An Intelligent Approach for CRC Models Based Agile Software Requirement Engineering Using SBVR

Hina Afreen(✉) and Umer Farooq

The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan
hinaafreen02@gmail.com, umer.bwnl@gmail.com

Abstract. In requirement engineering (RE) for agile software development, the Class-Responsibility-Collaborator (CRC) models are used as important brainstorming tool. However, manual generation of such CRC models by analyzing the requirements is a difficult and time-consuming task due to ambiguity and informal nature of natural languages-based software requirements. This paper introduces an improved requirement engineering technique based on CRC models that can help in specifying and analyzing software requirements in a better and faster way and curtailing difficulties associated with the traditional RE analysis technique. The proposed technique employs Semantics of Business Vocabulary and Rules (SBVR) to capture and specify software requirements in a controlled natural language. The SBVR representation is processed to extract object-oriented information and map the extracted information to CRC models in both textual and visual form. The proposed approach is implemented as an Eclipse plugin prototype SBVR2CRC as a proof of concept and the results of the experiments validate the effectiveness of the presented approach. Results show that such automated approach not only saves certain time and effort but also assists in generation of better CRC models and simplifies the CRC models based agile software development.

Keywords: Agile modeling · Requirement engineering · CRC models · SBVR

1 Introduction

In recent years, adoption of agile methodologies for software development has resulted in not only a rapid development of high-quality software but also delivering the value to customer [1]. Agile methods are totally different from standard SE process and based on face-to-face communication and iterative development [2]. Agile requirement engineering is a flexible, quicker and modern way of requirements elicitation that helps to make rapid delivery of software [3]. There are different reasons of agile requirements engineering success in software development process. One of the key reasons is face to face communication that makes requirements clearer. Moreover, agile requirement engineering allows customer to freely communicate with team throughout software development process while traditional requirement engineering only allow early stage communication of customer with development team that creates a gigantic gape in requirement understanding.

I. S. Bajwa et al. (Eds.): INTAP 2019, CCIS 1198, pp. 372–384, 2020.
https://doi.org/10.1007/978-981-15-5232-8_32

The requirement expert people find requirements elicitation techniques like brain-storming are lengthy and laborious. The Class-Responsibility-Collaborator (CRC) [4, 5] models are typically used brain-storming tool for Agile based requirements elicitation and specification [6]. These days, the agile teams often use CRC models in design of a distinct challenging user story. However, analysis of software requirements and CRC cards based modeling can be a challenging task due to ambiguity and informal natural of natural languages typically used for capturing software requirements. Semantics of Business Vocabulary and Rules (SBVR) [7] has emerged into a widely adopted standard in the recent years, originally introduced by Object Management Group (OMG). SBVR can be useful in capturing and specifying software requirements in a controlled natural language such as Structured English with underlying semantics. In this paper, we present an intelligent and novel approach to generate Class-Responsibility-Collaborator (CRC) models by automatic extraction of object oriented information from SBVR based software requirements and then transform the extracted information into CRC models. CRC cards exhibit an effective way to represent requirement specifications that plays very vital role in software development process.

The rest of this paper is structured into following sections: Sect. 2 provides related work. Section 3 depicts the framework of presented prototype tool, SBVR2CRC. Section 4 explains an experiment and results. Section 5 describes the conclusion and highlights the future work.

2 Related Work

The use of artificial intelligence techniques and methodologies to automate and improve the traditional practices and processes of software engineering is not a new idea. One of the applications of such work is automated generation of class models from natural language text such as Overmyer [8], Harmain [9], Gomes [10], Bajwa [11], Deeptimahanti [12], Sagar [13], Gulia [14], and Arora [15]. Similarly, object oriented analysis of SBVR to generate UML class models has also been partially achieved previously such as SBVR to UML models (Raj [16]), SBVR to class models (Bajwa [17]), SBVR to UML models (Njonko [18]) and extraction of SBVR from class models (Skersys [20]). However, the object-oriented analysis discussed in the mentioned researches is partial and addresses subset of UML models. A few of these above-mentioned works are discussed in the following text to highlight the research gap.

Raj et al. [16] presented an approach to transform SBVR business design into three UML diagrams i.e. activity diagram, sequence diagram and class diagram. Raj's work to generate UML class diagram is at very early stage and reuse some existing transformation rules that are described in official release of SBVR specifications.

