Developing Data Quality Metrics for Power System Modeling

Merlin Bögershausen - 317962 -
merlin.boegershausen@rwth-aachen.de

Supervisor: Prof. Dr. Stefan Decker
2nd Supervisor: Prof. Dr. Bernhard Rumpe

Advisor: Dr. Oya Deniz Beyan, Dr. Christoph Lange and Dr. Oliver Scheufeld

A thesis presented for the degree of
Bachelor of Science

The present work was submitted to:
Chair of Computer Science 5 - Information Systems
RWTH Aachen
Germany
10.June 2020

Contents

Abstract

The electrical power system is part of the European critical infrastructure. The European Network of Transmission System Operators for Electricity (ENTSO-E) coordinates cross border cooperation, exchange of information and empowers the integration of the 43 Transmission System Operators (TSO) around Europe. To ensure high reliability of the whole system, high-quality information about the system state is crucial. The Common Grid Model Exchange Specification (CGMES) provides the exchange format for the individual grid model (IGM) of each TSO. A Common Grid Model (CGM) is the combination of IGMs and provide all needed information for operational and planning processes with varying quality. However there is no automated method for execution of these quality specifications. Currently, the quality of information reviewed manually, and due to the complexity of the represented information, the process is slow and error prone. This work presents a scalable and efficient way to assess the quality of IGMs and CGMs represented in the CGMES format, which is based on RDF Schema and a subset of UML. The approach uses semantic web technologies to assess the quality of CGMES data in a semi-automatic way. It produces SHACL shapes, which can assess the data quality of most electrical power system models, but we also point out limitations of SHACL in this setting. The outcome of this work can be adopted by ENTSO-E members for improving the planning software by validating the quality of incoming data from different sources.

Chapter 1

Problem Description

The first chapter provides an introduction to the aspects of interconnected power systems, the demands on them and derives the aim of the thesis.

A stable and secure electrical power system in Europe is the aim of Regulation (EC) No. 714/2009 of the European Parliament and of the Council of 13 July 2009 on *conditions for access to the network for cross-border exchanges in electricity and repealing Regulation* [7] from 2009. As a reaction to this EU regulation the European transmission operators founded the European Network of Transmission System Operators for Electricity (ENTSO-E). Since July 2009 the ENTSO-E has been acting as the central institution for coordination of the implementation of the EU's energy policy. Regulation (EU) 2019/943 of 5 June 2019 on *the internal market for electricity* [8] extended the mandate of the ENTSO-E and makes information exchange more important.

1.1 Electrical Power System

The European electrical power system splits into many control zones each managed by one transmission system operator organization (TSO). These control zones form individual grid models (IGM) and merge into a common grid model (CGM). The grid model divide information into different categories:

- Physical Equipment includes Terminals, Lines and Generations.
- Topology Information defines the topology of the system. Topological nodes and edges represent the physical equipment.
- Electrical Information describes values such as resistance, minimal or maximal values of load, voltage or temperature
- General State Information describes states for the system and subsystems

The information within the CGM and the current market situation serve as input for regulation and planning processes. Due to the complexity of this system, its analysis and optimization are only possible with the help of decision-making software, which requires high-quality data for high-quality results.

1.2 Exchange Of Power System Models

Developing an efficient merge and exchange process for IGMs is one of the goals of ENTSO-E and their predecessors. One predecessor of ENTOS-E released in 2003 the ASCI based *UCTE data exchange format for load flow and three-phase short circuit studies* (UCTE-DEF) [17]. Each line of a UCTE-DEF file defines one instance of a specific type listed below:

1. Nodes: are 129 characters long and consist of an eight characters name followed by 17 data fields
2. Lines: have nine data fields and is 65 characters in length
3. 2-Windings Transformer: 90 characters total length with 13 data fields
4. Exchange Powers: four data fields in a 26 character string

The data fields have an assigned datatype and resolution. This leads to straightforward checks. These checks are validity checks and no quality evaluations. The result only determines the fitness for purpose. Extending the released definition with new components is impossible.

