Transdisciplinarity and the Future of Engineering
B.R. Moser et al. (Eds.)

doi:10.3233/ATDE220632

Use of Semantic Web Technologies to Enable System Level Verification in Multi-Disciplinary Models

Daniel DUNBAR[1], Thomas HAGEDORN, Mark BLACKBURN and Dinesh VERMA
Stevens Institute of Technology, Hoboken, USA

Abstract. Integration of data from multiple sources into a single, project wide view is a necessity to keep up with increasing complexity and transdisciplinary considerations in engineering projects. Semantic Web Technologies (SWT) provide a unique way of linking and reasoning upon data from disparate sources to gain insights on the data viewed as a whole. By ingesting project data into a tool-agnostic repository and applying targeted reasoning, SWT can be used to perform system level verification tasks, such as providing a Key Performance Indicator (KPI) of completeness that gives project design and status insights to key stakeholders. This paper reports research creating a Semantic System Verification Layer (SSVL) as an extension to an existing Digital Engineering framework that utilizes SWT. This process and procedure are applied to a relevant use case to demonstrate and clarify the functions.

Keywords. model-based systems engineering, semantic web, ontologies, model validation and verification, knowledge representation, digital engineering

Introduction

Today's systems are often highly complex and utilize advancements in technology and theory across a multitude of different domains. This expands the capabilities of systems as well as their adaptability, but it also introduces unique challenges that come from transdisciplinary collaboration. The move to model-based engineering design and analysis has enabled higher levels of fidelity and consistency within domain specific design tasks. However, model-based engineering by itself does not begin to solve problems of interoperability and transdisciplinary collaboration; the data is still stored in individual models that often do not interact with other models. To perform these interactions, separate processes must be designed, implemented, and maintained.

Digital Engineering (DE) is a field that seeks to address these processes using "authoritative sources of system data and models as a continuum across disciplines to support lifecycle activities from concept through disposal" [1]. DE seeks to integrate model-based systems engineering (MBSE) with domain specific model-based engineering (MBE). This integration, and the higher-level assessments enabled by it, begin to address cross-model project concerns apparent in a transdisciplinary design and analysis environment.

[1] Corresponding Author, Mail: ddunbar1@stevens.edu.

Central to the DE concept is the notion of the Digital Thread (DT) [2]. The DT is the full set of digital engineering tools, models, and generated data comprising the full lifecycle of a system. The DT seeks to promote the traceability of system data and facilitate analysis throughout that lifecycle. These goals are achieved through the creation of an Authoritative Source of Truth (AST) [1]. The AST functions as a central store of data that is shared among all aspects of the DT, enforcing a consistent and accessible representation of the system across its lifecycle. With all aspects of the DT residing within an AST, the AST thus serves as a basis for both data accessibility and traceability as well as potentially facilitating system verification and artificial intelligence applications.

One promising approach to implementing such an AST is the use of ontologies. Ontologies are human and machine-readable lexicons comprising a taxonomy of terms and logical restrictions placed upon them. In the context of a broader suite of Semantic Web Technologies (SWT), ontologies serve as both a basis to meaningfully tag data for knowledge representation and a basis for automated inferencing [3]. Ontologies facilitate a number of applications potentially of interest to a DE environment, such as automated inferencing and enhanced search. Notably, when used in a DE environment, ontologies permit the collation of meaningfully tagged data in a tool-agnostic repository which can serve as an AST. In this capacity then, the use of ontologies and SWT promote both interoperability and consistency of data across tools in the DT [4], [5].

This paper proposes the use of an SWT based Semantic System Verification Layer (SSVL) that is added onto a previously reported framework for data integration in a DE context [6]. This framework uses SWT to address data integration concerns inherent in transdisciplinary efforts, and the SSVL utilizes features of the SWT stack to perform verification checks upon a system represented in an ontology aligned AST. This paper demonstrates the SSVL by applying the broader framework to an Information Technology (IT) use case and producing a completeness metric that assesses data at the system level.

