SBVR2UML: A Challenging Transformation

Hina Afreen
Department of Computer Science & IT
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
hina.afreen@hotmail.com

Imran Sarwar Bajwa, Behzad Bordbar
School of Computer Science & IT
University of Birmingham
Birmingham, UK
i.s.bajwa@cs.bham.ac.uk

Abstract— **UML is a de-facto standard used for generating the software models. UML support visualization of the software artifacts. To generate a UML diagram, a software engineer has to collect software requirements in a natural language (such as English) or a semi-formal language (such as SBVR), manually analyze the requirements and then manually generate the class diagrams in an available CASE tool. However, by automatically transforming SBVR Software requirements to UML can seriously share burden of a system analyst and can improve the quality and robustness of software modeling phase. The paper demonstrates the challenging aspect of model transformation from SBVR to UML. The presented approach takes input the software requirements specified in SBVR syntax, parses the input specification, extracts the UML ingredients such as classes, methods, attributes, associations, etc and finally generate the visual representation of the extracted information. The presented approach is fully automated. The presented approach is explained via an example.**

Keywords- ***Automated Software Modelling, UML, SBVR***

I. INTRODUCTION

The emergence of Object Oriented Analysis and Design in software engineering has led to the semi and fully automation of various phases of software modeling. Automated generation of Unified Modeling Language (UML) [1] diagrams from the natural language software specifications is a major break through in the field of automated software modeling [2], [3]. Examples of such work are LIDA [4], GOOAL [4], CM-Builder [6], Re-Builder [7], NL-OOML [8], UML-Generator [9], etc. However, most of the tools are at their preliminary stages and do not provide very high accuracy [6]. A primary reason attributed by the researchers is the ambiguous nature of English that makes machine processing intricate and complex.

Where the trends in software modeling and software programming are rapidly changing, major advancements have been taken place in the field of software requirements elicitation and specification [10]. One of the recent advancement is use of OMG's is new standard SBVR (Semantic of Business Vocabulary and Rules) [11] in specification of the software requirements. The use of SBVR in capturing the software requirements not only advances the process of software specification but also simplifies the process of analyzing and designing the software models. Moreover, SBVR is based on higher order logic and hence easy to machine process. We want to exploit these salient features of SBVR and aim to machine processing SBVR and transform SBVR to UML class models using the model transformation technology.

A real challenge in SBVR to UML class model transformation was to deal with the un-addressed issue in available approaches for SBVR to UML transformations. For example, the approaches presented by Raj [12] and Nemuraite [13] do not support the automated parsing of SBVR specifications. A user should have the pre-parsed SBVR specification to use these approaches. Another drawback of these approaches is that they perform partial mapping in SBVR and UML class model. These approaches also lack support for extracting class associations, aggregations and generalizations [12], [13]. We aim to address these issues and present a tool that can automatically parse SBVR and can perform complete mapping from SBVR to UML.

In this paper, the major contribution is threefold. Firstly, model transformation based a novel approach is presented to perform syntactic and semantic analysis of SBVR specification of software requirements to extract object oriented elements as classes, attributes, operations, associations, generalizations, etc. Secondly, we report the structure of the implemented tool SBVR2UML that is able to automatically generate UML class models from SBVR software requirements specifications. Thirdly, we have solved a case study with our tool and compared the results with other tools (used for automated OOA) for the sake of performance evaluation.

The remaining paper is structured into the following sections: Section 2 describes background and the related work. Section 3 illustrates the architecture and workflow of the presented tool, SBVR2UML. Section 4 presents a solved case study from the domain of library information systems. Finally, the paper is concluded to discuss the future work.

II. BACKGROUND

In this section, a brief introduction to the basic concepts of the MDA, UML and SBVR is provided. The elaborated concepts are used in the rest of the paper.

A. Model Driven Architecture

The presented approach to model transform input SBVR

978-0-7695-4625-4/11 $26.00 © 2011 IEEE
DOI 10.1109/FIT.2011.14

specification to UML specification is based on the Model Driven Architecture (MDA) [14]. MDA is a software design approach typically involved in model based software system development. A key concept in the MDA is the notion of metamodels [15]. Model transformation can be defined in the MDA by mapping elements of the source metamodel to the target metamodel. On the basis of these mappings a set of rules are defined, typically called *Transformation Rules*. By employing these transformation rules, every model, which is an instance of the source metamodel, can be automatically transformed to an instance of the destination metamodel [16].