To be able to have a high model resolution and the ability to add new power system components easily, the *Common Grid Model Exchange Standard* (CGMES) [4] based on the Resource Description Framework (RDF) is currently in adoption and test phase. It is expected to be the productive exchange format in 2021. The extendable basic CGMES model consists of seven packages:

1. Core: contains the domain's base classes, including *IdentifiedObject*, *ACDCTerminal* and *EquipmentContainer*
2. Topology: defines components that describe the underlying topology of the system, like *ConnectivityNode*
3. Wires: extends core and topology packages and adds electrical characteristics, like *CurrentFlow* and *VoltageLevel*
4. DC: contains direct current components, *DCTerminal* as an extension
5. Operational Limits: defines limits for power system equipment
6. Load Model: defines energy consumers and loads for equipment

Table 1.1: Prefix bindings often used with the CGMES standard

Prefix	URL
cim:	`http://iec.ch/TC57/2013/CIM-schema-cim16#`
cims:	`http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#`

7. State Variables: introduces concepts to describe the state of the power system

Each CGMES class has a set of mandatory and optional RDF properties. As a consequence of the inheritance hierarchy, these sets overlap. Additional properties and classes are easy to define.

Together with CGMES, the ENTSO-E also released quality definitions for power system modelling [5]. These standards define quality as fitness for purpose for a particular process. These processes cover a wide range from planning to operation to historical analysis. Each process comes with different requirements and therefore different specifications of the quality requirements. The quality requirements span seven levels, the first three of which address file structure and naming. The following two focus on to constraints to objects and the consistency – this is the focus of this work. The last two deal with robustness and cross IGM inconsistencies. The assessment of data quality w.r.t. these requirements is still not solved.

To shorten the IRIs, we use well-known prefixes as defined by `http://prefix.cc/` and the two additional CGMES related prefixes given in Table 1.1. The *cims:* prefix is the RDFS extension made by the IEC. This extension defines multiplicities and other UML terms that are not part of RDFS. The *cim:* prefix is the default prefix for the CGMES model.

1.3 CGMES Quality Assessment Use Case

Quality assessment is crucial for processes based on CGMES because the quality defines the fitness for purpose. The CGMES processes give rise to two main use cases for quality assessment. **UC1 (import):** At the beginning of a process, the dataset needs to be compliant; this happens once per process instance. So it is necessary to assess each triple in the independent IGM contained in the CGM once. **UC2 (manipulation):** data manipulations by TSO engineers have to preserve the quality. As consequence each manipulation triggers a re-assessment of the changed data; this happens multiple times per process instance. Typical manipulations are updates of connections or load values; the amount of triple to check is below 100.

1.4 Aim of the Thesis

This work extracts the quality requirements from the Quality of CGMES and Definition of CGMES [4, 5] groups them and tries to find a categorization. A formal definition of the rule evaluating the requirement category is transformed into a quality metric. To evaluate the metrics they are implemented with the Shape Constraint Language [14] (SHACL[1]) and executable with a standard-compliant SHACL interpreter. From some metrics, pure SHACL is not sufficiently expressive. This work discusses how the implementation of such metrics become possible with extensions to SHACL. The remainder of this thesis is structured as follows: Chapter 2 discusses related work and the evaluation of data quality. It also gives a short introduction to the fundamental concepts of the technologies in this work. Chapter 3 explains the extraction of quality requirements and test datasets from the CGMES specification. Chapter 4 analyses the topic of data quality with respect to CGMES and extract quality requirements. Chapter 5 showd the implementation of the proposed solution. Chapter 6 evaluates the performance of the solution in a realistic setting and give a correctness proof. Chapter 7 concludes the results and outlines the planned adoption and ideas for further work.

[1] http://w3.org/TR/shacl/

Chapter 2

Related Work

Since the first mention of linked data by Tim Burners Lee the quality of such data is of interest, but there is no conses until today on how to evaluate. Since RDF became a W3C recommendation together with SPARQL in 1997, approaches to fulfil the requirements for a validation framework are proposed from time to time.