1. Background

The use of ontologically aligned data as the AST in a DE context has been explored in prior research. A major topic in the literature has been in the creation of an AST that contains all system data in an ontologically aligned format. Moser et al. formulate an Engineering Knowledge Base (EKB) that acts as a central hub for information [7]. Other efforts have focused on the question of interoperability among external toolsets [6], [8]. In these applications the AST serves as a predictable, neutral representation of system information which each tool independently accesses via an appropriate interface. Thus, the role of the ontology is to facilitate the creation of those interfaces while the external tools are permitted to work from the same store, promoting coordination and interoperability [7].

The use of ontologies and MBSE models such as Systems Modeling Language (SysML) system models has also been a focus. Interestingly, most of these efforts have converged upon the use SysML stereotypes to inject a SysML model with ontologically relevant terms [6], [8]–[11], though they differ in both method and objectives. One approach [12] uses an Ontology Modeling Language (OML) that simultaneously specifies both a SysML model and an underlying domain ontology. Doing so permits the execution of large numbers of semantically enhanced design checks. Tudorache and

Alani use SysML as an inspiration for ontology development work, although they stop short of addressing specific mapping functionality between a SysML model and an ontologically aligned dataset [9]. Other efforts [4]–[6], [8] coordinate a suite of ontologies aligned under a single top level [13], which are then used to create an extensible, project and domain agnostic framework for coordinating the exchange of system data across tools. Alongside appropriate software interfaces [6], the DE framework described in these works is used to facilitate a DT across various engineering workflows.

Verification using ontologies in the DE realm is less studied. Hagedorn et al. [14] describe the use of an ontology-based DE architecture for performing system-wide design evaluations. While the authors demonstrate a simple weight roll-up example, they stop short of directly inspecting a system against some set of rules or requirements. As noted above [12], ontologically backed design checking in the context of MBSE models has also been explored via the OML language. However, this requires the creation of specialized OML models as opposed to more common MBSE tools. Steyskal and Wimmer reported the use of the Shapes Constraint Language (SHACL) to perform constraint checking [15]. SHACL [2] is a protocol that permits the expression of verification checks as graph "shapes," with matching data subjected to logical tests. This permits evaluation of ontologically aligned data, which is typically created under an "open world" assumption, under a closed world assumption. Using this approach, the authors constructed consistency checks between viewpoints, in the context of multiple Resource Description Framework (RDF) graphs with mappings between them.

Still missing from relevant literature reviewed is a holistic assessment of a DE system model as a system-level representation of aggregate parts that are informed or populated by MBE and MBSE components. This paper's SSVL seeks to fill that gap by extending an existing DE framework [6] to include a system level verification component. This verification layer uses SHACL to evaluate the information associated with a system. A simple case study focusing on the completeness of data is used to demonstrate the potential of the proposed SSVL.

2. Methods

2.1. Description of the DE Framework

The method presented in this paper builds on previous research using ontologies and the SWT stack as foundational technologies for building a tool-agnostic AST [6].

2.1.1. Structure and Interfaces

The SSVL extends a method that uses SysML to characterize the system under analysis. Said method provides clear guidance on how external tools can interact with the ontologically aligned data by defining three different types of notional interfaces to the data store: the Mapping Interface, the Specified Model interface, and the Direct Interface (Figure 1) [6].

[2] https://www.w3.org/TR/shacl/

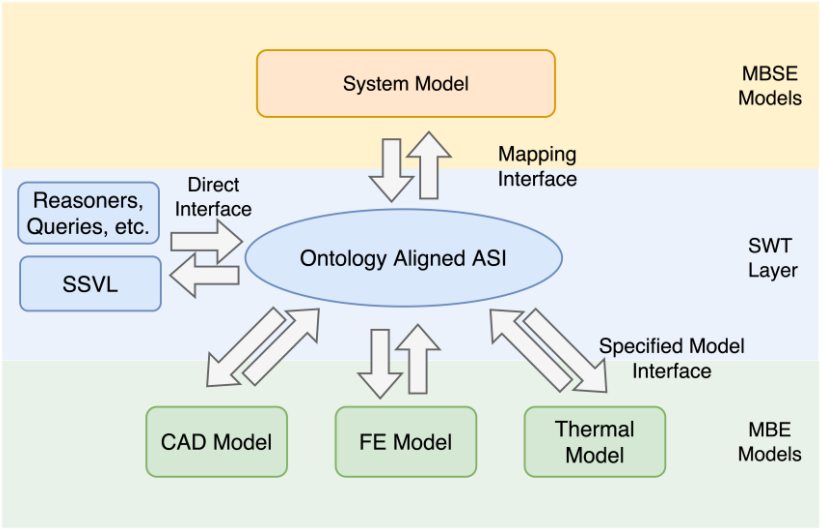


Figure 1. The DE Architecture and Interfaces Proposed in [6].