B. UMLClass Models

The Unified Modeling Language (UML) [1] supports a set of diagrams to visually represent various software artifacts. One of the most common and widely used diagrams is the UML class diagram. The class diagrams are typically used to represent the structural information of a software model. The key element in a class model is *class*. A *class* can have sub-elements such as *attributes*, *methods*, and *associations*.

C. SBVR Specification

A typical SBVR representation is based on SBVR business vocabulary and SBVR business rules [11]. In SBVR, all the specific terms and definitions of concepts used by an organization or community in course of business are treated as vocabulary. Common examples of SBVR vocabulary are object types, individual concepts, characteristics, fact types, etc. In SBVR, a formal representation under a business jurisdiction [11] is called a SBVR rule. SBVR rules can be of two types: Strucutral Rule (used to expresses structure or operation of a particular business entity) and Behavioural Rule (used to express the conduct of a business entity).

III. MODEL TRANFORATION FROM SBVR TO UML

This section explains the used approach to automatically map SBVR representation i.e. SBVR business rules to a UML class model. To map SBVR to a UML class model, we have to extract SBVR vocabulary from given SBVR rules and then map the SBVR vocabulary to basic elements of a UML class model (such as classes, associations, etc.) and finally generate a graphical representation of class model. The used approach works in following 5 phases:

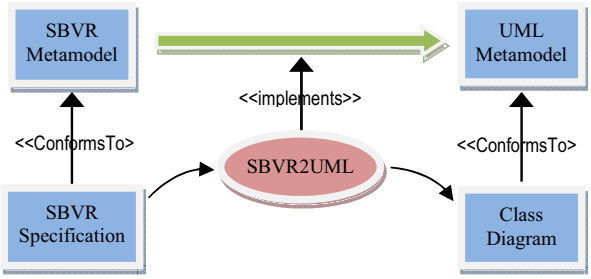

Figure 1. SBVR to UML Transformation Framework

All these five distinct phases are explained in detail in the remaining part of the section.

A. Pre-Processing SBVR Specficiation

The input SBVR specification is pre-processed to find out the candidates for possible SBVR vocabulary. We have user the Stanford parts-of- speech (POS [17] tagger v3.0 to tokenize and POS tag the input SBVR specification.

Then, the Stanford parser was used to syntactically analyze the POS tagged SBVR specifications. The Stanford parser generates the parse tree and typed dependencies. We have used the extracted information by the Stanford parser for the detailed semantic analysis. The semantic analysis of SBVR specification is described in next section.

B. Semantic Analysis of SBVR Specifications

To identify the SBVR vocabulary, semantic role labeling is performed. Semantic role labeling or thematic role labeling is a common approach used in shallow semantic parsing. The SBVR elements such as noun concept, individual concept, object type, verb concepts, etc are identified from the SBVR input. All these elements are shown in the extract of SBVR (meaning) metamodel shown in figure 2:

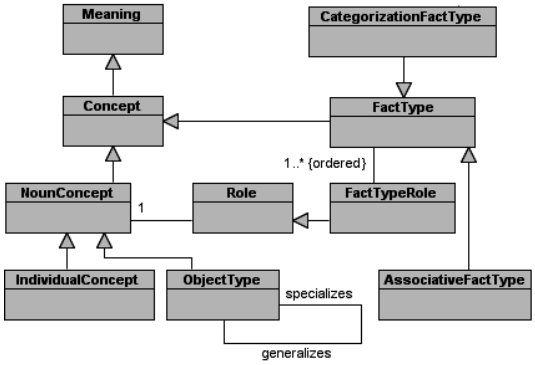

Figure 2. A constrcut of SBVR (meaning) metamodel

Following set of mappings are used to extract the various SBVR elements:

1) Extracting Object Types: In SBVR, the common nouns (actors, co-actors, thematic objects, or beneficiaries) are mapped as the object types [11] e.g. belt, user, cup, etc.

2) Extracting Individual Concepts: The proper nouns (actors, co-actors, thematic objects, or beneficiaries) are mapped to the individual concepts [11].

3) Extracting Fact Types: In SBVR specification, the auxiliary and action verbs are represented as verb concepts. To constructing a fact types, the combination of an object type/individual concept + verb forms a unary fact type e.g. "vision system *senses*". Similarly, the combination of an object type/individual concept + verb + object type forms a binary fact type e.g. belt *conveys* part is a binary fact type.

