Tool Support to Automate Transformations from SBVR to UML Use Case Diagram

Imane Essebaa and Salima Chantit
Computer Laboratory of Mohammedia (LIM), Faculty of Sciences and Techniques Mohammedia, Hassan II University of Casablanca, Mohammedia, Morocco

Keywords: Model Driven Architecture, Model Transformation, Semantic Business Vocabulary and Business Rules, UML Use Case Diagram.

Abstract: Model transformation becomes a very important technique in software engineering as it helps to guarantee traceability between models and develop software applications quickly, In this context, the purpose of this work is an approach that allows generating UML Use Case Diagram (UCD) from Semantic Business Vocabulary and Business Rules (SBVR) which is a standard introduced by OMG that can be used to capture software requirements in structured English. The paper gives a set of rules that map SBVR element into UCD elements. This approach is a part of our works that focus on automating Model Transformations to generate the application code following standards of Model Driven Architecture approach. Particulary we use SBVR standard and UCD to model the Computation Independent Model (CIM) which is the first level of abstraction. The paper describes also an implementation of this approach as an Eclipse plug-in that automates the transformation rules defined using QVT language. In order to well illustrate our approach, we apply it on RentalCarAgency system.

1 INTRODUCTION

System designing is an essential step in software development process. It acts as a bridge between the developer, the software designer and a simple user of the system. There exist different ways to ensure a good analysis of system requirements. Use Case Diagram (UCD) is an essential artefact in systems functional requirements analysis and specification. Indeed UCD provides a visual representation of possible interactions between system actors and their functionalities. In order to have a good design of the system that may be used to generate other models, it is necessary to have a well-described requirements, that is why it is recommended to use a structured language. In this paper, we focus on structured English language using SBVR.

Generating models from others using model transformation rules becomes an important technique to ensure traceability and link between system requirements and their design. Several approaches were proposed in this context, to transform UML diagrams, however few of them focused on generating UML diagrams, from structured requirements using SBVR (especially generating UCD).

Existing works concerning transformations between SBVR and UML diagrams focused on the mapping to generate Class Diagram and Activity Diagram. Furthermore few works have considered the transformation from UCD to SBVR.

In this paper we propose an approach to generate automatically UCD from SBVR. This approach is a part of our works that aim to automate transformations between different levels of MDA, where we have represented CIM level by SBVR and UCD. Then from CIM, we generate other MDA levels to obtain finally a generated application source code. Generating UCD from SBVR ensure traceability between requirements and other levels. In order to ensure automated transformations between levels, we present in this paper an approach that defines a set of transformation rules, developed with QVT language that we implement in an Eclipse plug-in.

This paper is organized as follows; section 2 gives an overview of concepts used to implement our approach. In section 3, we discuss the related works and in section 4, we describe the proposed approach and its implementation. In section 5, we present the application of the approach on a RentalCarAgency system example and in section 6, we discuss and evaluate our

Essebaa, I. and 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)*, pages 525-532
ISBN: 978-989-758-300-1

approach Finally, we finish by a conclusion and some perspectives.

2 OVERVIEW OF CONTEXT

2.1 Model Transformation

The execution of transformations between models ensures traceability of links between different models. This link is a guarantee of quality in the software development process. The transformation process proposed takes as input source models, applies transformation rules, and generate target models. This process conforms to Model Object Facility (MOF, 2009) which is an OMG standard that allows describing metamodels and their manipulation. Figure 1, illustrates the process of model transformation. A model transformation is like a function that takes as input parameters a set of models and returns in output another set of models. Those models, input and output, are structured by meta-models that specify the model transformation executed on models.

The MOF (Meta Object Facility) is normalized by OMG, it allows the definition of transformation rules and modeling languages, it also specifies the structure and syntax of meta-models. In this context, the OMG proposes a standard transformation language QVT(Query/view/transformation) to define transformation rules from models to models.

