



## **An Extended Rule of the SysML Requirement Diagram Transformation into OWL Ontologies**

**Ahmad Ashari<sup>1\*</sup>    Anny Kartika Sari<sup>1</sup>    Helna Wardhana<sup>1,2</sup>**

<sup>1</sup> *Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia*

<sup>2</sup> *Department of Informatics, Universitas Bumigora, Lombok, Indonesia*

\* Corresponding author's Email: [ashari@ugm.ac.id](mailto:ashari@ugm.ac.id)

---

**Abstract:** The System Modeling Language (SysML) used the Requirement Diagram to model non-functional requirements, such as response time, size, or system functionality, which cannot be accommodated in the Unified Modeling Language (UML). SysML Requirement Diagram, in its implementation, integrates with several diagrams describing the requirements, which are referred to as additional elements. The absence of transformation rules for these additional elements to become OWL ontology causes difficulties in reading, understanding, and tracking the requirements. In this research, an extended rule of the Requirement Diagram transformation is proposed to solve the problems. First, some transformation rules are defined to make requirements easier to trace and realize the ontology generation's automatic transformation. Second, the time required during transformation processing to prepare and generate the OWL file shows the proposed model's performance. The ontology components produced from this research, such as class, subclass, object property, and data property, can be viewed in Protégé.

**Keywords:** SysML diagram, Requirement diagram, Ontology, OWL, Transformation.

---

### **1. Introduction**

The Requirement Diagram is 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.

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 SysML benefits [1]. SysML Requirement Diagram, in its implementation, integrates with several diagrams in describing the system's requirements, such as activity, internal block, interaction, state machine, and use case. These diagrams become additional elements in the requirement diagram. The absence of transformation rules for these additional elements causes difficulties in understanding and tracking the requirements.

An ontology is generally defined as an explicit and formal representation of knowledge [2, 3]. The use of ontology enables system engineers to model metadata concepts and semantic contexts that can be used in model inference and transformation rulemaking [4-5]. Furthermore, the relevant concepts of a domain are reflected by the ontology [6]. Therefore, the transformation of the SysML Requirement Diagram into an ontology is needed. In [7], we proposed an automatic transformation of the SysML Requirement Diagram into the Web Ontology Language (OWL) ontology. Using the predefined transformation rules and algorithm, the transformation process from elements of SysML Requirement Diagram into ontology components runs well. It can produce an OWL ontology displayed through Protégé.

Although the transformation from SysML to OWL has been proposed by [8] and [6], the transformation is still done manually. In this research, an extended rule of the SysML Requirement Diagram transformation is proposed. First, some

transformation rules are defined, not only to complete the previous rules but also to make requirements easier to trace and realize the automatic transformation. Second, the time required during transformation processing for the generation of the OWL file shows the proposed model's performance. The main difference between our proposed model with the existing approaches is that the model suggests an automatic transformation from SysML Requirement Diagram into OWL. Through automatic transformation, this research is expected to increase the use of requirement diagram to support object-oriented system modelling, which models the system's non-functional requirements and expresses the structure and behavior of a system.

The rest of this paper is organized as follows. The related researches with this study are explained in Section 2. Section 3 presents the transformation rules and algorithms. Section 4 describes the results and discussion of the test. In section 5, the conclusion of this work is presented.

## 2. Related work

Research on the transformation of SysML Diagrams into OWL has been proposed by [8] and [6], but the transformation process is still done manually. Research by [8] uses several diagrams in SysML to analyze and present scenarios about system model change from a formal perspective, namely how to add, remove, and modify model elements in response to changes in a system's design. Ontology is applied to formalize transformations in the influence of the relationship between the system model's requirements, behavior, and structure. Another research was conducted by [6], translating the block diagram into OWL, which created an OWL knowledge base that could represent a system's structural design information.

Several other researchers [9-15] have proposed UML translation models into OWL automatically using the class diagram. The aim of [9] 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 from UML. Research conducted in [10] uses eXtensible Stylesheet Language (XSL) style sheets to transform UML models, producing applications that automatically transform class diagrams into OWL. 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 [11]. Other research is carried out in [12-13], which has revised the transformation rules and proposed the verification rules to check the UML class diagram's suitability with the ontological domain in OWL. An automatic translation of UML into OWL is proposed in [14] through an approach that analyzes UML models' consistency and satisfaction using logical reasoning for OWL. The design and software development that uses a model-based approach to produce OWL-based Web Service ontologies from the UML model is proposed [15].

