBVR metamodels. Several other papers followed presenting similar approaches [12–15]. Others published their results on the research of extracting SBVR BV&BR from BPMN process models [16,17] and the synchronization of these models [18]. Yet another study introduced an approach for generating UML activity diagrams from SBVR BR [19]. While OMG SBVR specification [2] already provides mapping examples between UML Class models and SBVR BV concepts, other transformations from SBVR BV and BR to UML have been explored as well, including Use Case models [20] and OCL constraints in UML models [21]. Process models were also considered as targets for SBVR transformations [22,23].

In the requirements engineering area, a considerable amount of significant research was carried out on the junction of use case modeling and the application of NL patterns that are semantically similar to SBVR specifications. Al-Hashemi et al. [24] presented an approach for developing UML UCM from natural language-based textual documents. It utilized a large set of UML concepts resulting in the development of a more comprehensive UML UCM. Deeptimahanti et al. [25] further explored the possibilities of the extraction

of UCM from the documents written in unstructured natural language. It was argued that the developed solution was capable of generating use cases from an unstructured text by properly structuring the text and then identifying the most meaningful items residing in those structures. The main steps of the approach included the identification of roles and the determination of use cases for each identified role. Roles were presented as subjects in sentences (identified as nouns) and use cases were presented as predicates (verb phrases) following the identified subjects; prepositions played the role of identifiers for relationships. From the point of view of SBVR, such an approach provides means similar to the identification of general concepts and verb concepts. The authors of [26] investigated possibilities to automate software requirement elicitation and specification and proposed a pipeline to extract structured requirements in the SBVR format. Furthermore, [27] used SBVR as a mediator representation in the process of transforming business requirements in NL into executable models.

Several other works dealt with the acquisition of conceptual models from the requirements specified in NL [28,29]. These papers provided relevant insights on using patterns and rules for defining use case model concepts by using text-based structures. In fact, it proved to be beneficial to transform requirements (written in NL) to SBVR, which helped to reduce ambiguity and obtain specifications that could be later processed by computers [30], as well as acquire business rules from regulatory texts [31].

Interesting results were presented in [32], where the authors attempted to relate (synchronize) NL specifications with visual UML models. The main idea was to recognize changes and propagate those changes throughout the whole model. The presented approach dealt with UML Class, State Machine, and Activity diagrams. The authors of that paper also argued that such an approach could provide different viewpoints to the problem domain, while at the same time enhancing the communication and quality assurance in various system development activities. Object Process Methodology (OPM) provides its own controlled natural language to ensure synchronization between visual artifacts and their textual representations [33]; however, it does not provide any compatibility with UML-based modeling.

For a more extensive overview of the related work, we refer to [6] which provides an exploration of a more comprehensive set of published research surpassing the threshold of the last decade.

3. Concepts and Definitions

In this section, the following aspects are briefly introduced: relevant concepts of SBVR standard; the developed UML profiles to support the developed DnD transformation metamodel underlying this research; a set of text processing operators relevant to the extraction of SBVR business vocabulary concepts from UML use case models.

3.1. SBVR Business Vocabulary

While the UML use case diagrams do not need many introductions to the IT community, our personal experience and observations show that the case with SBVR is very much different; that is, SBVR still requires an introduction to the audience.

SBVR facilitates the sharing of knowledge between various interested parties within IT projects and beyond [2]. To achieve this, SBVR provides means to formalize knowledge using natural language-based structures or other preferred forms of representation while constrained by formal logic. Together with UML, BPMN, and other OMG's visual modeling standards, SBVR is built upon OMG's Meta-Object Facility (MOF) [3] and provides interchange capabilities, which makes SBVR a part of the OMG's MDA.

Next, we will introduce the core elements of SBVR essential to the research (Figure 1):

- A *noun concept* is a generalization of a *general concept*, an *individual concept*, and a *role* (the latter is irrelevant in this paper). A general concept is a noun concept classifying things based on their common properties. An individual concept is a noun concept that corresponds to only one object (thing).

- A *verb concept* defines some kind of relationship between two or more noun concepts or a characteristic of a noun concept. A verb concept has specializations: association, property association (also: is-property-of verb concept), partitive verb concept (also: part-whole verb concept), and characteristic (also: unary verb concept).

Further categorization of SBVR concepts comprising an SBVR BV is irrelevant to the research presented in this paper and therefore will be elaborated no further.

SBVR standard also defines a *business rule* (BR). It should be noted that even though business rules do not comprise SBVR business vocabularies, they are an essential part of the overall SBVR model and, therefore, we believe it is important to define them while introducing SBVR as well. While SBVR defines a business rule as a 'rule that is under business jurisdiction', we prefer a more comprehensive definition presented in [34]. In that study, a business rule is defined as a part of the whole business model of an organization defining or constraining certain business aspects in certain situations or contexts and ensuring the achievement of one or more business goals.

SBVR model is usually comprised of one or more business vocabulary and business rulesets. The vocabulary is organized as a set of entries representing noun concepts and verb concepts relevant to some domain in a glossary-like manner. An entry begins with a primary presentation denoting the concept's name; each concept may be defined in more detail using non-mandatory fields, e.g., Definition, Concept Type, Synonym, Note, and Example. The development of SBVR models is based on the business rules' 'mantra' [35]. In SBVR terminology, it states that business rules are based on verb concepts and verb concepts are based on noun concepts.

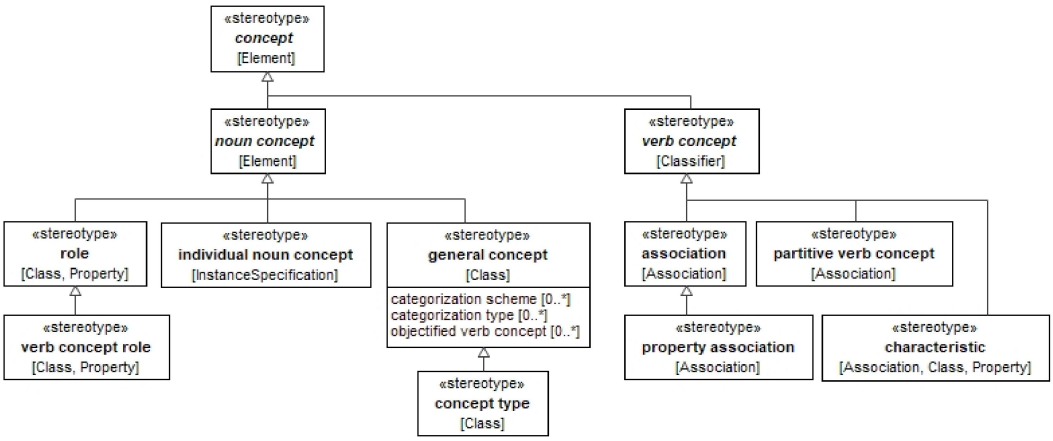


Figure 1. Fragment of UML profile for SBVR Business Vocabulary [36].

In SBVR, the following types of formatting are used to represent SBVR elements in controlled natural language (English, Lithuanian, or any other language of one's choice):

- The 'term' formatting represents a general concept, e.g., 'customer' and 'personal data'. General concepts are presented in singular form using lower case letters.
- The 'Name' formatting represents an individual concept, e.g., 'EU' and 'Kaunas University of Technology'. Individual concepts are usually proper nouns and, therefore, their first letter is capitalized. Individual concepts may also be numbers, e.g., '2020'.
- The '*verb*' formatting is for verbs or prepositions or a combination thereof used in verb concepts. A verb is formed in a singular active form and its synonymous form is passive (e.g., 'customer *provides* personal data' and its synonymous form ' personal data is *provided by* customer').
- The 'keyword' formatting represents linguistic symbols, e.g., 'the', 'then', 'that', 'It is mandatory that', and 'less than'. Keywords are not used directly in expressing noun concepts or verb concepts, but rather in their definitions and other properties

supplementing the concepts' primary presentations. Keywords are also used in expressing business rules.

It is important to note that SBVR business vocabulary may also be represented visually in the form of one or more diagrams. While the SBVR standard itself does not adopt any graphical notation, various existing visual languages and custom DSLs may be developed and utilized for this task. However, as SBVR is a part of the OMG's MOF cluster, it would seem only natural to use a language that is also based on MOF. For our SBVR-related research purposes, we developed a UML profile for SBVR Business Vocabulary and a plug-in for the CASE system MagicDraw. That solution allows modeling SBVR business vocabularies using visual diagrams [36].

For more details on SBVR BV and BR as well as their implementation, refer to [7]. There, we introduced implementation aspects of semantically rich SBVR business vocabularies within the actual modeling environment.

3.2. Architecture of Profiles Supporting DnD Transformations

Model transformation provides the means to develop a target model from a source model. Furthermore, we will provide a brief overview of the architecture of the developed UML profiles to support the DnD transformation metamodel, which was first presented in [7]; the metamodel itself will not be presented here to avoid unnecessary repetition of the already published material. The main engineering developments of the approach were implemented in the CASE system MagicDraw.

Figure 2 presents the main components of the overall implementation architecture. Here, the core of DnD transformation is deployed in the dark-painted *M2M Transformations* profile; native elements of MagicDraw containing other important DnD transformation-related properties are left unpainted.

One of the main parts of DnD transformation is a *transformation pattern* («*TransformationPattern*»). It is a structured class holding two parts in its structure compartment: a *source part* («*Source*») and a *target part* («*Target*»). The source part contains the concept types (together with their relevant properties) of a source model. The target part contains the concept types (together with their relevant properties) of a target model. Within each part, the included concept types may be interconnected with *mapping connectors* («*MappingConnector*»). Connectors are also used to define mappings between elements deployed in different parts (source and target)—these mappings show actual transformation mappings among concept types of the source and target models.

«*Customization*» is a stereotype denoting customization classes. These classes invoke the execution of drag-and-drop action specifications. Property *allowedDragAndDrops* specifies a set of transformation specifications that can be executed after a user drags and drops a source element (*SourceElement*) on the pre-defined target element (*customizationTarget*). Only one transformation specification can be executed per each drag-and-drop action activation. That is, there is more than one transformation specification defined in the *allowedDragAndDrops* property and a user will be asked to select one specific transformation from the contextual menu to be executed.