This section gives an overview of related work in the field of data quality. The end gives a short introduction into the used technologies of his work.

2.1 Definition of Data Quality

Weidemann et al. [26] characterized five different dimensions of data quality and used them together with a pedigree matrix to describe the data quality assessment result. The dimensions can be categorized into two groups: study independent ones (*reliability* and *completeness*), and *temporal correlation*, *geographical correlation* and *technological correlation*, which depend on the study performed.

Pundt [24] showed that the perception of data quality differs across consumers. Different interpretation and weighting of data quality levels led to this observation. The different usage of the same structure by two different producers or consumers intensifies this effect.

Pipino et al. [20] showed once more that data quality is multidimensional and dependent on the task to perform. They concluded that there is no "one size fits all". The authors observed that people are more likely to trust a single result than a set of results. To achieve single results they proposed aggregations on metrics to understand data quality easier. They proposed *Simple Ratios* for freeness of errors, completeness and consistency; *Min/Max* of believably, amount of data, timeliness and accessibility and *Weighted Average* as an alternative to min/max but the data need to be normalized.

Fürber and Hepp [10] worked on data quality on the Semantic Web. They defined generic SPARQL queries to detect missing datatype annota-

tion, illegal values and functional dependencies within RDF data.

In their survey about linked data quality, Zaveri et al. [27] defined the terminology and compared different approaches for data quality assessment of linked data. Within six proposed dimension groups, they gathered 23 dimensions of linked data quality and defined, within these dimensions, well-formed metrics.

- Intrinsic dimensions are independent of the use case and indicate the general usability of a dataset. All metrics of this group are objective, and therefore does not need information from the end-user.
- Accessibility dimensions are metadata quality as they centre around availability and the type of availability of datasets. The data to evaluate these metrics are with the dataset or is not present at all.
- Representational Dimensions are trying to measure how "good" the representation of the data is and how good the representation is understandable. Not all metrics in this group are objective.
- Contextual Dimensions are study dependent and data amount dependent. They attempt to approximate the solve-ability of the task at hand with the dataset at hand.
- Dataset dynamic Dimensions are time-related and depend on the point in time where the study takes place. Because the metrics in this dimension only take the presence of data into account, the evaluation is objective.
- Trust Dimensions depend on personal opinion and need a prior servery to achieve a correct weighting. Nearly half of the metrics in this dimension group are objective.

2.2 ISO/EIC 25012 Data Quality in Software Products

The ISO[1] in cooperation with IEC[2] standardized data quality in the context of software development. The ISO/IEC 25012 [12] standard organizes dimensions into *Inherent Data Quality*, a *System-Dependent Data Quality* and a group between. The standard gives a basic structure for data quality in general.

- Accuracy shows whether the data has attributes that represent values correctly. It splits into syntactic and semantic accuracy which defines the precision of the model in different levels.

[1] http://www.iso.org
[2] http://www.iec.com

- Completeness defines if each subject of an entity can represent all expected values.
- Consistency shows whether the data is free of contradictions and if it is coherent with other data.
- Credibility is the degree to which data has properties considered as true, including the concept of authenticity.
- Currentness shows whether the data is up to date with an accepted delay.
- Availability checks whether authorized users or systems can retrieve the data.
- Portability shows whether the data can be installed, replaced or moved into other systems.
- Recoverability is the degree to which the data maintain a minimum level of quality even in failure cases .
- Accessibility shows whether the data is accessible with different technologies.
- Compliance is the degree to which the data uses values which comply to standards, regulations or accepted conventions.
- Confidentiality checks whether the data is only accessible by authorized users, as defined in ISO/IEC 13335-1:2004.
- Efficiency shows whether the data provided properties with which the approximation of the performance of the dataset is possible.
- Precision degree to which the data is exact or provide discrimination.
- Traceability shows whether auditing the changes made to the dataset is possible.
- Understandability shows if the data is express with appropriated language, symbols and units to be interpretable by users.