## 3. Transformation rules and algorithm

This section presents the extent of the transformation rules and algorithms [7] used to change the Requirement Diagram into an OWL.

### 3.1 Transformation rules for additional elements

This research uses a set of rules to transform the SysML Requirement Diagram into ontology, as described in [7]. The transformation rules for most of the SysML Requirement Diagram elements have been discussed in [7]. This study proposes rules for <<testcase>> and notes, as shown in Table 1. The <<testcase>> element is transformed to become a class like a requirement, but there is a label given to add remark that the class is a <<testcase>>. In addition to notes, <<rationale>> and <<problem>> are also included in the notes group. Notes are transformed into the label in the ontology that contains information or notes like written in the symbols.

The rules are also constructed to transform elements from other diagrams, hereinafter referred to as additional elements used in the SysML Requirement Diagram, as shown in Table 2. Several other diagrams are used in conjunction with the SysML Requirement Diagram to describe the system's non-functional requirements [16]. The diagrams are activity, internal block, state machine, interaction, and use case. External documents or artifacts include additional elements used as a reference for tracking by the SysML Requirement Diagram element. All additional elements are transformed into class in ontology by explaining the label to differentiate from the class formed from <<requirement>>.

### 3.2 Algorithm

This research still uses the transformation algorithm *S2OTransformation* [7] with several

Table 1. Transformation rules of SysML requirement diagram into OWL ontologies

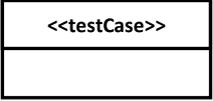
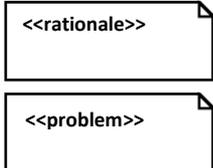
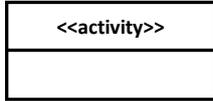
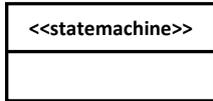
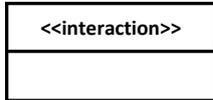
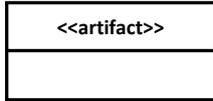
SysML requirement diagram element	SysML requirement diagram graphical symbol	Corresponding OWL ontology component	OWL representation	Additional remark on the label
testCase		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="TestCase-..." rdf:label="testCase"/&gt;</code>	testCase
Notes/comments		a label	<code>&lt;rdfs:comment&gt;..... &lt;/rdfs:comment&gt;</code>	According to the content of notes in the graphical symbol

Table 2. Transformation rules of additional elements in SysML requirement diagram into OWL ontologies

Additional elements from other diagrams	SysML requirement diagram graphical symbol	Corresponding OWL ontology component	OWL representation	Additional remark on the label
activity from Activity diagram		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="Activity-..." rdf:label="Activity Diagram"/&gt;</code>	From Activity diagram
block from Internal block diagram		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="Block-..." rdf:label="Internal Block Diagram"/&gt;</code>	From Internal block diagram
statemachine from Statemachine diagram		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="Statemachine-..." rdf:label=" Statemachine Diagram"/&gt;</code>	From Statemachine diagram
interaction Interaction diagram		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="Interaction-..." rdf:label=" Interaction Diagram"/&gt;</code>	From Interaction diagram
useCase from useCase diagram		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="useCase-..." rdf:label="useCase Diagram"/&gt;</code>	From useCase diagram
document from An external document		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="Document-..." rdf:label="External Document"/&gt;</code>	Refer to an external document
artifact from An external document		an OWL class (an entity class)	<code>&lt;owl:Class rdf:about="Artifact-..." rdf:label=" Artifact/External Document"/&gt;</code>	Refer to an artifact/ An external document

modifications to transform the additional elements became the classes by making classname and labelname to accommodate that

transformation. For more details, the fundamental steps for transforming the SysML Requirement Diagram into an OWL file are presented in the

flowchart in Fig. 1 and Fig. 2. We divide the flowchart drawing into two, namely first to describe the preparation of the OWL file generation, as shown in Fig. 1, and secondly, about the generation of the OWL file itself, as shown in Fig. 2.

#### 4. Result and discussion

This study uses the SysML Requirement Diagram model [17] to prove the additional elements' transformation into classes.