Moreover, this approach does not address various object-oriented elements such as methods, interfaces, data types, super lass, sub class etc.

Another automated approach is presented by Bajwa et al. [17] for object-oriented analysis of SBVR based software requirements to generate UML class models. The used approach systematically analyzes SBVR text and then formally transforms object-oriented information into UML class diagrams. However, similar to other approaches,

this approach performs partial object-oriented analysis and does not cover extraction of super-class, sub class, categorization, data types, etc.

Another contribution is made by Nemuraite et al. [21] by presenting a tool VeTIS that provides editing facility of SBVR vocabularies and rules and can also partially transforms them to UML and OCL models. Nemuraite's work specifies requirements by using use cases and performs modeling of business process with the help of activities. This tool is integrated with Magic Draw UML case tool.

Another theoretical approach was presented by Awasthi et al. [19] to generate UML class diagram from SBVR based business design. An algorithm was used in presented research to partially extract object-oriented information and VeTIS tool is used to evaluate its results.

One more automated transformation approach was proposed by Bonais et al. [22] to produce the structural design models by defining a subset of SBVR specifications according to the elements of UML class diagram. A formal meta-model of identified SBVR subset is defined and used to generate UML class diagram for the defined subset using a formal transformation. However, this work also generates partial UML class models as the support to extract super-class, sub-class, visibility, multiplicity, interfaces, etc. is not provided.

3 The SBVR2CRC

This section elaborates the presented approach to automatically generate CRC cards from software requirements specification that are described in English text. SBVR2CRC approach consists of two main steps. In first step, SBVR specifications are generated by processing software specifications that are taken as input text. Generated SBVR specifications consist of business vocabulary and business rules. We have only required business vocabulary to generate CRC model. At second step, CRC model is generated using object-oriented information (classes, inheritance, generalization, aggregation etc.) that is extracted from SBVR business vocabulary. Here, we also define transformation rules that transform OO information into CRC model. Remaining part of this section explains all of these steps in detail.

3.1 Generate SBVR Model

To generate SBVR model, SBVR2CRC tool takes English software requirement as input and process these requirements to generate elements of a SBVR model and these elements are further processed to extract SBVR vocabulary and SBVR rules to complete a SBVR model as depicted in Fig. 1.

The processing of software requirements specifications text is divided into three steps like lexical, syntax and semantic analysis to generate SBVR based specifications. Lexical analysis phase starts with the lexical processing of text file that contains software requirements specifications. Lexical processing is process of producing stream of characters and it is referred to as scanning.

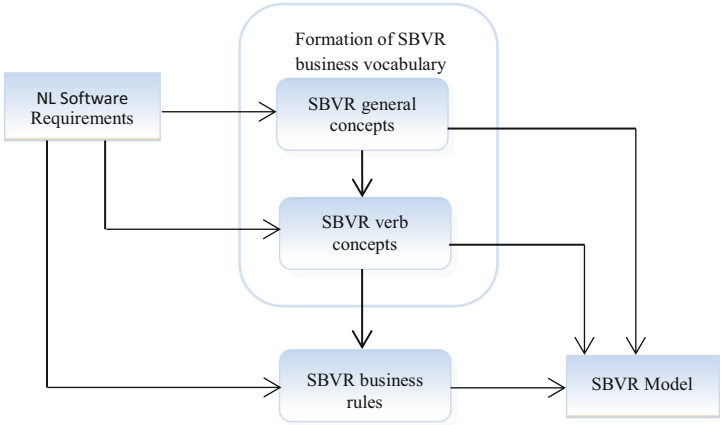

Fig. 1. SBVR model generation from NL text

The lexical analysis phase is further categorized four phases that are sentence splitting, tokenization, POS tagging and lemmatization. After performing lexical analysis, now we want to define the sentence structure of the input text. To accomplish this, we have used Stanford parser [23] that generates a parse tree. After syntactic analysis of text, generated parse tree is further used to perform semantic analysis. Shallow semantic parsing performed in our approach is similar to the Sematic Role Labeling (SRL) [24]. In the used approach, semantic roles are automatically assigned to each token in a sentence with the help of Stanford NER (Named Entity Recognition) also called CRFClassifier [25]. Since, Stanford CRFClassifier provides partial Semantic Role Labelling (SRL); a set of rules are user to identify complete set of semantic roles.