2.3 Validating RDF Data

The now deprecated SPARQL Inferencing Notation (SPIN) constituted an early attempt at formal validation of RDF datasets. SPIN used an RDF vocabulary to represent queries. SPIN was first introduced in 2009 and submitted to W3C in 2014 [13].

SHACL, inspired by SPIN, became a W3C Recommendation in 2017 [14]. SHACL is a language to express statements about the shape of RDF data. The shapes are represented in RDF and can thus be stored next to the data. Most modern RDF triple stores support the execution of SHACL. The W3C Shape Expression Working Group proposed ShEx, an alternative approach to SHACL [23], in 2018. ShEx rule execution starts at a specific node defined in the target node map.

The following subsection give an overview about the key-concepts of SHACL that are crucial to understand the proposed approach.

2.3.1 SHACL Focus Node

Focus nodes are the point in the data graph from which the relative property path starts. There are different ways to define focus nodes:

- *sh:targetNode*: which can be used to refer a specific node by its identifying IRI
- *sh:targetClass*: is used like *sh:targetNode* but match all instances of the given class and its sub-classes
- *sh:targetSubjectsOf*: matches all nodes that have a connection to an arbitrary property to a specified subject
- *sh:targetObjectsOf*: like *sh:targetSubjectOf* but matches the object nodes instead

2.3.2 SHACL Node Shape

Node shapes define the shape of data connected to the focus node. Usually, they define a focus node and have several property shapes. Node shapes can have two properties which configure the interpreter instance:

- *sh:closed*: signals the validator that every additional property is a violation.
- *sh:ignoredProperties*: used to tell the interpreter which properties should not be validated, its range is a list of *rdf:Property*
- *sh:deactivated*: defines a shape as deactivated and excluded from the evaluation, its range is *xsd:boolean*

2.3.3 SHACL Property Shape

Property shapes define the shape of a property path, starting from the focus node. The *sh:path* property of a property shape defines which property is under test. The definition is feature-rich:

- Property Path: defines the shape of property directly starting at the focus node
- Sequence Path: defines an arbitrary long path to the property under test, semantically equivalent to the SPARQL Path syntax
- Alternative Path: used to show that either the one or another given path fulfils the shape
- Inverse Path: used for shapes that need to give a constraint on a property where the focus node is the object

2.3.4 Combining SHACL Shapes

In some cases, node shapes and property shapes need to be combined logically, for this SHACL has a basic set ob logical combinations:

- *sh:not*: negates a shape completely
- *sh:and*: conjunctive operation on n shapes, equal to $\bigwedge_{i=1..n} s_i$
- *sh:or*: disjunctive operation on n shapes, equal to $\bigvee_{i=1..n} s_i$
- *sh:xone*: exclusive or operation in n shapes, equal to $\bigotimes_{i=1..n} s_i$

2.3.5 Data based Validation with SHACL

The rules used to achieve this are so-called ConstraintComponents and can use a parameter to receive information from the shape. The *$this* keyword for example, denotes the current target. We can use ASK and SELECT based validators for such constraints.

2.4 Data Analysis Framworks

The Luzzu framework by Debattista et al. [1] follows a data-oriented rather than a shape-oriented RDF validation approach. Luzzu includes implementations of more than 40 general-purpose metrics. Simple additional, usually domain-specific metrics can be defined in a domain-specific language [3], advanced metrics in Java.

The Semantic Analytics Stack (SANSA) is an approach which tries to combine the research of distributed analytic and semantic technologies. SANSA split into four different layers where implementations can be replaced as needed and is actively developed by Lehmann et al. since 2017 [16].

2.4.1 Knowledge Distribution and Representation Layer

The Knowledge Distribution and Representation layer connects the distributed data world with semantic web technologies. With the API of the layer it is possible to load RDF and OWL data from Apache SPARK[3] or Apache FLINK[4] which both make efficient distributed analytics and computations possible. The layer only loads and performs read/write operations on large distributed datasets. The layer itself is not capable of complex querying. Ermilov et al. [6] show how distributed data management work together with SANSA.