##### 4.1 Example of SysML requirement diagram

Fig. 3 shows the Requirement Diagram, created with Visual Paradigm Modeler v16.2. The requirement diagram is showing the traceability of the *Maximum Acceleration* requirement. The traceability to a text-based requirement includes the design elements to satisfy it, other requirements derived from it, and a testcase to verify it. The rationale for the deriveReq relationship based on parametric

analysis is also shown. This case example illustrates the use of all the SysML Requirement Diagram elements and the additional elements to represent all transformation rules' application to all components in the ontology. Fig. 3 contains the following elements:

- ❖ requirements such as Maximum Acceleration and Engine Specification
- ❖ requirement containment such as Engine Power
- ❖ dependencies between requirements such as trace, verify, satisfy, derive and refine,
- ❖ item id and text in Maximum Acceleration and Engine Power requirements
- ❖ testCase such as Max Acceleration
- ❖ note such as rationale
- ❖ the additional elements such as activity, block, and artifact

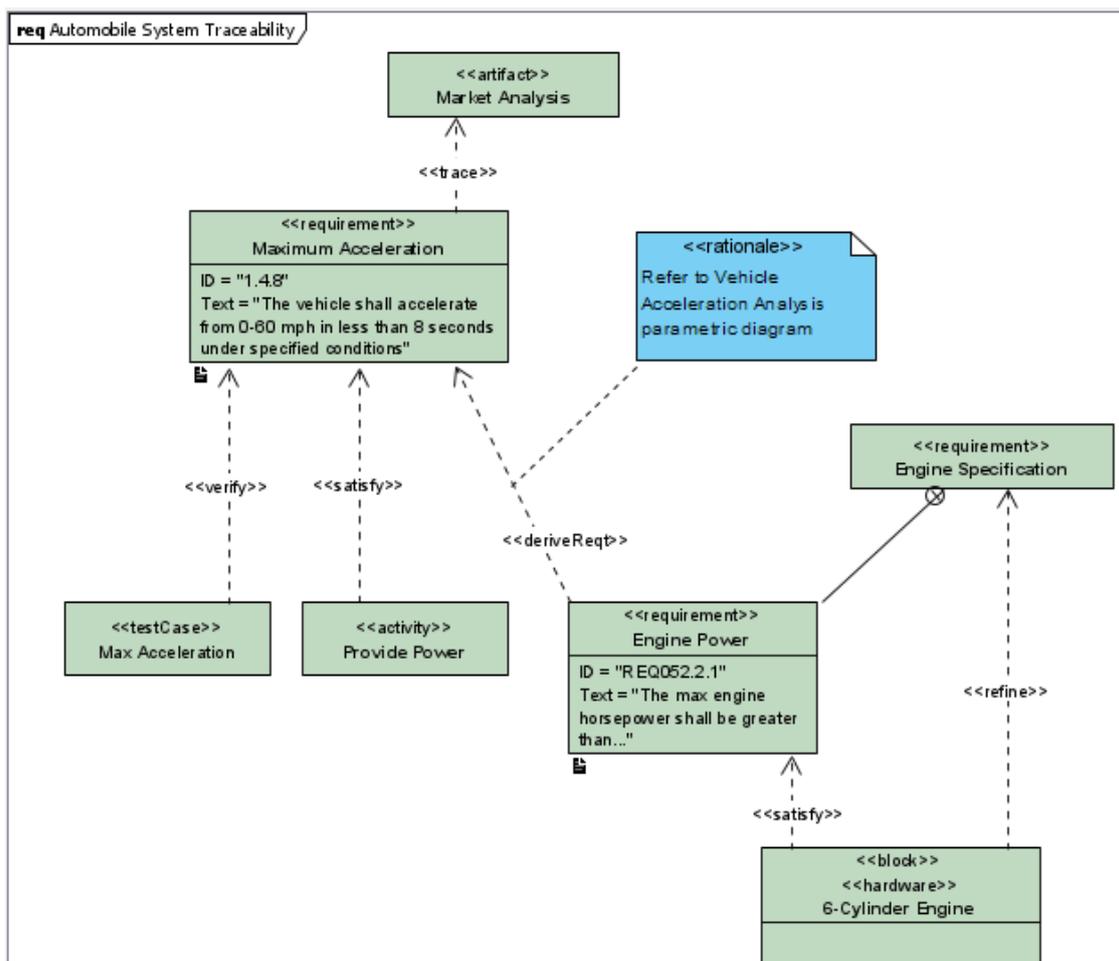


Figure. 3 Requirement diagram modeled using the visual paradigm

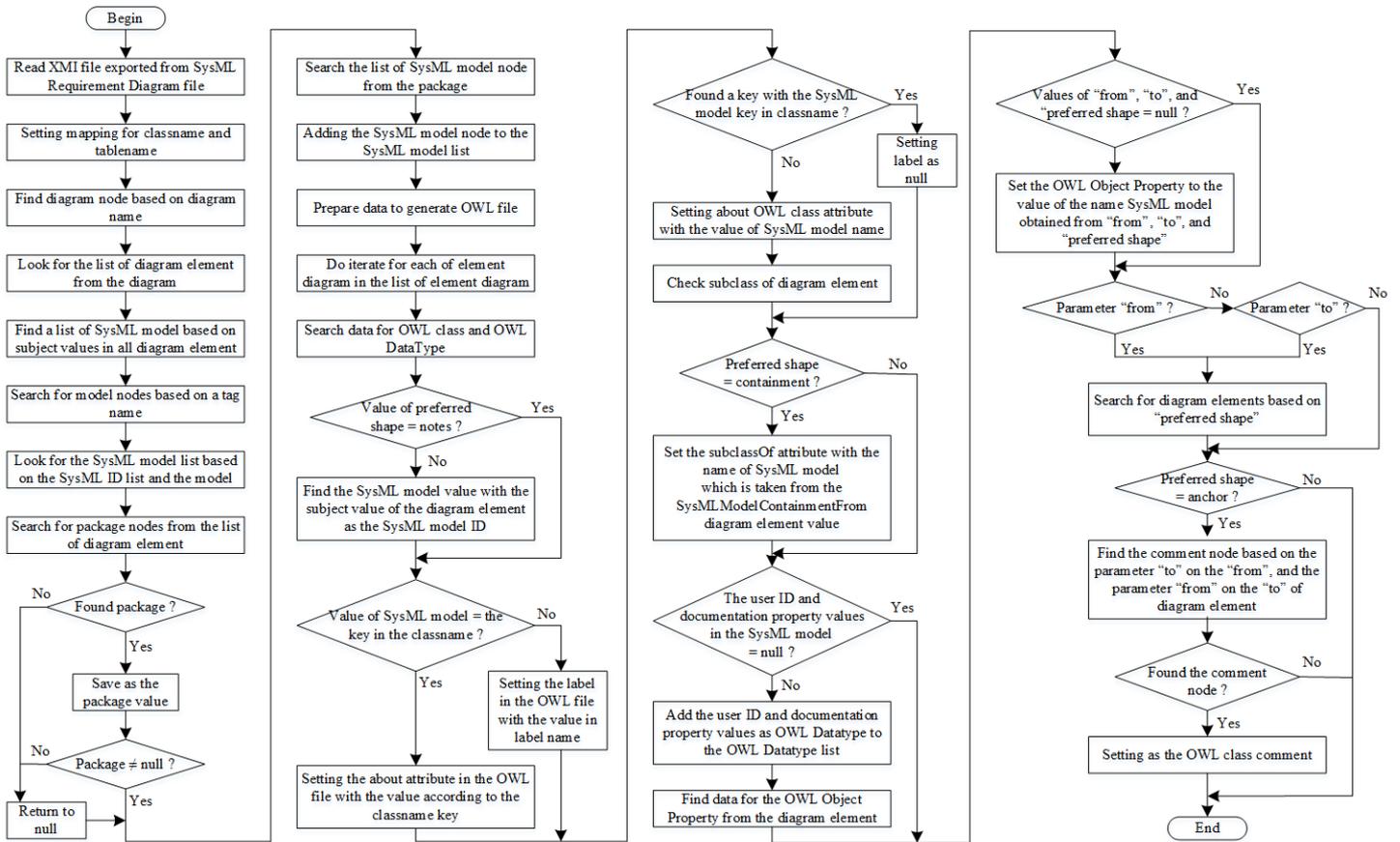