3.2 Transformation of SBVR Model to CRC Model

A complete description of the process starts from taking English software requirements to finally CRC model generation is shown in Fig. 2. The last step is the validation of overall transformation result i.e. CRC model. Validation of CRC model is an optional and during validation stage, one can go back and modify the concepts formed in the preceding stages if necessary.

A CRC card has two sides: front side and back side. The front side of card represents class name, responsibilities and collaborators. The class name represents the name of class for which CRC card is drawn and shown on top of the card. The responsibilities depicts knowing and doing of a class and are shown on left column of a CRC card as depicts in Fig. 3. Each class has some responsibility to fulfil. If a class has sufficient information to perform a responsibility, then it does not need collaborator class. The collaborator classes are needed when a class does not sufficient information to fulfill a job. The collaborators of a class are mentioned on right-hand column of a class [26].

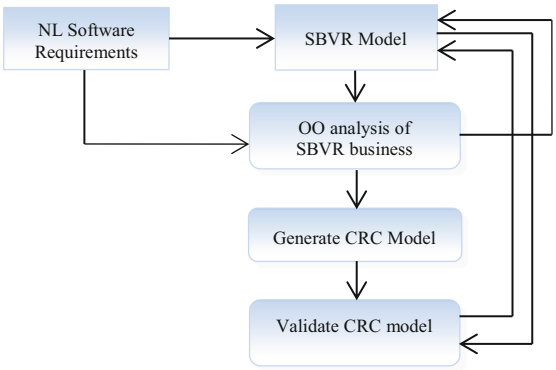

Fig. 2. Transformation of SBVR to CRC model

The backside of CRC card represents attributes and associations. The associations are represented as is - a relationship, aggregation, generalization, etc. It is important to note that consider front side of CRC card as the public information and the back side as encapsulated implementation details.

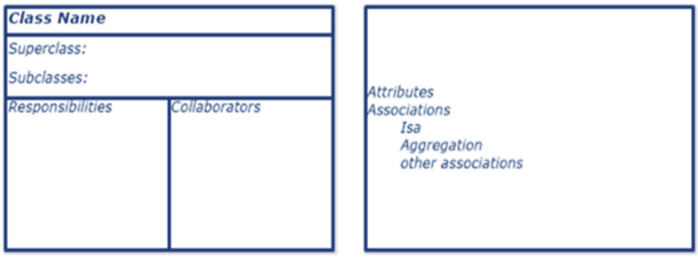

Fig. 3. A template of a CRC card [generated by software ideas modeler]

The NL software requirement has been processed in previous step to generate SBVR vocabulary and rules that define SBVR model. To generate CRC model, there is need to identify SBVR model elements and their corresponding CRC elements as shown in Table 1. After that perform their mapping using transformation rules that are explained in next sub section.

By using the SBVR to CRC mappings given in Table 1, a set of transformation rules are defined that are used to generate metadata of the target CRC model. Each Transformation rule consists of two parts. One is Left-hand-side (LHS) for source pattern and second is Right-Hand-Side (RHS) for target pattern.

Table 1. Mapping of SBVR model elements to CRC model elements

SBVR element	CRC model - front side	CRC model - back side
General noun concept	Class name	–
Verb concept	Responsibility	–
General noun concept	Collaborator	–
Is-property-of or unary verb concept	–	Attribute
Associative verb concept	–	Association
Partitive verb concept	–	Aggregation
Categorization verb concept	Generalization (superclass, subclass)	–

Extracting Class Name: CRC modeling provides a simple, easy and powerful way to identify and organize classes relevant to system requirements. CRC card has been divided into three sections [4]. The top section of CRC card contains class name. A collection of similar objects represents a class. In SBVR vocabulary, all object types are used identify to classes. The Rule1 contains three main parts of CRC card i.e. class name, responsibility and collaboration. To describe each part of CRC card, a transformation rule is needed. So Rule1 consists of three more rules i.e. Rule 2, Rule3 and Rule4. Rule1 explains how CRC card designed is given below:

Extracting Responsibility: Anything that a class knows or does is responsibility. Each class knows about its basic information and functionality that a class can perform. Every class must have some responsibility to be performed. If a class has sufficient information to accomplish assigned responsibility then it does not need any collaborator class otherwise it needs collaboration [27].