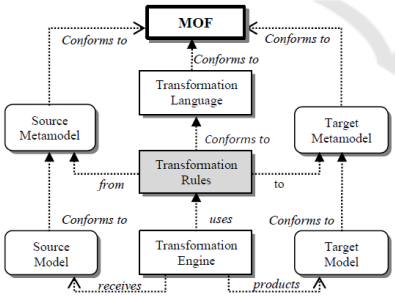

Figure 1: Relationship between the model transformation and meta-models.

2.2 Query / View / Transformation (QVT)

To implement our transformations, we have to use a transformation language. There exist many models of transformations language, but QVT is the unique proposal of the OMG. The QVT standard defines three model transformation languages: QVT Operational, QVT Relational and QVT core. We chose QVT Operational because it supports bidirectional transformations, both horizontal and vertical transformations, and ensures automatic traceability(MOF, 2009).

3 RELATED WORKS

Model Transformations are considered as an important approach. Several works were made in this context. However, most of them focused on transformations between different UML diagrams, and just a few ones were focused on the transformations from SBVR to UML diagrams. In this section, we present our analysis of these works:

T.Skersys and al. propose in their paper (Skersys et al., 2014) an approach that generate well-structured business vocabularies and rules using SBVR from the formalized requirements specifications expressed via Use Case diagrams. The paper propose an algorithm that defines UML2SBVR transformations which were implemented in MagicDraw tool. These transformations however are semi-automatic.

In their paper (Afreen et al., 2011) H. Afreen and al. proposed an approach to extract manually Object-Oriented informations from SBVR then to generate Class model. The paper proposes a conception of an eclipse plug-in called SBVR2UML that may generate Class model but it does not contain any description on how the link is made between these elements.We also note that this approach does not automate transformations between SBVR and UML Class Model. Indeed the Object-Oriented informations are generated manually. Moreover, the paper defines just a conception of an eclipse plug-in called SBVR2UML. We note also that this approach does not define how to generate other diagrams such as Sequence Diagram or Activity Diagram.

A.Raj and al. present in their paper (Raj et al., 2008), an approach to transform SBVR in more than just one UML diagrams: Class Diagram and Activity Diagram. We note that this approach describes for each generation, the algorithm to follow in order to get UML Diagrams. But this generation is still limited as it does not, for example, define how to get parameters of Class operations. Furthermore, in order to generate Activity Diagram, the approach takes into account just ”if-then” conditions but it does not explain how it will be transformed and how the diagram is generated.In this paper, authors consider the SBVR standard in the CIM level of MDA, while generated UML diagrams (Class Diagram, Activity Diagram and Sequence Diagram) represent the PIM level. We also note that the Activity Diagram contains the same information as the Sequence Diagram mo-

deled differently and their use at the same level does not add more information.

4 PROPOSED APPROACH

In this section, we present our proposed approach to generate UCD from SBVR that represent system requirements in a structured English.

4.1 CIM Level in Our Approach

The CIM level is the most abstract level in MDA approach. This level is considered as the most important because it models system requirements, and from this model, we can generate the other levels of MDA. Thus, other models are affected if any changes are made in CIM level.

In order to respect OMG recommendations to well model CIM level and to ensure traceability between models, we propose to represent it by an extended Use Case Diagram generated from OMG standard SBVR.

4.2 SBVR Meta-model

The OMG standard Business Vocabulary and Business Rules (SBVR), is a specification that was accepted by the OMG in 2005. The basic idea behind SBVR is to express requirements in a structured language (SBV, 2008). It allows the business analyst or a business person to express system requirements using structured English instead of using its own language.

Using SBVR to express business requirements affect a semantic to business rules and business vocabulary and give more details about the system, such as Business Management Rules that describes the interaction between system elements. A meta-model diagram of main fragments of SBVR is presented in figure 2.