Figure. 1 Flowchart for preparation of OWL file generation

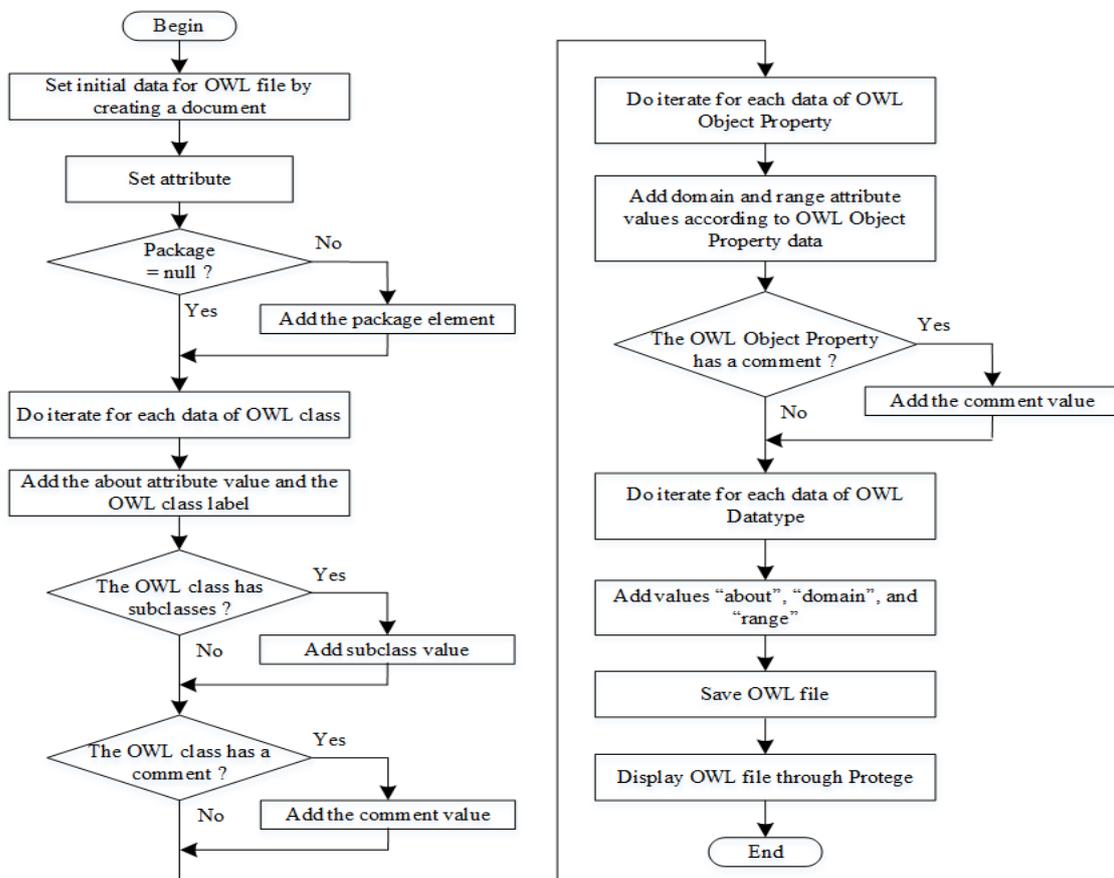


Figure. 2 Flowchart for generating of OWL file

### 4.2 The resulted ontology

According to the Requirement Diagram shown in Fig. 3, the S2OTransformation algorithm [7] is applied. The resulted ontology is shown in Fig. 4, Fig. 5, and Fig. 6.

Fig. 4 and Fig. 5 show the class hierarchy, which is the product of the transformation of requirements and the additional element, as shown in Fig. 3. Labeling for classes that are transformed from the additional elements such as activity diagram and internal block diagram is also visible.