Mapping interfaces enable the capture of external model data in ontologies by establishing connections between the model data and an ontologically aligned version of that model. This is inherently tool-specific and requires a mechanism for matching data contained in the tool model with ontologically relevant terms. For example, stereotypes in the SysML specification can be used to tag model elements with ontology terms for mapping to an ontology aligned representation. The Specified Model interfaces the DT itself within the broader system model. The System model characterizes a given MBE model by its input and output parameters, which are in turn defined throughout the system model. The Specified Model interface uses the ontology-aligned AST to afford these parameters to external tools. This aggregation can occur independently of the data source. For example, a cost model analysis may input parameters associated with a Computer Aided Drawing (CAD) model for part numbers, information derived from a vendor parts list for pricing, and parameters populated by a project management tool for man time information. In addition, related Finite Element Analysis or Thermal analyses might affect those CAD parameters, with the outputs from those analyses representing metadata stored as value properties in the SysML system of analysis model. Thus, a single model pulls information that originated from two or more different tools whose data is present in the ontological representation of the system model.

The Direct Interface refers to interacting with semantic data directly with standard SWT tools. This might include directly manipulating or accessing data through semantic queries or performing constraint checking using semantic rules or constraint languages. While this technology stack may be used in previous interface types, the distinction lies in the level of abstraction at which the user interfaces with the full stack of models

2.1.2. Semantic System Verification Layer

While the DE framework described above provides a means to repeatably access information in an ontological AST via third party tools, it does not afford a direct means to inspect that data. Issues introduced via the model interface might be propagated

through to third party tools and cause subsequent errors or inaccuracies. The proposed SSVL seeks to mitigate this by adding a verification layer.

Utilizing the DE framework described above, there are at least two possible approaches to extending the framework to include system level verification tasks. The first involves a definition of a model via the specified model interface that directly deals with a specific verification task. For example, a system level verification on multiplicity of certain component types could be represented within the model, and subsequently exported to an external verification tool via one of the interfaces to the AST. This independent model in the DT could be beneficial for certain system-level analysis tasks. However, this approach could quickly become overly burdensome to address some system-level verification tasks and is not inherently reusable.

The second approach to system-level verification tasks would be the use of the SWT stack itself. This approach creates a separate layer in the methodology apart from the DT aspects of DE that is focused solely on system verification. It performs operations on the system data in tool-agnostic form and takes advantage of SWT benefits such as forward chaining, extensibility, and reusability.

The SSVL follows the second approach by allowing the use of the SWT stack to prove system verification capability in a DE context. The SSVL is implemented using SHACL shapes, which are executed as a set of verification rules using an appropriate reasoning engine. Since the ontology-aligned data can involve complex patterns of data representation, the SHACL-SPARQL syntax is used. SHACL-SPARQL allows highly targeted verification based upon the SPARQL Protocol and RDF Query Language (SPARQL). In this syntax, a query is used to target specific patterns of data representation, yielding a set of entities that are then subjected to verification in the form of logical comparison to some desired state. When SSVL is activated by running an appropriate SWT software against the AST, the SHACL shapes produce a Boolean value indicating whether the constraints were violated or not. If they were violated, further details of the failures are provided in the form of a verification report written in the RDF language. As this all relies upon tool and potentially project agnostic data representation, this layer is primed for automation and reusability.

The SSVL thus utilizes the Direct Interface discussed above to access the SWT stack directly. Because these shapes act upon the entirety of the AST, they may be applied throughout the entirety of a DT for data verification and guidance. In practice, this means that shapes may act at both a domain specific and a domain agnostic level. For example, the SSVL might perform simple completeness checks at a domain agnostic level and perform highly sophisticated checks based upon the input of subject matter experts at the domain specific level.

