A Novel Approach to Maintain Traceability between Safety Requirements and Model Design

Qian Wang[†], Jing Liu[†*], John Zhang[‡*], Hui Dou[‡], Haiying Sun[†], HongTao Chen[‡], Xiaohong Chen[†], Jifeng He[†]
[†]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
[‡]Huawei Technology, Shanghai, China
[*]Corresponding authors: Jing Liu (Email: jliu@sei.ecnu.edu.cn), John Zhang (Email: john.zhangyh@huawei.com)

Abstract—**One of the major challenges confronting System Modeling Language(SysML) is that it cannot always provide verifiable guarantees of formalization and rigorousness. To verify model designs, the research of transformation from SysML to ontology emerges because of ontology's formal standards and verifiability obtained by ontology reasoners. However, existing transformation approaches are mostly limited to a single view without traceability or lack a clear process so that it can't be automated. In this paper, we propose a novel approach to maintain precious traceability between requirements and model multi-views design based on ontology. In addition, our approach contains a normative process of ontology building in support of an automated implementation. We use this approach to obtain the ontology of a safety-critical system and carry out the ontology evaluation experiment, whose results demonstrate the feasibility and efficiency of our approach.**

Index Terms—**Knowledge Engineering, Traceability, SysML, Description Logic, Ontology, Model Transformation**

I. Introduction

As a knowledge representation tool, System Modeling Language (SysML) has been used in many safety-critical systems. In the most common SysML usage mode, i.e., *SysML-as-Pretty-Pictures*, it uses various diagrams to express the requirements, structure, and behavior of the system. Unfortunately, this mode lacks complete formalization and rigorousness, and users pay relatively little attention to the well-formedness of SysML and its underlying simulatable and executable semantics [1]. Hence, the generated models are difficult to drive dynamic behavior simulations to confirm that the system behavior satisfies the safety requirements. The challenge will bring in more time and labor costs [2].

To address the aforementioned challenge, the research of transformation from SysML to ontology and verification based on ontology emerged. Ontology is extended based on description logic which has formal standards; therefore, ontology reasoners can find errors in ontology rules and facts while drawing inferences [4]. If a SysML model is transformed into an ontology with the same semantics, we can conduct inferences to verify the design of models.

In this paper, to address the problem, we propose a novel approach to maintain precious traceability between requirement and model design based on ontology. The establishment of maintaining traceability from risk to overall safety requirements, to safety-critical system component design, is of great significance for behavior simulations [7]. Specifically, we first construct the high-level ontology to formally represent concept elements and semantics of SysML using Web Ontology Language 2(OWL 2) [8]. Then, models' instances are added into the high-level ontology to generate low-level ontology, which can be applied to verify and query using well-developed OWL reasoners. Finally, we conduct experiments on a real-world safety-critical system. The experimental results show that the ontology generated by the proposed approach is of good overall quality, and there are no pitfalls that affect the consistency, reasoning, and applicability of ontology.

DOI reference number: 10.18293/SEKE2022-140

The contributions of this paper are threefold: (1)an innovative approach to maintaining traceability using a phased transformation, (2)a normative process of ontology building in support of an automated implementation, (3)an experimental evaluation of a real-world safety-critical system with 23 requirements showing feasibility and efficiency of our approach.

II. Transformation Approach

A. Ontology Generation

Transformation in this paper concentrates on SysML's four semantic dimensions: (1) requirements, (2) structure, (3) behavior, and (4) state. After the reference to [9] where shows a normative ontology building process to guarantee efficiency, the transformation consists of six steps. The first four steps create the upper-level ontology(written $\bar{O}$ for ease of understanding), and the last two steps derive the lower-level ontology(written o) of a specific model m from $\bar{O}$. The following are the detailed steps.

Step 1. Determine the domain and scope of the ontology. The way of starting the development of an ontology is to answer several questions:

- What is the domain? SysML's four semantic dimensions.
- For what we are going to use? Complete and correct expression of the domain's semantics, and then the users can query and infer about m from o.
- What types of information can the ontology provides? Models' basic and traceability information.
- Who are the ontology's users? Software engineers.

