Received September 11, 2020, accepted September 21, 2020, date of publication September 25, 2020,
date of current version October 8, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3026941*

# Converting OCL and CGMES Rules to SHACL in Smart Grids

**MOHAMED LARHRIB[1], MIGUEL ESCRIBANO[2], CARLOS CERRADA[1], AND JUAN JOSE ESCRIBANO[1]**

[1]Department of Software Engineering and Computer Systems, Universidad Nacional de Educacion a Distancia (UNED), 28040 Madrid, Spain
[2]Department of Forecast and Load Scheduling, Red Eléctrica de España, 28109 Alcobendas, Spain

Corresponding authors: Juan Jose Escribano (jjescri@issi.uned.es) and Mohamed Larhrib (mlarhrib4@alumno.uned.es)

This work was supported in part by the Spanish Ministry of Economy and Competitiveness under Project DPI2016-77677-P, and in part by the RoboCity2030-DIH-CM Madrid Robotics Digital Innovation Hub ("Robótica aplicada a la mejora de la calidad de vida de los ciudadanos: Fase IV"; S2018/-NMT-4331), funded by the "Programas de Actividades I + D de la Comunidad de Madrid."

**ABSTRACT** Models are first-class elements in Model-Driven Engineering (MDE). In this paradigm, the most widespread approaches adopted by the development community are Object-Oriented and ontological, formalized using Unified Modeling Language (UML) and Resource Description Framework (RDF), respectively. However, Object Management Group (OMG) does not provide a specific standard language for validating UML models against Object Constraints Language (OCL) constraints; meanwhile, World Wide Web Consortium (W3C) has defined Shapes Constraint Language (SHACL) as a standard validation language. Although the transformation between UML and RDF can be performed at the structural level, no effort has been made to transform OCL to SHACL. This paper addresses the transformation of OCL and text-based constraints to SHACL shapes in the context of Common Grid Model Exchange Standard (CGMES), a UML-based standard for electric utilities in Europe. This paper presents several contributions to the software engineering community. First, solving the validation problem in a standardized way. Second, facilitating European Network of Transmission System Operators for Electricity (ENTSO-E) the construction of an ontology associated with the CGMES standard. Third, allowing developers to integrate the two complementary approaches. Finally, Promoting the adoption and integration of the ontological approach in the software community.

**INDEX TERMS** CIM for ENTSO-E (CGMES), OCL rules, ontology, RDF/RDFS, SHACL standard.

## LIST OF ACRONYMS

| | |
|---|---|
| **CGM** | Common Grid Model |
| **CGMES** | Common Grid Model Exchange Standard |
| **CIM** | Common Information Model |
| **CIMI** | Clinical Information Modeling Initiative |
| **CLP** | Constraint Logic Programming |
| **DMTF** | Distributed Management Task Force |
| **DSL** | Domain Specific Language |
| **EMF** | Eclipse Modeling Framework |
| **ENTSO-E** | European Network of Transmission System Operators for Electricity |
| **EQ** | Equipment |
| **FOL** | First Order Logic |
| **HTML5** | Hypertext Markup Language 5 |
| **HTTP** | Hypertext Transfer Protocol |
| **IEC** | International Electrotechnical Commission |
| **IGM** | Individual Grid Model |
| **MDE** | Model-Driven Engineering |
| **MVC** | Model View Controller |
| **OCL** | Object Constraints Language |
| **ODM** | Ontology Definition Metamodel |
| **OMG** | Object Management Group |
| **OWA** | Open-World Assumption |
| **OWL** | Web Ontology Language |
| **RDF** | Resource Description Framework |
| **RDFS** | Resource Description Framework Schema |
| **SHACL** | Shapes Constraint Language |
| **SPARQL** | SPARQL Protocol and RDF Query Language |
| **SSH** | Steady State Hypothesis |
| **SV** | State Variables |
| **SWRL** | Semantic Web Rule Language |
| **TP** | Topology |
| **TS** | Technical Specification |

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed F. Zobaa.

| | |
|---|---|
| **TSO** | Transmission System Operator |
| **UML** | Unified Modeling Language |
| **URI** | Universal Resource Identifier |
| **W3C** | World Wide Web Consortium |
| WDTF | Water Data Transfer Format |
| XAPI | Experience API |
| XMI | XML Metadata Interchange |
| XML | Extensible Markup Language |

## I. INTRODUCTION

MDE is a paradigm where models are first-class elements throughout the software development life cycle. Object-Oriented is an MDE approach where the key elements to model the world are the class and the object. UML [1] is a visual modeling language that addresses modeling process with an Object-Oriented approach. OCL [2] defines added constraints on the objects in the UML model. RDF is a semantic web technology. In this approach everything is a resource with a unique identifier. Models are built from the perspective of resources, and predicates. Thus, the atomic knowledge representation is a triple in the form of (subject predicate object). Models are graphs. SHACL is a W3C standard validation language for validating conditions against RDF data graphs, in addition it can be used as a modeling language.

In MDE, models can be addressed from different approaches; this work focuses on the Object-Oriented (UML/OCL), and ontological (RDF/SHACL) approaches, as shown in Fig. 1.

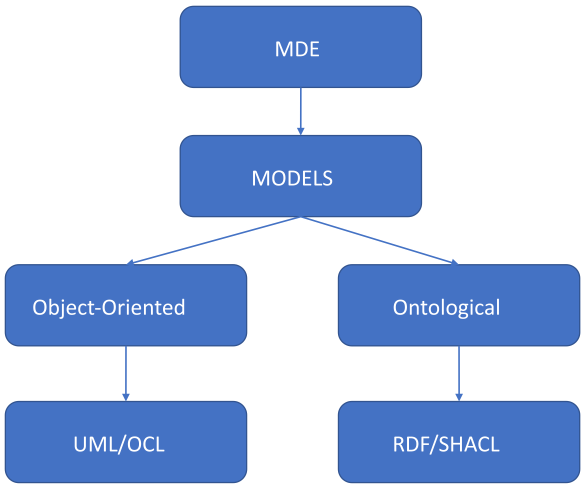


**FIGURE 1.** **Object-oriented and ontological approaches for MDE.**

Regardless of the modeling paradigm that is adopted, a model consists of a metamodel, rules, and instances. The transformation from a model in the source paradigm to another model in the target paradigm involves transforming lexical, syntactic, and semantic aspects from the source paradigm to the target paradigm, as indicated in Fig. 2.

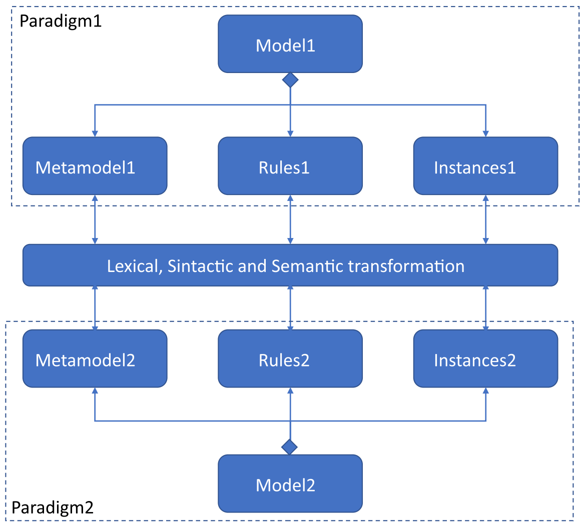


**FIGURE 2.** **Transformation of models between two paradigms.**

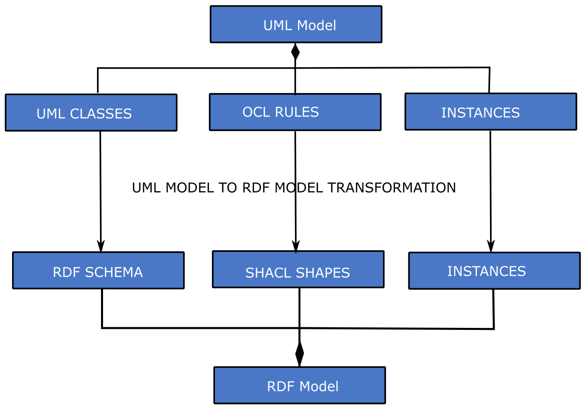


**FIGURE 3.** **Transformation between UML (object-oriented) models and RDF (ontological) models.**

In the case that concerns us, the transformation is performed between the Object-Oriented paradigm (UML) and the ontological paradigm (RDF), as shown in Fig. 3.