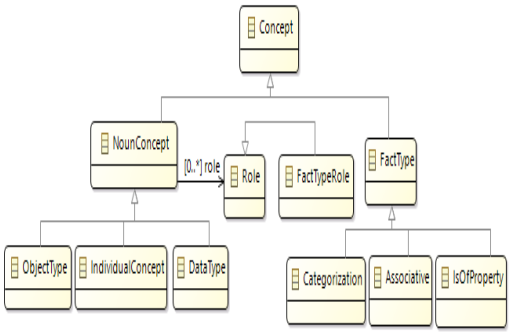

Figure 2: Main fragments of Semantic Business Vocabulary and Business rules Meta-model.

4.3 UML Use Case Diagram Meta-model

OMG (Object Management Group) defines a Use Case Diagram as the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system (OMG, 2011).

A Use Case Diagram is such a bridge that covers the gap between the simple user of the system and the software designer, indeed it gives a general view of the system, in our approach UCD is generated from SBVR in order to ensure traceability between system requirements and its design. A simplified version of the meta-model of the extended Use Case Diagram is represented in figure 3.

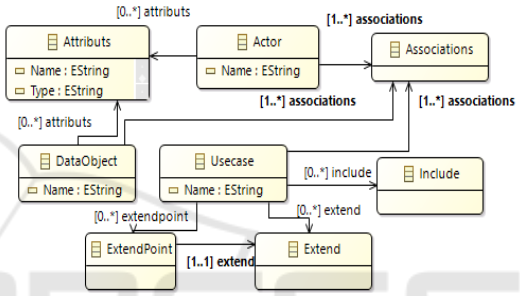

Figure 3: Main fragments of Use Case Diagram Meta-model.

4.4 Transformation Rules

In this section, we will define different transformation rules that allow the automatic generation of UCD from SBVR.

The transformation process proposed takes as input source models, applies transformation rules, and generate target models. This process conforms to MOF (Model Object Facility) which is an OMG standard that allows describing meta-models and their manipulation.

The Table 1 below describes transformation rules to automatically generate UML Use Case Diagram from SBVR.

4.4.1 Transformation Rules Description

This section gives more details on each transformation rule defined

1. IndividualConcept2System: The Individual Concept that has System as a general Concept is turned into a System in UCD.

Table 1: Transformation rules to generate UML UCD from SBVR.

N	Rule	Source Model	Target Model
1	IndividualConcept2System	Individual Concept	System
2	NounConcept_role2Actor	Noun Concept with Role as a general concept	Actor
3	NounConcept_objectType2DataObject	Noun Concept with Object Type as a general concept	Data Object
4	Verb_AssociativeFactType2UseCase	Verb of Association Fact Type	Use Case element
5	AssociativeFactType2Association	Association Fact Type	Association
6	IsObligatory_BusinessRule2Include	Is Obligatory Business Rule	Include
7	IsPossible_BusinessRule2Extend	Is Possible Business Rule	Extend
8	IsPermitted_BusinessRule2Generalization	Is Permitted Business Rule	Generalization
9	IsNecessary_BusinessRule2 Association_UseCase_DataObject	Is Necessary Business Rule	Association between Use Case element and Data Object

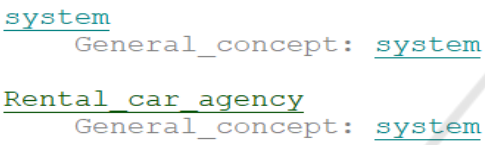

Figure 4: Examples of individual concept.

2. NounConcept_role2Actor: The Noun Concept in SBVR that has Role as a general concept is turned into an Actor in UCD.

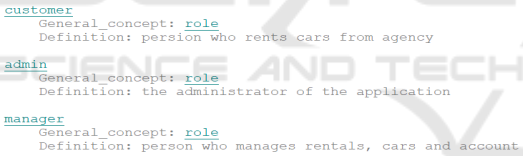

Figure 5: Examples of Role Noun Concept.

3. NounConcept_objectType2DataObject: The Noun Concept in SBVR that has Object Type as general concept is turned into a DataObject in UCD.

Figure 6: Example of Object Noun Concept.

