Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 189 (2021) 328–334

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on AI in Computational Linguistics

A review of the generation of requirements specification in natural language using objects UML models and domain ontology

Alaa Abdalazeim[a1] and Farid Meziane[b]

[a]*University of Bahri, Sudan*
[b]*University of Derby, UK*

Abstract

In the software development life cycle, requirements engineering is the main process that is derived from users by informal interviews written in natural language by requirements engineers (analysts). The requirements may suffer from incompleteness and ambiguity when transformed into formal or semi-formal models that are not well understood by stakeholders. Hence, the stakeholder cannot verify if the formal or semi-formal models satisfy their needs and requirements. Another problem faced by requirements is that when code and/or designs are updated, it is often the case that requirements and specifically the requirements document are not updated. Hence ending with a requirements document not reflecting the implemented software.
Generating requirements from the design and/or implementation document is seen by many researchers as a way to address the latter issue. This paper presents a survey of some works undertaken in the field of generation natural language specifications from object UML model using the support of an ontology. and analyzing the robustness and limitations of these existing approaches. This includes studying the generation of natural language from a formal model, review the generation of natural language from ontologies, and finally reviews studies about check to generate natural language from OntoUML.

Keywords: Requirements Specification, Natural Language Generation, Object UML Model, Ontology.

* Corresponding author. *E-mail address:* alaaabdelazheim@yahoo.com

1877-0509
10.1016/j.procs.2021.05.102

1. Introduction and motivation

In the software development lifecycle, requirements engineering is an important phase because it determines client wishes, and the remaining development phases are based on it [1]. Hence, any error in requirements means reworks that consume time and require additional costs [1]. The requirements phase is a collaborative process involving end-users, domain experts, and analysts to write the requirements in natural language that may be incomplete and inconsistent then the analysts will convert them into formal or semi-formal models which are often not well understood by end-users but favored by the designers and developers [2]. Furthermore, modifications in the final stages are often not reflected in the requirements specifications [1]. This creates inconsistencies between requirements specification documents and system (design/implementation) documents, therefore, the design of a framework to generate specification in natural language from object model specified in the Unified Modeling Language (UML) and an ontology to align the modifications that would have been made in the code or design with the requirements and validate the consistency is a significant step to address the issue of consistency. Most recent works provide systems and frameworks to automate requirements specification transformation into formal models such as UML use cases and class diagrams which means checking the consistency in a forwarding approach. Little research attempted to align the modification of design or code to requirements specification.

This survey paper reviews the generation of specification in natural language from the object UML model using an ontology that enables communication with the stakeholders to reduce ambiguities in the use of terminologies and facilitate communication between the stakeholders. The remaining of the paper is organized as follows. Section2, will provide some background to the current study and will introduce the requirements specification phase, Natural Language Generation (NLG) and ontologies. In section 3, we review some of the most important works in generating Natural language requirements from various models. We discuss the outcome of the review in section 4 and conclude in section 5.

2. Background

In this section, we will introduce the background information that is needed to understand this review paper. We will particularly introduce software requirements specification, natural language generation systems and ontologies.

2.1. An overview of software requirements specification

A software requirement specification (SRS) is a document that represents an agreement between the client and the software developer. It specifies the client's requirement and can also be used to validates the final product [3]. Furthermore, IEEE defined SRS as "A specification for a particular software product, program, or set of programs that perform certain functions in a specific environment" [4]. In addition, [4] described a good SRS document as one that is Correct, Unambiguous, Complete and Consistent. In the majority of cases, SRS are written in natural language as this is probably the only language that clients can understand and are comfortable to sign. It is reported that up to 20-25% of the project time is spent in defining the requirements [3]. The remaining development phases of software are based on SRS. SRS are transformed into formal or semi formal specifications before they are further transformed to design models and then implementation using programming languages.

2.2. A brief introduction to natural language generation systems

Natural language generation systems (NLGS) are computer systems that can produce understandable texts in human languages from some underlying non-linguistic representation of information [5,6]. They are therefore a mapping from some kind of representation to natural languages. The achieve this goal, NLGS are structured into a set of tasks. The number of tasks and sometimes subtasks differ from one approach to another. Reiter and Dale [5] divided the process of NLGS into six distinct tasks and these are summarized as follows:

- ***Content determination.*** This is the process of determining which information will be included in the text [7]. The output of this task is usually a set of objects data and their relationships which are referred to as messages. Most recent research use data-driven techniques and a cluster approach with content determination [6].