Common Information Model (CIM)[1] is a UML-based standard adopted by electrical utilities for exchanging network models between companies. It consists of a set of International Electrotechnical Commission (IEC) standards; Table 1 shows a subset adopted in Europe by European Network of Transmission System Operators for Electricity (ENTSO-E).

In the context of ENTSO-E, Common Grid Model Exchange Standard (CGMES) is a standard aimed at exchanging power system models between Transmission System Operators (TSO).

CGMES is a standard for interoperability based on the UML/OCL standard and can be considered an MDE approach. However, although OMG provides UML together with OCL as languages for modeling, it lacks a standard

[1]The CIM acronym can be found in two domains: Electricity domain officially adopted by IEC, and IT environment domain officially adopted by Distributed Management Task Force (DMTF). Our case is the first one.

**TABLE 1. IEC standards adopted by ENTSO-E.**

| IEC Standard | Standard Name |
|---|---|
| IEC 61970-552 | CIM XML Model Exchange Format |
| IEC 61970-301 | Common Information Model (CIM) Base |
| IEC 61970-302 | Common Information Model (CIM) for Dynamics Specification |
| IEC 61970-452 | CIM Static Transmission Network Model Profiles |
| IEC 61970-453 | Diagram Layout Profile |
| IEC 61970-456 | Solved Power System State Profiles |
| IEC 61970-457 | Common Information Model (CIM) for Dynamics Profile |
| IEC 61968-4 | Application integration at electric utilities |

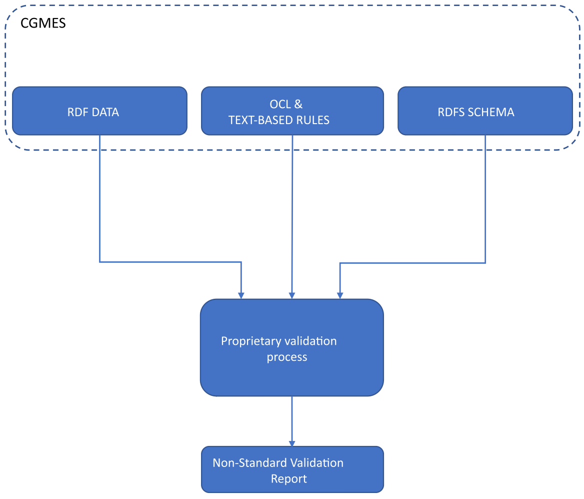

CGMES
RDF DATA
OCL & TEXT-BASED RULES
RDFS SCHEMA
Proprietary validation process
Non-Standard Validation Report

**FIGURE 4. Proprietary approach for validation process.**

validation language to validate UML models against OCL constraints. Consequently, validation is performed using proprietary, non-standard methods, see Fig. 4, thereof resulting in a negative impact on essential software factors such as quality and sustainability, among others.

The solution proposed in this work is to adopt SHACL as a language to validate models against the rules in the CGMES context, see Fig. 5. The effort consists of (i) the elaboration of a methodology to transform the OCL and text-based rules to SHACL (ii) the conversion of all CGMES rules to SHACL (iii) the development of an application that allows validating any model of a TSO, member of ENTSO-E, against the CGMES rules.

This work presents a series of contributions: (i) A methodology to convert OCL to SHACL, which has never been addressed before. (ii) Provide ENTSO-E stakeholders with an RDF/SHACL ontological model to validate their models and guide them in developing their own rules in SHACL, either by converting existing ones or creating new ones. (iii) A web application to validate model compliance with the CGMES rules. (iv) Since a big effort is being made from different

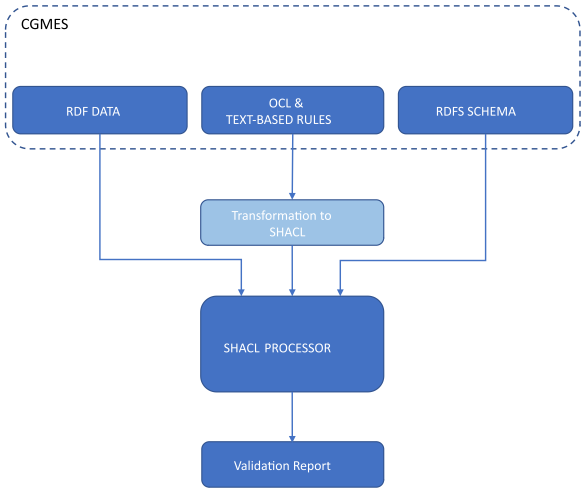


**FIGURE 5. Validation using SHACL in CGMES context.**

communities to promote the adoption of ontological engineering through semantic web technologies, our contribution by addressing the conversion from OCL to SHACL bridges the gap between Object-Oriented and ontological paradigms.

The target audience of this paper is domain experts in utilities, and software developers in the context of model validation.

This paper is split into nine parts. In part 2, related work is presented. In part 3, CGMES standard documents are described. Part 4 explores the reasons to adopt SHACL as a validation language. Part 5 shows a basic modeling rule in SHACL. Part 6 describes the methodology to transform OCL and CGMES rules into SHACL shapes. Part 7 presents a web application to validate CGMES models. Finally, part 8 contains conclusions and future works.

## II. RELATED WORK

In the last decades, the two most promising approaches for modeling are Object-Oriented and the ontological. The software community has adopted (UML/OCL) for the Object-Oriented modeling approach. On the other hand, the ontology modeling approach is performed using semantic web technologies (RDF/OWL/SHACL) [3]. This section is mainly divided into two parts: (i) a presentation of works related to Object-Oriented modeling, (ii) and another one that includes works related to modeling with ontologies.

A set of publications regarding UML/OCL approaches has been found. In [4], a group of automation strategies for OCL rules is presented. The meta-model evolution has a significant impact on the OCL rules. This work presents a semiautomatic approach throughout a strategy and a technique for OCL rules evolution concerning class diagram changes. In [5], a web-based tool has been developed using a user-friendly language at a higher level of abstraction. This language allows for rules definition. Therefore, knowledge of OCL syntax is not required. In this work [6], authors have used predicates logic to transform UML/OCL to validate UML models.

This methodology reduces the abstraction level towards First-Order Logic (FOL) for validating purposes.

In [7], it is created a logical model around UML/OCL considering preconditions, postconditions, states, and transitions between them. This approach adds new constraints (modifies only statements) to validate and verify UML models, taking into account all possible states and transitions between them, thus, reasoning about behavioral validation and verification. [8] demonstrates a conversion from UML/OCL model to the constraint logic programming (CLP) paradigm to allow for validation, reasoning, and verification. Only OCL invariants are considered. A new framework has been developed for automating transformation between class diagram to logic specification (Formula). With a reasoner, satisfiability is determined within certain bounds, i.e., if an instance model can be created.

In [9], a framework is developed with a repository of models. A common data model is created to abstract and integrates heterogeneous sources such as Extensible Markup Language (XML), ontologies, and so on via transformation to query and homogeneously visualize them. It is an independent modeling tool and can be integrated with other modeling tools. In state-of-the-art, it is possible to validate rules in the ontology paradigm. A set of publications regarding ontologies approaches has been found.