2.1.3. Implementation of the Semantic System Verification Layer

As the SSVL relies upon off-the-shelf SWT software, the implementation of it is highly environment specific. Since relatively few RDF repositories support within-tool use of SHACL-SPARQL, the SSVL was implemented parallel to the AST. The open-source pySHACL Python package [16] was used as a SHACL engine. PySHACL implements a SHACL verification engine and report generation tool. It works alongside the open-source RDFLib package[3], which facilities the parsing, manipulation, and serialization of RDF graphs within a Python environment.

[3] https://github.com/RDFLib/rdflib

The workflow thus proceeds as follows (Figure 2). The AST is populated via the model and mapping interfaces described in [6]. At any point prior to the activation of the SSVL, SHACL shapes are defined (either using an SWT tool or simply by writing an RDF document) as a SHACL graph. At the time of activation, the contents of the AST are downloaded as an RDF document. A python script retrieves this document, parses the data into an RDFLib graph object, and then activates the pySHACL engine. The engine evaluates the SHACL graph against the contents of the AST, and then generates a verification report which may be used to diagnose any noted issues.

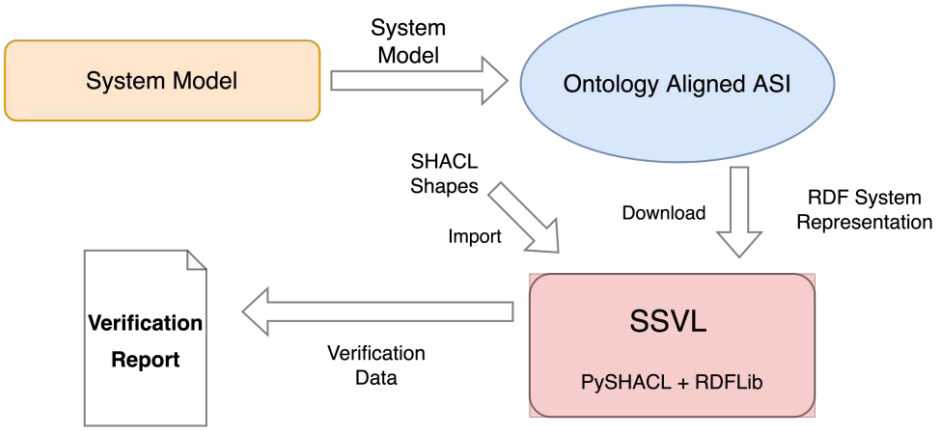


Figure 2. Workflow for the SSVL implementation.

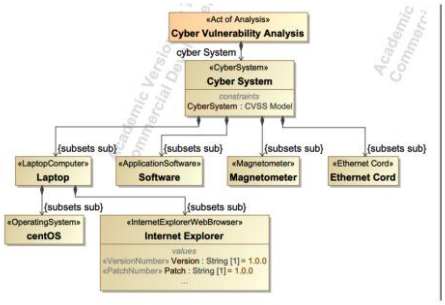
2.2. Implementation Case Study

To demonstrate the SSVL discussed above, it is applied to an IT architecture use case based upon one reported in previous research [6], [17]. The use case focuses on a simple cyber system with a hypothetical security vulnerability. In the previously reported case [6], a MATLAB script performs the necessary analysis to populate a Cyber Vulnerability Scoring System (CVSS)[4] vector and assign a summary score to the system. In this case we first evaluate the representation of the system in the AST by using the SSVL to check the completeness and formatting of the model data. Note, the specific tools represent implementation choices aimed at demonstrating the coordination of multiple tools via an AST, not an underlying requirement of the method or its implementation in software.

2.2.1. Structure and Interfaces

Per the method described in [6], the system is defined using a SysML block definition diagram, shown in Figure 3. The model uses stereotypes to bind ontological terms to the SysML model, permitting automated generation of an ontology-aligned representation of the system through the mapped interface. The system comprises a set of software and hardware elements each of which has a CVSS property indicating a summary vulnerability state.