4) Extracting Characteristics: In SBVR, the characteristic [11] (section:11.1.2.2) or attributes are typically represented

using *is-property-of* fact type e.g. "name *is-property-of* customer". Moreover, the use of possessed nouns (i.e. pre-fixed by *'s* or post-fixed by *of*) e.g. student's age or age of student is also characteristic.

5) Extracting Quantifications: In SBVR specification, all indefinite articles (*a* and *an*), plural nouns (prefixed with s) and cardinal numbers (2 or two) are mapped to the quantifications.

6) Extracting Associative Fact Types: The associative fact types [11] (section 11.1.5.1) are identified by associative or pragmatic relations in English text. In English, the binary fact types are typical examples of associative fact types e.g. "The belt conveys the parts". In this example, there is a binary association in belt and parts concepts. This association is one-to-many as '*parts'* concept is plural. In conceptual modeling of SBVR, associative fact types are mapped to associations.

7) Extracting Partitive Fact Type: The partitive fact types [11] (section 11.1.5.1) are identified by extracting structures such as "*is-part-of*", "*included-in*" or "*belong-to*" e.g. "The user puts two-kinds-of parts, dish and cup". Here 'parts' is generalized form of '*dish'* and '*cup'*. In conceptual modeling of SBVR, categorization fact types are mapped to aggregations.

8) Extracting Categorization Fact Types: The categorization fact types [11] (section 11.1.5.2) are identified by extracting structures such as "*is-category-of*" or "*is-type-of*", "*is-kind-of*" e.g. "The user puts two-kinds-of parts, dish and cup". Here 'parts' is generalized form of '*dish'* and '*cup'*. In conceptual modeling of SBVR, categorization fact types are mapped to generalizations.

C. Mapping SBVR to Class Diagram

In this phase, overview of the transformations rules is presented. These transformations rules are incorporated to transform SBVR to UML class models. In Table I. the informal correspondence in elements of SBVR and UML class model is shown. Finally the SBVR rule is further processed to extract the OO information.

TABLE I. INFORMAL MAPPING BETWEEN SBVR AND UML METAMODEL ELEMENTS

SBVR metamodel element	UML metamodel element
Object Type	Class
Individual Concept	Object
Characteristic	Class Attribute
Verb Concept	Class Method
Fact Type	Association
Partitive Fact ct Type	Generalization
Categorization Fact Type	Aggregation
Quantifications	Cardinalities

The mapping of the each SBVR vocabulary item to the respective UML class element is described below:

1) Mapping Object Type to *Class:* In a SBVR rule, all the object types are mapped to classes in a UML class model.

2) Mapping Individual Concepts to Objects: We are mapping all individual concepts in a SBVR rule to the objects in a UML class model.

3) Mapping Characteristics to Attributes: All the SBVR characteristics or unary fact types (without action verbs) associated to an object type are mapped to the attributes in a UML class diagram.

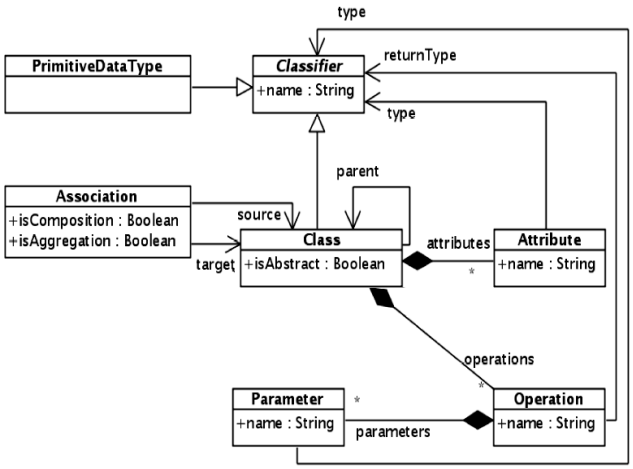

Figure 3. A constrcut of UML (Class Diagram) metamodel

4) Mapping Verb Concepts to Methods: All the SBVR verb concepts (action verbs) associated to a noun concept are mapped to methods for a class e.