Step 2. Enumerate important terms. This step focuses on writing down important terms. The semantics of a dimension

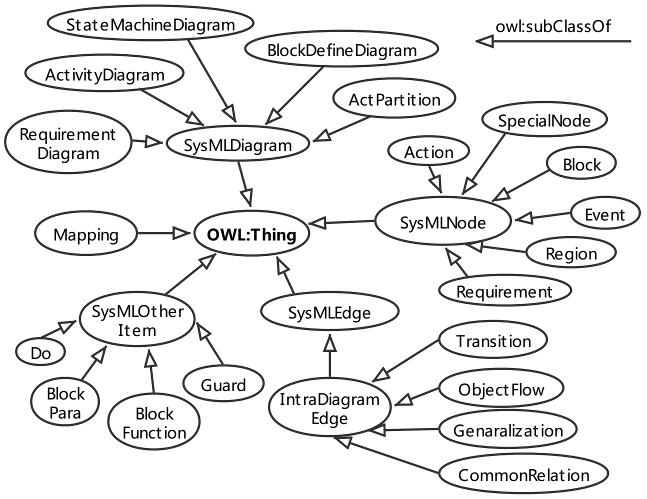

Fig. 1. Main parts of class hierarchy

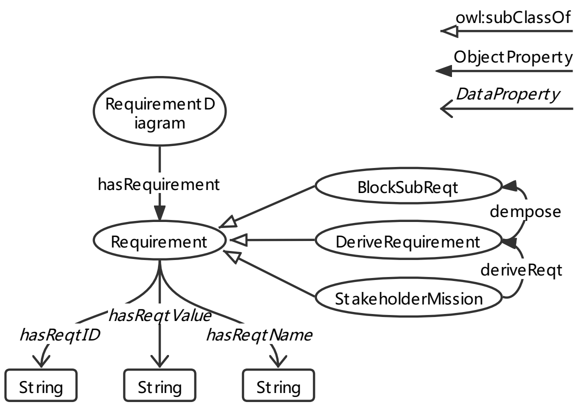

Fig. 2. Semantic representation of req

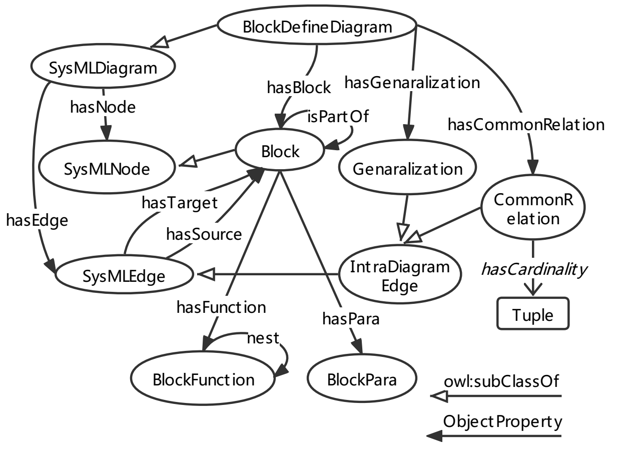

Fig. 3. Semantic representation of bdd

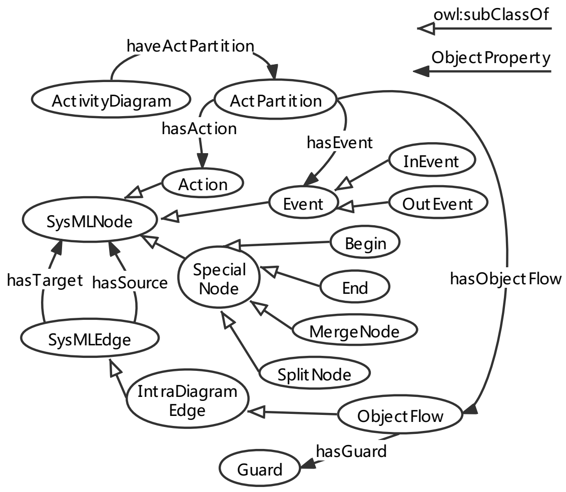

Fig. 4. Semantic representation of act

are usually visualized as a kind of diagram. Each diagram contains nodes and edges which represent various components of SysML and relationships between components. In addition to the vertical semantics of each diagram, there are the horizontal semantics between diagrams expressed through the cross-cutting mechanism. [7] extends cross-cutting mechanism with a data structure called Mapping for a more detailed traceability information model(TIM). To sum up, important terms include SysML diagram, node in diagram, edge in diagram, mapping, and other item of SysML component.