«*DragAndDropSpecification*» and its specialization «*DragAndDropSpecificationExtension*» realizes the rest of the properties of DnD transformation specification that do not fall into the transformation pattern or customization classes. Those are discussed in detail in [7].

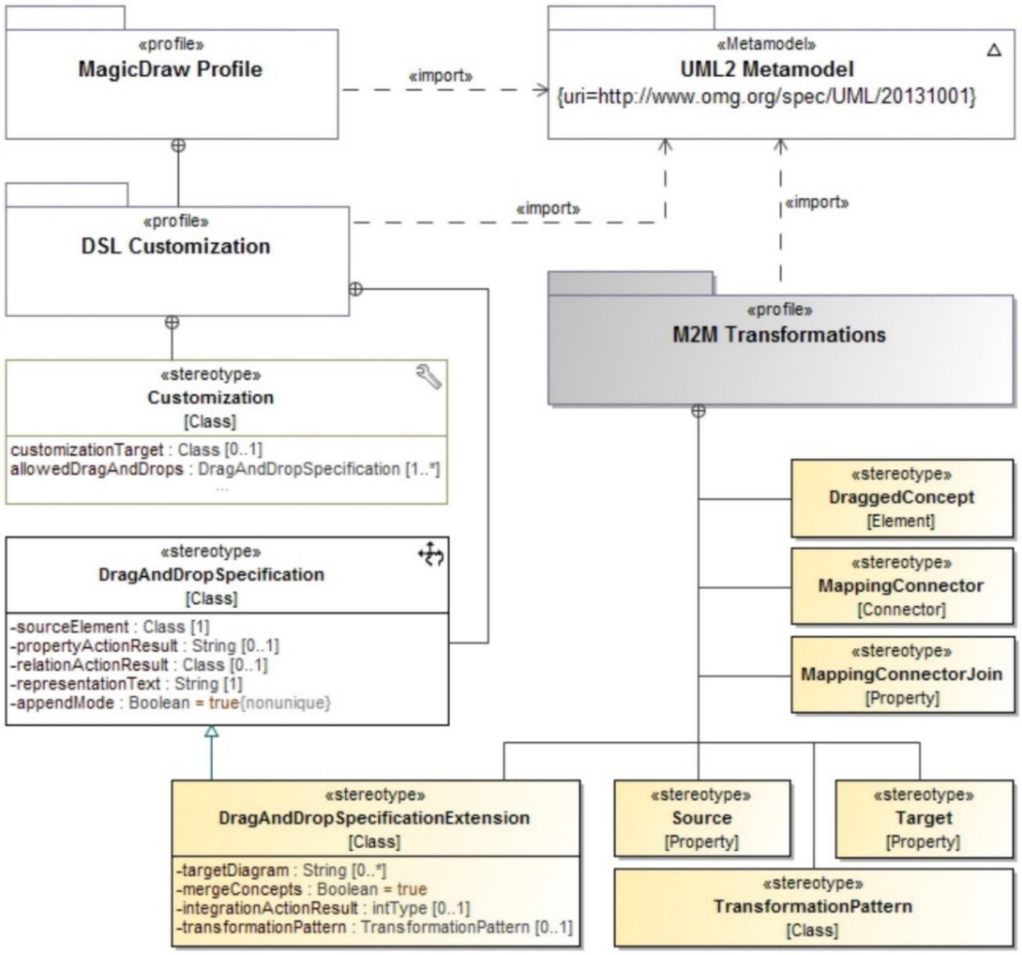


Figure 2. The overall architecture of UML profiles for supporting DnD transformation [7].

The prototype DnD transformations solution is implemented as a plug-in for the CASE system MagicDraw (the latest DnD implementation is accessible at https://bitbucket.org/pauliusdan/magicdraw-d-d-transformations). This CASE system features a UML extension mechanism via UML profiling and extensibility of both the CASE system functionality and DSL engine (via API) [37], which are mandatory to support this approach. Among other components, it includes the *SBVR Modeler* plug-in for MagicDraw, which uses a UML profile for SBVR to enable visual modeling functionality [36], model integration solver plugin, and the SBVR editor. Aside from *Model Integration profiles* and other previously developed transformations, the *DnD Transformations* plugin also contains the library of newly developed UML UCM → SBVR BV transformations, which are introduced further in this paper. The core of the DnD transformations functionality is implemented as the *M2M Transformations Core* framework, which provides abstract and technology-agnostic implementation within our research boundaries. The overall architecture also contains the *NLP component*, which is currently under intense investigation in our research [8,38] but is not used and, hence, is not covered in this paper. For more details about the implementation aspects of the solution, we refer to [7].

3.3. DnD Transformation Specification Explained by Example

Let us assume that a user (e.g., system analyst) has created or somehow obtained, a UML use case model (Figure 3, panel 1), which is represented in the UML use case diagram (Figure 3, panel 3). Having this model, he would then prefer to use it as a source of knowledge for the development of SBVR business vocabulary for that particular business domain.

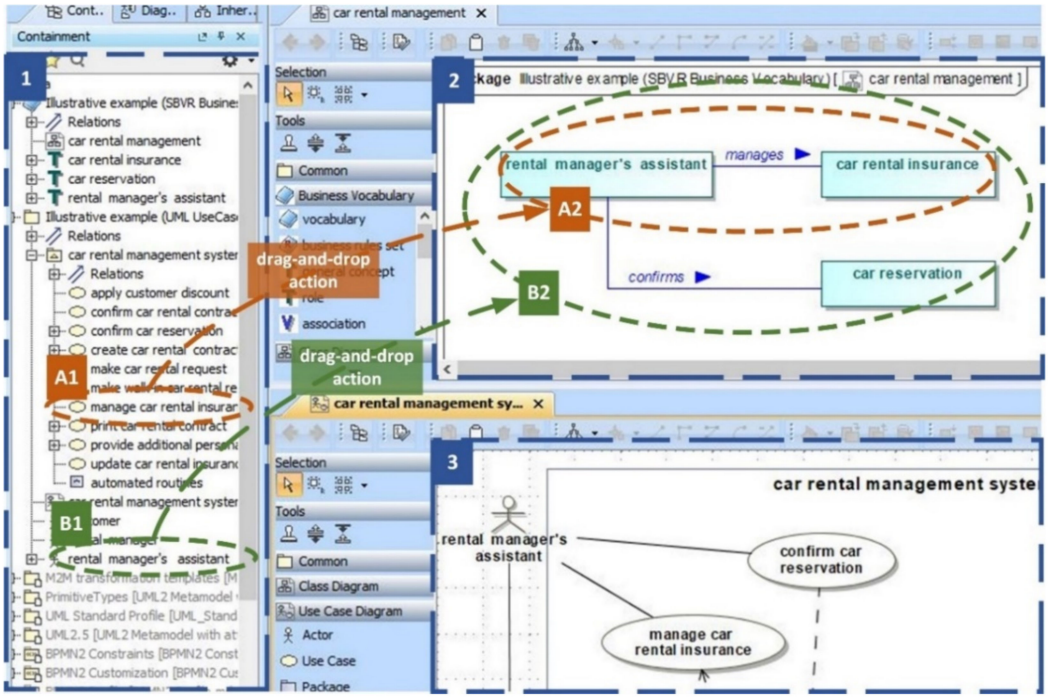


Figure 3. An illustrative example of DnD transformation in action (refer to [6] to view the full specification of the UML use case model and SBVR business vocabulary presented in this example).

While such vocabulary could be developed manually from scratch, its development could also be automated to some degree by utilizing the model transformation functionality provided by the modeling environment. Our presented solution enables the user to use drag-and-drop actions selectively and intuitively on certain use case model elements by triggering predefined transformation actions to generate a set of one or more related business vocabulary elements and represent those elements in the opened SBVR business vocabulary diagram (Figure 3, panel 2).

Next, based on the example presented in Figure 3, let us discuss two illustrative scenarios of using drag-and-drop actions that will enact specific DnD transformations (Figure 4) producing certain results:

1. If we select a use case 'manage car rental insurance' (Figure 3, panel A1) and drag-and-drop it onto the business vocabulary diagram, then a verb concept 'rental manager's assistant *manages* car rental insurance' will be automatically created in the SBVR business vocabulary (Figure 3, panel A2). This occurs because we specified a transformation rule to transform a use case and its actors to a corresponding number of verb concepts (a verb concept for each triplet <*UseCase-Association-Actor*>) when that use case is dropped onto the business vocabulary diagram (Figure 4, panel 2).
2. If we select a use case 'rental manager's assistant' (Figure 3, panel B1) and drag-and-drop it onto the business vocabulary diagram, then two verb concepts, namely 'rental manager's assistant *manages* car rental insurance' and 'rental manager's assistant *confirms* car reservation', will be automatically created in the SBVR business vocabulary (Figure 3, panel B2). This happens because we specified a DnD transformation rule instructed to transform an actor and all use cases associated with it to a corresponding number of verb concepts (one verb concept for each triplet <*Actor-Association-UseCase*>) when that actor is dragged and dropped onto the business vocabulary diagram (Figure 4, panel 2).