[4] https://nvd.nist.gov/vuln-metrics/cvss#



#	Name	av : Attack Vector	score : Real	vs : String
1	centOS	Physical		
2	cyber System	Physical		"default"
3	ethernet Cord	Physical		
4	internet Explorer	Network		
5	laptop	Physical		
6	magnetometer	Physical		
7	software	Physical		

Figure 3. Cyber System Block Definition Diagram (left) and Partial Instance Table (right)

The model was instantiated to facilitate analysis, and the scoring for the Cyber System score was intentionally left blank, while another field was seeded with an invalid CVSS vector. The first omission poses a potential risk of error for downstream domain models to raise errors, which might require diagnosis of the full DT. The latter indicates that an analysis might not have been completed, again indicating possible issues for downstream analysis. Thus, the objective of the case study is to catch this omission using the described SSVL to facilitate integration of the MBSE and MBE models. The grayed-out portions of the table indicate values that are not related to the Cyber System CVSS Analysis.

2.2.2. Design Validation

The SSVL was configured with SHACL shapes specifying constraints that should be enforced in the data graph. Shapes are defined to check two aspects of the model: the completeness of the model insofar as there are no blank fields and the fidelity of the included data. These checks are implemented via two verification rules:

1. No blank (null) fields are acceptable in a complete system model
2. The CVSS Vector should follow the regular expression pattern: AV:[NALP]\/AC:[LH]\/PR:[NLH]\/UI:[NR]\/S:[UC]\/C:[NLH]\/I:[NLH]\/A:[NLH] (Figure 4)

```
ex:VectorShape
    a sh:NodeShape ;
    sh:target [
        a sh:SPARQLTarget ;
        sh:select """
            PREFIX pre: <http://exampledata.org/pre.owl#>
            SELECT DISTINCT ?this
            WHERE {
                ?s pre:name 'vs' .
                ?s pre:has_value ?this .
                ?s pre:relevant_to / pre:name 'cyber_System' .
            }
            """ ;
    ] ;
    sh:datatype xsd:string ;
    sh:pattern "AV:[NALP]\\/AC:[LH]\\/PR:[NLH]\\/UI:[NR]\\/S:[UC]\\/C:[NLH]\\/I:[NLH]\\/A:[NLH]" .
```

Figure 4. SHACL Shapes Graph for Cyber Completeness Metric. Prefixes are excluded from the figure for brevity.

Once the AST is populated, the SHACL engine evaluates these verification rules and generates a summary report. Note that the RDF graphs are extensible, and so

potentially many SHACL shapes can be applied to the AST in addition to the ones shown. Indeed, different sets of shapes might be applied at different phases of a DT.

3. Results

Before performing an analysis via MATLAB, the SSVL is invoked, and a compliance report is generated (Figure 5). As expected, the SHACL shapes identify two issues in the AST data derived from the SysML model - that the system model is not complete and that some of the data does not conform to the proper format. The blank value must be remedied in the SysML instantiation, and the CVSS vector either fixed in model or corrected with an appropriate MBE tool.

```
[] a sh:ValidationReport ;
    sh:conforms false ;
    sh:result [ a sh:ValidationResult ;
            sh:focusNode <http://testontology.org/cyber_mapped/88ebed44-798d-4ca6-8510-
            d597b66c5e7b.owl#5109752e-7966-4fdd-a986-f2f7473ca529_spec> ;
            sh:resultMessage "Less than 1 values on <http://testontology.org/cyber_mapped/88ebed44-
            798d-4ca6-8510-d597b66c5e7b.owl#5109752e-7966-4fdd-a986-f2f7473ca529_spec>->[
            sh:inversePath pre:conforms_to ]" ;
            sh:resultPath [ sh:inversePath pre:conforms_to ] ;
            sh:resultSeverity sh:Violation ;
            sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
            sh:sourceShape ex:NoBlanksShape ],
        [ a sh:ValidationResult ;
            sh:focusNode "default" ;
            sh:resultMessage "Value does not match pattern 'AV:[NALP]\\/AC:[LH]\\/PR:[NLH]\\/UI:
            [NR]\\/S:[UC]\\/C:[NLH]\\/I:[NLH]\\/A:[NLH]'" ;
            sh:resultSeverity sh:Violation ;
            sh:sourceConstraintComponent sh:PatternConstraintComponent ;
            sh:sourceShape ex:VectorShape ;
            sh:value "default" ] .
```

```
[] a sh:ValidationReport ;
    sh:conforms true .
```

Figure 5. Validation Report – Incomplete (left) and Complete (right).