5) Mapping Associative Fact Types to Associations: A unary fact type with action verb is mapped to a unary relationship and all associative fact types are mapped to binary relationships. The use of quantifications with the respective noun concept is employed to identify *multiplicity* e.g. User(s) and database will have one to many association in Figure 4. The associated verb concept is used as caption of association.

6) Mapping Partitive Fact Types to Generalization: The partitive fact types are specified as generalizations. The subject-part of the fact type is considered the main class in generalization and object-part of the fact types is considered as the sub class.

7) Mapping Categorization fact Types to Aggregations: The categorization fact types are mapped to aggregations. The subject-part of the fact type is considered the main class in aggregation and object-part of the fact types is considered as the sub class.

D. Drawing UML Class Model

This phase draws a UML class model by combining class diagram symbols with respect to the information extracted of the previous phase. To draw a class diagram, three rectangles were combined: one for class name, one for the class attributes and one for the class methods. All the classes were intelligently grouped. The classes having associations and other relationships were drawn close to each other. Finally, the

associations were also captioned with the titles and cardinalities. The graphics functions in Java such as (drawrect(), drawline(), etc) are used to draw the class diagram and other symbols.

IV. A CASE STUDY

An example of the software requirements for KeePass Password Safe [18] is presented here. KeePass Password Safe is an OSI Certified Open Source Software available under the terms of the GNU license Ver. 2. Following is the problem statement of the case study.

> *KeePass consists of a database which contains data for one or more users. Each user's data are divided into groups and subgroups so that they are organized in a form that serves right the user. Every user has a unique Master Key which can be simple or composite and its combination opens uniquely the database. If lost there is no recovery. Groups and subgroups contain entries with usernames, passwords URLs etc that can be sent or copied to websites, application and accounts. There is also the ability for a onetime key creation to be used once in a transaction without the risk of reused by others for any reason.*

We generated the SBVR rule representation of the above input of KeePass Password specification. We have used NL2SBVR tool [20] for generating SBVR representation. The SBVR specification after extracting SBVR vocabulary is as follows:

> KeePass *consists* of a database which *contains* data for one or more users. It is necessary that each user's data are *divided* into groups and subgroups so that they are *organized* in a form that *serves* right the user. It is obligatory that every user *has* a unique Master Key which *can* be simple or composite and its combination *opens* uniquely the database. If lost there *is* no recovery. It is necessary that Groups and subgroups *contain* entries with *usernames*, *passwords*, *URLs* etc that can be *sent* or *copied* to websites, application and accounts. It is possibility that there *is* also the ability for a onetime key creation to be used once in a transaction without the risk of *reused* by others for any reason.

Afterwards, the extracted SBVR vocabulary was mapped to the UML class elements. Following information was extracted in OO analysis phase:

TABLE II. OBJECT ORIENTED ANALYSIS RESULTS

Example	Count	Details
Classes	13	User, Database, Data, MasterKey, Simple, Composite, Groups, SubGroups, Applications, Webpage, Accounts, OneTimeKey, Form
Attributes	03	user name, password, url
Methods	02	send(), copy()
Associations	06	database for user, form serves user, user has Masterkey, its open database, Group and Subgroup send or copy entries to websites, applications, and accounts, OneTimeKey used in a transaction
Generalizations	01	database contains data
Aggregations	02	data is divided into groups and subgroups, MasterKey can be simple or composite
Instances	01	KeePass

There are two pieces of information such as "If lost there is no recovery", "without the risk" could not be processed as this information should be translated to formal constraints such as OCL. This could be achieved by using our other available tool NL2OCL*via*SBVR [19] available for free download. Another issue was OneTimeKey should be sub-type of MasterKey but this information has not been shown in generated UML class model. Moreover, one construct "form serves the user" was totally missed by the tool. By considering all these errors and omissions the calculated recall was and calculated precision was