In [10], instead of validating RDF data against a schema, a model is validated against RDF data. The purpose is validating model credibility or giving the capability of ontology learning. Validation is performed using axiom scoring heuristics based on possibility theory.

In [11], Semantic Web Rule Language (SWRL) is used for data exchange applied to the water domain using Water Data Transfer Format (WDTF) for the definition of integrity constraints and using Web Ontology Language (OWL) for modeling this domain. Despite OWL and SWRL consider Open-World Assumption (OWA), authors have used OWL/SWRL for model validation. Validation is performed by transforming axioms to queries on the logic domain and transforming these queries into SPARQL Protocol and RDF Query Language (SPARQL), which are machine-readable. However, the authors recognize that SHACL is suitable in the validation contexts.

As can be seen in the state-of-the-art, there are several non-standardized validation approaches [10], [11]. All of them are in a web semantic context. In 2017 SHACL language has been released as an RDF validation standard in the ontology modeling domain. With our present state of knowledge, no works have been found related to the transformation of text-based and OCL rules to SHACL.

In [12], several validating RDF approaches are described. One of them is SHACL, the first standard validation language for RDF graphs. In [13], an online validation tool for SHACL is available. In [14], an open-source standard Experience API (XAPI) is a starting point for the e-learning domain in collecting learning data. Authors have transformed the XAPI standard model to an ontology, and they have used SHACL for validating the constraints defined in the XAPI standard. JSON is used to serialize data. In [15], authors have transformed Clinical Information Modeling Initiative (CIMI) to ontologies, using SHACL to model and define rules with validation purposes. In [16], a collaborative ontology is created from information streams coming from different sources, both public and private, in the domain of environmental sensors. Semantic mapping, along with data enrichment, is performed. Data are validated using SHACL defining collaborative rules. For that purpose, authors have developed a framework named LSane. In [17], the quality evaluation of a couple of Knowledge Bases (KB) is performed based on completeness and consistency by using a semantic web framework. This evaluation has been performed using SHACL and SPARQL queries. Since these KBs are RDF-based and public, a statistical analysis has been realized. The main contribution of this work is the quality assessment of KBs subject to constant evolution. In [18], the first attempt to use SHACL for modeling and validating CGMES models has been realized but limited to RDF schema validation without addressing OCL rules and text-based CGMES rules.

The domain of this work is data exchange for utilities. As it is shown in previous paragraphs, there is a need from different domains to transform the modeling process and models into an ontological approach with RDF/OWL and SHACL. This work is focused on the most tedious task for this transformation between different approaches: transforming OCL rules to SHACL shapes. For transforming UML to RDF or OWL, OMG specifies this transformation in Ontology Definition Metamodel (ODM) and in MOF to RDF Mapping, which is a new beta specification. However, no work has been found so far concerning the transformation between OCL rules and SHACL. W3C has specified SHACL as standard for RDF validation. In this work a methodology is presented for: (i) the conversion of OCL constraints to SHACL dealing with key lexical, syntactic, and semantic elements in both grammars, (ii) the transformation of text-based rules to SHACL has been addressed both manually and semi-automatically given the ambiguity of natural language. Moreover, with this methodology, the authors have modeled/implemented all the CGMES rules, namely OCL and text-based ones. Additionally, they have developed a microservices-based application for model validation.

## III. CGMES STANDARD DOCUMENTS

"Common Grid Model Exchange Specification (CGMES) is an IEC Technical Specification (TS) based on the IEC CIM family of standards. It was developed to meet necessary requirements for TSO data exchanges in the areas of system development and system operation" [19].

ENTSO-E CGMES provides a set of documents for the standard specification. The Key documents that are of interest for this work are shown in Table 2.

- QoDCRules.xsd is an xsd file for defining rules model, see Fig. 6.

**TABLE 2. Primary CGMES documents.**

| Document | Purpose |
|---|---|
| Quality_of_CGMES_Datasets_and_Calculations.pdf | Rules description |
| QoDCRules.xsd | Rules model |
| QoDCRules level1.xml to level7 | Report validation structure |
| RDF schema files | RDFS for each profile |
| OCL files | OCL file for each profile |
| IEC Standards Documents for serialization | See Table 1 |

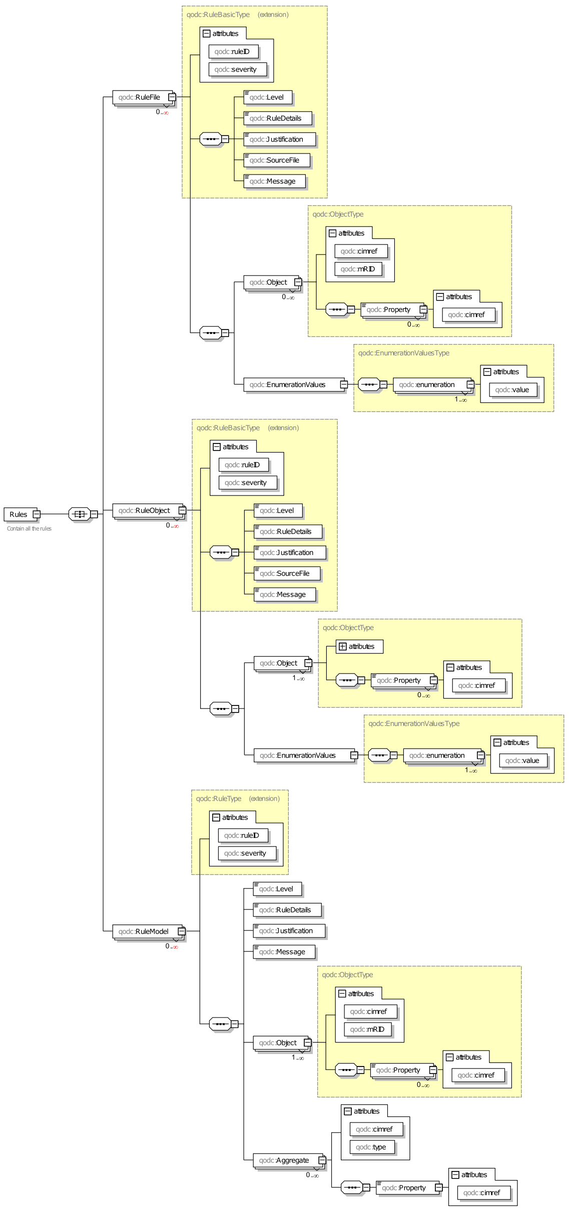


**FIGURE 6. CGMES rules model.**

- QoDCRules level1.xml to level7 define how validation reports should be produced for the seven validation levels.
- Resource Description Framework Schema (RDFS) and OCL files for models corresponding to Equipment (EQ) diagram layout, dynamics, equipment boundary, core operation, short circuit, geographical location, State Variables (SV), Steady State Hypothesis (SSH), topological boundary and Topology (TP) profiles.
- Two IEC standards used by CGMES for UML to RDF serialization are: (i) IEC 61970-552 [20] defines the document structure, header, syntax, and metadata. (ii) IEC 61970-501 [21] that defines the mapping of concepts between CIM (UML) and RDF.

## IV. WHY SHACL

In the context of modeling, especially in validating models for quality in interoperability when exchanging models, there is a need for a standard validation language. "SHACL is a language for validating RDF graphs against a set of conditions" [22]. W3C provides SHACL to allows for the model's validation against the rules. However, no standard validation language is provided for validating models defined in UML (Object-Oriented). A key aspect of MDE is the model's transformation. Thus, transforming UML/OCL model to RDF/SHACL and validating with semantic web technologies.

A particular case is the CIM standard, whose foundations lie on Object-Oriented techniques (UML) [23]. Therefore, models are class and object diagrams, and with the purpose of model serialization, CIM standard has adopted the W3C RDF language.