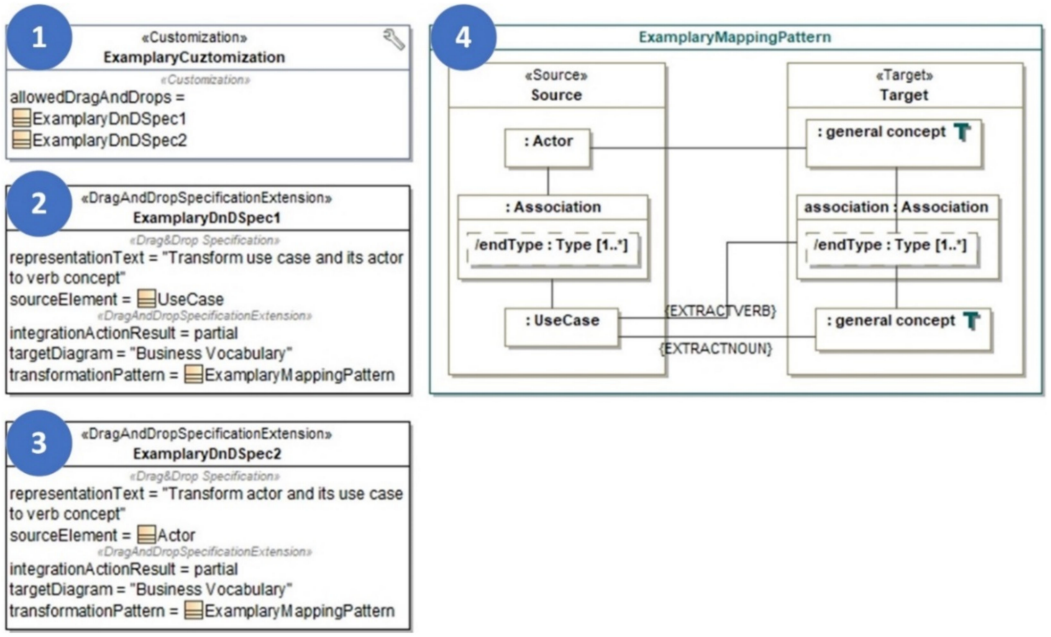


Figure 4. Example of DnD transformations specification.

Figure 4 presents the DnD transformation specification, which is executed by the transformation engine upon performing the above-mentioned scenarios.

The DnD transformation specification is composed of a customization class *ExamplaryCustomization* (Figure 4, panel 1), DnD action specifications *ExamplaryDnDSpec1* and *ExamplaryDnDSpec2* (Figure 4, panels 2,3), and a transformation pattern *ExamplaryMappingPattern* (Figure 4, panel 4). *ExamplaryCustomization* refers to the specified *ExamplaryDnDSpec1* and *ExamplaryDnDSpec2* allowing these specifications to be executed by the transformation engine when a user drags a *sourceElement* and drops it onto a *customizationTarget*. In our case, the *customizationTarget* is not defined because the target element is the target diagram itself defined in the *targetDiagram* property.

In this case, the *ExamplaryCustomization* together with *ExamplaryDnDSpec1* (Figure 4, panel 2) defines that the transformation will be executed after a user drags an instance of *UseCase* from the use case model and drops it onto the *Business Vocabulary* diagram. Accordingly, the same *ExamplaryCustomization* and the *second* DnD action specification *ExamplaryDnDSpec2* (Figure 4, panel 3) defines that the transformation will be executed after a user drags an instance of an *Actor* from the source model and drops it onto the target diagram. Note that both *ExamplaryDnDSpec1* and *ExamplaryDnDSpec2* define the same *ExamplaryMappingPattern*, which means that both transformations will refer to the same transformation pattern upon their execution. The *ExamplaryMappingPattern* defines the following:

- *Source part*, which specifies UML *UseCase Model* pattern holding *Actor*, *UseCase*, and *Association*;
- Target part, which specifies the SBVR Business Vocabulary pattern holding two general concepts related to one another using association. The triplet <general concept-association-general concept> forms a verb concept;
- *Connectors* that act as *mappings* between inner elements of the source and target parts. These mappings define atomic transformations on the element-to-element level;
- *Connectors* among the inner elements themselves within the source and target parts. Relationships among the inner elements are defined based on the underlying meta-models of UML and SBVR.

In this case, after a user drags the use case 'manage car rental insurance' from the UCM and drops it onto the diagram of a business vocabulary (Figure 3, panel A1), all actors

associated with the dragged use case will be collected and transformed to corresponding SBVR elements together with the use case. The transformation result will comprise a general concept 'rental manager's assistant', association with verb wording '*manages*', and another general concept 'car rental insurance'. The transformation also includes NLP operators EXTRACTNOUN() and EXTRACTVERB() for extracting noun/noun phrase and verb/verb phrase correspondingly (see Section 3.4 for more information). Following a similar logic, another specified DnD action specification (Figure 4, panel 3) is executed when selecting the actor 'rental manager's assistant' and dropping it into the business vocabulary diagram. To see a full set of the specified DnD transformations, refer to Section 4.

3.4. Linguistic Text Processing

Text processing operators not only help to improve the quality of results of *automatic* M2M transformations but also speed up the process of *semi-automatic* transformations by adding a degree of automation to text analysis, which would otherwise have to be performed by a user himself. In this section, we apply a set of linguistic text processing operators that improves text processing tasks; these operators were previously introduced in [38].

The following operators were defined for the linguistic text processing in the UML UCM → SBVR BV approach:

- EXTRACTNOUN(*phrase*, '*all*')—extracts nouns/noun phrases from the *phrase*. If the second parameter is set to '*all*', all possible nouns/noun phrases will be extracted; otherwise, the most general noun phrase will be extracted;
- EXTRACTVERB(*phrase*)—extracts a verb wording from the *phrase*;
- CONCAT(*phrase*, . . .)—concatenates multiple text phrases, including the names of the source elements or conventional text chunks (which are represented using single quotes such as '*text*').

In this paper, we did not consider advanced NLP functionality as one of our objectives. Therefore, we present the simplified versions of EXTRACTNOUN() and EXTRACTVERB(), which were applied for text processing in this paper:

- EXTRACTVERB() is simplified to extracting the first word in the phrase, relying upon the fact that this operator will mostly be applied for processing Use Case element names, which are considered to be named using the pattern <VERB><NOUN> | <NOUN PHRASE>. This pattern is generally considered as a good naming practice for naming activity-like UML model elements, such as use cases in use case diagrams or activities in activity diagrams.
- EXTRACTNOUN() is simplified to extracting the remaining part of the given phrase.

These restrictions do not impact evaluation results presented in this paper as the focus lays on the *structural soundness* of the transformation output. In other words, we seek to verify if a correct number of corresponding target elements are generated, rather than *semantic soundness*, which aims at verifying the semantic validity of the acquired output (e.g., whether the label *actually* represents a valid noun phrase or a verb phrase).

4. Specifications of Transformation Rules for UML UCM → SBVR BV Transformation

This section introduces a set of model-based transformation rule specifications for the UML UCM → SBVR BV approach.

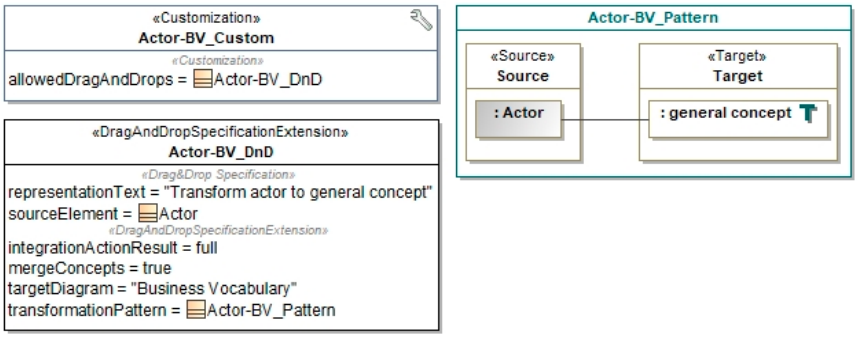
Note that the formal definition of the set of identified transformation rules was already presented in [6]; therefore, to save space and to avoid unnecessary repetition, we will omit the formal presentation of rules in this paper.

4.1. DnD Transformation Rules for Extracting SBVR Noun Concepts

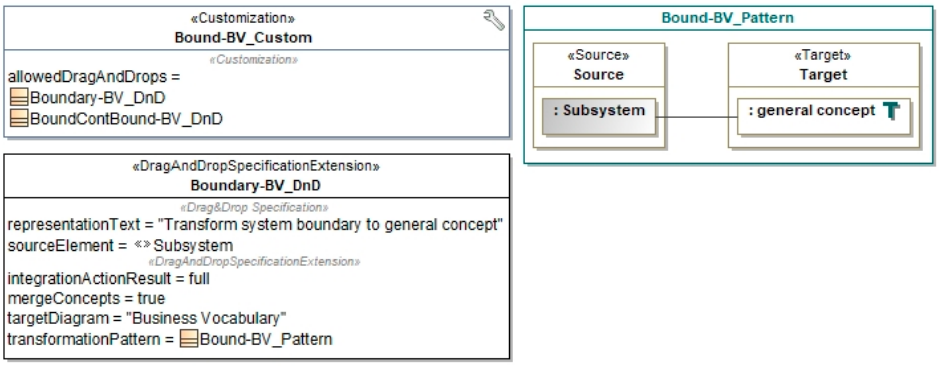
Listing 1 presents the specification of transformation rules for extracting SBVR general concepts from UML use case models.

Listing 1. Transformation rules for extracting SBVR general concepts from UML use case models.