- ***Text structuring (Discourse planning).*** In this task we define an order for the content defined and generated in the previous task [5]. The techniques used in this phase include handcrafted rules, domain-dependent structuring rules and the Rhetorical Structure Theory [6].

- ***Sentence aggregations.*** The ordered messages generated in the previous task are aggregated in this task to generate meaningful sentences. The sentences themselves can be aggregated to make the text more readable, remove redundancies and enhance the fluency of the text [5,7]. Early systems used domain dependent relationships and knowledge to perform sentence aggregation but more recent works data-driven and machine learning approaches.

- ***Lexicalization.*** Lexicalization is the process of deciding which specific words and phrases should be chosen to express the information [5]. This is a complex task as in Natural language some events and processes can be described in different ways. It is reported in [5] that lexicalization is particularly important when generating multilingual texts.
- ***Referring expression generation***. Reiter and Dale [5] defined referring expression generation as "the task of selecting words or phrases to identify domain entities" and to distinguish it from the lexicalization phase added that this should be a discrimination task "where the system needs to communicate sufficient information to distinguish one domain entity from other domain entities". Some approaches uses for this task are described it [6].
- ***Linguistic realizations.*** In this phase, grammar, orthography, and morphological rules are used to produce correct sentences. There are many approaches that are used in linguistic realization and this include [7] human-crafted templates, human-crafted grammar-based systems and statistical.

2.3. Ontologies

In 1993, Gruber defined an Ontology as an "explicit specifications of conceptualizations" [8] which defines a common vocabulary for a knowledge domain [9]. An Ontology involves a set of concepts such as entities, attributes, process, and their relationship and this is referred to as the conceptualization of a specific domain [10]. Furthermore, Ontologies support the sharing of information structures, the reuse of domain knowledge, and the clarification of domain assumptions [10]. Ontologies are classified into different types depending on their usage and this include:

- Top-level Ontology: being in support of a wide view of the world appropriate for several target domains.
- Reference Ontology: aims to the structuring of ontologies derived from them.
- Core Ontology: derives from the definition of a super domain.
- Application Ontology: appropriate to use in reasoning engines or software packages.

Valaski et al. (2016) claim that an ontology can assist in (i) identifying problems in specification and models; (ii) improving communication; (iii) building models more accurate and (iv) permitting traceability among artifacts and enhances the requirements identification quality [11]. Within the context of this paper, we are more interested application ontologies and specifically, those that are designed and developed to support the requirements engineering phase of software development.

3. Generating natural language specifications from software models: state of the art

In this section, we will review two types of works. First, we will review systems that attempted to generate natural language specifications from Software models. In the second subsection, we will review systems that

generate natural language from ontologies. These two approaches are complementary when used to generate natural language specifications.

3.1. Transformation formal model into NL specification

Hudaib et al. [12] suggested a bidirectional model to convert natural language requirements written in English into UML and then generate natural language specifications from UML models. The natural language specification is preprocessed to generates XML representation to input the natural language requirements and XML schema representation is used to generate both UML diagrams and natural language specification. They used the MIMB tool (http://www.metaintegration.com/Products/MIMB/SupportedTools.html?find=cosort) to convert XML into UML and vice versa and the GATE (https://gate.ac.uk/) tool for natural language processing. To evaluate their system, they collected a set of requirements from academic research that have been used for the evaluation of similar systems. One of the limitations of this work is the use of an XML dataset for the evaluation of their system. In practice, models are not expressed in XML.

Fockel and Holtmann [13] proposed a bidirectional model to convert between requirements-based model and a controlled natural language (CNL) and reverse. Their system is developed into a full methodology taking the advantages of both models namely model-based and CNL. They have used an automotive example as a case study to illustrate their methodology. They evaluated the transformation between the two models by measuring the execution time of the transformation runs in different use cases. One of the limitations of this work is its evaluation as it measures just the execution time of the transformation without measuring the quality of the generated natural language requirements.

In [14] Goto et al. present an approach to create scenarios from a conceptual model by generating an event list of scenarios from Use Case Correspondence Models (UCCM). Each generated event contains a flag as a checkpoint to identify requirements. Eventually, software engineers and clients describe the requirement based on a flag. They evaluated their approach by comparing the requirements created from the event list and the manual requirements created by the software engineers.