It is necessary to note a characteristic of high relevance: the "executability" of the model. It can be defined as follows. A model M formalized in a language L under a paradigm P is executable if there exists a processor that allows reasoning tasks on the model for a specific objective without any model transformation. (e.g. P = Object-Oriented, P = ontology, L = RDF; L = UML).

Since UML is a visual language, models formalized with it, are not directly executable. Therefore, it is necessary a transformation to another language that allows the executability feature. However, models defined in RDF are directly executable, in the sense that they support reasoning, see Fig. 7. ENTSO-E has created CIM profiles to specify the CGMES standard. These profiles are mainly class diagrams, together with additional rules written in OCL or text-based. With this specification, it becomes clear that the model is not directly executable. A crucial task such as validation in this context can be performed through: (i) Transforming the model to another paradigm that allows model executability. (ii) Using a general-purpose language, but the text-based and OCL rules require a considerable effort to be implemented, which negatively affects the software quality. (iii) Using Eclipse Modeling Framework (EMF) with OCL, but the returned validation report lacks elementary information since it contains only a reference to the object and the invariant; in essence, OCL is not a validation language.

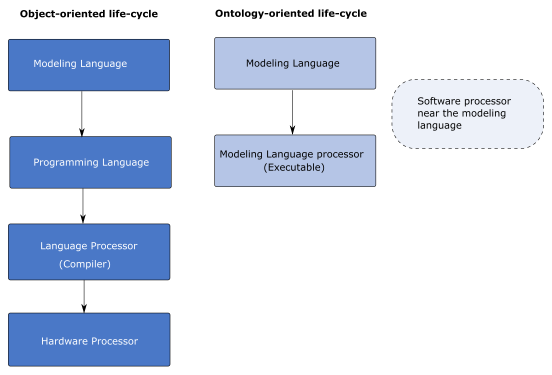

Object-oriented life-cycle
Ontology-oriented life-cycle
Modeling Language
Programming Language
Language Processor (Compiler)
Hardware Processor
Modeling Language
Modeling Language processor (Executable)
Software processor near the modeling language

**FIGURE 7.** Object-oriented life-cycle vs ontology-oriented life-cycle.

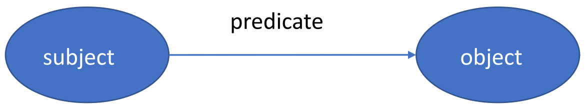


**FIGURE 8.** The RDF graph corresponding to (subject predicate object) triple.

W3C, in response to the community's need for a standard language for validation, has provided SHACL for this purpose.

The arguments for adopting SHACL are: (i) Incorporating SHACL as a standard validation language for the CGMES standard to solve the validation problem. (ii) The creation of an ontology as part of CGMES, representing the structural part in RDFS and the rules in SHACL. Therefore, providing a reusable and shared knowledge (reasoning, processing, querying) and being both machine-readable and human-readable, through semantic web technologies. (iii) RDF/SHACL models are directly executable, which translates into a considerable saving of time and effort in coding into a general-purpose programming language, and into an improvement of process and product quality, as this approach is more aligned with MDE [24]–[26] than UML/OCL.

Additionally, the authors have performed two comparisons concerning: (i) The presence of Object-Oriented and ontologies approaches in key modeling aspects in the CGMES standard, see Table 3. (ii) Features offered by the OCL and SHACL languages, see Table 4.

## V. BASIC MODELING OF A RULE IN SHACL. BASIC CONSTRUCTS

This section aims to illustrate the essential elements of SHACL from an architectural, syntactic, and semantic point of view. In the RDF world, everything is a resource. The atomic model in RDF is a triple (subject predicate object) that corresponds to an underlying graph, see Fig. 8.

[2]XML metadata interchange
[3]Universal resource identifier (URI)

**TABLE 3.** The implication of object-oriented and ontological paradigms in different CGMES modeling aspects.

| **CGMES modeling aspects** | **Ontological paradigm** | **Object-Oriented Paradigm** |
|---|---|---|
| Structural modeling | Out of scope of CGMES standard | UML |
| Serialization | RDF provides unique and persistent identifiers | • XMI[2] only for class diagrams<br>• Serialization of object diagram with relationships between objects is not allowed in (UML/XML)<br>• XML does not allow link between two elements that are not parent or child |
| Validation | SHACL/SPARQL (not used) | OCL rules are part of CGMES standard |
| Rules definition | Rules are not formalized in SHACL | OCL and text-based rules |
| Reasoning support | Yes | No |
| Metamodel | RDF SCHEMA or OWL (unintentional Ontology has been created) | Class diagram |
| Instances | RDF data | Object diagram |
| Standard for validation | SHACL (not used) | It is a standard for modeling but not for validation |

**TABLE 4.** SHACL vs OCL.

| **FEATURE** | **OCL** | **SHACL** |
|---|---|---|
| Reasoning | Not supported | Supported |
| Validation report | Only provides invariant and object | Object URI[3] and rule ID violated sh:message |
| Severity | Not supported | sh:warning, sh:Violation, sh:info |
| Dependent constraints | Not supported | Dependency between shapes is supported |
| Inconsistencies repair | No side effects | sh:rule constructs provide support to repair inconsistencies |
| Inter-model constraints | Not supported | SPARQL enables inter-model constraint |
| Flexibility in the context definition | Limited to meta-class | SPARQL-based targets provide flexibility in the context definition |
| URI/ID | It is not part of the Object-Oriented paradigm | Part of the RDF paradigm |

SHACL is a language to validate data models against restrictions. It should be noted that both the data and the language are represented in RDF language. Therefore, they are graphs. SHACL processor has two inputs, RDF data and

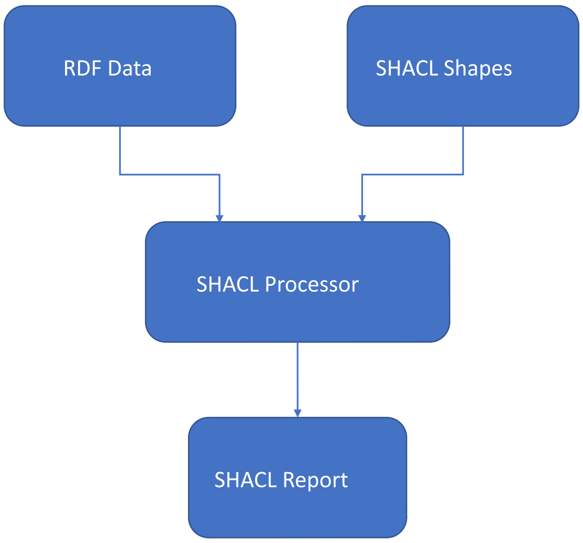


**FIGURE 9. Architecture of SHACL processor.**

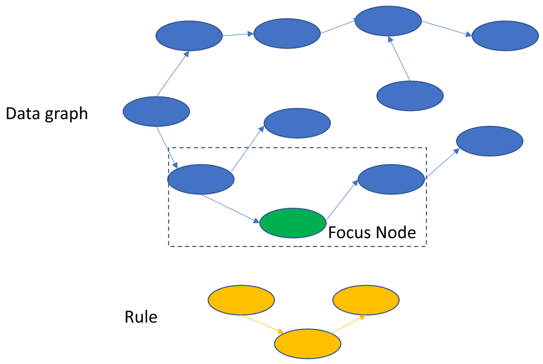


**FIGURE 10. SHACL processor traversing RDF graph for rule validation.**

```
:Peter  a :Person ;
        :idCode "123456789A" ;
        :idCode "123456789B" .
```

**LISTING 1. RDF data example.**

a set of rules. It returns a validation report in RDF, which contains the nodes of the data graph where the rules have been violated, see Fig. 9.