By simply navigating to the sh:focusNode instances indicated in the report, one can quickly traverse the data in the AST to identify the specific elements in the SysML model that pose potential problems. Thus, when deployed in the broader DE environment the SSVL provides both a means to anticipate problems in the MBE models and obtain information about their provenance.

After the modeler remedies the SysML, MATLAB runs the SME model for CVSS scores and updates the triple store with the results of the analysis. A subsequent run of the SSVL indicates no verification errors, and thus the model is complete, and the data adheres to the specified standards according to the verification.

4. Discussion

The addition of the SSVL to a digital engineering framework expands the functionality of the AST beyond a DT connecting various domain and system models; it allows for system wide assessment. The use of SWT to implement the framework and SSVL extension takes advantage of the inherent reasoning capabilities of a knowledge base rooted in formal logic. Thus, the SSVL enables emergent behavior made possible by the implementation of a DE framework.

The IT use case presented, while simplistic, illustrates the potential uses of the SSVL. The method itself is domain agnostic and intended to be translated to many different domains. Because the SSVL acts directly upon the AST, it is thus possible to view the entirety of a DT at once and thus craft powerful system verification rules. The use of standard recommendations like SHACL enables a broader development community and greater reuse opportunity across projects and organizations. SHACL graphs are

inherently extensible, meaning that an expanding ecosystem of verifications can be incrementally developed, shared, and reused across engineering projects and domains. As with the broader ontology space, the potential for reusability and expressive logical tests is a significant strength of this approach. Combined with an overall DE framework, the SSVL offers significant potential to realize a highly detailed, automated, and reusable system verification tool. Future research should explore this potential within a more detailed domain context.

Some limitations should be noted. As is commonly the case with RDF, scale and speed are factors that limit the scope of the SSVL. The example presented in this paper is very simple – more complex systems using variants of this methodology do show higher computational costs. These may be mitigated through any number of software solutions both within the SWT stack and using external tools, and optimal approaches should be investigated. The use of SHACL-SPARQL dramatically limits the toolset available for implementation – indeed this implementation was forced to implement the SSVL in parallel to the AST rather than integrated into it as would be preferable. While this may change in time, it does pose a challenge for creating a highly automated DE workflow. Moreover, the use of SWT for verification subjects the verification to the limits of SWT: applications involving significant numerical computation would thus be best implemented using an external tool and an appropriate interface to the AST. A hybrid approach is likely necessary for many systems. Future research should explore such an approach.

5. Future Work and Conclusion

Further research can be done to integrate SHACL constraint checking into a broader methodology in three important ways: (1) SHACL constraint checking can be developed to be accessed and modified programmatically as part of a larger workflow; (2) SHACL constraint checking can be expanded to capture other types of compliance checks such as consistency and satisfiability; and (3) a compliance ontology can be created and automatically populated by SHACL results. The latter integration would allow for the use of SHACL result parameters in additional implementations of the DE environment's interfaces, which would enable more flexible connections to broader design and analysis tasks.

In conclusion, this paper presents a novel Semantic System Verification Layer (SSVL) that uses SWT to perform verification on data sourced from multiple models connected via a Digital Thread and captured in an Authoritative Source of Truth. It demonstrates usage of the SSVL by performing an assessment of completeness based on system data aggregated in an AST through the use of a mapping interface. It uses SHACL and a direct interface to the ontology aligned data to perform constraint checking to establish a binary assessment of whether the system model was complete according to its context dependent definition. The use case results show a progressive example that shows both an incomplete model and how the SSVL can help remediate it.

Acknowledgement

This research was sponsored by the Systems Engineering Research Center (SERC), a University Affiliated Research Center (UARC) housed at Stevens Institute of Technology.

References

[1] N.N., *Digital Engineering Strategy*, Department of Defense, Office of the Deputy Assistant Secretary of Defense for Systems Engineering, Jun. 2018. https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf, accessed June, 20 2022.

[2] W. Roper, *There is no spoon: The new digital acquisition reality*. United States Air Force, 2020. https://software.af.mil/wp-content/ uploads/2020/10/There-Is-No-Spoon-Digital-Acquisition-7-Oct-2020-digital- version.pdf, accessed June, 20 2022.