Step 3. Define the classes and their hierarchy This step first defines the most common concepts in the domain, and then specializes those concepts, i.e., a top-down development process [9]. As shown in Fig. 1, upper-level ontology $\bar{O}$ contains five main classes: `SysMLDiagram`, `SysMLEdge`, `SysMLNode`, `SysMLOtherItem`, and `Mapping` that implements traceability.

Step 4. Define the properties of classes. The classes alone can't provide enough information to answer the competency questions in Step 1. `ObjectProperty` and `DataProperty` are required to describe the internal structure of concepts. Fig. 2, 3, 4, 5 show the classes and properties that enable the semantics of requirement diagram(req), block definition diagram(bdd), activity diagram(act), and state machine diagram(stm), respectively.

We use `ObjectProperty` to represent edges with simple semantics e.g., `isPartOf` indicates edges in bdd that represent combination or aggregation. If using such a restriction on edges with complex semantics, such as transitions, we would need OWL Full. To keep the ontology as an OWL 2 DL ontology, transition is defined as a subclass of SysMLEdge, as shown in Fig. 1. Then transitions' extra semantics are represented by subclasses of `SysMLOtherItem`, then use the corresponding properties to connect them as shown in Fig. 5.

We need the second phase to complete the transformation from m to o.

Step 5. Enrich the class and class hierarchy. In a system, each block will have several particular instances; consequently, all blocks are modeled as children of `Block`.

Step 6. Create individuals. The final step is to create

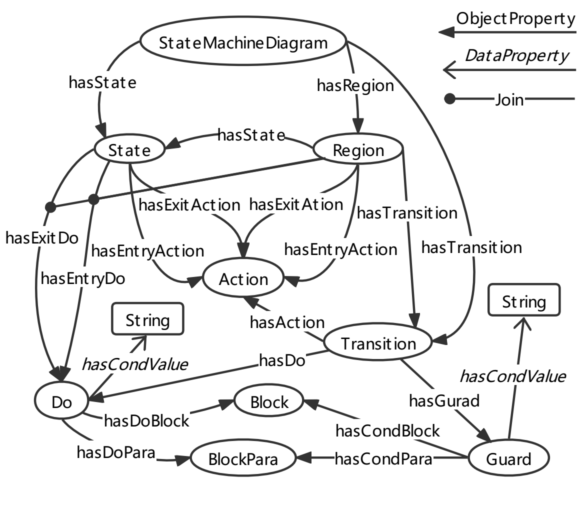

Fig. 5. Semantic representation of stm

individuals for the remaining instances in m. Creating an individual of a class requires (1) selecting a class, (2) creating an individual of that class, and (3) adding property values to that individual.

B. Traceability Information Model

The model in Fig. 6 specifies the well-formedness criteria for the potential traceability links between requirements and components. In practice, the ontology-based implementation of TIM provides at least two benefits [10]:

1) As tracing is a complex task, but the approach provides a guideline that simplifies its building and allows for flexible changes.
2) As traceability is implemented in a way that is friendly to people who did not create it and who only need to know some of the semantics of `ObjectProperty`.

The class hierarchy in *Requirement Concepts* demonstrates a process of modeling the analysis of requirements for safety-critical systems. `trace` establishes the traceability link between `Mapping` and `Block`, which is also the link point between *Requirement Concepts* and *Design Concepts*. If there exists a block b that satisfies the following axioms, then b is said to be a system component that satisfies the functional requirement q.

$$Block(?b) \wedge Mapping(?map) \wedge DeriveRequirement(?q) \\ \wedge hasMapping(?q, ?map) \wedge trace(?map, ?b)$$

Of course, it is possible that several b meet q. map have `mapTo` series of object properties to concretize the satisfaction. The details include that: (1)a block satisfies a requirement by a block function concretized by `mapToBF`; (2)a block satisfies a requirement by modifying its parameter concretized by `mapToBP`; (3)a block satisfies a requirement before or after an activity is performed in its activity partition concretized by `mapToAE`. `allocate` links a block to its activity partition, while `mapToBF` links a block function and an activity. These