The SHACL processor traverses the graph, visiting the focus nodes to evaluate the rule. In case of non-compliance, information about the coordinates where the rules are violated is added to the validation report with additional information such as a feedback message indicated in the rule, see Fig. 10.

W3C has divided the SHACL specification into SHACL-CORE, which contains the basic constructs, and SHACL-SPARQL, which provides constructs for a more advanced expressivity, based on SPARQL.

The following example illustrates a simple rule, its implementation with SHACL-CORE, and the output of the validation report. The RDF data is described in Listing 1.

Where *:Peter* object of type *:Person*, is declared *(:Peter a :Person;)* with two identification codes *"123456789A"*

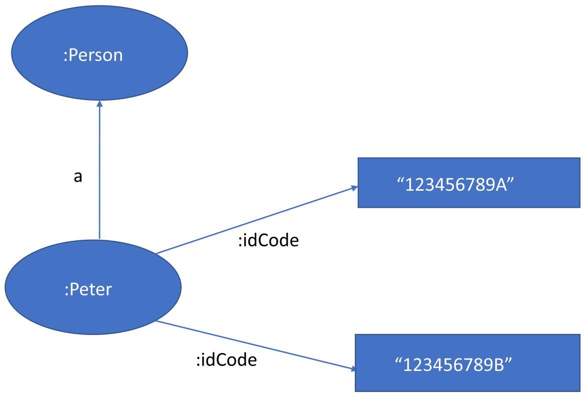


**FIGURE 11. The RDF graph corresponding to RDF code in listing 1.**

```
# The rule identifier is :uniqueCodePerson
:uniqueCodePerson       a sh:PropertyShape ;

        # The Class of the focus node
        sh:targetClass :Person ;

        # Non-conformance message
        sh:message "Only one value for idCode is allowed";

        # The constraint applies to the object of the :idCode predicate
        sh:path :idCode;

        #  The maximum number of values for :idCode must be 1
        sh:maxCount 1 ;

        #  The minimum number of values for :idCode must be 1
        sh:minCount 1.
.
```

**LISTING 2. SHACL rule for uniqueness of idCode.**

```
[ a      sh:ValidationReport ;
sh:conforms  false ;
sh:result     [ a         sh:ValidationResult ;
        sh:focusNode            :Peter ;
        sh:resultMessage        "Only one value for idCode is allowed" ;
        sh:resultPath           :idCode ;
        sh:resultSeverity       sh:Violation ;
        sh:sourceConstraintComponent    sh:MaxCountConstraintComponent ;
        sh:sourceShape          :uniqueCodePerson
        ]
] .
```

**LISTING 3. Validation report.**

and *"123456789B"*. The graph corresponding to the previous RDF is shown in Fig. 11.

In this rule sample, all instances of type "*:Person*" should have a unique value for "*:idCode*". SHACL code that defines the rule is depicted in Listing 2.

Listing 3 shows the validation report obtained once SHACL processor is executed with input data corresponding to Listing 1 (RDF data) and Listing 2 (SHACL shapes).

In the RDF validation report depicted in Listing 3, the RDF data violates the rule because there are two "idCodes" for the same person. The report shows violation coordinates in terms of the node and the rule where non-compliance has occurred. Additional information, such as severity and a feedback message, is presented.

## VI. METHODOLOGY TO TRANSFORM OCL AND CGMES RULES TO SHACL SHAPES

Object-Oriented and ontological paradigms are formalized by UML/OCL and RDF/SHACL, respectively.

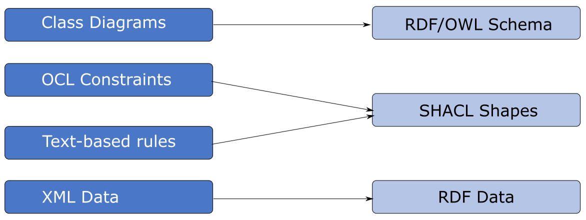


**FIGURE 12.** Transformation from object-oriented (UML + OCL) and text-based rules to (RDF + SHACL).

The transformation from UML/OCL to RDF/SHACL is shown in Fig. 12, and involves the following steps:

1. Class diagrams are transformed into RDF/OWL schemas by mapping UML concepts to RDF concepts.
2. OCL constraints are transformed into their SHACL shapes counterparts.
3. Text-based rules are defined in SHACL constraints
4. UML Objects are converted to RDF data.

The focus of this work is on transforming OCL and text-based rules to SHACL.

### A. OCL TO SHACL

This section aims to illustrate how to perform the transformation from OCL to SHACL by presenting the key ideas that facilitate UML transformation with semantic web technology applied to the domain of electrical utilities (CGMES).

Some requirements must be taken into consideration during the activity of the transformation from OCL to SHACL as follows:

a. How the SHACL processor runs through the data graph with the constraint graph.
b. The main task is to produce triples from the components of the OCL expressions.

An invariant in OCL is defined as follows:

"Context ClassName inv: invariantName oclExpression" defines a constraint identified by *invariantName*, and where *ClassName* represents the name of the class of the objects on which the boolean condition indicated in *oclExpression* is applied.

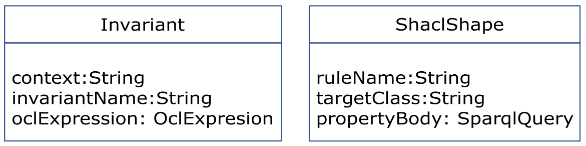


**FIGURE 13.** UML classes for OCL invariant and SHACL shape.

An OCL invariant and a SHACL shape can be represented by the UML classes in Fig. 13.

The mapping between Invariant and ShaclRule is presented in Fig. 14.

The process of transforming OCL to SHACL consists of the following stages:

1. Extract the main concepts from the OCL invariant and build with them the initial structure of the

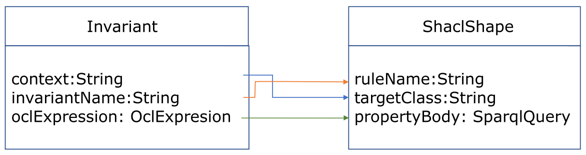


**FIGURE 14.** Mapping between OCL invariant and SHACL shape.

```
Ocl2Shacl(ocl_invariant,CIM_UML_RDF_TABLE, schemaDomain)
        # Initial structural transformation
        ruleName<- invariantName;
        targetClass<-context
        # oclExpresion to propertyBody transformation
        shaclShape<-
                "qodc:"+ruleName + " a sh:NodeShape;"
                # uml2rdf(umlConcept) is a simple function that returns
                # the rdf concept corresponding to uml concept
                # from table CIM_UML_RDF_TABLE
                "sh:targetClass " +  uml2rdf(context) " ;" +
                "sh:sparql [\n" +
                        " a sh:SPARQLConstraint ;\n" +
                                oclExpression2Sparql(oclExpression,
                                CIM_UML_RDF_TABLE, schemaDomain) +
                "]."
End
```

**LISTING 4.** Transformation from OCL invariants to SHACL algorithm.

**TABLE 5.** uml2rdf an excerpt from the domain concept mapping table between UML and RDF in CIM.

| CIM-UML concept | CIM-RDF concept |
|---|---|
| ... | ... |
| IEC61970::Base::Wires::PowerTransformer.name | cim:IdentifiedObject.name |
| ... | ... |

SHACL rule, as illustrated in the *Ocl2Shacl()* algorithm shown in Listing 4. As OCL is used to define constraints on UML class diagrams, and SHACL is a language to validate RDF data against RDF shapes graphs, the *Ocl2Shacl()* algorithm needs as a parameter *CIM_UML_RDF_TABLE*, a table that contains the mapping of the domain concepts between UML and RDFS. The mapping, which is an inherent part of the CGMES standard, is shown in Table 5.