Extract Class Attribute: Each class has some basic information. To represent this information, all characteristics or unary verb concept (excluding action verbs) that are extracted from SBVR vocabulary are used.

Extracting Class Methods: A CRC card contains attributes and method of a class. Methods describe basic functionality that a class can perform [27]. There may be some responsibility that a class does not perform and to perform this responsibility class needed collaboration with other class that has ability to perform required functionality. Now, we want to extract methods for a class. In CRC card diagram, left section contains responsibilities that describe what a class knows or does. OO analysis of SBVR vocabulary shows that verb concepts describe actions that all common nouns can perform. So, all the verb concepts (associated to noun concept) are extracted from SBVR vocabulary is mapped to methods of a class.

After identifying the collaborator class, it is necessary to add responsibilities and check whether this class can perform its responsibilities. If identified collaborator class does not fulfill its responsibility then it requires another collaborator class. There are different types of generic relationship between classes that helpful in identification of collaborators.

- Extract is-part-of relationship: There are number of classes that are extracted from software requirements specifications and have some relationship with other classes in terms of functionality. The is-part-of relationship is used to connect all the classes that are part of an aggregate class. This relationship may be a composition or aggregation between classes.
- Find Aggregation: In this type of relationship, a part exists independently of whole and also whole is not especially responsible for parts. Now, we want to express relationship that exist between these classes so that CRC cards effectively show requirements specifications that leads to agile software development. To find relationship between these classes, the categorization verb concepts are mapped to aggregation. The main class is subject part of verb concept while subclass is object part of verb concept.
- Find Composition: If a part is always a part of single whole then it shows composition. The part exists only if the whole exist and whole is responsible for creation of other objects.
- Extract Is-kind-of relationship: Sometimes a class uses attributes and operations of another class additional to its own attributes. And that class does not exist without main class. There exists a relationship of is-kind-of between two classes that identify generalization between these classes.
- Extract Association: Sometimes, there is a more general relationship between classes that is known as association. In association, one class relates to another class. Association creates a links between classes. After performing OO analysis of SBVR based software requirements specifications, we have find that the association is formed from all associative verb concepts.
- Multiplicity: An association can be further elaborated by representing multiplicity. By using OO information that has extracted from software requirements specifica tions, we can represent multiplicity by using quantification with noun concept.

3.3 Drawing CRC Cards

After generating metadata of CRC model in the previous phase, the extracted metadata is used to draw a CRC model. The CRC cards can be generated from the extracted metadata in two ways; by XMI with a professional CASE tool or draw from scratch by using graphics libraries in Java. In our approach, the first way is opted as it generates professional CRC cards in a formal way. The processes of drawing a CRC card starts from a XMI template of an empty CRC card. In our implementation, such XMI template is generated by drawing an empty CRC card in the Visual Paradigm tool and exporting its XMI representation.

4 Experiment and Results

In this section, we have solved an example of library system originally presented by [28]. This example is later solved by Harmain to generate UML class models. Our proposed prototype tool takes software requirements specifications as natural language text and then generates SBVR based software requirements specifications. The SBVR software

requirements are further processed to extract business vocabulary as explained in Sect. 3. The SBVR software requirements are further processed to extract business vocabulary and all the extracted SBVR elements from solved example are shown in Table 2.

Table 2. Total number of SBVR elements extracted from the running example

Set no.	SBVR element	Number of entries
1	General noun concepts	42
2	Individual concepts	6
3	Verb concepts	11
4	Associative verb concept	07
5	Partitive verb concept	03
	Total number of entries	69

Following text explains the working of the prototype tool with screen shots. The NLbased software requirements specifications are given as input to the prototype tool SBVR2CRC as shown in Fig. 4.

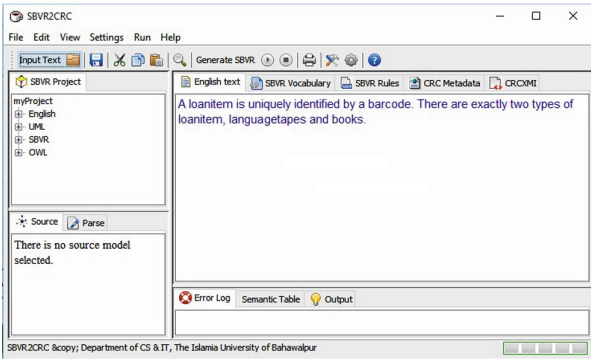

Fig. 4. The input given to the SBVR2CRC tool

When user clicks 'GenerateSBVR' button in menu bar, the SBVR specifications are generated that consists of SBVR vocabulary shown in Fig. 5.