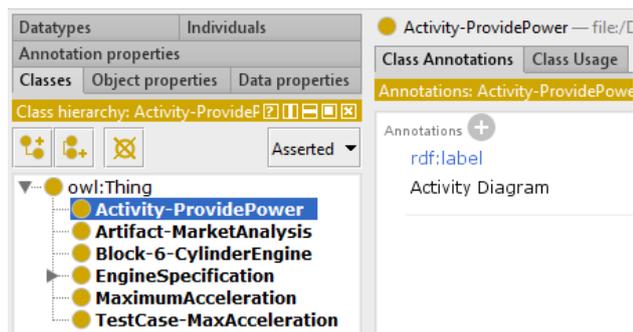


Figure. 4 The OWL class resulted from the transformation for activity

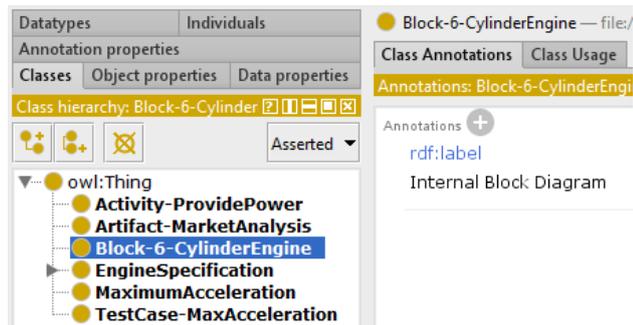


Figure. 5 The OWL class resulted from the transformation for block

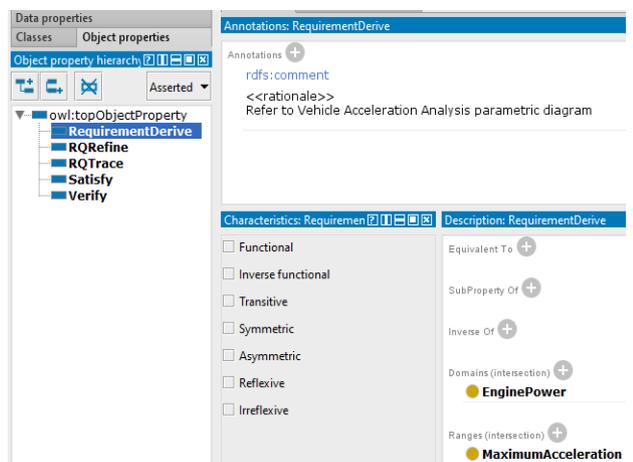


Figure. 6 The OWL object properties resulted from the transformation

Fig. 6 shows object properties as the results of the transformation process of the <<derive>>, <<trace>>, <<satisfy>>, <<refine>>, and <<verify>> dependencies. Fig. 6 also shows the source (domain) and destination (range) of each dependency. For example, the domain (derive from) of the derive object property is class Engine Power, while the range (towards) is class Maximum Acceleration. The label, as the results of the transformation process of the rationale, is also shown.

### 4.3 Algorithm efficiency test

The accuracy and speed of the transformation process of all SysML Requirement Diagram elements into an OWL ontology can show the good performance of the proposed model. Ten SysML Requirement Diagrams created with Visual Paradigm tools were tested in the experiments, as listed in Table 3. This study uses ten examples of SysML Requirement Diagrams, taken from several references regarding using the SysML Requirement Diagram [16–20]. The reason for choosing this case example is because these examples can represent the entire transformation process of all the elements of the SysML Requirement Diagram and the additional elements.

Table 3 shows the number of each SysML Requirement Diagram element contained in each case study, namely requirement (Req), containment (Cont), dependencies such as trace (T), copy (C), derive (D), verify (V), refine (R), and satisfy (S). Table 3 also shows the number of items, testcase, notes, and the additional elements related to the SysML Requirement Diagram, such as activity diagram, internal block diagram, state machine diagram, interaction diagram (Intr), use case diagram, artifact, and external document (Doc).

The verification result of the transformation, as shown in Table 3, shows that all elements in the SysML Requirement Diagram case examples have been transformed into ontology components. According to the rules defined, the transformation results are in the form of OWL files containing classes, subclasses, object properties, data properties, and labels. The transformation process's success is 100% of all the example diagrams show that a fully automatic ontology transformation can be achieved.

The algorithm's efficiency was tested to determine the transformation algorithm's actual

execution time of the SysML Requirement Diagram with different diagram sizes. The size of each diagram ( $N$ ) is determined based on its many elements, namely the number of requirements, the number of containments, the number of dependencies, the number of items, the number of testcases, the number of notes, and the number of other diagram elements. Eq. (1) shows the calculation of the size of the diagram.

$$N = N_R + N_C + N_D + N_I + N_T + N_N + N_E \quad (1)$$

where:

$N$  = Diagram size

$N_R$  = Number of *requirement*

$N_C$  = Number of *containment*

$N_D$  = Number of *dependency*

$N_I$  = Number of *item*

$N_T$  = Number of *testcase*

$N_N$  = Number of *note*

$N_E$  = Number of *other diagram elements*

Furthermore, the execution time will be calculated from the proposed model algorithm routine that runs the SysML Requirement Diagram. To see the algorithm's effectiveness, the calculation of time is divided into the execution time for the preparation of generating the OWL file and the execution time for the generation OWL file. The calculation of the two types of time is carried out to see which algorithm's performance during the process, which takes more time. Time measurement is done empirically. We use the millisecond (ms) as a unit of time in writing processing time calculations. The results of calculating the execution time for the entire SysML Requirement Diagram are shown in Table 4. Table 4 shows that the processing time required is based on each diagram's size in each case study to prepare and generate the OWL file. Based on experiments conducted on ten examples, as shown in Table 4, each diagram's processing time is greatly influenced by several things: the number of requirements, containment, and diversity of elements in each diagram.