4. Verb_AssociativeFactType2UseCase: The verb in Associative Fact Type in SBVR is turned into a Use Case element in UCD. The Name of the Use Case Element is a combination of the Verb and the Noun Concept that follows the Verb in the Associative Fact Type.

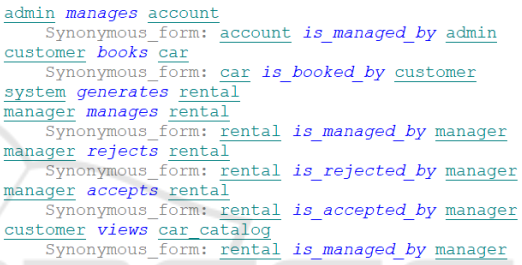

Figure 7: Example of Association Fact Type.

5. Associative_FactType2Association: The Associative Fact Type between two Noun Concepts is turned into an Association between the corresponding Use Case Element and an Actor, or between Use Case Element and DataObject if it exists.

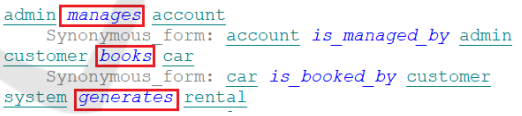

Figure 8: Examples of Association Fact Type verb.

6. Is_Obligatory_BusinessRule2Include: It Is_Obligatory business rules are turned into Include association between Use Case Elements generated from the two Associative Fact types used in the rule. The Include association is redirected from the Use Case Element generated from the second Associative Fact Type to the Use Case Element generated from the fist Associative Fact Type.

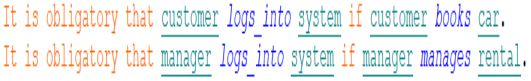

Figure 9: Examples of Is_Obligatory Business Rule.

7. IsPossible_BusinessRule2Extend: It Is Possible business rule is turned into Extend association if the rule is used with two Associative Fact Type. The Extend association is redirected from the Use Case Element generated from the first Associative Fact Type to the Use Case Element generated from the second Associative Fact Type.

Figure 10: Example of Is_Possible Business Rule.

8. IsPermitted_BusinessRule2Generalization: It Is Permitted business rule is turned into Generalization association if the rule is used with two Associative Fact Types. The Generalization association is redirected from the Use Case Element generated from the first Associative Fact Type to the Use Case Element generated from the second Associative Fact Type.

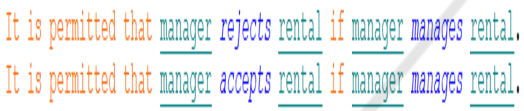

Figure 11: Example of Is_Permitted Business Rule.

9. IsNecessary_BusinessRule2Association_UseCase _DataObject: The association between a UseCase element and a DataObject element is generated from Is Necessary Business Rule between system generation fact type and a corresponding fact type that will be generated into the UseCase element.

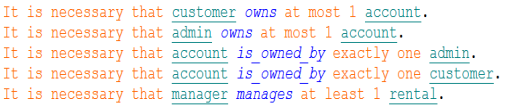

Figure 12: Example of is necessary Business Rule.

4.5 Implementation of Our Approach

To implement our approach, we need tools that allow creating input elements (Business Vocabulary and Business Rules). After analyzing and testing existing tools, we found that the unique tool that supports SBVR is an eclipse plug-in called Vetis which was implemented for an old version of Eclipse to support SBVR standard. Concerning UML editors we choose Papyrus because it supports all UML diagrams elements. According to this, we choose to implement our solution as an Eclipse plug-in to benefit from the existing tools and also to facilitate the use of our implemented solution by designers and developers.