Fig. 5. SBVR vocabulary generated by the SBVR2CRC tool

The proposed prototype tool then automatically extract CRC meta data from SBVR vocabulary and embed this metadata in a XMI of a blank CRC card that have exported from Visual Paradigm tool. We have also generated CRC XMI after embedding the metadata that is shown in Fig. 6.

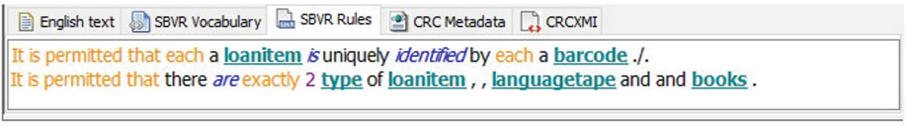

Fig. 6. SBVR rules generated by the SBVR2CRC tool

Our proposed tool has also extracted metadata from given input according to the transformation rules given in Sect. 3. The meatdata that is extracted is also shown in Fig. 7.

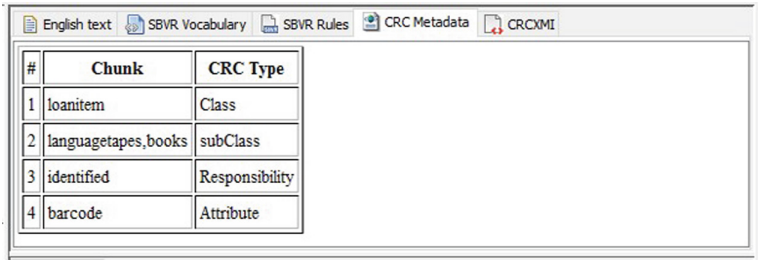

Fig. 7. SBVR model mapping to CRC model by the SBVR2CRC tool

The proposed prototype tool then automatically extract CRC meta data from SBVR vocabulary and embed this metadata in a XMI of a blank CRC card that have exported from Visual Paradigm tool. We have also generated CRC XMI after embeding the meatdata.

The CRC XMI consists of information about class name, its attributes, responsibilities and also super classes and sub classes of main class it have. A CRC is generated from the XMI is shown in Fig. 8.

Now, evaluation of proposed tool is performed by using statistical measures precision and recalls. To accomplish this, we compare our presented prototype tool with another tool presented by Harmain et al. [9] because a similar approach was presented to generate class diagram from natural language. To do this comparison, first of all there is needed to identify three sets. The precision and recall measures are defined on the cardinality basis of these three sets is given below. The precision and recall percentage of all CRC elements generated by our transformation and Harmain's transformation is given in Table 3.

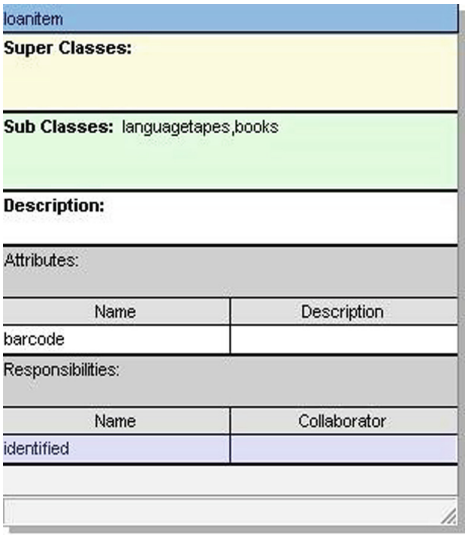

Fig. 8. A CRC card generate from the XMI using Visual Paradigm tool.