The preparation stage of creating an OWL file takes a longer time than it takes to generate an OWL file, although the time difference is not very significant. The time required to prepare the OWL file is due to several processes that must be carried out in the execution algorithm. These processes are:

- Mapping `classname` and `tablename`

- Finding the SysML model
- Prepare data to generate an OWL file.
- Checking subclass of diagram element
- Searching object property
- Searching property data.

Table 4 shows that the greater the number of requirements and containment in a diagram, the longer it will take for the preparation process to produce an OWL file. From the ten examples used, diagram example #7 takes the longest time to prepare and generate the OWL file, namely 315 ms for preparation and 72 ms to generate the OWL file. Diagram example #7 has the highest number of requirements and containment and contains several additional elements. Example diagrams #6 and #9 have the fastest processing time to prepare to generate OWL files, namely 250 for #6 and 248 for #9. These two examples have the smallest number of requirements for other examples. The time to generate OWL files is influenced by the many types of diagram elements involved. The more types of elements in a diagram, the longer it will generate the OWL file. Example of diagram #1 has the fastest time to generate OWL files, namely 60 ms, because this example only contains `item` elements other than `requirement` and `containment`.

## 5. Conclusion

In this paper, the automatic transformation from SysML Requirement Diagram into OWL ontology, especially for additional elements, has been fully achieved. The additional elements are activity diagrams, internal block diagrams, state machine diagrams, interaction diagrams, use case diagrams, artifacts, and external documents. Based on the experimental results, it can be concluded that the extent of transformation rules and algorithms can produce an ontology file.

The number of all elements in a SysML Requirement Diagram, especially the additional elements, greatly affects the length of time it takes to transform the diagram into an OWL ontology. The experimental results show that the actual execution time needed to prepare the OWL file generates longer than the time required to create OWL files. The number of requirements and the amount of containment determines the length of processing time needed to prepare the OWL file generation. The more the number of requirements and containment, the longer it will take for processing. Meanwhile, the time required to generate the OWL file is influenced by the many additional elements in the SysML Requirement Diagram. The more types of elements involved, the longer it will take.

Table 3. Transformation of all model elements into ontology

Case study	Number of SysML requirement diagram elements											Number of additional elements							Number of OWL ontology components					The percentage of successful transformation (%)
	Req	Cont	Dependency						Item	Test case	Notes	Activity	Block	State machine	Intr	useCase	Artifact	Doc	Class	Subclass	Object property	Data property	Label (notes)	
			T	C	D	V	R	S																
#1	9	7	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	9	7	0	2	0	100
#2	5	9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	5	9	0	2	0	100
#3	10	0	0	0	9	0	0	0	0	0	3	0	0	0	0	0	0	0	10	0	1	0	3	100
#4	2	2	0	0	2	0	0	1	4	0	2	0	1	0	0	0	0	0	3	2	2	2	3	100
#5	2	1	1	0	1	1	1	2	2	1	1	1	1	0	0	0	1	0	6	1	5	2	5	100
#6	2	0	0	0	1	1	1	1	0	1	0	0	1	0	0	1	0	0	5	0	4	0	3	100
#7	9	9	0	0	9	1	0	0	1	1	3	0	0	0	0	0	0	0	10	9	2	2	4	100
#8	4	0	1	0	3	0	0	0	0	0	1	0	0	0	0	0	1	0	5	0	2	0	2	100
#9	2	0	0	0	1	1	2	0	0	0	0	0	0	1	1	1	0	0	5	0	3	0	3	100
#10	14	0	0	0	12	0	0	0	14	0	0	0	0	0	0	0	0	0	14	0	1	2	0	100

Table 4. Transformation processing time

Case study	Diagram size	Number of elements							The processing time for preparation of generation OWL file (ms)	The processing time for generation OWL file (ms)
		Req	Cont	Dep	Item	Testcase	Notes	Additional elements		
#1	18	9	7	0	2	0	0	0	301	60
#2	15	5	9	0	1	0	0	0	304	63
#3	22	10	0	9	0	0	3	0	310	62
#4	14	2	2	3	4	0	2	1	295	65
#5	16	2	1	6	2	1	1	3	298	69
#6	9	2	0	4	0	1	0	2	250	61
#7	33	9	9	10	1	1	3	0	315	72
#8	10	4	0	4	0	0	1	1	256	62
#9	9	2	0	4	0	0	0	3	248	63
#10	40	14	0	12	14	0	0	0	312	64

The ontology components resulted from this research, such as class, subclass, object property, and data property, can be viewed through Protégé. Based on the predefined transformation rules, through automatic transformation, this research is expected to increase the use of requirement diagrams to support object-oriented system modeling, which not only models the system's non-functional requirements but also expresses the structure and behavior of a system.