***T*1. Transform *an actor* (UML) into a *general concept* (SBVR).**
Specification:

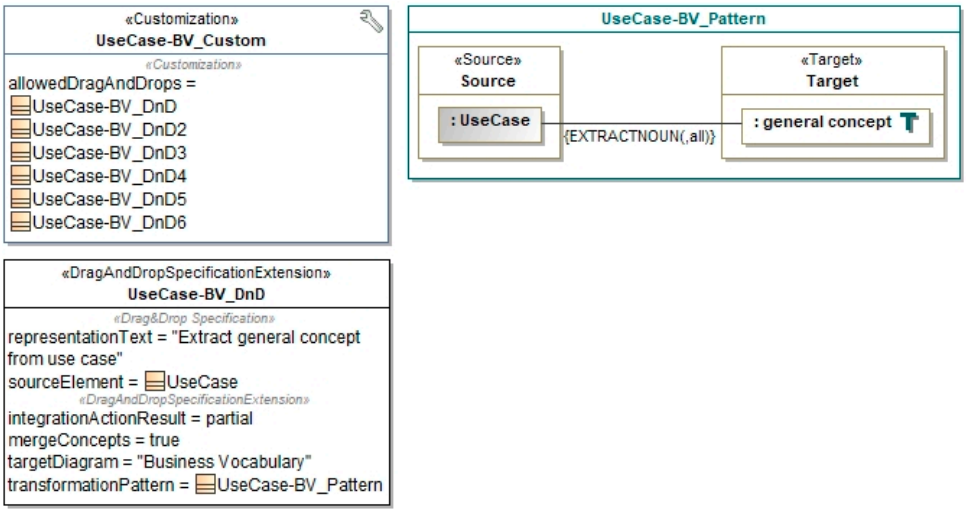


***T*2. Transform a *system boundary* (UML) into a *general concept* (SBVR).**
Specification:



Note that this transformation specification is comprised of two transformation rules T_2 and T_8 specified by a customization class *Boundary-BV_Custom* referencing two drag-and-drop specifications (*Boundary-BV_DnD*—for the T_2; *BoundaryContBoundary-BV_DnD*—for the T_8) and their corresponding transformation patterns (Sections 3.2 and 3.3 explains these dependencies in more detail). This is because both these rules have the same *customization target*, i.e., the target element, which is where the source element is being dragged and dropped onto. In this case, the customization target has no assigned value meaning that the source elements are being dragged and dropped directly onto *targetDiagram*, which is equal to 'Business Vocabulary'. To avoid excessive details that are irrelevant for the *particular* transformation rule T_x, we will present only those classes that are relevant to that particular T_x. In the context of T_2, classes related to T_8 are irrelevant and therefore are filtered out from this specification view. The same principle holds for all DnD transformation specifications presented in Listings 1 and 2.

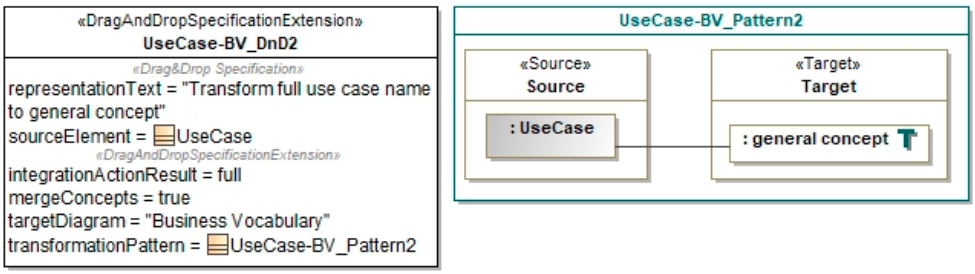
***T*3. Form one or more *general concepts* (SBVR) by text-processing the name of a *use case* (UML).**
Text processing. Extract one or more nouns/noun phrases from the name of a use case:
EXTRACTNOUN(*UCM(use_case: UseCase), all*) → *{noun_nounphrase}*.
The extract *{noun_nounphrase}* will be formed into one or more general concepts upon the execution of the T_3 transformation rule.
Specification:



***T*4. Form a *general concept* (SBVR) from the *whole* name of a use case (UML).**

The acquired general concept are used for constructing verb concepts representing specializations and <<*include*>> | <<*extend*>> relationships among use cases.

Specification:



Note: *Customization* class for this transformation is presented in T3.

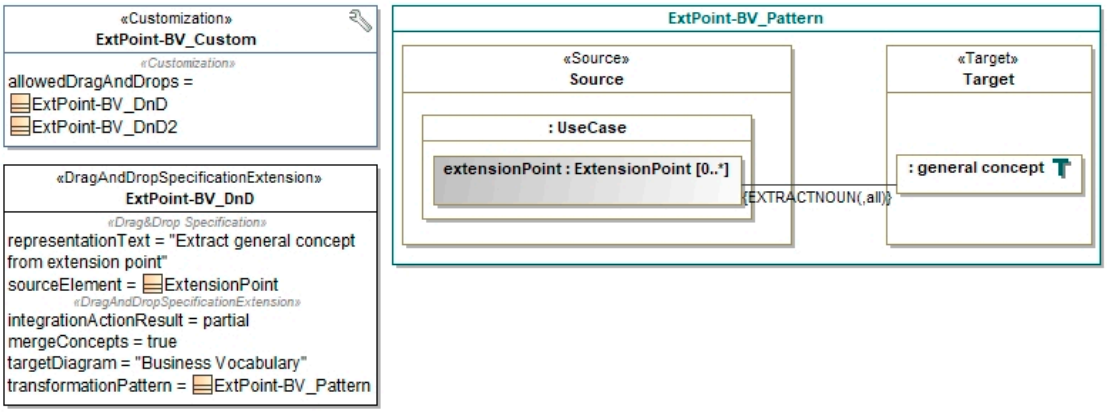
***T*5. Extract one or more *general concepts* (SBVR) from the text expression representing an *extension point* (UML).**

Text processing. Extract one or more nouns/noun phrases from the extension point:

EXTRACTNOUN(*UCM(ext_point: UseCase), all*) → *{noun_nounphrase}.*

The extracted *{noun_nounphrase}* will be formed into one or more general concepts upon the execution of the T5 transformation rule.

Specification:



***T*6. Extract additional, more general, or otherwise relevant *general concepts* (SBVR) from the general concepts already formed by the transformation rules T3 and T5.**

Specification: no separate DnD transformation is specified for the rule T6 because this rule is redundant as all possible nouns and noun phrases are to be extracted by performing T3 and T5 transformation rules, which will invoke NLP operators in their pre-processing stage.

4.2. DnD Transformation Rules for Extracting SBVR Verb Concepts

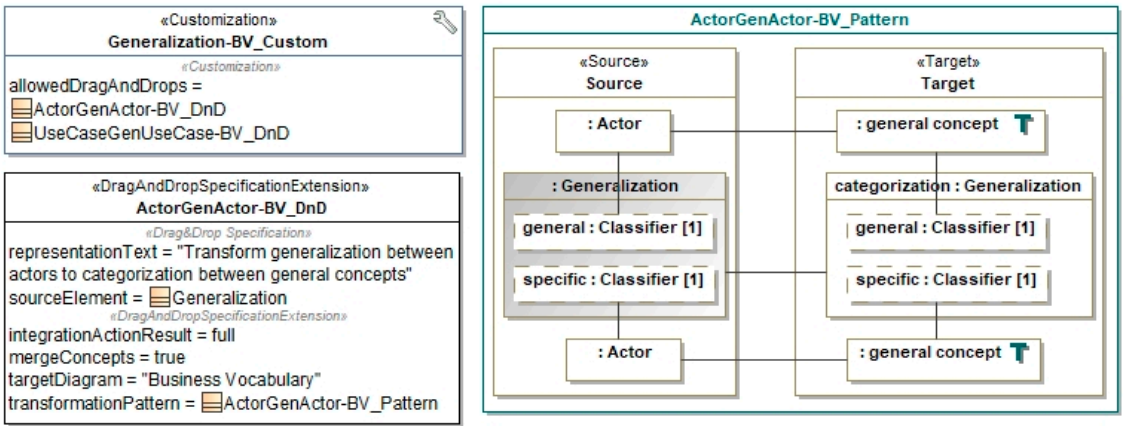
According to the business rules 'mantra' [35], verb concepts are built upon noun concepts. Therefore, any of the following DnD transformation specifications implementing T_7–T_{20} rules will incorporate patterns from DnD transformation specifications implement-

ing T_1–T_6 rules, which were relevant for the extraction of noun concepts. Next, Listing 2 presents specifications of transformation rules for extracting SBVR verb concepts from UML use case models.

Listing 2. Transformation rules for extracting SBVR verb concepts from UML use case models.

***T7*. Form a *verb concept* (SBVR) by transforming a *generalization* relationship between two *actors* (UML). The two general concepts representing actors are bound using a reserved verb wording '*generalizes*' in the verb concept (i.e., general concept1 *generalizes* general concept2).**

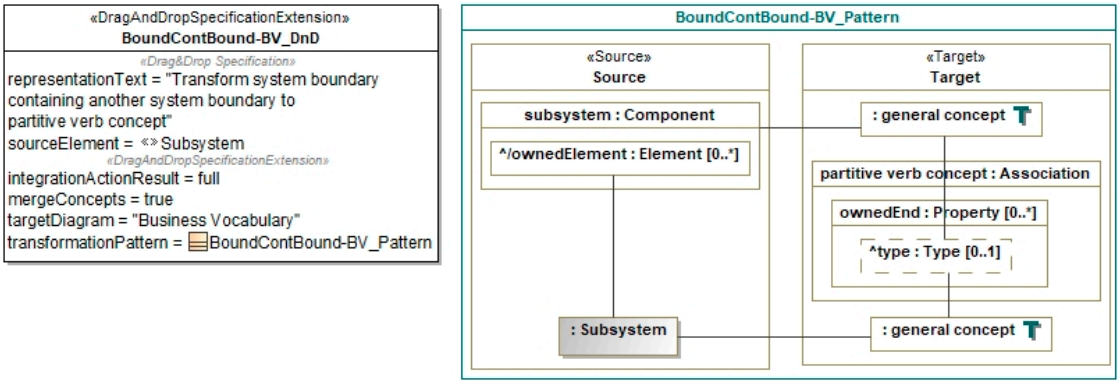
Specification:



Note that in the *Target part*, the name '*categorization*' is used for the *Generalization* element. Such a naming convention is used to link concept type *A* of the UML profile for SBVR (i.e., '*categorization*' in this particular case) with the concept type *B* of the UML itself (i.e., '*Generalization*'), from which that element *A* is derived. This is more of a technical specificity of the CASE system on how it handles UML profiles rather than the rule imposed by us. This element naming convention holds for all other relevant cases in this set of DnD transformation specifications.