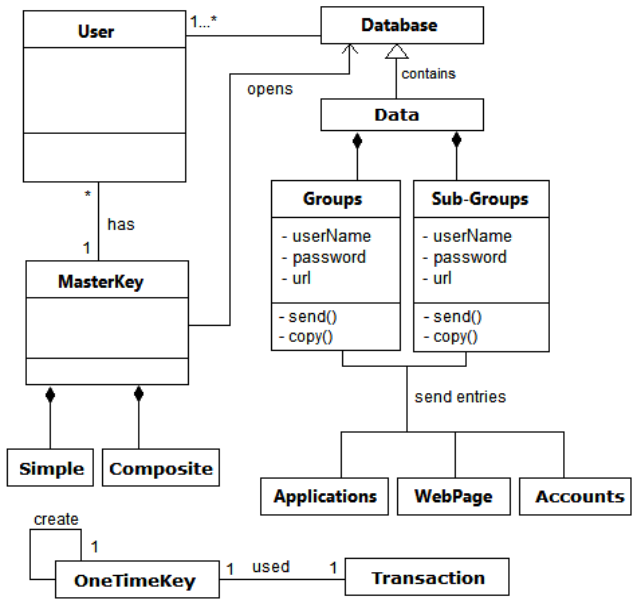

Figure 4. A class model of case study generated by SBVR2UML

Figure 4 shows a screen shot of a class model generated from extracted object oriented information of the input case study "KeyPass Password Safe".

V. EVALUATION

To evaluate the performance of SBVR2UML tool, a set of case studies including the case study discussed in section VI were solved. The results of these case studies were used to calculate recall and precision values as shown in table III.

Average recall for SBVR requirement specifications to UML class diagram transformation is calculated 83.82% while average precision is calculated 91.017%. These results are very encouraging for the future enhancements.

TABLE III. SBVR2UML EVALUATION RESULTS

Example	N_{sample}	$N_{correct}$	$N_{incorrect}$	$N_{missing}$	***Rec%***	***Prec%***
Example 1	33	28	2	3	84.84	93.33
Example 2	40	37	2	1	92.50	94.87
Example 3	54	46	3	5	85.18	93.86
Example 4	38	31	3	4	81.57	91.17
Example 5	24	18	4	2	75.00	81.82
				Average	83.82	91.01

We have also compared the results of SBVR2UML with other available tools that can perform automated analysis of the NL requirement specifications. Recall value was not available for some of the tools. We have used the available recall and precision values of the tools for comparison as shown in table IV:

TABLE IV. A COMPARISON OF PERFORMANCE EVALUATION – SBVR2UML VS OTHER TOOLS

NL Tools for Class Modelling	***Recall***	***Precision***
CM-Builder (Harmain, 2003)	73.00%	66.00%
GOOAL (Perez-Gonzalez, 2002)	-	78.00%
NL-OOML (Anandha, 2006)	-	82.00%
LIDA (Overmyer, 2001)	71.32%	63.17%
UML-Generator (Bajwa, 2009)	-	83.66%
SBVR2UML	83.82%	91.01%

Here, we can note that the accuracy of other NL tools used for information extraction and object oriented analysis is well below than SBVR2UML. Moreover, the various tools' functionalities (if available, is automated or user involved) are also compared with SBVR2UML as shown in Table IV:

TABLE V. COMPARISON OF SBVR2UML WITH OTHER TOOLS

Support	**CM-Builder**	**LIDA**	**GOOAL**	**NL-OOML**	**UML 2SBVR**
Classes	Yes	User	Yes	Yes	Yes
Attributes	Yes	User	Yes	Yes	Yes
Methods	No	User	Yes	Yes	Yes
Associations	Yes	User	Semi-NL	No	Yes
Multiplicity	Yes	User	No	No	Yes
Aggregation	No	No	No	No	Yes
Generalization	No	No	No	No	Yes
Instances	No	No	No	No	Yes

Table IV shows that besides SBVR2UML, there are very few tools those can extract information such as multiplicity, aggregations, generalizations, and instances from NL requirement. Thus, the results of this initial performance evaluation are very encouraging and support both the approach adopted in this paper and the potential of this technology in general.

VI. CONCLUSIONS

In this paper, we addressed couple of challenging issues in model transformation of SBVR specifications to UML class models. First issue of SBVR parsing was addressed by using typical NLP approaches. Second issue of transformation of SBVR metamodel elements to UML class diagram metamodel elements was addressed by using model transformation technology. Sitra library was used for the purpose of the model transformation. Moreover, the automated object oriented analysis of SBVR specifications of software requirements was performed. The results show that the presented approach is a better approach as compared to the other available approaches. Moreover, the SBVR2UML tool provides a higher accuracy as compared to other available NL-based tools. Besides better accuracy, SBVR has also enabled to extract OO information such as association multiplicity, aggregations, generalizations, and instances as other NL-based tools can't process and extract this information.

