Transformation of SysML Requirement Diagram into OWL Ontologies

Helna Wardhana[1], Ahmad Ashari*[2], Anny Kartika Sari[3]
Department of Informatics, Universitas Bumigora, Lombok, Indonesia[1]
Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia[1, 2, 3]

***Abstract*—The Requirement Diagrams are used by the System Modeling Language (SysML) to depict and model non-functional requirements, such as response time, size, or system functionality, which cannot be accommodated in the Unified Modeling Language (UML). Nevertheless, SysML still lacks the capability to represent the semantic contexts within the design. Web Ontology Language (OWL) can be used to capture the semantic context of system design; hence, the transformation of SysML diagrams into OWL is needed. The current method of SysML Diagrams transformation into OWL is still done manually so that it is very vulnerable to errors, and the translation process requires more time and effort for system engineers. This research proposes a model that can automatically transform a SysML Requirement Diagram into an OWL file so that system designs can be easily understood by both humans and machines. It also allows users to extract knowledge contained in the previous diagrams. The transformation process makes use of a transformation rule and an algorithm that can be used to change a SysML Requirement Diagram into an OWL ontology file. XML Metadata Interchange (XMI) serialization is used as the bridge to perform the transformation. The produced ontology can be viewed in Protégé. The class and subclass hierarchy, as well as the object properties and data properties, are clearly shown. In the experiment, it is also shown that the model can conduct the transformation correctly.**

Keywords*—*SysML Diagram; Requirement Diagram; ontology; OWL; transformation

I. INTRODUCTION

The current system engineering process still tends to be centered on documents and uses various engineering diagrams that are sometimes inconsistent. Therefore the use of modeling languages is needed to determine the complexity of a system, including the non-software components. The need for this modeling language cannot be fulfilled by the Unified Modeling Language (UML). Therefore a System Modeling Language (SysML) profile was developed from UML. SysML is an extension of UML created by the Object Management Group (OMG) to support the modeling of a complex system involving humans and components of hardware and software.

SysML itself is becoming one of the most popular modeling languages. It is a widely accepted, object-oriented graphic software modeling language [1]. SysML reuses some diagrams in UML. SysML also provides other modeling capabilities, namely the requirements and the relationships of parametric, adding activities of UML, internal block diagram, and block definition diagram. According to the Meta-Object Facility (MOF), although SysML is a formal language, most types of diagrams in SysML are relatively easy to understand because of the graphical user interface.

The requirements and parametric constraints are modeled by SysML by expanding its semantics to support performance analysis and requirements engineering [2]. Use Case Diagram in UML can be used to model system functional requirements, but UML does not have elements that can describe non-functional requirements explicitly [3]. SysML can accommodate the deficiencies contained in UML because SysML using Requirement Diagram to depict and model non-functional requirements, such as response time, size, or system functionality in defining several elements. Nevertheless, SysML still lacks the capability to represent the semantic contexts within the design.

The development of integrated models in information modeling, where the model elements in one diagram can be related to the model elements in other diagrams, is one of the benefits of SysML [4]. SysML Diagrams also enable modeling systems that can be used at an early design stage that supports specifications as well as during design updates [5]. The semantic gap between heterogeneous systems and various disciplines can be bridged by the SysML-based system model because of the interoperability nature of SysML through the use of the XML Metadata Interchange (XMI) format. Tracking any changes in artifacts between requirements and specifications can also be done by the SysML-based information model. Interoperability among various tools for analyzing needs, structural, behavior, and system constraints can also be enhanced by this model. Thus, the SysML Requirement Diagram can capture the requirements as well as the functional, design, and process relationships between those requirements. This is achieved through the types of dependency relationships that exist in SysML, namely, `satisfy`, `verify`, `refine`, `derive`, `trace`, and `copy`. SysML Requirement Diagrams, as one of the kinds of the new diagram in SysML, enables the depiction of system requirements to be of high quality because it makes the description of requirements more easily to be understood and ensure traceability of system development.

The availability of a well-defined system model for carrying out all design tasks, including adjustments and evaluation of the system, is crucial for the system engineers and stakeholders in the acquisition of the system. The use of ontology enables system engineers to not only model metadata concepts but also semantic contexts that can be used in model inference and transformation rulemaking [6]–[8]. Ontology facilitates the process of managing the data obtained because

*Corresponding Author

ontology allows the proper arrangement of the entire system [9]. Ontology is also able to infer generalization and specialization between classes based on constraints imposed on the property of the class definition [10]. Furthermore, the appropriate concepts of a domain are reflected by the ontology [11]. Therefore, the transformation of the SysML Requirement Diagram into an ontology is needed. The purpose of each dependency relationship contained in the SysML Requirement Diagram can be shown in the form of object property in the ontology.

The aim of developing the ontology is to share a general understanding of the structure of information [12] and to have a common controlled vocabulary for various statements about the complexity of systems. The benefits from the development of ontology are the use of controlled vocabulary, durable information storage, information exchange without loss, integration of interdisciplinary information, analysis of automation, and manufacturing of the product.