2. The mapping between OCL and SHACL concepts lies in the observation that the primary or main element of an OCL expression is navigability, and it starts from the object whose type is indicated by default in the *context* statement of the OCL constraint, or explicitly, such as *Class.allInstances().count()* where *Class* is any class in the class diagram that represents the model.

The main idea or task is the creation of triples (subject predicate object) from OCL expressions of the form *"a.x"* representing the navigability starting from an object on which the rule is applied to the feature *x*. The *oclExpression2Sparql()* algorithm illustrates a high-level way to realize this task (see Listing 5). Table 5 shows the semantic and syntactic mapping between OCL and SHACL. Listing 6 depicts an excerpt of OCL grammar, defined by OMG, and Fig. 15 shows its corresponding abstract syntax where the primary element has the following form:

exp ::= a opNav b (params)? (where opNav is '.' or '− >')

```
oclExpression2Sparql(oclExpression, CIM_UML_RDF_TABLE, schemaDomain)
.....
        # OCL to RDF transformation using navigability.
        foreach a_i  in a_1.a_2.....a_i.a_{i+1}.....a_{n-1}.a_n
                # create triple as follows
                ?a_i        a_i.a_{i+1}     ?a_{i+1}
        end foreach
.....
# For example a transformation of an expression like
#  a_1.a_2.....a_i.a_{i+1}.....a_{n-1}.a_n > 500 is:
# FILTER (?a_n > 500)

        FILTER (boolean condition with the corresponding pattern elements)

End
```

**LISTING 5. Transformation from OCL expression to SPARQL query.**

```
OclExpressionCS ::= CallExpCS
CallExpCS ::= FeatureCallExpCS
FeatureCallExpCS ::= OperationCallExpCS
FeatureCallExpCS ::= PropertyCallExpCS
FeatureCallExpCS ::= NavigationCallExpCS

OperationCallExpCS ::= OclExpressionCS[1] simpleNameCS OclExpressionCS[2]
        | OclExpressionCS '->' simpleNameCS '(' argumentsCS? ')'
        |OclExpressionCS '.' simpleNameCS '(' argumentsCS? ')'
        | simpleNameCS '(' argumentsCS? ')'
        |OclExpressionCS '.' simpleNameCS isMarkedPreCS '(' argumentsCS? ')'
        |simpleNameCS isMarkedPreCS '(' argumentsCS? ')'
        | pathNameCS '(' argumentsCS? ')'
        | simpleNameCS OclExpressionCS
        | OclExpressionCS '.' pathNameCS '::' simpleNameCS '(' argumentsCS?
               ')'
        | OclExpressionCS '.' pathNameCS '::' simpleNameCS isMarkedPreCS '('
               argumentsCS? ')'

PropertyCallExpCS ::= OclExpressionCS '.' simpleNameCS isMarkedPreCS?
        | simpleNameCS isMarkedPreCS?
        | pathNameCS
        | OclExpressionCS '.' pathNameCS '::' simpleNameCS isMarkedPreCS?

NavigationCallExpCS ::= PropertyCallExpCS
        | AssociationClassCallExpCS

AssociationClassCallExpCS ::= OclExpressionCS '.' simpleNameCS ('['
       argumentsCS ']')? isMarkedPreCS?
        | simpleNameCS ('[' argumentsCS ']')? isMarkedPreCS?
argumentsCS[1] ::= OclExpressionCS ( ',' argumentsCS[2] )?
simpleNameCS ::= NameStartChar NameChar*
        | '_' #x27 StringChar* #x27
simpleNameCS[1] ::= simpleNameCS[2] WhiteSpaceChar* #x27 StringChar* #x27

NameStartChar ::= [A-Z] | "_" | "$" | [a-z]
| [#xC0-#xD6] | [#xD8-#xF6] | [#xF8-#x2FF]
| [#x370-#x37D] | [#x37F-#x1FFF]
| [#x200C-#x200D] | [#x2070-#x218F] | [#x2C00-#x2FEF]
| [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD]
| [#x10000-#xEFFFF]
```

**LISTING 6. An excerpt of OCL grammar specified by OMG.**

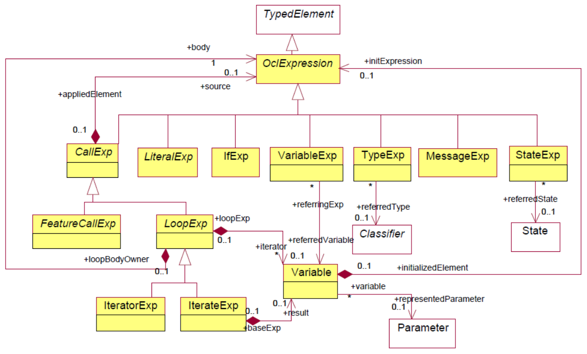


**FIGURE 15. The basic structure of the abstract syntax kernel metamodel for expressions (OMG).**

## *B. FROM TEXT-BASED RULES TO SHACL*

The algorithm's parameters are a text-based rule and a schema domain; Text-rule is specified in natural language in the context of the domain, in this case, electrical utilities;

```
textBasedToShaclManual(text_rule,schemaDomain)
        TBR = {is a set of the concepts in the Text-Based Rule}
        CC = {CIM Concepts initially empty}
        RC = {set of pair of related concepts, initially empty}
        Foreach c in TBR
                # dc is the domain concept defined in RDF schema.
                # rdfsConcept is a manual task performed by the model expert
                # to find the RDF concept that corresponds to textual
                # concept c.

                dc= rdfsConcept (c)
                CC->add(dc)
        End Foreach

        # constructPattern is a modeler manual task which consists of
        # searching for a subgraph that interconnects the concepts in CC.

        pattern = constructPattern(CC, schemaDomain)

        # getOperationsInTextBasedRules is a modeler manual task which
        # consists of extracting from text_based the operations and their
        # associated operands involved in the rule.

        OO = {(Operation,{Operands})} = getOperationsInTextBasedRules(
               text_rule)

        # createSHACLShape is a modeler manual task which consists of
        # building SHACL shape from the pattern and the operations stated
        # in text-based rule

        SHACLShape = createSHACLShape (pattern,OO)
End textBasedToShaclManual
```

**LISTING 7. Text-based to SHACL manual algorithm.**

```
textBasedToShaclSemiAutomatic(text_rule,schemaDomain)

        # First, similarity-based processing is performed to determine the
        # RDF schema concepts involved in the rule SC = { is a set of
        # RDF schema concepts}

        SC = getSimilarities(text_rule, schemaDomain);

        # The domain expert intervenes to select the most appropriate match
        # with the task selectAppropriateConcepts

        CC = selectAppropriateConcepts(SC);

        # Second, the pattern corresponding to the rule is built by searching
        # for a subgraph  that interconnect the concepts found in the
        # previous step using any Pathfinding algorithm over schemaDomain
        # graph

        SG=getSubGraph(CC, schemaDomain);

        # With SG, the modeler builds the SHACL shape

End textBasedToShaclSemiAutomatic.
```

**LISTING 8. Text-based to SHACL semi-automatic algorithm.**

The schema domain is the metamodel domain defined in RDF. Concepts in the text-based rule are translated to CIM concepts and are connected using the schema domain to build the pattern that corresponds to the original rule.

The conversion from the concept of the domain in natural language to its corresponding one in CIM/XML can be realized with two approaches: (i) manual, with the collaboration of the domain expert or (ii) semi-automatic using Natural Language Processing NLP technologies in two steps. First, similarity-based processing is performed to determine the RDF schema concepts involved, and the domain expert intervenes to select the most appropriate match. Second, the pattern corresponding to the rule is built by searching for the subgraphs that interconnect the concepts found in the previous step and involving the domain expert to select the most appropriate pattern.