In future, we aim to integrate our SBVR to OCL transformation plugin [21] with SBVR to UML plugin, so that a user may generate both UML and OCL with the same ease and simplicity.

REFERENCES

[1] Bryant B.R, Lee, B.S., et al. 2008. From Natural Language Requirements to Executable Models of Software Components. In Workshop on S. E. for Embedded Systems:51-58.

[2] Ilieva, M.G., Ormandjieva, O. 2005. Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation. in proc. of Natural Language Processing and Information Systems LNCS- 3513/2005:427-434mda/

[3] OMG. (2007). Unified Modelling Language (UML) Standard version 2.1.2. Object Management Group, Available at: http://www.omg.org/

[4] Overmyer, S.V., Rambow, O. 2001. Conceptual Modeling through Linguistics Analysis Using LIDA. 23rd International Conference on Software engineering, July 2001

[5] Perez-Gonzalez, H. G., Kalita, J.K. 2002. GOOAL: A Graphic Object Oriented Analysis Laboratory. 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '02), NY, USA: 38-39.

[6] Harmain, H. M., Gaizauskas R. 2003. CM-Builder: A Natural Language-Based CASE Tool for Object- Oriented Analysis. Automated Software Engineering. 10(2):157-181

[7] Oliveira, A., Seco N. and Gomes P. 2006. A CBR Approach to Text to Class Diagram Translation. TCBR Workshop at the 8th European Conference on Case-Based Reasoning, Turkey, September 2006.

[8] Anandha G.S., Uma G.V. 2006. Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification. PRICAI 2006: Trends in Artificial Intelligence, LNCS 4099/2006: 1155-1159

[9] Bajwa I.S., Samad A., Mumtaz S. 2009. Object Oriented Software modeling Using NLP based Knowledge Extraction. European Journal of Scientific Research, 35(01):22-33

[10] Ashfa Umber, Tayyiba Bashir, M Shahid Naweed, Imran Sarwar Bajwa (2011) Requirements Elicitation Methods In: 2nd IEEE International MIMT Conference (MIMT 2011) 541-545 Singapore

[11] OMG. 2008. Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0. Object Management Group, Available: http://www.omg.org/spec/SBVR/1.0/

[12] Raj, A. , Parbharkar, T.V. Hendryx, S. 2008. Transformation of SBVR Business Design to UML Models. India Software Engineering Conference (ISEC 2008), ACM, Hyderabad, India

[13] Nemuraite, L., Ceopniene, l., et al. 2008. Representation of Business Rules in UML&OCL models for developing Information Systems. First IFIP Working Conference, PoEM 2008, Stockhom, Sweden.

[14] Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture{Practice and Promise. The Addison-Wesley Object Technology Se-ries. Addison-Wesley (2003)

[15] OMG: MOF Core v. 2.0 Document Id: formal/06-01-01. http://www.omg.org.

[16] Akehurst, D.H., Bordbar, B., Evans, M.J., Howells, W.G.J., McDonald-Maier, K.D.: SiTra: Simple transformations in java. In Nierstrasz, O., Whittle, J., Harel, D., Reggio, G., eds.: Model Driven Engineering Languages and Systems, 9th In-ternational Conference, MoDELS 2006. Volume 4199 of LNCS., Genova, Italy,Springer (2006) 351{364

[17] Toutanova. K., Manning, C.D. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: 63-70.

[18] Elia Kouzari, 2008. Software Requirements Specifications for KeePass Password Safe. Software Engineering, Aristotle University Thessaloniki, Available at: http://keepass.info/extensions/base/docs/Software RequirementsSpecification-KeePass-1.10.pdf

[19] Imran S. Bajwa, Behzad Bordbar, Mark G. Lee [2010] "OCL Constraints Generation from Natural Language Specification", in EDOC 2010 - IEEE International EDOC Conference 2010, Vitoria, Brazil, Oct 2010, pp:204-213

[20] Imran S. Bajwa, Mark G. Lee, Behzad Bordbar [2011] SBVR Business Rules Generation from Natural Language Specification. in proceedings of AAAI 2011 Spring Symposium -AI4BA, San Francisco, USA, Mar 2011, pp:2-8

[21] Imran S. Bajwa, Mark G. Lee [2011] "Transformation Rules for Translating Business Rules to OCL Constraints", in ECMDA-FA 2011-7th European Conference on Modelling Foundations and Applications, Birmingham, UK, Jun 2011, pp:132-143