SysML provides graphical syntax that is very useful for human understanding, but SysML does not have formal semantics. Web Ontology Language (OWL) and SysML are different languages, but both have terminology for instances, classes, and properties. OWL has construction terms for classes that are not owned by SysML, and SysML has terminology for operations that are not owned by OWL [7].

The development of manual ontologies using the OWL ontology editor at this time, such as Protégé, is still a fairly complex work, requires more understanding of the language of ontology, and is at risk of experiencing problems in the acquisition of knowledge [6]. Therefore, approaches and tools are needed that enable reducing efforts and adapting ontologies automatically or semi-automatically using existing sources of knowledge.

Existing researches on modeling language are more focusing UML to OWL transformation, both manually and automatically. Some researchers who have proposed the transformation model of UML into OWL automatically use the same type of diagram, namely the class diagrams [6], [10], [13]–[17]. Research about the transformation of SysML Diagrams into OWL has been performed by [18] and [7]. However, the transformation process is still done manually, so it is very susceptible to errors and requires more time and effort for system engineers because they have to repeat the same work as in the system development. Manual translation also results in the system engineers or other users not being able to extract the knowledge contained in the previous diagram [6].

This research proposes a model that is able to transform a SysML Requirement Diagram into an OWL ontology file automatically. The main contribution of this paper is the transformation rule and the algorithm that is used to change a SysML Requirement Diagram into an OWL ontology file. The resulting OWL file can be displayed through Protégé, which can clearly show the hierarchy of classes and subclasses, object properties, data properties, including their ontograf, to show the dependencies used in the SysML Requirement Diagram.

The rest of the paper is organized as follows. Related work is described in Section 2, while Section 3 explains the proposed model to transform a SysML Requirement Diagram into an OWL file. The transformation rule and the algorithm are described in Section 4, while Section 5 presents evaluation and discussion. The paper is concluded in Section 6.

II. Related Work

Research on the transformation of SysML Diagrams into OWL files is carried out by [18] and [7]. However, the transformation process in the research is still done manually. Research conducted by [18] uses several SysML diagrams, namely, requirements diagrams, activity diagrams, block definition diagrams, and internal block diagrams. It is to analyze and present scenarios about system model change from a formal perspective. Changes to the intended system model, for example, how to add, delete, and modify the model elements in response to changes in the design of a system. Ontology is applied to formalize transformation in the influence of the relationship between requirements, behavior, and structure of the system model so that its semantics can be understood by humans and can be read by machines. From the experiments using case studies of water distillation systems, [18] it is proven that identification of information on the impact of changes can help system designers to complete modifications in a short time and with higher quality.

Another research that transforms SysML into OWL is performed by [7]. The translation of block diagrams into OWL by [7] produces a method for creating an OWL knowledge base that can represent structural design information such as the decomposition of parts and connectivity structures of a system.

Several other researchers [6], [10], [13]–[17] have proposed translation models of UML into OWL automatically using the same type of diagram, which is the class diagram. The goal of [6] is the establishment of an appropriate conceptual correspondence between UML and OWL through the semantic-preserving scheme translation algorithm. The algorithm proposes an approach that automatically extracts OWL ontology from UML class diagrams and as formal evidence for semantic preservation that can also use to analyze the time complexity of the algorithm.

Research conducted in [10] uses eXtensible Stylesheet Language (XSL) style sheets to transform UML models, producing applications that automatically transform class diagrams into OWL ontologies based on the proposed transformation rules. An eXtensible Stylesheet Language Transformations (XSLT)-based architecture for automated OWL development consisting of Metamodel Definition of Ontology that is defined using the Meta-Object Facility (MOF) has been proposed by [13]. Other research on the automatic translation of UML into OWL is carried out in [14], [15] which has revised the transformation rules identified in the literature review and proposed the verification rules to check the suitability of the UML class diagram with the ontological domain in OWL through an algorithmic method.

An automatic translation of UML class diagrams and statechart diagrams into OWL is proposed in [16] through an approach that analyzes the consistency and satisfaction of UML models using logical reasoning for OWL. The design and

software development that uses a model-based approach to produce OWL-based Web Service ontologies (OWL-S) from the UML model is proposed in [17]. The proposed method is based on the UML profile, which represents the characteristics of OWL-S.

Research related to translating UML into OWL files manually was carried out in [19]–[22]. The importance of the role of ontology in developing e-learning platforms is increasingly becoming a reason of [19] to build OWL Moodle that can make the data exchanged therein can be processed by machines. Other research conducted by [20] proposes ontology development methodologies to facilitate the decision making process about water management systems used in a web-based Decision Support Systems. Manually changing data and information on repository publications are addressed in [21], [22] through the use of structured knowledge that is based on ontology design with dynamic domains.