***T8*. Form a *partitive verb concept* (SBVR) by transforming an is-part-of relationship between two *system boundaries* (UCM). The two general concepts representing system boundaries are bound using a reserved verb wording '*contains*' in the verb concept (i.e., general concept1 *contains* general concept2).**

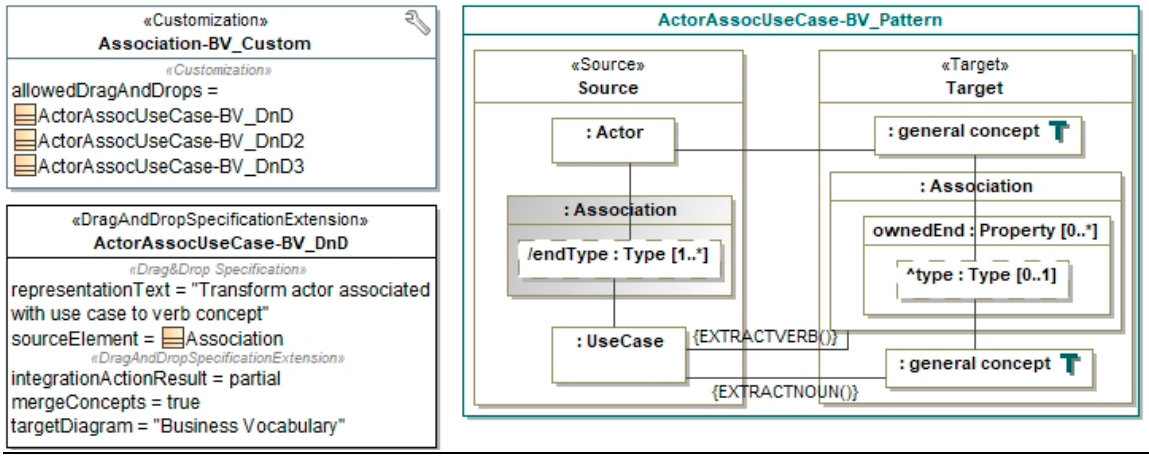
Specification:



Note: *Customization* class for this transformation is presented in T_2.

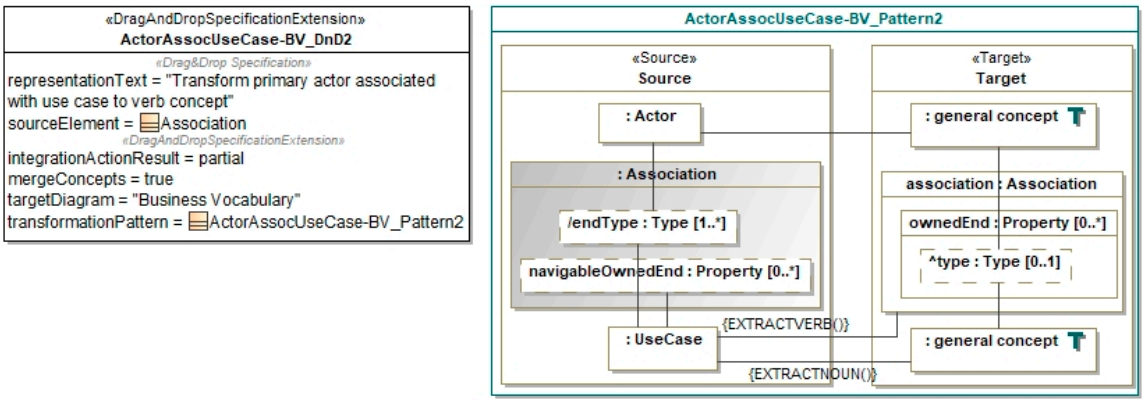
***T9*. Form a *verb concept* (SBVR) by transforming a *use case* performed by an *actor* (UCM).**

Specification:



***T10*. Form a *verb concept* (SBVR) by transforming a *use case* performed by a *primary actor*. The rule's logic is similar to the one defined for the transformation rule T9.**

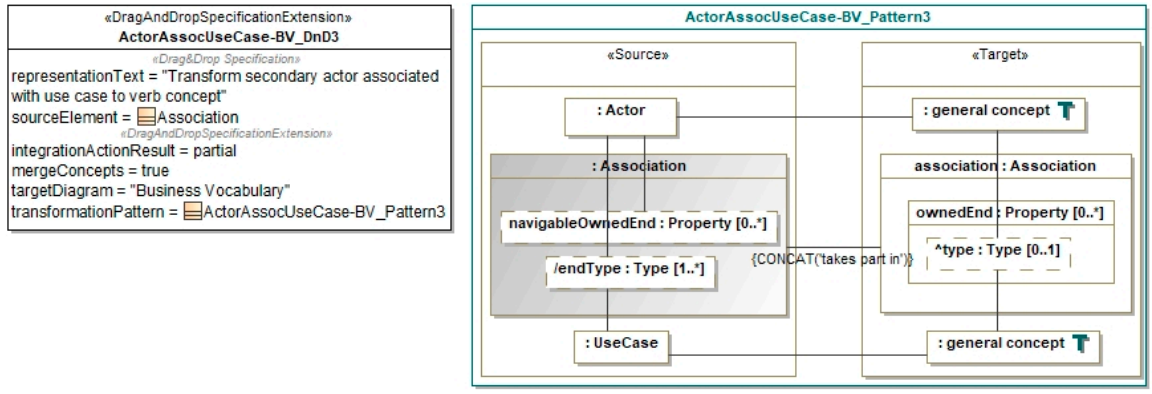
Specification:



Note: *Customization* class for this transformation is presented in T9.

***T11*. Form a *verb concept* (SBVR) by transforming a relationship between a *use case* and a *secondary actor* (UCM). Two general concepts representing a secondary actor and a use case are bound using a reserved verb wording *'takes part in'* in the verb concept (i.e., general concept1 *takes part in* general concept2).**

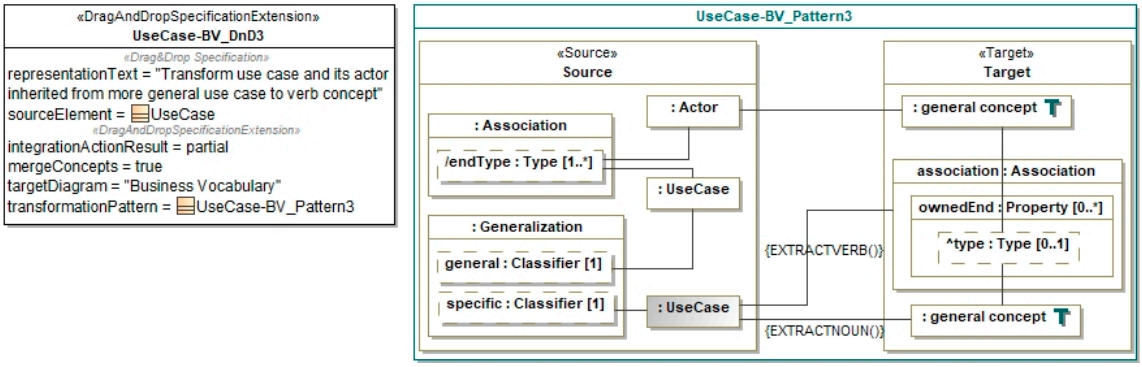
Specification:



Note: *Customization* class for this transformation is presented in T9.

***T12*. Form a *verb concept* (SBVR) by transforming a specialized *use case* performed by an *actor* inherited from a more general *use case* (UML).**

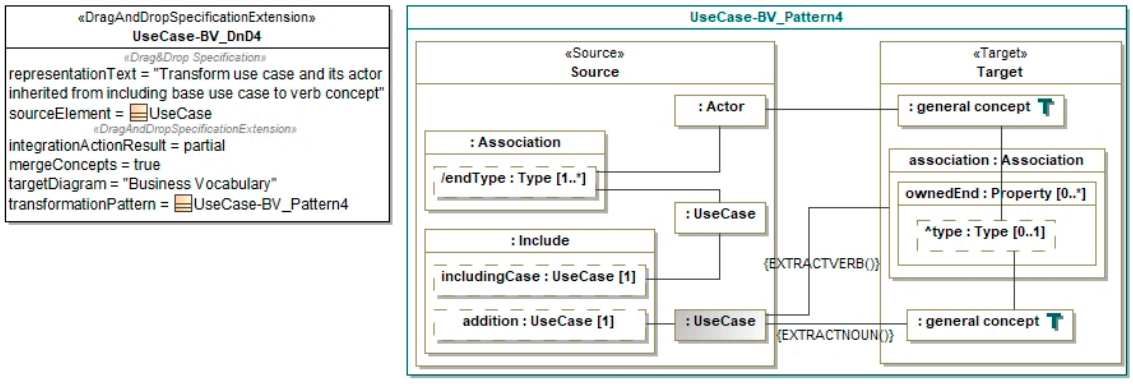
Specification:



Note: *Customization* class for this transformation is presented in T3.

***T13*. Form a *verb concept* (SBVR) by transforming an included *use case* (the addition) performed by an *actor* inherited from an including (the base) *use case* (UML).**

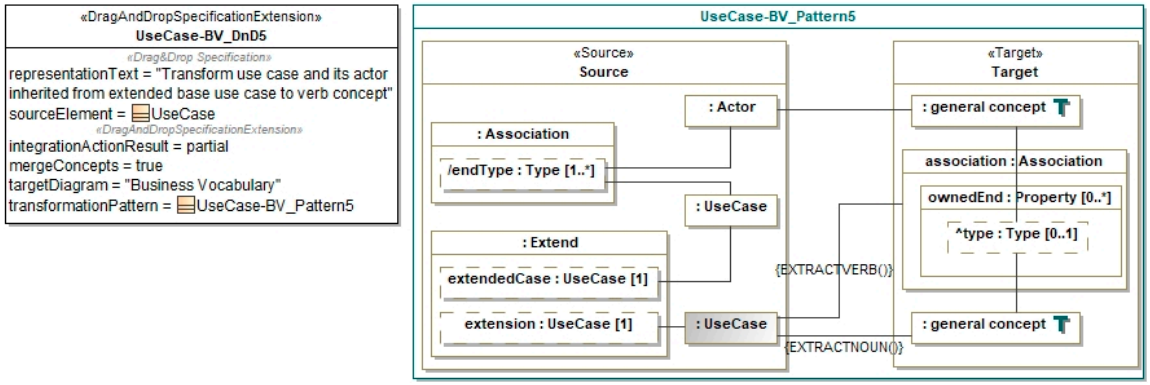
Specification:



Note: *Customization* class for this transformation is presented in T3.