According to these two approaches, *textBasedToShaclManual()* describes the manual method (see Listing 7), and *textBasedToShaclSemiAutomatic()* describes the semi-automatic method (see Listing 8).

## VII. TOOL DEVELOPMENT FOR VALIDATING CGMES RULES

Authors have developed a Model View Controller (MVC) web application as a set of microservices to validate models from utilities. Hypertext Markup Language 5 (HTML5) is used to implement the view part; the model layer is represented through RDF/RDFS/SPARQL/SHACL; the controller part consists of a set of Java classes that manage Hypertext Transfer Protocol (HTTP) requests and perform validation tasks invoking SHACL processor.

Fig. 16 shows the architectural layer of the developed application.

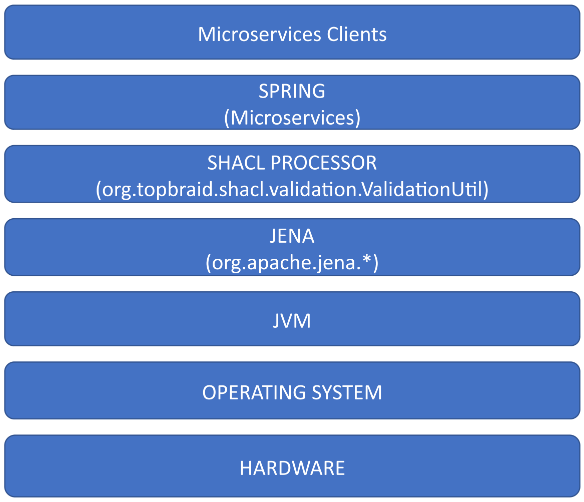


**FIGURE 16.** Layers architecture for developed application.

A more detailed architecture, representing the application components and the communication among them is shown in Fig. 17.

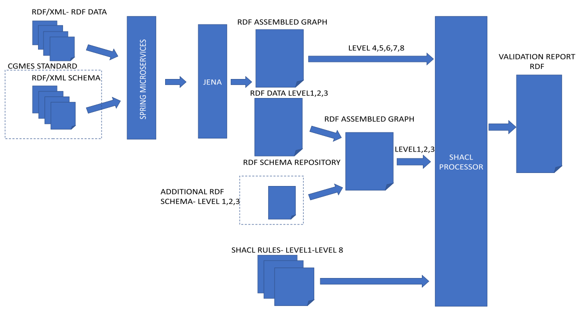
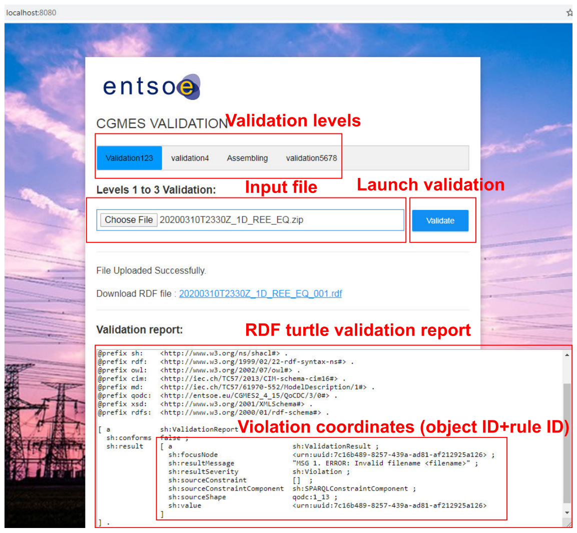


**FIGURE 17.** Detailed architecture of the application.

For validation purposes, the system uses RDF/XML and schema as input data, representing domain models from electrical utilities. Validation levels have been grouped into three categories: (i) A microservice copes with validating RDF files at levels 1, 2, and 3. (ii) A second microservice deals with level 4 validation (OCL). (iii) Finally, a third microservice deals with levels 5 to 8. The reason for this separation is that levels 1, 2, and 3 deal with file naming, XML well-formedness and XML structure, namespaces, headers, and file packaging. Level 4 deals with OCL rules defined by the standard. Levels 5 to 8 deal with cross-profile consistency, business rules related to load flow, cross Individual Grid Model (IGM) consistency, and Common Grid Model (CGM) plausibility.

A microservice is provided for the assembling process, which consists of producing a graph from the four files that represent an IGM (EQ, TP, SV, and SSH together with the Boundary Set) and the corresponding RDFS.

The user interface of the application is shown in Fig. 18.



Levels 1 to 3 Validation:

20200310T2330Z_1D_REE_EQ.zip

File Uploaded Successfully.

Download RDF file : 20200310T2330Z_1D_REE_EQ_001.rdf

Validation report:

```
@prefix sh:    <http://www.w3.org/ns/shacl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix cim:   <http://iec.ch/TC57/2013/CIM-schema-cim16#> .
@prefix md:    <http://iec.ch/TC57/61970-552/ModelDescription/1#> .
@prefix qodc:  <http://entsoe.eu/CGMES2_4_15/QoCDC/3/0#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

[ a            sh:ValidationReport ;
  sh:conforms  false ;
  sh:result    [ a                             sh:ValidationResult ;
                 sh:focusNode                  <urn:uuid:7c16b489-8257-439a-ad81-af212925a126> ;
                 sh:resultMessage              "MSG 1. ERROR: Invalid filename <filename>" ;
                 sh:resultSeverity             sh:Violation ;
                 sh:sourceConstraint           []  ;
                 sh:sourceConstraintComponent  sh:SPARQLConstraintComponent ;
                 sh:sourceShape                qodc:1_13 ;
                 sh:value                      <urn:uuid:7c16b489-8257-439a-ad81-af212925a126>
               ]
] .
```

**FIGURE 18.** Application user interface.

Another objective of this application is to evaluate (i) The contribution of adopting RDF/SHACL within the MDE approach. (ii) The impact of using models that are directly executable in software development. (iii) The improvement in scalability, maintainability, and agility, among others, in the software process.

For developing an application in electrical utilities context with this approach, it is necessary: (i) Background in CIM, UML, RDF/S, SPARQL, and SHACL. (ii) The mapping of concepts from natural language to CIM concepts. (iii) The model and semantics of a rule. (iv) The navigability within the metamodel (RDFS) and the models (RDF) of the domain. (v) Graph pattern building, corresponding to the rule that will be translated into a SPARQL query which will be embedded into SHACL.

For developing the application, Spring microservices, SHACL validation library (developed with JENA library), HTML5, and a Javascript library are required.

## VIII. CONCLUSION AND FUTURE WORKS

In this work, a standardized solution for the validation problem has been presented for UML/OCL-based models, adopting SHACL as the validation language. To this purpose, a methodology for converting OCL and text-based rules to SHACL has been specified. Since CGMES is based on UML/OCL, a rule transformation to RDF/SHACL is required.

The arguments for adopting SHACL as a validation language have been presented through a comparison between the two approaches (UML/OCL vs. RDF/SHACL) considering essentials criteria for the model validation task such as violation reporting, reasoning, among others.

The basic concepts and constructs of SHACL have been presented, as a first approach to the semantic and syntactic foundations of this validation language within the modeling domain.

The methodology for the transformation from OCL and text-based rules to SHACL has been presented as algorithms. For the conversion from OCL to SHACL, key elements have been considered in lexical, syntactic, and semantic terms within the scope of the two grammars. The text-based transformation to SHACL has been addressed from the linguistic perspective towards the syntax and semantics of SHACL.

Authors have implemented all the rules, defined by CGMES, in SHACL. They have also developed a web application with Spring microservices to validate the eight validation levels of CGMES specifications. They will make them available to the user community, both electrical and software engineering.