TABLE I. RESEARCH RELATED TO MODELING LANGUAGE INTO OWL TRANSFORMATION

Research	SysML to OWL Manual	UML to OWL		Diagram Type	Case Studies
		Automatic	*Manual*		
[18]	√			Requirement Diagram, Activity Diagram, Block Definition Diagram, and Internal Block Diagram	Water distillation system
[7]	√			Block Diagram	Vehicle
[15]		√		Class Diagram	
[14]		√		Class Diagram	
[16]		√		Class Diagram and State Chart Diagram	Content Management System
[6]		√		Class Diagram	University
[10]		√		Class Diagram	
[17]		√		Class Diagram, Sequence Diagram, and Activity Diagram	Publication
[13]		√		Class Diagram	Wine ontology
[19]			√	Class Diagram	Social Learning Net. Analysis
[20]			√	Class Diagram	Water management system
[21]			√	Activity Diagram	Repository of university publications
[22]			√	Use Case Diagram and BPMN Diagram	Repository of university publications

Table I summarizes related work and classifies the existing work based on the type of modeling language, the diagram types within each modeling language, and the case studies to be used in the experiment. The main difference between the existing researches with the proposed model is that the model suggests an automatic transformation from SysML into OWL. Although the transformation from SysML into OWL has been introduced in [18] and [7], the transformation is still done manually. With automatic transformation, this research supports the opportunity to increase the use of requirements diagrams to support object-oriented system modeling that incorporates not only software, but also people, materials, and other physical resources and can express the structure and behavior of a system.

III. PROPOSED WORK

This research proposes a transformation model from SysML Requirement Diagram into OWL ontologies automatically. The proposed model takes the XMI serialization of the SysML Requirement Diagram, as the input and then produces the appropriate OWL ontology in the RDF/XML syntax as the output. In general, there are four main processes in the proposed model, as shown in Fig. 1.

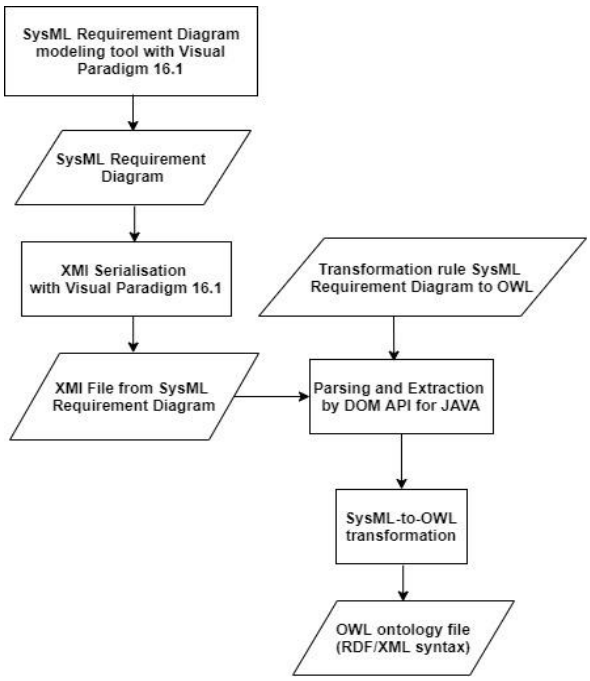


Fig. 1. The Architecture of the Proposed Model.

The first process is the modeling of the SysML Requirement Diagram using Visual Paradigm tool. Visual Paradigm Modeler is one of the modeling tools that can be used to create a SysML Requirement Diagram. The case examples used in this research are the ones that are presented in several references, including in the SysML International Standards document published by OMG. The second process is to export the SysML Requirement Diagram file to obtain the XMI serialization file. The XMI serialization extracted from SysML Requirement Diagrams is then transformed into an OWL ontology representation. The third process is the parsing and extraction XMI file from SysML Requirement Diagram by

Document Object Model (DOM) Parser Application Program Interface (API) for Java. The last process is the transformation of the SysML Requirement Diagram into an OWL document represented in Resource Description Framework/Extensible Markup Language (RDF/XML) syntax according to predefined transformation rule. The transformation rules will change the elements in the SysML Requirement Diagram into ontology components. A package is transformed into an ontology, a requirement is transformed into a class, a containment is transformed into a subclass, a dependency is transformed into a relationship (object property), an item is transformed into an attribute (data property). The complete explanation about the transformation rules is presented in Section IV. This transformation process generates OWL files that can be visualized through Protégé.

IV. Transformation Rules and Algorithm

This section presents the transformation rules and algorithms that are used to change the SysML Requirement Diagram in graphical symbols into OWL in RDF/XML syntax that can be displayed through Protégé.

A. SysML-to-OWL Transformation Rule

The proposed model is realized according to several transformation rules, as shown in Table II. This research proposes a set of rules for transformation of class, subclasses, associations, and almost all elements of the SysML Requirement Diagram. The rules are designed based on previous studies related to UML to OWL transformation, as proposed in [6], [10], [15]. SysML is an extension of UML so that SysML Requirements extends UML classes and dependencies [2], therefore, some elements in SysML Requirement Diagrams have common semantic correspondence with UML diagrams.