4.5.1 Papyrus Modeling Tool

Papyrus is an Open Source UML tool based on Eclipse. It was developed by the Laboratory of Model Driven Engineering and Embedded Systems and can be used as a standalone tool or as an Eclipse Plug-in. Papyrus is designed to be easily extensible because it is based on UML profile. In our approach, we used Papyrus as an Eclipse plug-in that we extend to support DataObject elements in Use Case Diagram. As papyrus is based on EMF (Eclipse Modeling Framework) tool that allows creating UML diagrams, we start by extending the EMF plug-in to define DataObject element and its specifications to be used by Use Case Diagram. Then, we configure Papyrus in order to add DataObject element in Use Case Diagram, so that it can be accessible from the palette menu.

4.5.2 Vetis Plug-in

Vetis tool was implemented to support transformations from SBVR to UML&OCL. It was based on Eclipse platform 3.4.1 and it supports UML 2.0. SBVR 1.0 and ATL. In our approach we use Vetis plug-in to recognize SBVR elements and their structure and to benefit from the editor of SBVR standard.

4.5.3 Our Eclipse Plug-in

Plug-in is a software component used in computing that adds a specific features to an existing computer program. The plug-in we have developed, defines transformations rules based on QVT (Query View transformation). These rules are automatically executed from a Java program. The plug-in takes as input a Papyrus Project that contains Business Vocabulary and Business Rules expressed using SBVR standard. The ”Transforms” feature added in the menu, allow the execution of transformations defined automatically. As result of transformations, an out folder is created in the Papyrus Project that contains the output generated Use Case Diagram. The figure 13 below shows the ”transforms” menu and the architecture of Papyrus project.

5 CASE STUDY

In order to illustrate our approach and the transformation rules defined, this section shows how to apply the approach on a RentalCarAgency application. The example is described as follow: The application to realize must guarantee the following services:

- Visualization of available cars.

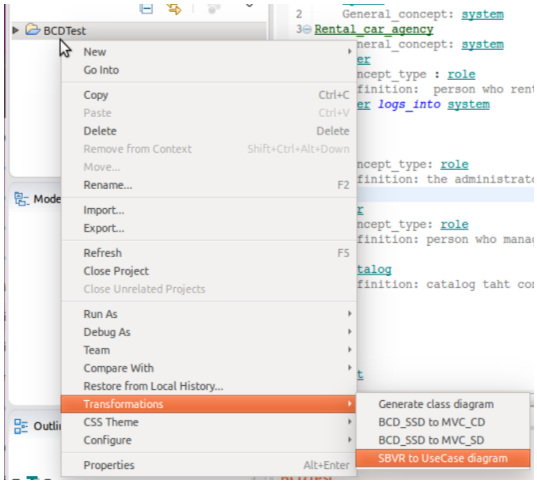

Figure 13: Papyrus Project Architecture and "Transforms" menu.

- Customers subscription.
- Booking of cars.
- Visualization of reservations.
- Management of reservations (accept/decline).
- Management of cars.
- Management of customer accounts.
- Management of accounts Managers.

The application has three users profiles that have different privileges :

- Customer: A person who can view the cars available in the agency, rates and promotions and may subscribe. Once registered, the visitor becomes a client of the Agency. A client must authenticate in the system to search for available cars and book a car by indicating the reservation date and time.
- Manager: A Manager must also authenticate to view all cars, add, edit or remove cars and view the bookings made by customers waiting for validation to decide to accept or refuse them.
- Administrator: Once authenticated into the system, the administrator has the privilege of modifying and deleting a customer account, as well as the management of managers account (add, change or delete).

We can define some management rules as below:

- A customer can rent several cars.
- A car can be rented by 0 or several customers.
- A manager can manage 1 or more cars.
- A car is managed by 1 or more managers.
- An administrator can manage 1 or several customer accounts.
- An administrator can manage 1 or more account managers.