Burke and Johannsson [15] developed a tool to transform formal software specifications into natural language. They added formatting and automatic generation of grammar modules for domain-specific vocabulary [17]. Meziane et al. [1] developed the GeNLangUML system to generate natural language specification from UML models by developing rules and use of WordNet and a linguistic ontology to remove ambiguities. They have evaluated their system using academic case studies. Brosch and Randak [16] developed a tool to enable automatic generation of textual specification from a class diagram using standard phrases into the natural language using the Eclipse Plugin. Their study is developed to only assist students in the software engineering field to generate textual specifications from a class diagram model.

Burden and Heldal [17] used a hotel reservation system to illustrate their system that generates natural language from Executable and Translatable UML (xtUML). They considered only the static part of the platform-independent model. Their method for transformation contains two steps: (1) the class diagram xtUML model was transformed into an intermediate linguistic model using the Grammatical Framework. (2) The linguistic model is transformed into a natural language text. They used xtUML - which is more restricted than UML - with Bridge Point as xtUML tool.

Landhaußer et al. [18] designed a Requirements Engineering Feedback System (REFS) to ensure the model and the natural language specification are consistent when the models change. They have defined two models, the initial model (Mi) and the changed model (Mc) these are continually compared and synchronized using EMFCompare of the Eclipse Modeling Framework to compute the difference between Mi and Mc. They used the WHOIS protocol

specification as a case study and evaluate their approach by experiment on three requirement specification. Besides, Pattanotai [19] proposed a method to automatically generate class description in two steps: (1) the class diagram component transmitted into a tree data structure (2) generate a class description for the class component tree. He evaluated his method by implementing a desktop application using C# language which generates class descriptions from class diagram. Zontek [20] developed a new software tool FLOOR which integrates domain ontology and requirements ontology to enable real-time feedback concerning to quality of new requirements.

3.2. Generation natural language from Domain ontologies

In the requirements process the Ontology ensures consistency and aid reasoning, therefore, an ontology-based requirements specification tool could assist in reducing misunderstanding and misinformation [9]. It is worth mentioning that Bontcheva [21] stated that an Ontology should be engaged with the requirements engineering phase. Reviewing this type of work is important as ontologies are the backbone for the NLGS when applied to requirements engineering.

Bontcheva and Wilks [22] developed the MIAKT approach (Medical Imaging and Advanced Knowledge Technologies) which uses NLG tools for generating reports from a domain ontology encoded in semantic Web standards such as OWL and RDF. Androutsopoulos et al. [23] developed three methods to generate a textual description of objects from symbolic information by employing OWL ontology and using M-PIRO's multilingual generation system as an example. Schutte [24] proposes a system to generate natural language descriptions for classes of Ontology. Angeli et al. [25] designed a generation system that executes content selection and surface realization in a domain-independent framework using three domains Robocup sportscasting, technical weather forecasts, and common weather forecasts the system evaluated by BLUE score and human evaluation.

Stevens et al. [26] proposed an application to generates text-based definition of Ontology entities using natural language generation NLG processes. Their application for Experimental Factor Ontology (EFO) was evaluated using a survey to test the fluency of generated text. Androutsopoulos et al. [27] developed the Natural OWL System which is an open source that support English and Greek language to generate fluent and coherent multi-sentence texts which describe individuals or classes of OWL ontologies with better-generated text with the domain-dependent linguistic resources.

Cimiano et al. [28] designed a Natural Language Generation system that used RDF data to generate Natural language text depending on Ontology and Ontology lexicon through the natural language generation process. They use the cooking domain as a case study and evaluated the fluency of the generated text with cooking novices and advanced cooks. Similarly, Dong and Holder [29] designed a Natural language engine to generate English natural language from the RDF graph.

Braga and Almeida [30] proposed an approach to assess conceptual models by initiating narratives stories about a subject domain so that these stories can assist in conceptual model validation. They used domain of Software Configuration Management as an example to view the approach. Furthermore, Cojocaru et al. [31] developed an application to generate natural language from ontology using Rhetorical Structure Theory which structures hierarchically the ontological content to be in human-like. Elliott and Allen [32] developed a methodology to generate software requirement specification using an Ontology in IEEE standard format. Their methodology offered seven use cases and ontology framework and evaluated using three case studies.