B. Transformation Algorithm

The proposed model for extracting the OWL ontology from a SysML Requirement Diagram can be implemented using the transformation algorithm `S2OTransformation` based on the transformation rules. The algorithm performs transformation for each element of the SysML Requirement Diagram into OWL in RDF/XML syntax automatically. The algorithm below can be applied to produce OWL in RDF/XML syntax so that ontologies can be directly displayed through Protégé.

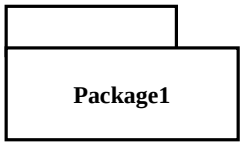
The algorithm has been implemented in Java programming language based on J2SE 1.8.0 platform. As can be read from the algorithm, the input is the XMI file produced from the serialization of the Requirement Diagram, while the output is the OWL file as the result of the transformation process.

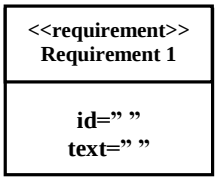
Algorithm S2OTransformation

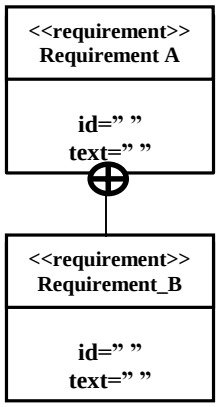
Input: XMI file from SysML Requirement Diagram
Output: OWL file displayed through Protégé
Begin
1. read XMI file exported from SysML Requirement Diagram file
2. find a node of the diagram based on diagram names
3. look for the list element diagram from the diagram
4. find a list of SysML model IDs based on subject values in all element diagrams
5. search for model nodes based on a tag name
6. look for the SysML model list based on the SysML ID list and the model
7. search for package nodes from the list element diagram
8. if the package found, then save as package value
9. if the package not found, then return null.
10. if the package node is not the same as null, proceed by searching the list SysL model node from the package
11. adding the SysML model node to the SysML model list
12. prepare data to generate OWL file
13. do iterate for each element diagram in the element diagram list
14. search data for OWL Class in diagram element
15. if the value of preferred shape = "Requirement",
16. look for the value of the SysML model with the subject value of the element diagram as ID SysML model
17. set attribute of OWL Class with the name of value SysML model
18. check subclass of diagram element
19. if the preferred shape = "Containment",
20. set the subclassOf attribute with the name of SysML model which is taken from the SysMLModelContainmentFrom diagram element value
21. if the userID and documentation property values in SysML model are not equal to null,
22. add the userID and documentation property values as OWL Datatype to the OWL Datatype list
23. search data for OWL Object Property from the diagram element
24. if the values of `from`, `to` and `preferred shape` are not equal to null
25. then set the OWL Object Property to the value of the name SysML model obtained `from`, `to` and `preferred shape`
26. generate OWL file
27. set initial data for OWL file by creating a document
28. set attribute
29. if the package is not the same as null,
30. add the package element
31. do iterate for each data of OWL Class
32. add the attribute value about OWL class
33. if the OWL class has subclasses, add subclass values
34. if the OWL class has a comment, add the comment value
35. do iterate for each data of OWL Object Property
36. add domain and range attribute values according to OWL Object Property data
37. if the OWL Object Property has a comment, add the comment value
38. do iterate for each data of OWL Datatype
39. add values about domain and range
40. save OWL file
41. display OWL file through Protégé

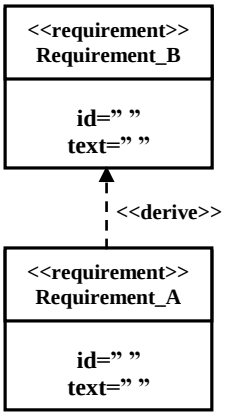
End

TABLE II. TRANSFORMATION RULE OF SYSML REQUIREMENT DIAGRAM INTO OWL ONTOLOGIES

SysML Requirement Diagram Element	SysML Requirement Diagram graphical symbol	Corresponding OWL Ontology element	OWL representation
a SysML package		an OWL ontology	`<owl:Ontology rdf:about="Package1">`

a SysML requirement
 	
an item in a requirement (id, text)
		an OWL class (an entity class)
		
		an attribute (data property)
			`<owl:Class rdf:ID="Requirement 1"/>`
			`</owl:Class>`
			
			`<owl:DatatypeProperty rdf:ID="id">`
			`<rdfs:domain rdf:resource="#Requirement 1"/>`
			`<rdfs:range rdf:resource="&xsd;string"/>`
			`</owl:DatatypeProperty>`
			
			`<owl:DatatypeProperty rdf:ID="text">`
			`<rdfs:domain rdf:resource="#Requirement 1"/>`
			`<rdfs:range rdf:resource="&xsd;string"/>`
			`</owl:DatatypeProperty>`

a requirement containment		a subclass	`<owl:Class rdf:ID="Requirement _B">`
			`<rdfs:subClassOf rdf:resource="Requirement_A"/>`
			`</owl:Class>`

a dependency notation		a relationship class (object property)	`<owl:ObjectProperty rdf:ID="derive">`
			`<rdfs:domain rdf:resource="#Requirement_A"/>`
			`<rdfs:range rdf:resource="#Requirement_B"/>`
			`</owl:ObjectProperty>`

V. Experimental Results

This research used `HSUVSpecification` model [2] of the SysML Requirement Diagram to do an experiment to evaluate the performance of the proposed model.