Table 3. Precision and recall measures of CRC metadata

CRC element	Expected CRC element	Our Transformation					Harmain's Transformation				
		TP	FN	FP	Precision	Recall	TP	FN	FP	Precision	Recall
Classes	10	10	0	0	100%	100%	10	0	5	0.67	100%
Attributes	10	10	0	0	100%	100%	8	2	5	0.62	100%
Methods	11	11	0	0	100%	100%	9	2	4	0.69	100%
Association	07	07	0	0	100%	100%	5	2	5	0.44	71%
Aggregation	0	0	0	0	100%	100%	0	0	0	0	0
Super class	1	1	0	0	100%	100%	0	1	0	N/A	0
Sub class	2	2	0	0	100%	100%	0	2	0	N/A	0
Visibility	21	19	2	2	95%	95%	0	21	0	N/A	0
Primitive Data Type	17	16	1	1	100%	94%	0	17	0	N/A	0
Composition	02	01	0	1	50%	100%	1	1	1	50%	50%
Overall	**81**	**77**	**3**	**4**	**94.5%**	**96%**	**33**	**48**	**20**	**62%**	**41%**

We have also plotted the clustered column chart that compares values of precision and recall of SBVR2CRC prototype tool and Harmain's transformation shown in Fig. 9. Harmain's transformation did not generate several important elements like aggregation, super class, sub class, visibility and primitive data type that also greatly influenced the overall precision of their transformation. Our presented transformation efficiently identifies all these important elements that are used to generate CRC model and makes a great contribution in this aspect that leads towards generation of quick and accurate software development.

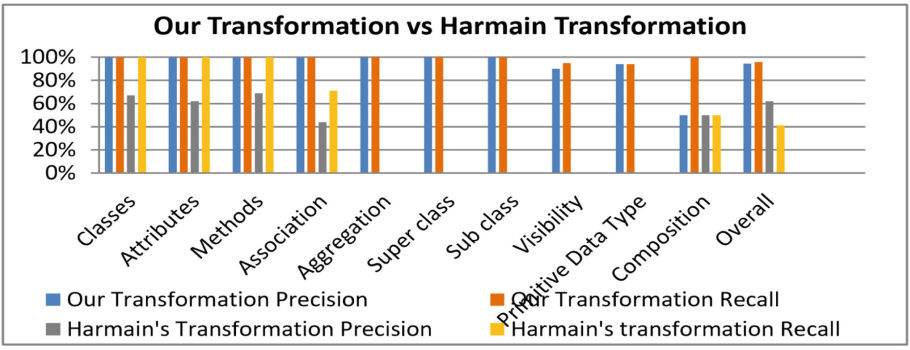

Fig. 9. The precision and recall clustered column chart

5 Conclusion and Future Work

The main objective of paper was to address the challenging aspect of software requirements specifications captured in natural language that are difficult to machine process and produced low quality software. We presented a controlled representation of software requirements that has greatly improved the accuracy of machine processing and generate high quality software in an agile way. Our proposed approach starts from generation of SBVR based requirements specifications. After that perform automatic analysis of SBVR based specifications, CRC metadata like association, super class, subclass, visibility, primitive data type is extracted that all other tools cannot extract. Finally, SBVR2CRC prototype tool maps this extracted information into CRC cards that represent software requirements specifications not only in an efficient way but also improve the accuracy of software as well as faster the development process. Moreover, our automated tool can be helpful for software developers to efficiently represent software requirements specifications that results into agile software development and also increase overall efficiency of development process.

Our presented approach also identifies all CRC elements that are not considered by any other approach. There is not a single approach to generate CRC model of software requirements specification. For the future work, we have planned to extend our approach to handle alias, interface. We expect that our presented approach could play

an important role in capturing software requirements specification in the form of CRC model that not only speed up the whole development process but also produce high quality software.