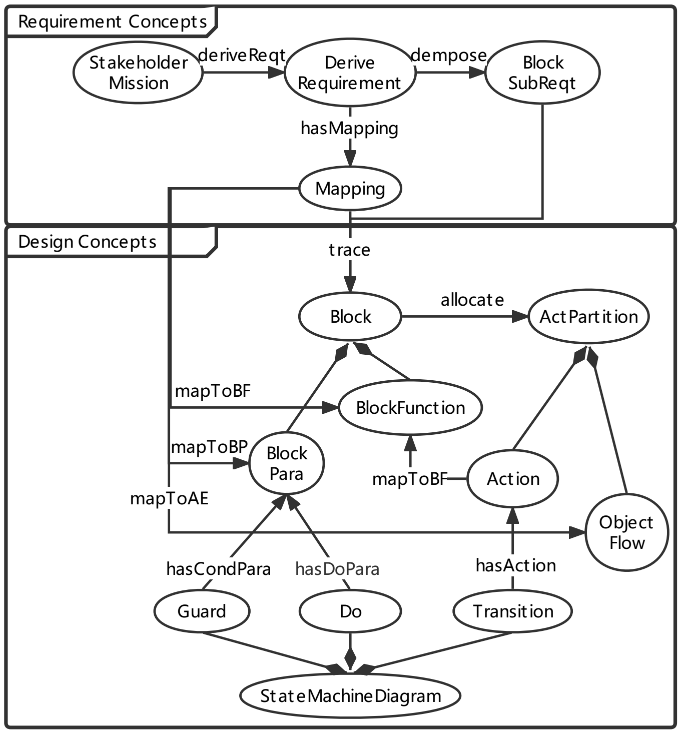

Fig. 6. Ontology_based TIM

mean that the function of a block can be performed as an activity in its activity partition. Conversely, functions that the block does not have cannot be performed. The final `has` series of object properties pass the link to the design of state machine diagrams.

III. Evaluation

A. Experiment Design

We first select a safety-critical system, Production Cell System (PCS) [11], and use the approach described above to obtain its lower-level ontology o_{PCS} as the experimental object. Production Cell System (PCS) is a well-known paradigm for embedded systems and was previously used as a baseline to evaluate the capabilities of various specification methods for safety analysis and verification. From the complete SysML requirements and design specification given in [11], it is known that PCS consists of 23 safety-related requirements and 6 main blocks, each with corresponding activity partitions and state machine diagrams.

We chose a comprehensive quantitative ontology evaluation method in [12], which has a study of metrics implemented in the popular quality frameworks. There are 8 sub-characteristics $RROnto$, $ANOnto$, $LCOMOnto$, $INROnto$, $CROnto$, $NOMOnto$, and $CBOnto$ which count the proportion of classes, properties, and individuals from different perspectives to measure the rationality of ontology design. Each sub-characteristic has a calculated value and a score out of five, based on each value. Based on the eight sub-characteristics, four more metrics are summarized to measure overall quality:

TABLE I
QUANTITATIVE EVALUATION RESULTS

Sub-characteristics		
Name	Value	Score
$RROnto$	0.64	4
$ANOnto$	1.00	5
$LCOMOnto$	2.44	4
$INROnto$	1.80	5
$CROnto$	14.80	5
$NOMOnto$	0.31	5
$CBOnto$	1.13	5
Metrics		
Name	Avg(scores)	
SE_v	4.33	
FAE_v	4.67	
ME_v	4.67	
Glo_v	4.50	

TABLE II
QUALITATIVE EVALUATION RESULTS

Dimension	Desription	Importance
Completeness	Missing domain or range in properties	Important
	Creating unconnected ontology elements	Minor
	Inverse relationships not explicitly declared	Minor
Compliance	No license declared	Important
Accuracy	Using a Miscellaneous Class	Minor

structural metric SE_v, functional adequacy metric FAE_v, maintainability metric ME_v, and global metric Glo_v. For details, see the framework presented in [12]. The qualitative evaluation tool we chose in [13] extends previous work on modeling errors and 41 pitfalls are identified with importance levels(critical, important, or minor). Experimental documents can be found at https://github.com/ch-wq81404/Experimental-documents.