A. Example of SysML Requirement Diagram

Fig. 2 shows the `HSUVSpecification` Requirement Diagram, created with Visual Paradigm Modeler v16.1. This case example illustrates the use of SysML Requirement Diagrams for the development of car manufacturing, particularly specification of a Hybrid Sport Utility Vehicle (HSUV), which contains the following elements:

- A package, namely, HSUVSpecification package.
- Requirements such as Eco-Friendliness, Performance, Ergonomic, Qualification, Capacity, Zero-emissions, MaxAcceleration, Range and SizeSeatBelt.
- Requirement containments such as Emissions, Braking, FuelEconomy, OffRoadCapability, Acceleration, SafetyTest, CargoCapacity, FuelCapacity and PassengerCapacity.
- Dependencies between requirements such as copy, derive, trace and refine.
- Item id and text in Emissions, Zero-emissions, and SizeSealtBelt requirements.

B. The Produced Ontology

To the HSUVSpesification Requirement Diagram shown in Fig. 2, the S2OTransformation algorithm is applied. The produced ontology is shown in Fig. 3, 4, 5, and 6.

Fig. 3 shows the class and sub-class hierarchy, which is the result of the transformation of packages and requirements, shown in Fig. 2. The name of the package, i.e., HSUVSpecification, becomes the name of the ontology in Fig. 3.

Nine requirements become nine classes in ontology, i.e., Eco-Friendliness, Performance, Zero-emissions, MaxAcceleration, Range, Ergonomics, SizeSealtBelt, Qualification, and Capacity. Nine requirement containments become the nine subclasses, i.e., Emissions, Braking, FuelEconomy, Acceleration, SafetyTest, CargoCapacity, OffRoadCapability, Fuel Capacity, and PassengerCapacity subclasses.

The class and subclass hierarchy in the produced ontology is in accordance with the hierarchy of requirements and containments shown in Fig. 2.

Fig. 4 shows object properties as the results of the transformation process of the <<derive>>, <<trace>>, <<copy>>, and <<verify>> dependencies. Fig. 4 also shows the source (domain) and destination (range) of each dependency. For example, the domain (derive from) of the derive object property is class Range, while the range (towards) is class FuelCapacity and class FuelEconomy.