2.4.2 Query Layer

On top of the representation, the Query layer uses the distributed data and enables querying. The layer provides SPARQL Endpoints as well as direct querying of the SPARK or FLINK instances. The execution of the queries combines a SPARQL to SQL rewriting tool and custom partitioning algorithms. Stadler et al. [25] demonstrate how the SANSA query layer and big datasets can work together.

2.4.3 Inference Layer

The Inference layer of SANSA makes use of the schema information contained in OWL and RDF data. With this information the calculation of efficient execution plans is possible. Analyzing and transforming the data to get a better scale-able is also possible with the layer.

2.4.4 Machine Learning Layer

Machine learning algorithms can use this layer to use semantic information. This additional information can lead to more human-understandable solutions. This layer is under ongoing development and is part of the overall goal of SANSA.

2.5 Object Constraint Language

The Object Constraint Language (OCL) [11] is a language to define general conditions in software development. The language is part of the Unified Modeling Language (UML) [19] and covers the definition of invariant in class diagrams.

[3] https://spark.apache.org/
[4] https://flink.apache.org/

2.5.1 Constraint Types

OCl defined six different types of constraints which fulfil a different purpose.

- Invariant: constraints that must hold over the whole lifespan of an object
- Pre-/Postcondition: must hold before/after the execution of an operation
- Initial-/derived-values: must hold for all objects in the initial data and all data that is derived
- Definition: allows defining attributes of operations that are not part of the model
- Guards: must hold before a state transition is possible

In the context of CGMES, only invariants are of interest.

2.5.2 Definition of Invariants

An invariant needs a context definition. This context corresponds to the class and defines for which objects the invariant must hold. To define a context, the keyword *context* is followed by the fully qualified class name.

In one context an arbitrary number of invariants is possible. An invariant definition starts with the *inv* keyword, optionally followed by an invariant name. A combination of statements forms the invariant rule. With the keyword *self* the instance under evaluation is accessible. With the dot-notation members are accessible.

2.5.3 Examplary OCL Rule

An exemplary age tracking software stores the name and age of Persons. The domain model has two requirements on instances of the class Person:

1. The age of a person must be positive or zero
2. The name if a person must be set

Listing 1 shows the OCL invariants defining the requirements in UML. The first line set the context to Person. There is an invariant named personAge that defines that the age of a person must be greater or equal to zero. The other invariant is named personName and defines that the name of a person must differ from null.

Listing 1: OCL invariant definition for a Person with positive age and set name

```
context Person
 inv personAge: self.age >= 0
 inv perosnName: self.name <> null
```

2.6 Validate Grid Models

In 2007, Power Info LLC developed CIMSpy [22] as an analysis and engineering tool for the Common Information Model [9]. CIMDesk [21] is a further evolution of CIMSpy for CGMES from 2008. As model validation is not their primary use case, functionality is limited.

In 2018 Nenadić et al. developed an approach to evaluate the shape of CGMES datasets [18]. In the approach, SHACL is used to evaluate the shape of CGMES Datasets, but the more complex OCL quality definitions were out of scope.

Chapter 3

Methodological Approach

This chapter demonstrates the procedure used to obtain quality metrics for power system modelling. The used data and techniques to evaluate the approaches are given together with the sources of the requirements.

For each group of requirements from the Definition of CGMES and Quality of CGMES [5, 5] a formal definition is given. From this definition a blueprint is derived and implemented with SHACL. To test the proposed metrics datasets from the CGMES Conformity Assessment Scheme[1] are supplemented with pre-defined violations.

3.1 Source of the Assessment Data and Quality Requirements

This section explains the transformation process from a textual requirement definition to executable metrics this paper uses. The section ends with a presentation of the datasets for validation.