B. Evaluation Results

Results of quantitative and qualitative evaluation are shown in Tab. I and Tab. II respectively. Tab. II shows the pitfalls and their importance in o_{PCS}. This will allow the user to correct the ontology and transform it into a better ontology.

From the results, almost all sub-characteristics get the highest score of 5, except for $RROnto$ and $LCOMOnto$. Anyway, the high scores of other metrics indicate the following:

- Rationalization of class richness portrayed by $CROnto$.
- Vertical and horizontal coordination of the class hierarchy portrayed by $LCOMOnto$ and $CBOOnto$.
- Quantitatively sufficient properties portrayed by $NOMOnto$, $INROnto$, and $ANOnto$.

It can be seen from Tab. II that o_{PCS} does not have problems of critical importance, which will affect the usage of ontology. All of these demonstrate the feasibility and efficiency of our approach.

IV. RELATED WORK

Existing transformation approaches are mostly limited to a single view [2], [5], [6] without traceability. [2] proposed the idea of using ontology to verify the dynamic behavior for complex systems. The work of [3] is closest to ours, and its ontology is used to analyze system change propagation. But they does not share the file so that our approach cannot be compared to theirs. All of them lack an automated tool.

V. CONCLUSION

We have presented a novel approach to maintain precious traceability between requirements and model multi-views design based on ontology. And the experimental results can be used by other relevant researchers to compare different transformation approaches or for other purposes. In future work, we will implement an automated tool to derive ontology model from SysML model, and select a different type of system as a case to explore the universality of our transformation method. In addition, we can use semantic inference tools of ontology to perform verification.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development under Project 2019YFA0706404, the NSFC under Project 61972150, and the Fundamental Research Funds for Central Universities.

REFERENCES

[1] O. M. Group, "How should SysML be applied to a MBSE project? How is SysML commonly abused?," https://sysml.org/sysml-faq/sysml-applied-mbse.html.

[2] C. Ruirui, Y. Liu, and X. Ye, "Ontology based behavior verification for complex systems." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 51739, 2018.

[3] H. Wang, V. Thomson, and C. Tang, "Change propagation analysis for system modeling using semantic web technology," *Advanced Engineering Informatics(AEI)*, vol. 35, pp. 17–29, 2018.

[4] S. Jenkins and N. Rouquette, "Progress on integrating owl and sysml,", NASA, 2012.

[5] H. Wardhana, A. Ashari, and A. Sari, "Transformation of sysml requirement diagram into owl ontologies," *Int J Adv Comp Sci Appl*, pp. 11, 2020.

[6] H. Graves, "Integrating sysml and owl," *Proceedings of OWL: Experiences and Directions*, 2009.

[7] B. Lionel, F. Davide, N. Shiva, S. Mehrdad and Y. Tao, "Traceability and sysML design slices to support safety inspections: a controlled experiment," in *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 1, pp. 1–43, 2014.

[8] W. W. W. Consortium(W3C), "Owl 2 web ontology language: Direct semantics (second edition)," https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/, 2012.

[9] N. F. Noy, D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology,", 2001.

[10] P. Mader, O. Gotel, and I. Philippow, "Getting back to basics: Promoting the use of a traceability information model in practice," In *Workshop on Traceability in Emerging Forms of Software Engineering(ICSE)*, pp. 21–25, 2013.

[11] T. E. Klykken, "A case study using sysml for safety-critical systems," *Master's thesis*, 2009.

[12] G. R. Rold an-Molina, D. Ruano-Ord as, V. Basto-Fernandes, and J. R. M endez, "An ontology knowledge inspection methodology for quality assessment and continuous improvement," *Data Knowledge Engineering(DKE)*, vol. 133, pp. 101889, 2021.

[13] P. María, A. Gómez-Pérez, and M. C. Suárez-Figueroa, "Oops!(ontology pitfall scanner!): An online tool for ontology evaluation." *International Journal on Semantic Web and Information Systems(IJSWIS)*, vol. 10, no. 2, pp. 7–34, 2014.