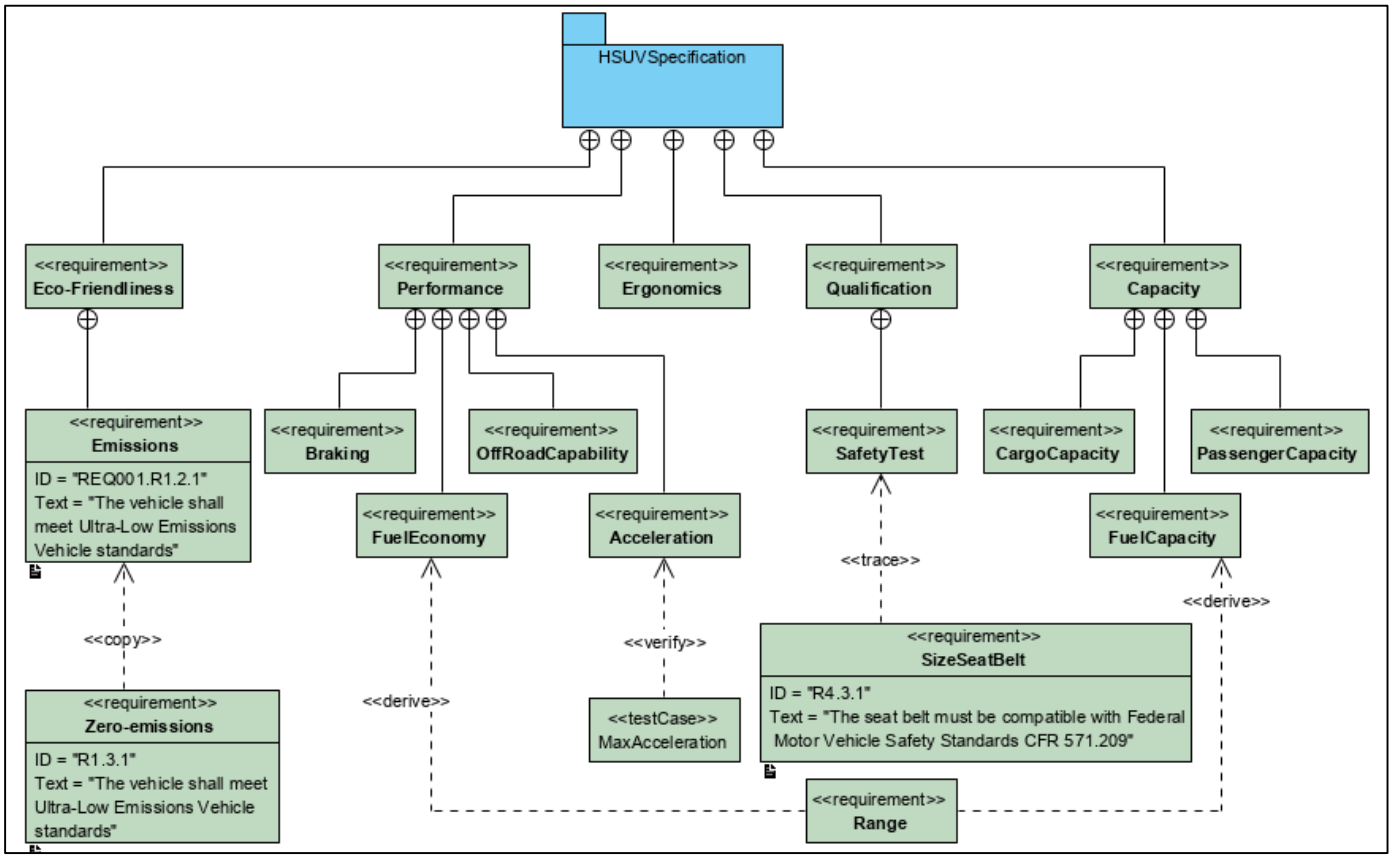


Fig. 2. `HSUVSpecification` Requirement Diagram Modeled using Visual Paradigm.

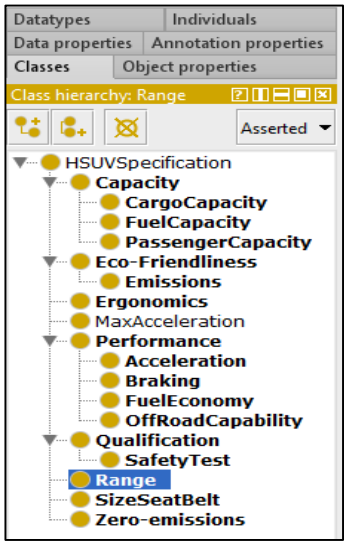


Fig. 3. The OWL Classes and Subclasses Produced from the Transformation.

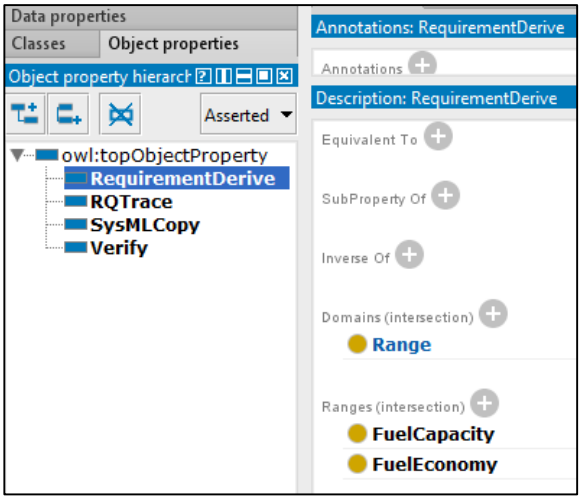


Fig. 4. The OWL Object Properties Produced from the Transformation.

Fig. 5 shows the produced data properties as the results of the transformation of item <<id>> and item <<text>>. The domains in Fig. 5 shows which requirements have the id and text attributes, while ranges show the data type of the attribute, namely string.

Fig. 6 shows the ontograf of the produced ontology, which is a depiction of the class hierarchy along with existing object properties. The straight blue lines indicate subclass (i.e., Eco-Friendliness, Performance, Qualification, Zero-emissions, Range, MaxAcceleration, Ergonomic, SizeSealtBelt, and Capacity), and the dashed lines indicate object properties (i.e., copy, derive, trace and verify). The experiment of case studies denotes that the proposed model works well, and can produce fully automatic ontological transformations.

C. Verification of Transformation Result

Testing the results of transformation is one of the crucial processes carried out to determine the performance of the proposed model that has been offered. The produced ontology file is tested for the accuracy of the design and its validity to the system design contained in the SysML Requirement Diagram. The testing of the proposed model is carried out by verifying the successful transformation of each element contained in the SysML Requirement Diagram into the appropriate ontology component. The testing is aimed to demonstrate the correctness of the proposed model and to show that a fully automatic ontology transformation can be achieved.

As shown in the above experiment, the produced ontology contains all elements contained in the SysML Requirement Diagram in Fig. 2. This research also does an experiment on 4 other Requirement Diagrams. To save space, this paper only displays the verification result of the transformation, as shown in Table III, which shows that all elements contained in the SysML Requirement Diagram case examples have been transformed into ontology components according to rules that have been defined in Table II.