4. Discussions

In this paper, we presented a comprehensive review of the generation of natural language from the object models, domain ontology, XML, and OntoUML and discussed various issues related to each approach. The paper organizes the previous work into three categories: (1) the transformation of the formal model into natural language

specification; (2) Generation of natural language from domain ontology; (3) Generation of natural language by reverse XML to UML. We finally reviewed the studies that propose the OntoUML model as the way to generate natural language. We found that most research in transforming software models into natural language specification propose tools or systems to generate natural language using XML schema, developed rules, created templates and tree data structure. The use and introduction of ontologies to generate text has provided the rigour that other approaches lacked in the past. The most successful systems are those that are domain dependent and use a domain ontology. This makes the generalisation of such approaches difficult. One way of dealing with this issue could be the use of several ontologies where for each domain a specific and related ontology will be selected. There are many lessons that can be learned from the systems that generated natural language texts from ontologies which are encoded in OWL and RDF. Eventually, several studies start to evaluate the expressiveness of OntoUML compared to UML also assess the possibility of generating natural language from OntoUML.

5. Conclusions

Keeping consistency between software requirements is an important aspect in the software development process as this keep the mapping between the different phases. This also helps software maintenance and particularly when the developers involved in the initial system have left the organization. Natural language generation has gained maturity over the last few years and the generation of software specifications from various design models are gaining in maturity. The introduction of ontologies and particularly domain specific ontology has improved the quality of the generated text and provided consistency between the various models. .

References

[1] F. Meziane, N. Athanasakis and S. Ananiadou (2008), “Generating Natural Language Specifications from UML Class Diagrams”, *Requirements Engineering Journal*, **13(1)**:1-18.

[2] Gašević D., Kaviani N., Milanović M. (2009) “Ontologies and Software Engineering”. In: Staab S., Studer R. (eds) *Handbook on Ontologies. International Handbooks on Information Systems*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-92673-3_27.

[3] R. Beg, Q. Abbas and A. Joshi, "A Method to Deal with the Type of Lexical Ambiguity in a Software Requirement Specification Document," 2008 First International Conference on Emerging Trends in Engineering and Technology, Nagpur, India, 2008, pp. 1212-1215, doi: 10.1109/ICETET.2008.160.

[4] J. Doe (2011), "Recommended Practice for Software Requirements Specifications (IEEE)”, available at: https://www.midori-global.com/downloads/jpdf/jira-software-requirement-specification.pdf (accessed: 11/03/2021).

[5] Ehud Reiter and Robert Dale (1997), “Building applied natural language generation systems,” *Natural Language Engineering*, 3(1): 57–87.

[6] A. Gatt and E. Krahmer (2018), "Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation," *Artificial Intelligence Research*, (61): 65 – 170.

[7] John Bateman and Michael Zock (2005), Natural Language Generation In: Ruslan Mitkov (Ed), The Oxford Handbook of Computational Linguistics (1 ed.), Oxford University Press, DOI: 10.1093/oxfordhb/9780199276349.013.0015.

[8] T.R. Gruber T.R (1993), “A translation approach to portable ontology specifications”, Knowledge Acquisition, 5(2):199–220

[9] Chen J., Xue X., Huang L., Ren A. (2019) An Overview on Visualization of Ontology Alignment and Ontology Entity. In: Krömer P., Zhang H., Liang Y., Pan JS. (eds) Proceedings of the Fifth Euro-China Conference on Intelligent Data Analysis and Applications. ECC 2018. Advances in Intelligent Systems and Computing, vol 891. Springer, Cham. https://doi.org/10.1007/978-3-030-03766-6_42.

[10] Diego Dermeval, Jéssyka Vilela, Ig Ibert Bittencourt, Jaelson Castro, Seiji Isotani, Patrick Brito & Alan Silva (2016). “Applications of ontologies in requirements engineering: a systematic review of the literature”. Requirements Engineering, 21, 405–437. https://doi.org/10.1007/s00766-015-0222-6

[11] Valaski, J; Reinehr, S; Malucelli, A. (2016), “Which Roles Ontologies play on Software Requirements Engineering? A Systematic Review”, Proceedings of the International Conference on Software Engineering Research and Practice (SERP); Athens: 24-30. Athens: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[12] Amjad Hudaib, Bassam Hammo and Yara Alkhader (2007), “Towards Software Requirements Extraction Using Natural Language Approach”, Proceedings of the 6th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, Corfu Island, Greece, February 16-19, 2007 pp. 155-160.

[13] M. Fockel and J. Holtmann (2014), "A requirements engineering methodology combining models and controlled natural language," 2014 IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE), Karlskrona, Sweden, 2014, pp. 67-76, doi: 10.1109/MoDRE.2014.6890827.

[14] K. Goto, S. Ogata, J. Shirogane, T. Nakatani and Y. Fukazawa (2015). "Support of scenario creation by generating event lists from conceptual models," 2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), Angers, France, 2015, pp. 376-383.