***T14*. Form a *verb concept* (SBVR) by transforming an extending *use case* (the extension) performed by an *actor* inherited from an extended (the base) *use case* (UML).**

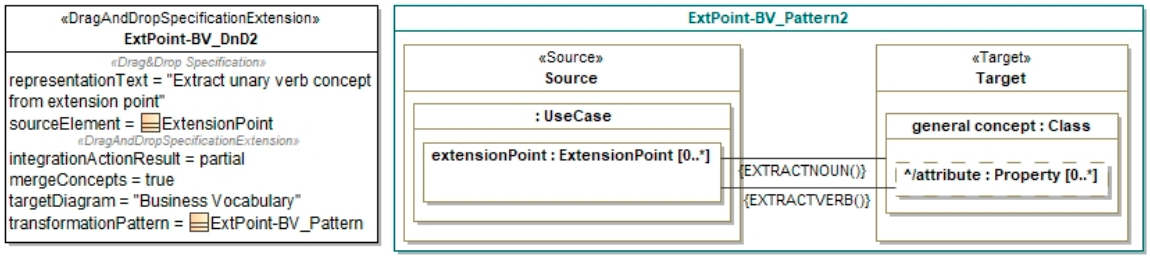
Specification:



Note: *Customization* class for this transformation is presented in T3.

***T15*. Form one or more *unary verb concepts* (*characteristics*) (SBVR) by transforming the condition of an *extension point* (UML).**

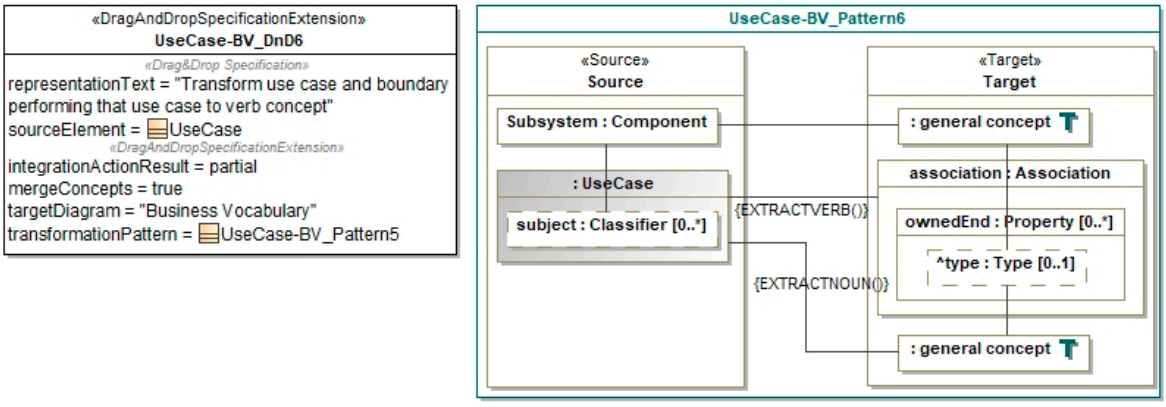
Specification:



Note: *Customization* class for this transformation is presented in T5.

***T16*. Form a *verb concept* (SBVR) by transforming a *use case* executed by a *boundary* in which that *use case* is contained (UML).**

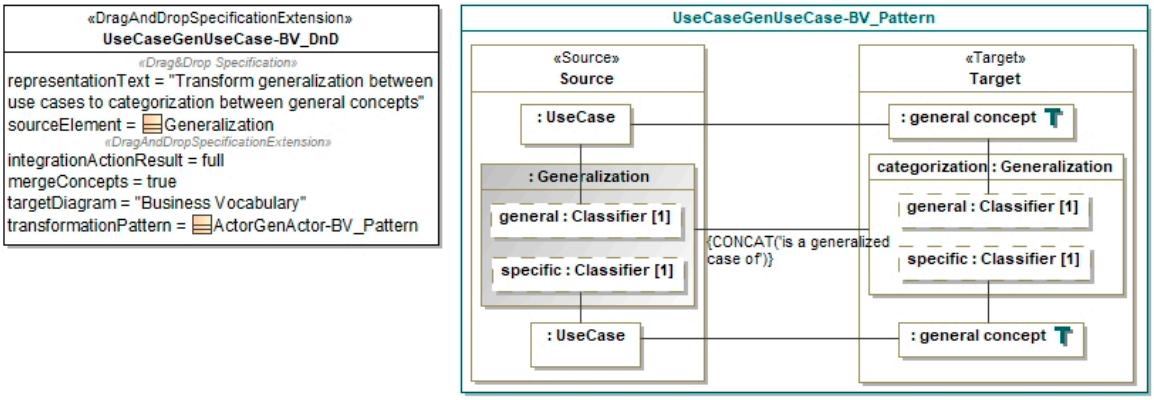
Specification:



Note: *Customization* class for this transformation is presented in T3.

***T17*. Form a *verb concept* (SBVR) by transforming a *Generalization* relationship between two *use cases* (UML). Two general concepts representing use cases are bound using a reserved verb wording *'is a generalized case of'* in the verb concept (i.e., general concept1 *is a generalized case of* general concept2).**

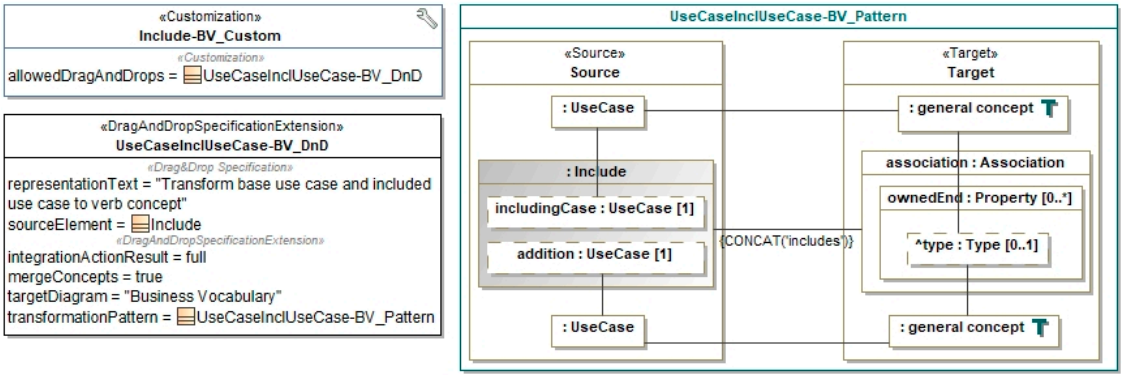
Specification:



Note: *Customization* class for this transformation is presented in T7.

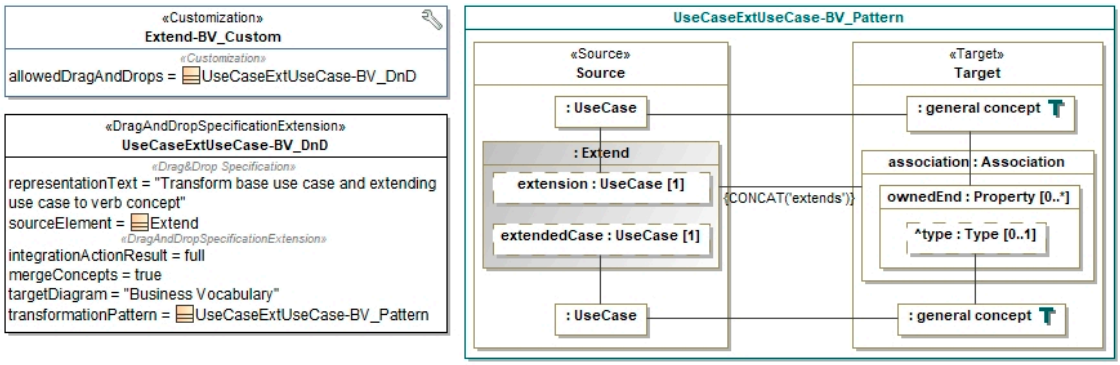
***T18*. Form a *verb concept* (SBVR) by transforming an *Include* relationship between two *use cases* (UML). Two general concepts representing use cases are bound using a reserved verb wording *'includes'* in the verb concept (i.e., general concept1 *includes* general concept2).**

Specification:



***T*19. The rule forms a *verb concept* (SBVR) by transforming an *Extend* relationship between two *use cases* (UML). Two general concepts representing use cases are bound using a reserved verb wording '*extends*' in the verb concept (i.e., general concept2 *extends* general concept1).**

Specification:



***T*20. The rule forms additional *verb concepts* (SBVR) by analyzing and transforming the relationships among the general concepts formed as a result of the T6.**

Specification: This rule cannot be modeled and executed using the current existing DnD transformation means. Such transformation is based purely on semantic text analysis (detection and resolution of synonyms, hyponyms, hypernyms, holonyms, meronyms, etc.).

Note that the applied techniques do not automatically validate the semantics of the extracted verb concepts. It is assumed that the extracted SBVR BV entries are to be checked by a user himself who is actively participating in the whole transformation process. The transformations satisfy some of the formal requirements of model transformation languages, such as determinism, termination, or typing, as they are assumed to always terminate and provide the expected set of outputs with the predefined types each time they are executed given valid source inputs. Nevertheless, due to the definition of partial transformations, the user himself is responsible for the full validation of the output elements and their semantic correspondence to the modeling objectives.

5. Experimental Evaluation

The main goal of the experiment was to find out the accuracy of DnD transformations (T_{DnD}) for extracting structurally sound SBVR BV from UML UCM by comparing that result to the result of manual extraction T_M, which is considered as a benchmark ('gold standard') in our experiment. It should be pointed out that the semantic correctness of the names (or labels) of output elements themselves are not considered in this experiment as this would require introducing advanced NLP techniques, which is out of the scope of this paper.