References

1. Sillitti, A., Succi, G.: Requirements engineering for agile methods. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 309–326 (2005)
2. Cao, L., Ramesh, B.: Agile requirements engineering practices: an empirical study. IEEE Softw. **25**(1), 60–67 (2008)
3. Batool, A., Hafees, Y.: Comparative study of traditional requirement engineering and agile requirement engineering. In: 15th International Conference on Advanced Communication Technology (ICACT), pp. 1006–1014 (2013)
4. Beck, K., Cunningham, W.: A laboratory for teaching object-oriented thinking. In: Proceedings of OOPSLA, pp. 1–6 (1989)
5. Wirfs, R., Alan, B.: Object Design: Brief Tour of Responsibility Driven Design. Addison Wesley (2003). book chapter 2, ISBN 0201379430
6. Inayt, I., Salwah, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51**, 915–929 (2015)
7. OMG: Semantics of Business Vocabulary and Rules (SBVR) Standard v.1.0. Object Management Group (2008). http://www.omg.org/spec/SBVR/1.0/
8. Overmyer, S.: Conceptual modeling through linguistic analysis using LIDA. In: Proceedings of ICSE 2001 23rd international conference on Software Engineering, pp. 104–410 (2001)
9. Harmain, H.M., Gaizauska, R.: CM-builder: a natural language based CASE tool for object-oriented analysis. Autom. Softw. Eng. **10**(2), 157–181 (2003)
10. Gomes, P., Pereira, F., Paiva, P., Carreiro, P., Ferreira, J.: Reuse of UML class diagrams using case-based composition. In: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE), pp. 20–24 (2004)
11. Bajwa, S., Mumtaz, S., Samad, A.: Object oriented software modeling using NLP based knowledge extraction. Eur. J. Sci. Res. **32**(3), 613–619 (2009)
12. Deeptimahanti, D., Ali, M.: An automated tool for generating UML models from natural language requirements. In: IEEE/ACM International Conference on Automated Software Engineering (2009)
13. Bhala, R., Sagar, V., Abirami, S.: Conceptual modeling of natural language functional requirements. J. Syst. Softw. **88**, 25–41 (2014)
14. Gulia, S., Choudhury, T.: An efficient automated design to generate UML diagram from Natural Language Specifications. In: 6th International Conference on Cloud System and Big Data Engineering (2016)
15. Arora, A., Sabetzadeh, L., Briand, M., Zimmer, F.: Automated checking of conformance to requirements templates using natural language processing. IEEE Trans. Softw. Eng. **41**(10), 944–968 (2015)
16. Raj, A., Prabhakar, T., Hendryx, S.: Transformation of SBVR business design to UML models. In: Proceedings of 1st Annual India Software Engineering Conference, ISEC, pp. 29–38 (2008)
17. Bajwa, S., I., Afreen, H.: Generating UML class models from SBVR software requirements specifications. In: Artificial Intelligence Conference Belgian/Netherlands (2011)
18. Njonko,: From natural language business requirements to executable models via SBVR. In: International Conference on Systems and Informatics (ICSAI), pp. 2453–2457 (2012)

19. Awasthi, S.: Transformation of SBVR business rules to UML class model. In: Pfeiffer, H.D., Ignatov, D.I., Poelmans, J., Gadiraju, N. (eds.) Conceptual Structures for STEM Research and Education. Lecture Notes in Computer Science (Including Subseries Lecture in AI and Lecture Notes in Bioinformatics), pp. 277–288. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35786-2_21
20. Skersys, T., Danenas, P.: Extracting SBVR business vocabularies and business rules from UML use case diagram. J. Syst. Sofw. (2018). https://doi.org/10.1016/j.jss.2018.03.061
21. Nemuraite, L.: Vetis tool for editing and transforming SBVR business vocabulary and rules into UML & OCL. Inf. Technol. 377–384 (2010)
22. Bonais, M., Kinh, N., Eric, P., Wenny, R.: Automated generation of structural design models from SBVR specification. Appl. Ontol. **11**, 51–87 (2016)
23. Manning, C.D.: Par-of-speech tagging from 97% to 100%: is it time for some linguistics? In: Proceedings of CICLing, vol. 1, pp. 171–189(2011)
24. Toutanova, K., Klein, G., et al.: Feature-rich part-of-speech tagging with acyclic dependency network. In: Proceedings of HLT-NAACL, pp. 252–259 (2003)
25. Finkel, R.J., Grenager, T., Manning, C.: Incorporating non-local information into extraction systems by Gibbs sampling. In: proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 363–370 (2005)
26. Fayad, M., Hamza, H., Sanchez, H.: A pattern for an effective class responsibility collaborator (CRC) cards. In: Proceeding of 5th IEEE Workshop on Mobile Computing Systems and Applications USA, pp. 584–587 (2003)
27. Schach, S.R.: Object-Oriented and Classical Software Engineering. WCB/McGraw-Hill, Boston (2011)
28. Callan, R.E.: Building Object-Oriented Systems: An Introduction from Concepts to Implementation in C++. Computational Mechanics Publications, Southampton (1994)