After analysing the application requirements, the designer should standardize the requirements with Structured English using Business Vocabulary and Business Rules with the OMG standard SBVR, the first step is to define the vocabulary of the application and specify General Concept for each element. A part of Business vocabulary is defined in the figure 14. After applying the transformation rules already described in previous sections we got the following results presented in table 2.

```
system
    General_concept: system

Rental_car_agency
    General_concept: system

customer
    General_concept: role
    Definition: persion who rents cars from agency

admin
    General_concept: role
    Definition: the administrator of the application

manager
    General_concept: role
    Definition: person who manages rentals, cars and account

rental
    General_concept: object_type

account
```

Figure 14: Papyrus Project Architecture and "Transforms" menu.

```
car

car_catalog

admin manages account
    Synonymous_form: account is_managed_by admin

customer books car
    Synonymous_form: car is_booked_by customer
system generates rental

manager manages rental
    Synonymous_form: rental is_managed_by manager
manager rejects rental
    Synonymous_form: rental is_rejected_by manager
manager accepts rental
    Synonymous_form: rental is_accepted_by manager

customer views car_catalog
    Synonymous_form: rental is_managed_by manager

manager owns account
    Synonymous_form: account is_owned_by manager

customer logs_into system

manager logs_into system
```

Figure 15: Example of Business Vocabulary of RentalCarAgency application.

```
It is obligatory that customer logs_into system if customer books car.
It is obligatory that manager logs_into system if manager manages rental.
It is possible that customer books car if customer views car_catalog.
It is permitted that manager rejects rental if manager manages rental.
It is permitted that manager accepts rental if manager manages rental.
It is necessary that system generates rental if customer books car.
```

Figure 16: Example of Business Rules of RentalCarAgency application.

Table 2: Application of transformation rules on RentalCarAgency system.

N	Rule	Source Model	Target Model
1	IndividualConcept2System	RentalCarAgency	RentalCarAgency System
2	NounConcept_role2Actor	- Customer - Manager - Admin	- Customer - Manager - Admin
3	NounConcept_objectType2DataObject	- Rental	- Rental
4	Verb_AssociativeFactType2UseCase	- admin manages account - manager manages rental - customer books car - manager rejects rental	- manages account - manages rental - books car - rejects rental
5	AssociativeFactType2Association	- admin manages account - manager manages rental - customer books car	- Association between ”admin” and ”manages account” use case - Association between ”manager” and ”manages rental” - Association between ”customer” and ”books car”
6	IsObligatory_BusinessRule2Include	- It is obligatory that customer logs_into_system if customer books car - It is obligatory that manager logs_into system if manager manages rental	- Include Association from ”customer books car” to ”customer logs_into system” - Include Association from ”manager manages rental” to ”manager logs_into system”
7	IsPossible_BusinessRule2Extend	- It is possible that customer books car if customer views car_catalog	- Extend Association from ”customer books car” to ”customer views car_catalog”
8	IsPermitted_BusinessRule2Generalization	- It is permitted that manager accepts rental if manager manages rental - It is permitted that manager rejects rental if manager manages rental	- ”manages rental” is a generalization of ”rejects rental” - ”manages rental” is a generalization of ”accepts rental”
9	IsNecessary_BusinessRule2 Association_UseCase_DataObject	- It is necessary that system generates rental if customer books car	- Rental is a dataObject of ”customer books car” use case

Table 2 defines elements of Use Case Diagram which are automatically generated using the developed plug-in which is based on Transformation rules previously defined, the figure 16 below presents the generated Use Case Diagram of RentalCarAgency system:

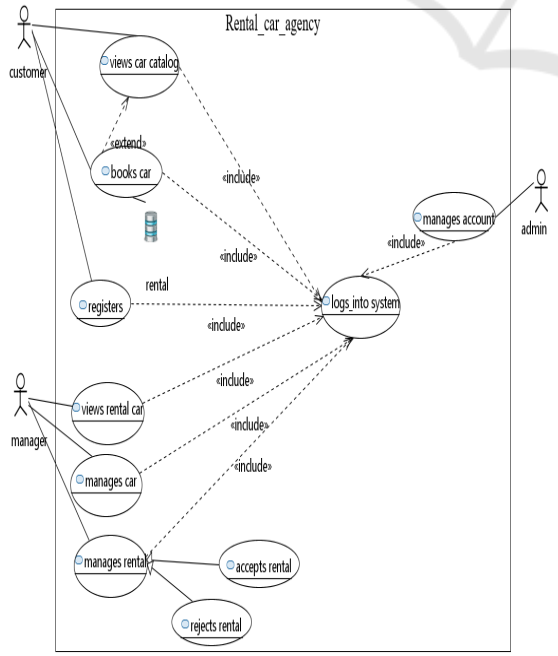

Figure 17: Generated Use Case Diagram of RentalCarAgency application.