The experiment was performed using 10 real-world UML use case models each containing one or more use case diagrams. Note that for both T_{DnD} and T_M, the same given

set of transformations (see T_1–T_5, T_7–T_{19}, Section 4) and the same set of UML use case models as in [6] were used to maintain the same level of objectivity between both conducted research works. Where needed, the refactoring of the source models was performed to eliminate any syntactical modeling errors. Both T_{DnD} and T_M performed by the authors of the paper as this did not expose any threats to the experiment validity; in other words, the human factor does not carry any impact on the performance of the DnD transformations.

Statistics of the input set *S* together with the benchmark results of T_M are provided in Table 1. Here, 'n/a' means that the UCM element, which was a source for the particular transformation T_{DnD}, was absent in the source model, meaning the transformation was not invoked for that particular case.

Table 1. Statistics of the input set *S* and benchmark results of T_M.

	Statistics of Source UCM									**Results of T_M**		
	Diagrams	**Actors**	**Use Cases**	**Associations**	**Boundaries**	**Include**	**Extend**	**Extension Points**	**Generalizations**	**General Concepts**	**Verb Concepts**	**Total Extracted**
Case 1	5	8	16	34	9	5	n/a	n/a	n/a	38	42	92
Case 2	3	12	49	52	7	7	11	n/a	n/a	101	81	215
Case 3	1	3	11	10	2	n/a	3	3	n/a	20	14	38
Case 4	2	6	24	11	3	4	5	n/a	n/a	43	28	80
Case 5	7	11	39	36	8	n/a	18	n/a	4	75	58	143
Case 6	2	11	35	34	7	n/a	3	n/a	n/a	78	44	127
Case 7	2	5	28	17	5	3	11	2	3	50	40	108
Case 8	2	8	33	24	8	n/a	9	n/a	4	63	44	111
Case 9	2	4	50	39	12	10	7	7	n/a	83	58	165
Case 10	2	2	39	7	8	9	24	n/a	1	80	61	170

Table 2 presents the results of the experiment in the form of the *accuracy* of the extracted results' *structural soundness* per each transformation rule. Here, we will follow the notation used in [38]. Formally, we define the output of a transformation T as *structurally sound* if, for each instance of the source model's concept type (I_{SMCT}) and any generated output element (instance of the target model's concept type I_{TMCT}), there exist one or more constrained mappings between any I_{TMCT} and any I_{TMCT} as the result of T and a bijective single trace relationship between any I_{TMCT} generated using T and any I_{SMCT}, i.e.: $\forall I_{SMCT}, \forall I_{TMCT} : \exists \mathcal{M}(I_{SMCT} \rightarrow I_{TMCT}) \wedge \mathcal{T}(I_{TMCT} \rightarrow I_{SMCT})$. That is, each generated element must be matched to any I_{SMCT} and there must exist one or more generated elements for each I_{SMCT}, unless this is restricted by predefined mapping constraints. Hence, we essentially perform checking if all the elements required to be generated by the predefined transformation were generated successfully.

Table 2. Transformation execution performance.

Rule	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
T_1	1	1	1	1	1	1	1	1	1	1
T_2	1	1	1	1	1	1	1	1	1	1
T_3	1	0.891	0.818	0.917	0.949	0.921	0.929	0.909	0.88	1
T_4	1	1	1	1	1	1	1	1	1	1
T_5	n/a	n/a	1	n/a	n/a	n/a	1	n/a	1	n/a
T_7	n/a	1	n/a	1	1	n/a	1	1	1	1
T_8	1	1	n/a	n/a	n/a	1	1	n/a	1	n/a
T_9	1	0.897	0.8	0.818	0.944	0.914	0.882	0.903	0.872	1
T_{10}	1	0.897	0.8	0.818	0.944	0.914	0.882	0.903	0.872	1
T_{11}	1	1	1	1	1	1	1	1	1	1
T_{12}	n/a	n/a	n/a	1	n/a	n/a	n/a	n/a	n/a	n/a
T_{13}	1	1	n/a	n/a	n/a	n/a	1	1	1	1
T_{14}	n/a	n/a	1	n/a	1	n/a	1	n/a	n/a	1
T_{15}	n/a	n/a	1	n/a	n/a	n/a	1	n/a	1	n/a
T_{16}	1	1	1	1	1	1	1	1	1	1
T_{17}	n/a	n/a	n/a	1	n/a	n/a	n/a	n/a	n/a	n/a
T_{18}	1	1	n/a	1	n/a	n/a	1	1	1	1
T_{19}	n/a	1	1	1	1	1	1	1	1	1

The accuracy for the output's structural soundness for the rule T for the model M is defined as follows: Given $\#^i_{gen}$ as the number of generated outputs, $\#^i_{actual}$ as the number of actual elements to be generated, and n as the number of transformations required to be performed using rule T_r and model M, we define accuracy as the proportion of correctly performed transformations that produced structurally sound outputs with the correct number of output elements:

$$Acc(M,\ T_r) = \frac{\sum_{i=1}^{n} \mathbf{1}\left(\#^i_{gen} = \#^i_{actual}\right)}{n} \tag{1}$$

The value of 1.0 is the maximum possible value, which indicates that all transformations were performed as expected. 'n/a' indicates that particular rules could not be invoked on these cases due to the absence of UML elements in the model required for execution. We excluded rules T_6 and T_{20} from evaluation due to certain NLP processing requirements, which are out of scope in this paper.

While the nature of the predefined transformations suggests that these values should be equal to 1.0, this is not always true for the transformations that apply two or more text processing operators on the model elements possessing ill-named labels. Table 2 shows that transformations comprising mappings with EXTRACTNOUN/EXTRACTVERB operators often failed to achieve maximum accuracy values. This is explained by the fact that to apply both these operations successfully, text labels must contain at least two words, which was not always the case. There were cases when a UseCase element name contained a single word (e.g., '*Login*' and '*Start*'), which resulted in the extraction of valid verbs, but also in failure to extract noun phrases required to generate general concepts for valid output tuples *(general concept-association-general concept)*. We addressed this issue in [38] by introducing DnD transformation specifications to support conditional branching, which could be used to handle such exceptions and to apply different processing logic of different cases. Therefore, it is safe to conclude that model element naming practices do influence the actual performance of the transformations. At the same time, it should be mentioned that the impact of some of the issues could be minimized to a certain degree by

utilizing advanced NLP techniques [38]. Nonetheless, the obtained results indicate that the developed DnD transformations provided structurally correct results for most of the performed drag-and-drop actions with the selected UCM elements from *S*.

Moreover, several other observations were made after the experiment:

- Drag-and-drop actions-based transformations proved to be useful and provide the required level of flexibility in both the visual development of transformations themselves as well as their actual application. In the future, the existing DnD transformations library could be further extended by adding additional draggable source elements to DnD specification classes referencing the already existing transformation patterns or the newly developed ones. This would allow a user to trigger transformations from an even larger set of draggable elements in the use case model.
- We argue that the DnD transformation approach may not be the best choice for performing full-scale M2M transformation on large models. The reason is that it would take a reasonable amount of time to drag-and-drop every required source model element to acquire the full target model based on the provided source model. For the full-scale M2M transformation, it is advised to use the automatic or semi-automatic M2M transformation approach presented in [6]. The DnD transformation approach is better suited for the agile and incremental development of visual models and/or when the models under development are of a smaller volume.
- The experiment provided a strong reassurance that our developed DnD transformation technology can be applied to the development of transformations for UML and other visual languages expressed via UML profiles.
- It can be safely concluded that data quality is one of the determining factors which must be considered while applying this approach. No NLP tool will be able to handle invalid names or bad modeling practices despite its performance in other fields.
- While NLP functionality was not considered in this paper, it can be useful to improve the semantic quality of the generated elements [8,38]. Furthermore, this would allow enabling the transformation rules T_6 and T_{20} that were described in this paper but excluded from our experimentation. For example, detection of synonyms could help improve merging (or semantical relations identification) of the synonymous concepts in the SBVR business vocabulary.
- Moreover, NLP capabilities enable the processing of 'one-to-many' type of transformation rules (i.e., the ones that generate multiple output elements for a given single input element). This is enabled by using the '*all*' parameter in the EXTRACTNOUN() operator, which would then extract *all* available noun phrases from the text instead of a single text chunk. This also is useful in processing disjunctive or conjunctive clauses containing multiple instances of verb and noun phrases. However, such processing of element labels is only possible when using advanced NLP techniques, such as dependency parsing or relation extraction.

6. Conclusions

Well-structured and formalized business vocabularies provide certain benefits to system developers by providing a shared vocabulary for the project and means to reuse existing artifacts, etc. However, manual development of such business vocabularies proves to be a human effort-consuming and time-consuming task. We argue that one of the methods of acquiring business vocabularies faster and with less effort is to automate the extraction of such vocabularies from the existing sources of knowledge.

In this paper, we presented conceptual and implementational aspects of the approach for extracting SBVR BV from UML UCM, which utilizes M2M transformation technology based on the drag-and-drop actions (*DnD transformation*) presented in [7]. One of the key features of DnD transformation technology is the model-driven development and customization of M2M transformations. Such an approach increases the extensibility, flexibility, and usability features of this kind of M2M transformation when used in a CASE system.

The conducted experiment shows that the developed library of DnD transformations correctly interprets and executes the full set of transformation rules for the extraction of SBVR business vocabularies provided in [6]. This provides our approach with the same transformation power and, at the same time, more flexibility to the overall modeling process when compared to our previous development of UML UCM → SBVR BV transformation. It also proves, once again, that our developed DnD transformation technology can be applied to transforming various models based on UML and UML profiles.

For future developments, we view improving the NLP component as the most promising research trend providing even greater level automation and semantic quality to the transformation results. Such an NLP component could then be adopted for other already developed M2M transformation approaches as well as future developments in this area, providing impact to the whole model-driven system development process. Yet another research could be focused on the development of DnD transformations library for generating UML use case models from the existing SBVR business vocabularies (possibly, from business rulesets)—this would be a model transformation to the opposite direction compared to the one presented in this paper. Such development would deliver a complete all-around transformation solution (UML UCM ←→ SBVR BV). From the DnD transformation technology point of view, this model transformation could be developed relatively easily; the harder part would be the identification of the full set of mapping rules for this new model transformation because such a bi-directional transformation would not be fully reflective.