[3] T. R. Gruber, The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases, *KR'91: Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning,* April 1991, pp. 601–602.

[4] T. J. Hagedorn, B. Smith, S. Krishnamurty, and I. Grosse, Interoperability of disparate engineering domain ontologies using basic formal ontology, *J. Eng. Des.*, vol. 30, no. 10–12, pp. 625–654, Dec. 2019, doi: 10.1080/09544828.2019.1630805.

[5] M. A. Bone, M. R. Blackburn, D. H. Rhodes, D. N. Cohen, and J. A. Guerrero, Transforming systems engineering through digital engineering, *J. Def. Model. Simul. Appl. Methodol. Technol.*, vol. 16, no. 4, pp. 339–355, Oct. 2019, doi: 10.1177/1548512917751873.

[6] D. Dunbar *et al.*, Driving Digital Engineering Integration and Interoperability Through Semantic Integration of Models with Ontologies, arXiv:2206.10454, 2022.

[7] T. Moser, The Engineering Knowledge Base Approach, in: S. Biffl and M. Sabou (eds.) *Semantic Web Technologies for Intelligent Engineering Applications*, Springer International Publishing, Cham, 2016, pp. 85–103. doi: 10.1007/978-3-319-41490-4_4.

[8] M. Bone, M. Blackburn, B. Kruse, J. Dzielski, T. Hagedorn, and I. Grosse, Toward an Interoperability and Integration Framework to Enable Digital Thread, *Systems*, vol. 6, no. 4, p. 46, Dec. 2018, doi: 10.3390/systems6040046.

[9] T. Tudorache and L. Alani, Semantic Web Solutions in the Automotive Industry, in S. Biffl and M. Sabou (eds.) *Semantic Web Technologies for Intelligent Engineering Applications*, Springer International Publishing, Cham, 2016, pp. 297–326. doi: 10.1007/978-3-319-41490-4_12.

[10] J. S. Jenkins and N. F. Rouquette, "Semantically-Rigorous Systems Engineering Modeling Using SysML and OWL, *5th International Workshop on Systems & Concurrent Engineering for Space Applications*, 2012, https://trs.jpl.nasa.gov/bitstream/handle/2014/43338/12-5065_A1b.pdf, accessed June 20, 2022.

[11] D. A. Wagner, M. B. Bennett, R. Karban, N. Rouquette, S. Jenkins, and M. Ingham, An ontology for State Analysis: Formalizing the mapping to SysML, *2012 IEEE Aerospace Conference*, Big Sky, MT, Mar. 2012, pp. 1–16. doi: 10.1109/AERO.2012.6187335.

[12] D. Wagner, S. Y. Kim-Castet, A. Jimenez, M. Elaasar, N. Rouquette, and S. Jenkins, CAESAR Model-Based Approach to Harness Design, *2020 IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2020, pp. 1–13. doi: 10.1109/AERO47225.2020.9172630.

[13] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with basic formal ontology*. MIT Press, Boston, 2015.

[14] T. Hagedorn, M. Bone, B. Kruse, I. Grosse, and M. Blackburn, Knowledge Representation with Ontologies and Semantic Web Technologies to Promote Augmented and Artificial Intelligence in Systems Engineering, *INSIGHT*, vol. 23, no. 1, pp. 15–20, 2020, doi: https://doi.org/10.1002/inst.12279.

[15] S. Steyskal and M. Wimmer, Leveraging Semantic Web Technologies for Consistency Management in Multi-viewpoint Systems Engineering, in: S. Biffl and M. Sabou (eds.) *Semantic Web Technologies for Intelligent Engineering Applications*, Springer International Publishing, Cham, 2016, pp. 327–352. doi: 10.1007/978-3-319-41490-4_13.

[16] Sommer, Ashley and Car, Nicholas, *pySHACL*. Zenodo, 2022. doi: 10.5281/ZENODO.4750840.

[17] M. R. Blackburn *et al.*, *Transforming Systems Engineering through Model-Centric Engineering*, Final Technical Report SERC-2020-TR-009, WRT-1008 (NAVAIR, Jun. 2020.