The contributions presented in this work are: (i) Transforming OCL and text-based rules to SHACL, a task never addressed before, through a methodology that allows a systematic implementation of the rules. (ii) Providing ENTSO-E stakeholders with an ontology that enables them to validate their models. Additionally, it can help as a guide for their rule implementation. (iii) Making available a web application, developed by the authors, to the community of electrical utilities, to validate CGMES models against the eight levels of validation specified in the standard. (iv) Bridging the gap between Object-Oriented and ontological paradigms by addressing the conversion of OCL to SHACL, since a considerable effort is being realized from different communities to foster the adoption of ontological engineering through semantic web technologies.

This work has focused on the CGMES standard release that is currently adopted by ENTSO-E members. A minor limitation of this work is the optimization of rules expressed in SHACL, in terms of performance for SHACL processor, and implementing new CGMES standard releases in the short term.

To further our research, we plan to develop a Domain-Specific Language (DSL) in the electrical utilities domain, enabling experts to write text rules to be transformed into SHACL automatically.

## REFERENCES

[1] OMG. (2017). *About the Unified Modeling Language Specification Version 2.5.1*. Accessed: Sep. 9, 2020. [Online]. Available: https://www.omg.org/spec/UML/About-UML/

[2] OMG. (2017). *About the object constraint language specification version 2.4*. Accessed: Sep. 9, 2020. [Online]. Available: https://www.omg.org/spec/OCL/About-OCL/

[3] M. Uschold, Y. Ding, and P. Groth. *Demystifying OWL for the Enterprise*. San Rafael, CA, USA: Morgan & Claypool, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8374125

[4] D. E. Khelladi, R. Bendraou, R. Hebig, and M.-P. Gervais, "A semi-automatic maintenance and co-evolution of OCL constraints with (meta)model evolution," *J. Syst. Softw.*, vol. 134, pp. 242–260, Dec. 2017.

[5] M. Hammad, T. Yue, S. Wang, S. Ali, and J. F. Nygård, "IOCL: An interactive tool for specifying, validating and evaluating OCL constraints," *Sci. Comput. Program.*, vol. 149, pp. 3–8, Dec. 2017.

[6] M. Gogolla, F. Hilken, and K.-H. Doan, "Achieving model quality through model validation, verification and exploration," *Comput. Lang., Syst. Struct.*, vol. 54, pp. 474–511, Dec. 2018.

[7] N. Przigoda, P. Niemann, J. G. Filho, R. Wille, and R. Drechsler, "Frame conditions in the automatic validation and verification of UML/OCL models: A symbolic formulation of modifies only statements," *Comput. Lang., Syst. Struct.*, vol. 54, pp. 512–527, Dec. 2018.

[8] B. Pérez and I. Porres, "Reasoning about UML/OCL class diagrams using constraint logic programming and formula," *Inf. Syst.*, vol. 81, pp. 152–177, Mar. 2019.

[9] M. S. Ángel, J. de Lara, P. Neubauer, and M. Wimmer, "Automated modelling assistance by integrating heterogeneous information sources," *Comput. Lang., Syst. Struct.*, vol. 53, pp. 90–120, Sep. 2018.

[10] A. G. B. Tettamanzi, C. Faron-Zucker, and F. Gandon, "Possibilistic testing of OWL axioms against RDF data," *Int. J. Approx. Reasoning*, vol. 91, pp. 114–130, Dec. 2017.

[11] Y. Shu, "A practical approach to modelling and validating integrity constraints in the semantic Web," *Knowl.-Based Syst.*, vol. 153, pp. 29–39, Aug. 2018.

[12] J. E. L. Gayo, E. Prud'hommeaux, I. Boneva, and D. Kontokostas, *Validating RDF Data*. San Rafael, CA, USA: Morgan& Claypool, 2017.

[13] J. E. L. Gayo. (2017). *Rdfshape: RDF Playground*. Accessed: Sep. 9, 2020. [Online]. Available: http://rdfshape.herokuapp.com/about

[14] J. C. Vidal, T. Rabelo, M. Lama, and R. Amorim, "Ontology-based approach for the validation and conformance testing of xAPI events," *Knowl.-Based Syst.*, vol. 155, pp. 22–34, Sep. 2018.

[15] C. Martínez-Costa and S. Schulz, "Validating EHR clinical models using ontology patterns," *J. Biomed. Informat.*, vol. 76, pp. 124–137, Dec. 2017.

[16] M. T. Frank, S. Bader, V. Simko, and S. Zander, "Lsane: Collaborative validation and enrichment of heterogeneous observation streams," *Procedia Comput. Sci.*, vol. 137, pp. 235–241, Dec. 2018.

[17] M. R. A. Rashid, G. Rizzo, M. Torchiano, N. Mihindukulasooriya, O. Corcho, and R. García-Castro, "Completeness and consistency analysis for evolving knowledge bases," *J. Web Semantics*, vol. 54, pp. 48–71, Jan. 2019.

[18] K. R. Nenadic, M. M. Gavric, and V. I. Durdevic, "Validation of CIM datasets using SHACL," in *Proc. 25th Telecommun. Forum (TELFOR)*, Nov. 2017, pp. 1–4.

[19] ENTSO-E. (2019). *Common Information Model*. Accessed: Sep. 9, 2020. [Online]. Available: https://www.entsoe.eu/digital/common-information-model/

[20] (2016). *IEC 61970-552:2016*. Accessed: Sep. 9, 2020. [Online]. Available: https://webstore.iec.ch/publication/25939.

[21] (2006). *IEC 61970-501:2006*. Accessed: Sep. 9, 2020. [Online]. Available: https://webstore.iec.ch/publication/6215.

[22] W3C. (2017). *Shapes Constraint Language (shacl)*. Accessed: Sep. 9, 2020. [Online]. Available: https://www.w3.org/TR/shacl/

[23] (2016). *IEC 61970-301:2016*. Accessed: Sep. 9, 2020. [Online]. Available: https://webstore.iec.ch/publication/31356.

[24] F. S. Parreiras, *Semantic Web Model-Driven Engineering*. Hoboken, NJ, USA: Wiley, 2012.

[25] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*. San Rafael, CA, USA: Morgan & Claypool, 2017.

[26] D. Gasevic, D. Djuric, and V. Devedzic, *Model Driven Engineering and Ontology Development*. 2009, doi: 10.1007/978-3-642-00282-3.

**MOHAMED LARHRIB** received the M.S. degree in computer engineering from the Universidad Nacional de Educacion a Distancia, Spain, in 2015. His research interests include software engineering, power systems data exchange formats, semantic Web, common information model, and model driven engineering.

**MIGUEL ESCRIBANO** received the M.S. degree in industrial engineering and the Ph.D. degree from the Polytechnic University of Madrid, Spain, in 1999 and 2005, respectively. Since 2005, he has been an Assistant Professor in financial economics, statistics and operations research, and actuarial science with the Universidad Complutense de Madrid, Spain, and he has been a Power Engineer with Red Eléctrica de España since 2008. His research interests include software engineering, power systems data exchange formats, semantic Web, and common information model.

**CARLOS CERRADA** received the M.S. degree in industrial engineering and the Ph.D. degree from the Polytechnic University of Madrid, Spain, in 1983 and 1987, respectively. He was a Fulbright Scholar at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, from 1989 to 1990. He is currently a Full Professor with the Department of Systems and Software Engineering, Universidad Nacional de Educacion a Distancia, Spain. His research interests include software engineering, robotics, pattern recognition, 3D object representation, and ubiquitous computing. He is a member of the IFAC.

**JUAN JOSE ESCRIBANO** received the M.S. degree in industrial engineering from the Polytechnic University of Madrid, Spain, in 1995, and the Ph.D. degree from the Universidad Nacional de Educacion a Distancia, Spain, in 2003. Since 2001, he has been an Assistant Professor with the Department of Systems and Software Engineering, Universidad Nacional de Educacion a Distancia, Spain. His research interests include software engineering, semantic Web, and common information model.