[15] Burke D.A., Johannisson K. (2005). "Translating Formal Software Specifications to Natural Language". In: Blache P., Stabler E., Busquets J., Moot R. (eds) Logical Aspects of Computational Linguistics. LACL 2005. Lecture Notes in Computer Science, vol 3492. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11422532_4.

[16] Petra Brosch, Andrea Randak (2010). "Position Paper: m2n—A Tool for Translating Models to Natural Language Descriptions", Electronic Communications of the EASST, vol. 34. http://dx.doi.org/10.14279/tuj.eceasst.34.593.622.

[17] Håkan Burden and Rogardt Heldal. (2011). "Natural language generation from class diagrams". In Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa). Association for Computing Machinery, New York, NY, USA, Article 8, 1–8. DOI:https://doi.org/10.1145/2095654.2095665.

[18] M. Landhäußer, S. J. Körner and W. F. Tichy, "Synchronizing domain models with natural language specifications," 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), Zurich, Switzerland, 2012, pp. 22-26, doi: 10.1109/RAISE.2012.6227965.

[19] N. Pattanotai, "Automatic class description generation," 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2016, pp. 105-109, doi: 10.1109/CompComm.2016.7924674.

[20] Edward Zontek-Carney (2017). "FLOOR (Framework for Linking Ontology Objects and Textual Requirements): A New Requirements Engineering Tool that Provides Real-time Feedback", Master thesis, University of University of Maryland, USA (https://drum.lib.umd.edu/handle/1903/19564).

[21] Bontcheva K. (2005) Generating Tailored Textual Summaries from Ontologies. In: Gómez-Pérez A., Euzenat J. (eds) The Semantic Web: Research and Applications. ESWC 2005. Lecture Notes in Computer Science, vol 3532. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11431053_36.

[22] Bontcheva K., Wilks Y. (2004) "Automatic Report Generation from Ontologies: The MIAKT Approach". In: Meziane F., Métais E. (eds) Natural Language Processing and Information Systems. NLDB 2004. Lecture Notes in Computer Science, vol 3136. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-27779-8_28.

[23] I. Androutsopoulos, S. Kallonis, and V. Karkaletsis (2005), "Exploiting OWL Ontologies in the Multilingual Generation of Object Descriptions," in Proceedings of the Tenth European Workshop on Natural Language Generation. (https://www.aclweb.org/anthology/W05-1617).

[24] Neils Schütte (2009). "Generating Natural Language Descriptions of Ontology Concepts", Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009), Athens, Greece, pp. 106–109. (https://www.aclweb.org/anthology/W09-0617/).

[25] G. Angeli, P. Liang, and D. Klein (2010). "A Simple Domain-Independent Probabilistic Approach to Generation," in Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP, Cambridge, MA. Pp. 502–512.

[26] Stevens R, Malone J, Williams S, Power R, Third A. (2011). "Automating generation of textual class definitions from OWL to English". Journal of Biomedical Semantics. 2011 May 17. doi: 10.1186/2041-1480-2-S2-S5.

[27] Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis (2013). "Generating natural language descriptions from OWL ontologies: the natural OWL system". Journal of Artificial Intelligence Research, 48, 1 (October 2013), 671–715.

[28] Philipp Cimiano, Janna Lüker, David Nagel, Christina Unger (2013). "Exploiting Ontology Lexica for Generating Natural Language Texts from RDF Data". Proceedings of the 14th European Workshop on Natural Language Generation, Sofia, Bulgaria, Association for Computational Linguistics, pp. 10–19.

[29] Ngan T. Dong and Lawrence B. Holder (2014). "Natural Language Generation from Graphs". International Journal of Semantic Computing, 8(3): 335-384.

[30] B.F.B. Braga and J.P.A. Almeida. (2015). "Modeling Stories for Conceptual Model Assessment". In: Jeusfeld M., Karlapalem K. (eds) Advances in Conceptual Modeling. ER 2015. Lecture Notes in Computer Science, vol 9382. Springer, Cham. https://doi.org/10.1007/978-3-319-25747-1_29.

[31] Cojocaru, Dragos Alexandru and Stefan Trausan-Matu (2015). "Text Generation Starting from an Ontology", RoCHI (2015).

[32] Robert A. Elliott Sr. and Edward B. Allen (2016). "Creating a Software Requirements Specification Document Using an Ontology Based Methodology", Vol. 3, Issue 9, September 2016.