Author Contributions: Conceptualization, T.S. and P.D.; methodology, T.S. and P.D.; software, P.D. and T.S.; validation, T.S., P.D., R.B., A.O. and J.C.; investigation, P.D., R.B., A.O. and J.C.; resources, P.D., T.S. and R.B.; writing—original draft preparation, T.S., P.D.; writing—review and editing, T.S., P.D., R.B., A.O. and J.C.; visualization, T.S.; supervision, T.S.; funding acquisition, R.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. OMG. Unified Modeling Language (OMG UML). Version 2.5. OMG Doc. Number: Formal/2015-03-01. Available online: www.omg.org/spec/UML/2.5 (accessed on 4 April 2020).
2. OMG. Semantics of Business Vocabulary and Rules (SBVR) v.1.4, OMG Doc. No.: Formal/2017-05-05. 2017. Available online: https://www.omg.org/spec/SBVR/1.4/PDF (accessed on 4 April 2020).
3. OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification v1.1, OMG Doc. No.: Formal/2011-01-01. 2011. Available online: https://manualzz.com/doc/47620449/meta-object-facility--mof--2.0-query-view--transformation (accessed on 4 April 2020).
4. OMG. Model Driven Architecture (MDA)—The Architecture of Choice for a Changing World. Available online: http://www.omg.org/mda/specs.htm (accessed on 4 April 2020).
5. Kapocius, K.; Skersys, T.; Butleris, R. The need for business vocabularies in BPM or ISD related activities: Survey based study. In Proceedings of the IEEE International Conference on Computer and Information Technology, Xi'an, China, 11–13 September 2014; pp. 622–629. [CrossRef]
6. Skersys, T.; Danenas, P.; Butleris, R. Extracting SBVR business vocabularies and business rules from UML use case diagrams. *J. Syst. Softw.* **2018**, *141*, 111–130. [CrossRef]
7. Skersys, T.; Danėnas, P.; Butleris, R. Model-based M2M transformations based on drag-and-drop actions: Approach and implementation. *J. Syst. Softw.* **2016**, *122*, 327–341. [CrossRef]
8. Danenas, P.; Skersys, T.; Butleris, R. Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams. In *Data and Knowledge Engineering*; Elsevier: Amsterdam, The Netherlands, 2020; Volume 128, pp. 1–19. [CrossRef]
9. Thakore, D.; Upadhyay, A.R. Development of Use Case Model from Software Requirement using in-between SBVR format at Analysis Phase. *Int. J. Adv. Comput. Theory Eng.* **2013**, *2*, 86–92.

10. Cabot, J.; Pau, R.; Raventós, R. From UML/OCL to SBVR specifications: A challenging transformation. *Inf. Syst.* **2010**, *35*, 417–440. [CrossRef]
11. Nemuraite, L.; Skersys, T.; Sukys, A.; Sinkevicius, E.; Ablonskis, L. VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models. In Proceedings of the International Conference on Information and Software Technologies, Kaunas, Lithuania, 21–23 April 2010; pp. 377–384.
12. Afreen, H.; Bajwa, I.S.; Bordbar, B. SBVR2UML: A Challenging Transformation. In Proceedings of the Frontiers of Information Technology, Islamabad, Pakistan, 19–21 December 2011; pp. 33–38. [CrossRef]
13. Awasthi, S.; Nayak, A. Transformation of SBVR business rules to UML class model. In Proceedings of the Conceptual Structures for STEM Research and Education 20th International Conference on Conceptual Structures, ICCS 2013, Mumbai, India, 10–12 January 2013; Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany, 2013; Volume 7735, pp. 277–288. [CrossRef]
14. Bajwa, I.S.; Lee, M.G. Transformation Rules for Translating Business Rules to OCL Constraints. In Proceedings of the Modelling—Foundation and Applications, 7th European Conference, ECMFA 2011, Birmingham, UK, 6–9 June 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 132–143. [CrossRef]
15. Bonais, M.; Rahayu, W.; Pardede, E. Integrating Information Systems Business Rules into a Design Model. In Proceedings of the 15th International Conference on Network-Based Information Systems, Melbourne, Australia, 26–28 September 2012; pp. 104–111. [CrossRef]
16. Malik, S.; Bajwa, I.S. Back to Origin: Transformation of Business Process Models to Business Rules. In *Business Process Management Workshops. BPM 2012*; Lecture Notes in Business Information Processing; Springer: Berlin/Heidelberg, Germany, 2012; pp. 611–622. [CrossRef]
17. Skersys, T.; Kapocius, K.; Butleris, R.; Danikauskas, T. Extracting business vocabularies from business process models: SBVR and BPMN standards-based approach. *Comput. Sci. Inf. Syst.* **2014**, *11*, 1515–1535. [CrossRef]
18. Tutkute, L.; Butleris, R.; Uzdanaviciute, V.; Sinkevicius, E.; Skersys, T.; Kapocius, K. Improving Quality of Business Models using Business Vocabulary based Synchronization Method. *Electron. Electrotech.* **2013**, *19*, 125–130. [CrossRef]
19. Iqbal, U.; Bajwa, I.S. Generating UML activity diagram from SBVR rules. In Proceedings of the Sixth International Conference on Innovative Computing Technology (INTECH), Dublin, Ireland, 24–26 August 2016; pp. 216–219. [CrossRef]
20. Essebaa, I.; Chantit, S. Tool support to automate transformations from SBVR to UML use case diagram. In Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018), Madeira, Portugal, 23–24 March 2018; pp. 525–532. [CrossRef]
21. Mohanan, M. Automated transformation of NL to OCL constraints via SBVR. *Int. J. Adv. Intell. Paradig.* **2020**, *16*, 229–240. [CrossRef]
22. Tantan, O.C.; Akoka, J. Automated transformation of business rules specification to business processes: From SBVR to BPMN. In Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE 2014), Vancouver, BC, Canada, 1–3 July 2014; pp. 684–687.
23. Steen, B.; Pires, L.F.; Iacob, M.E. Automatic generation of optimal business processes from business rules. In Proceedings of the IEEE International Enterprise Distributed Object Computing Workshop, Vitória, Brazil, 25–29 October 2010; pp. 117–126. [CrossRef]
24. Al-Hashemi, R.; Al-Jaafreh, M.; Al-Ramadin, T.; Al-Dmour, A. A Smart Algorithm for Use-Cases Production Based on Name Entity Recognition. *Comput. Inf. Sci.* **2015**, *8*, 51–55. [CrossRef]
25. Deeptimahanti, D.K.; Sanyal, R. Semi-automatic generation of UML models from natural language requirements. In Proceedings of the 4th India Conference on Software Engineering (ISEC '11), Thiruvananthapuram, India, 24–27 February 2011; ACM Press: New York, NY, USA, 2011; pp. 165–174. [CrossRef]
26. Umber, A.; Bajwa, I.S.; Asif Naeem, M. NL-based automated software requirements elicitation and specification. In Proceedings of the Communications in Computer and Information Science, Kochi, India, 22–24 July 2011; pp. 30–39. [CrossRef]
27. Njonko, P.B.F.; El Abed, W. From natural language business requirements to executable models via SBVR. In Proceedings of the 2012 International Conference on Systems and Informatics, ICSAI 2012, Yantai, China, 19–22 May 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 2453–2457. [CrossRef]
28. Wang, M. Requirements Modeling: From Natural Language to Conceptual Models Using Recursive Object Model (ROM) Analysis. Ph.D. Thesis, Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada, 2013.
29. Yue, T.; Briand, L.C.; Labiche, Y. A systematic review of transformation approaches between user requirements and analysis models. *Requir. Eng.* **2011**, *16*, 75–99. [CrossRef]
30. Ramzan, S.; Bajwa, I.S.; Ul Haq, I.; Naeem, M.A. A model transformation from NL to SBVR. In Proceedings of the 2014 9th International Conference on Digital Information Management (ICDIM), Bangkok, Thailand, 29 September–1 October; 2014; pp. 220–225. [CrossRef]
31. Guissé, A.; Lévy, F.; Nazarenko, A. From regulatory texts to BRMS: How to guide the acquisition of business rules? In *Rules on the Web: Research and Applications. Rule ML 2012*; Lecture Notes in Computer Science Series; Bikakis, A., Giurca, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7438, pp. 77–91. [CrossRef]
32. Landhäußer, M.; Körner, S.J.; Tichy, W.F. From requirements to UML models and back: How automatic processing of text can support requirements engineering. *Softw. Qual. J.* **2014**, *22*, 121–149. [CrossRef]
33. Dori, D. *Model-Based Systems Engineering with OPM and SysML*; Springer: New York, NY, USA, 2016. [CrossRef]
34. Von Halle, B. *Business Rules Applied: Building Better Systems Using the Business Rules Approach*; John Wiley and Sons: Hoboken, NJ, USA, 2002; ISBN 978-0-471-41293-9.

35. OMG. The Business Rules Manifesto. Available online: Businessrulesgroup.org/brmanifesto.htm (accessed on 4 April 2020).
36. Skersys, T.; Pavalkis, S.; Nemuraite, L. Implementing semantically rich business vocabularies in CASE tools. In Proceedings of the International Conference on Numerical Analysis and Applied Mathematics (ICNAAM-2014), Rhodes, Greece, 22–28 September 2014; AIP Publishing: New York, NY, USA, 2015; Volume 1648, pp. 1–4. [CrossRef]
37. No Magic, Inc. UML Profiling and DSL, v 18.1. Available online: www.nomagic.com/files/manuals/MagicDraw%20UMLProfiling&DSL%20UserGuide.pdf (accessed on 1 April 2020).
38. Danenas, P.; Skersys, T.; Butleris, R. Extending drag-and-drop actions-based model-to-model transformations with natural language processing. *Appl. Sci.* **2020**, *10*, 6835. [CrossRef]