For further research, this SysML Requirement Diagram transformation tool will be further developed and analyzed to transform other types of diagrams in SysML and integrate them with diagrams in Unified Modeling Language (UML).

### Conflicts of Interest

The authors declare no conflict of interest.

### Author Contributions

The contributions by the authors for this research are as follows: conceptualization, methodology, validation, formal analysis, Ahmad Ashari, and Anny Kartika Sari; software, investigation, resources, data curation, writing—original draft preparation, project administration, Helna Wardhana; writing—review and editing, visualization, supervision, Ahmad Ashari, and Anny Kartika Sari.

### Acknowledgments

This research was supported by Rekognisi Tugas Akhir (RTA) program from the Research Directorate of Universitas Gadjah Mada through contract number 2488/UN1.P.III/DIT-LIT/PT/2020.

### References

- [1] 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.
- [2] H. Wardhana, Y. Suyanto, and S. Priyanta, "Utilization Of Query Rewriting Over Ontology Change : A Review", *Int. J. Sci. Technol. Res.*, Vol. 7, No. 3, pp. 117–124, 2018.
- [3] A. K. Sari, "Mapping of Change Operations from Gene Ontology into Medical Subject Headings", *International Journal of Intelligent Engineering and Systems*, Vol. 13, No. 4, pp. 44–55, 2020, doi: 10.22266/ijies2020.0831.05.
- [4] A. Makwana, "A Known in Advance, What Ontologies to Integrate ? For Effective Ontology Merging Using K-means Clustering", *International Journal of Intelligent Engineering and Systems*, Vol. 11, No. 4, pp. 72–87, 2018, doi: 10.22266/ijies2018.0831.08.
- [5] F. Mary and H. Fernandez, "A Novel Analysis and Prediction of Students ' Behaviour Using Semantic Similarity-Based Improved J48 IL Algorithm in Personalized Library Ontology", *International Journal of Intelligent Engineering and Systems*, Vol. 11, No. 5, pp. 173–182, 2018, doi: 10.22266/ijies2018.1031.16.
- [6] H. Graves, "Integrating SysML and OWL", in *CEUR Workshop Proceedings*, 2009, Vol. 529, No. Owled.
- [7] H. Wardhana, A. Ashari, and A. K. Sari, "Transformation of SysML Requirement Diagram into OWL Ontologies", *Int. J. Adv. Comput. Sci. Appl.*, Vol. 11, No. 4, pp. 106–114, 2020.
- [8] 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, 2018, doi: 10.1016/j.aei.2017.11.004.
- [9] 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, 2012, doi: 10.1007/s11280-011-0147-z.
- [10] A. Belghiat and M. Bourahla, "Transformation of UML models towards OWL ontologies", In: *Proc. of 2012 6th International Conf. on Sciences of Electronics, Technologies of Information and Telecommunications, SETIT 2012*, pp. 840–846, 2012, doi: 10.1109/SETIT.2012.6482025.
- [11] D. Gasvic, D. Djuric, V. Devedzic, and V. Damjanovic, "From UML to ready-to-use OWL ontologies", In: *Proc. of Second IEEE International Conf. on Intelligent Systems*, 2004, No. June, pp. 485–490, doi: 10.1109/is.2004.1344798.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] Il-Woong Kim and Kyong-Ho Lee, “A Model-Driven Approach for Describing Semantic Web Services: From UML to OWL-S”, *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.)*, Vol. 39, No. 6, pp. 637–646, 2009, doi: 10.1109/tsmcc.2009.2023798.
- [16] OMG, “OMG Systems Modeling Language”, 2017, [Online]. Available: <http://www.omg.sysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf%5Chttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:OMG+Systems+Modeling+Language#8>.
- [17] S. Friedenthal, A. Moore, and R. Steiner, “An Automobile Example Using the SysML Basic Feature Set”, in *A Practical Guide to SysML*, Elsevier, 2015, pp. 53–81.
- [18] S. Friedenthal, A. Moore, and R. Steiner, *Water Distiller Example Using Functional Analysis*. 2015.
- [19] S. Friedenthal, A. Moore, and R. Steiner, *Modeling Text-Based Requirements and Their Relationship to Design*. 2015.
- [20] T. Weilkiens, *Systems Engineering with SysML/UML Modeling, Analysis, Design*. The OMG Press, 2008.