6 ANALYSIS AND DISCUSSION

In order to evaluate our approach, we make a comparison between results of existing approaches and ours. The comparison is based on two axes; the first one is about transformation rules and their automation, and the second one is about the integration of SBVR to UML transformations on MDA approach.

At the moment of preparing this work, we could not find works that directly deal with our approach. We note that most of the previous approaches propose transformations from SBVR to different UML diagrams; Class Diagram, Activity Diagram, Sequence Diagram. The only work that discuss transformations between SBVR and UCD is the work of T.Skersys and al. where they propose a semi-automatic approach to generate SBVR from UML Use Case Diagram. We mention that this proposed approach of automating transformations from SBVR to UML Use Case Diagram is a part of our works that concerns transformations between different level of MDA. The current work aims to automate transformation in CIM level to ensure traceability with PIM and PSM levels that will be generated through successive vertical transformations. Concerning previous works already sited in related works section, few of them have mentioned that their approaches of transformations between SBVR and UML are included in MDA approach. In the next

Table 3, we summarize the analysis done on our approach and existing ones that transforms SBVR to UML diagrams. We study if proposed methods ensure transformation rules to generate UML from SBVR and if these rules are implemented automatically.

Table 3: Analyse and discussion of different approaches of SBVR to UML transformations.

Methods	SBVR to UML transformations	
	Rules	Automation
Skersys and al.	P	P
Arfeen and al.	P	N
Raj and al.	P	N
Nemuraite and al.	P	N
Essebaa and al. (current approach)	Y	Y

legend: P: Partial, Y: Yes, N: No

As we show in table 3, all presented approaches even if they partially define transformation rules, they didn't ensure their automation. Our approach provides transformation rules to generate UML Use Case Diagram from SBVR, and this paper describes the implementation of these rules in an eclipse plug-in to ensure their automation.

7 CONCLUSION

In this paper, we have presented our approach concerning SBVR to UCD transformations in CIM level. This approach aims to ensure traceability between models in other MDA levels, as PIM and PSM ones are generated from the CIM level.

Furthermore, the UCD generated from SBVR can be used to eliminate the gap between a simple user and design analysts. This work is a part of our previous (Essebaa and Chantit, 2017) and future works focused on automating transformations between models in order to generate code. In our future works, we aim to propose approaches that aim to automate all the process of MDA until code generation.

REFERENCES

(2008). Toward an automatic approach to get pim level from cim level using qvt rules. In *Semantics of Business Vocabulary and Business Rules (SBVR)*.

(2009). Omg, meta object facility (mof)2.0query/view/ transformation specification. In *http://www.omg.org/spec/QVT/1.0/PDF*.

(2011). In *OMG Unified Modeling LanguageTM (OMG UML), Superstructure*.

Afreen, H., Bajwa, I. S., and Bordbar, B. (2011). Sbvr2uml: A challenging transformation.

Essebaa, I. and Chantit, S. (2017). Tool support to automate transformations between cim and pim levels. In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: MDI4SE,*, pages 367–378. INSTICC, SciTePress.

Raj, A., Prabhakar, V., T., Hendryx, and Stan (2008). Transformation of sbvr business design to uml models. pages 29–38. ACM.

Skersys, T., Danenas, P., and Butleris, R. (2014). *Approach for Semi-automatic Extraction of Business Vocabularies and Rules from Use Case Diagrams*, pages 182–196. Springer International Publishing, Cham.