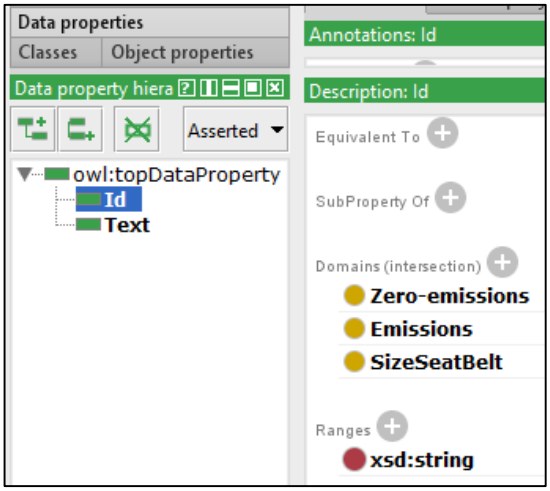


Fig. 5. The OWL Data Properties Produced from the Transformation.

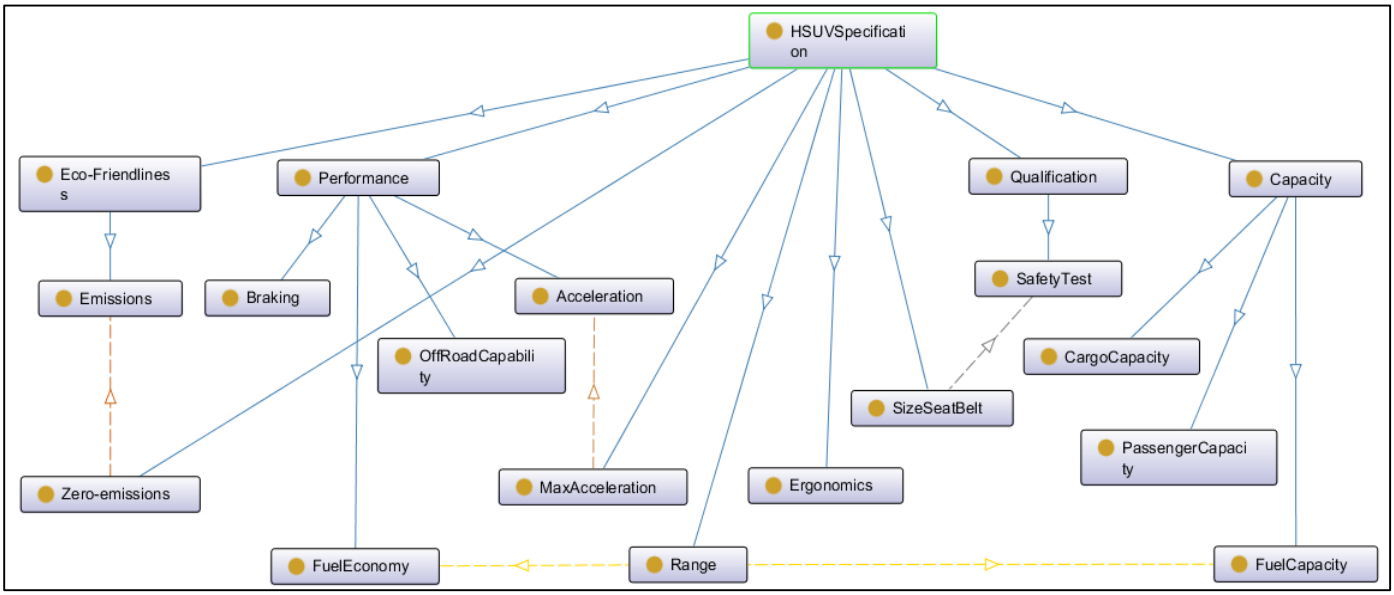


Fig. 6. OntoGraf of the produced ontology

TABLE III. VERIFICATION OF ELEMENT TRANSFORMATION RESULT

Case Study	Number of SysML Requirement Diagram Element					Number of OWL Ontology Element				
	Package	*Requirement*	*Containment*	*Dependency*	*Item*	*Ontology*	*Class*	*Subclass*	*Object Property*	*Data Property*
#1	1	9	9	5	2	1	9	9	5	2
#2		4		3			4		3	
#3		10		9			10		9	
#4		8	7		2		8	7		2
#5		4	1	3			4	1	3	

VI. CONCLUSION

In this paper, an automatic transformation of the SysML Requirement Diagram into OWL ontology has been proposed. From the experiment results, it can be concluded that the transformation of the SysML Requirement Diagram into OWL in RDF/XML syntax works well, and is able to produce an OWL ontology that can be displayed through Protégé. This is achieved using the transformation rules and algorithms that have been defined. The results of the transformation of several case studies have also been verified for correctness.

For further research, the proposed model will be developing and testing to transform other types of diagrams in SysML into OWL ontologies and then compare the results.

REFERENCES

[1] M. C. Hause, "The SysML Modelling Language," in Fifth European Systems Engineering Conference, 2006, vol. 9.

[2] Object Management Group, "An OMG Systems Modeling Language TM Publication OMG Systems Modeling Language v1.5," 2017. [Online]. Available: http://www.omg.org/spec/SysML/20161101. [Accessed: 28-Feb-2019].

[3] T. Weilkiens, Systems Engineering with SysML/UML Modeling, Analysis, Design. United States of America: Morgan Kaufmann Publishers, 2008.

[4] D. Wu, L. L. Zhang, R. J. Jiao, and R. F. Lu, "SysML-based design chain information modeling for variety management in production reconfiguration," J. Intell. Manuf., vol. 24, no. 3, pp. 575–596, 2013, doi: 10.1007/s10845-011-0585-6.

[5] S. C. Spangelo et al., "Model based systems engineering (MBSE) applied to Radio Aurora Explorer (RAX) CubeSat mission operational scenarios," in IEEE Aerospace Conference Proceedings, 2013, doi: 10.1109/AERO.2013.6496894.

[6] Z. Xu, Y. Ni, W. He, L. Lin, and Q. Yan, "Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach," World Wide Web, vol. 15, no. 5–6, pp. 517–545, Sep. 2012, doi: 10.1007/s11280-011-0147-z.

[7] H. Graves, "Integrating SysML and OWL," in CEUR Workshop Proceedings, 2009, vol. 529, no. Owled.

[8] E. K. Elsayed and D. R. Fathy, "Sign Language Semantic Translation System using Ontology and Deep Learning," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 1, pp. 141–147, 2020.

[9] S. M. Akhtar, M. Nazir, K. Saleem, H. M. Ul Haque, and I. Hussain, "An ontology-driven iot based healthcare formalism," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 2, pp. 479–486, 2020.

[10] A. Belghiat and M. Bourahla, "Transformation of UML models towards OWL ontologies," in 2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications, SETIT 2012, 2012, pp. 840–846, doi: 10.1109/SETIT.2012.6482025.

[11] Y. Traore and D. Bassole, "Multi-Label Classification using an Ontology," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 12, pp. 472–476, 2019.

[12] N. F. Noy and D. L. Mcguinness, "Ontology Development 101 : A Guide to Creating Your First Ontology," pp. 1–25, 2000.

[13] D. Gasvic, D. Djuric, V. Devedzic, and V. Damjanovic, "From UML to ready-to-use OWL ontologies," in Second lEEE International Conference on Intelligent Systems, 2004, no. June, pp. 485–490, doi: 10.1109/is.2004.1344798.

[14] M. Sadowska and Z. Huzar, "Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2," in Software Engineering: Challenges and Solutions, vol. 504, 2017, pp. 47–59.

[15] M. Sadowska and Z. Huzar, "Representation of UML Class Diagrams in OWL 2 on the Background of Domain Ontologies," e-Informatica Softw. Eng. J., vol. 13, no. 1, pp. 63–103, 2019, doi: 10.5277/e-Inf190103.

[16] A. H. Khan and I. Porres, "Consistency of UML class, object and statechart diagrams using ontology reasoners," J. Vis. Lang. Comput., vol. 26, pp. 42–65, 2015, doi: 10.1016/j.jvlc.2014.11.006.

[17] Il-Woong Kim and Kyong-Ho Lee, "A Model-Driven Approach for Describing Semantic Web Services: From UML to OWL-S," lEEE Trans. Syst. Man, Cybern. Part C (Applications Rev., vol. 39, no. 6, pp. 637–646, 2009, doi: 10.1109/tsmcc.2009.2023798.

[18] H. Wang, V. Thomson, and C. Tang, "Change propagation analysis for system modeling using Semantic Web technology," Adv. Eng. Informatics, vol. 35, pp. 17–29, Jan. 2018, doi: 10.1016/j.aei.2017.11.004.

[19] B. Bouihi and M. Bahaj, "An UML to OWL based approach for extracting Moodle's Ontology for Social Network Analysis," Sci. - Procedia Comput. Sci., vol. 148, pp. 313–322, 2019, doi: 10.1016/j.procs.2019.01.039.

[20] H. A. Salah, "Ontology development (OWL&UML) methodology of web-based Decision Support System for water management," in Proceedings of the 2014 6th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2014, 2014, pp. 11–22, doi: 10.1109/ECAI.2014.7090217.

[21] J. I. Olszewska, "UML Activity Diagrams for OWL Ontology Building," in Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), 2015, vol. 2, pp. 370–374.

[22] J. I. Olszewska, R. Simpson, and T. L. Mccluskey, "Dynamic OWL Ontology Design Using UML and BPMN," in Proceedings of the International Conferenceon Knowledge Engineering and Ontology Development (KEOD-2014), 2014, pp. 436–444, doi: 10.5220/0005159204360444.