The specification of CGMES [4] includes a definition of requirements for data quality [5], specifying the shapes that data should have using two RDFS meta-properties, namely *cims:multiplicity* and *cims:datatype*. The first defines the cardinality of a property with the enum values *rdfs:M:0..1*, *rdfs:M:1..1*, *rdfs:M:0..n* and *rdfs:M:m..n*. The second defines the expected data type of a literal, whereas CGMES only uses *rdfs:range* for the expected type of resources. Additional model invariants are defined in the Object Constraint Language (OCL), a subset of the Unified Modeling Language (UML).

This work groups, the requirements given in the RDFS and OCL are transformed into a formal definition. Each formal definition is an abstraction and covers a whole group of requirements. The mapping assigns a Boolean value to a subject. This value is true if the property meets the quality

[1] `https://www.entsoe.eu/digital/cim/cim-conformity-and-interoperability/`

requirements of CGMES for the type of the subject.

A blueprint for a SHACL shape is the result for each abstract rule definition. The parameter from the requirements replaces the placeholder in the template.

3.2 Different Power System Models for Assessments

This section gives the sources and an overview of the test data. The result is a data collection of test datasets for quality metrics development.

The assessment datasets are from CIM Conformity and Interoperability Assessment Framework in version 2 from 2017. This version contains five different grid sizes with different characteristics:

- MicroGrid: the smallest possible dataset concerning syntax and connectivity. It consists of three regions and is for interoperability testing.
- MiniGrid: to validate standardized calculation on power systems. Consists of two different representations with around 50 nodes.
- SmallGrid: a small part of the European power system. For testing of diagrams and visualizations. Consists of around 100 nodes for graphical representations.
- FullGrid: a subset of the Belgian power system with 100 to 144 Terminal instances and not intended to validate the analysis.
- RealGrid: represents a model of the European power system. A real power system with only the needed topology. Consists of around 33.000 nodes in one dataset and for overall testing.

For complete assessments concerning correctness and performance, the RealGrid dataset is the best choice. To verify that metrics are also applicable within European merging-processes the MircoGrid is the best. The MiniGrid and SmallGrids are not in the scope of this work, because they are not targeting the modelling aspect of CGMES.

3.3 Structure of Power System Models

Each grid dataset consists of different graphs realizing the separation of concerns. The graphs combine different for different purposes.

Figure 3.1 shows the coupling between the main graphs of a CGMES dataset.

The Equipment Model (EQ) graph is usually the biggest one because it provides information about the existence of components. All other graphs have dependencies into this one. Some of them are direct and others are transitive via another graph.

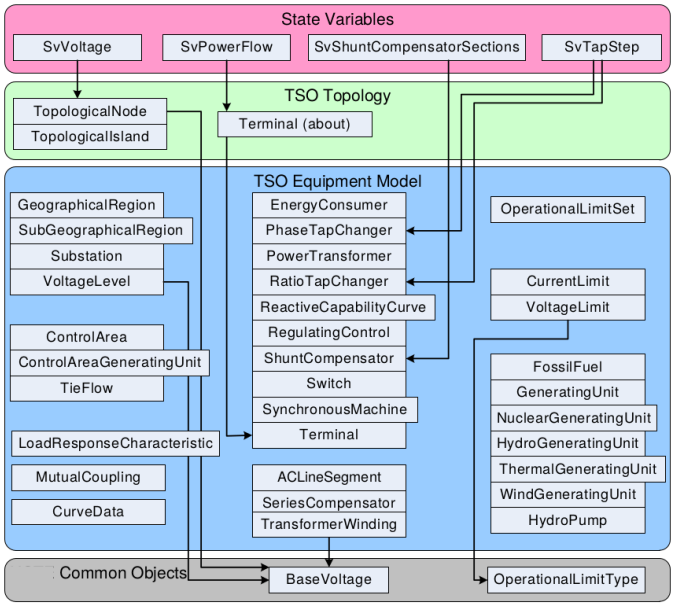


Figure 3.1: Extensive coupling of the three main graphs of CGMES. The grey squares indicate class instances in the profile and the arrows show dependencies to other instances.

The State Variables (SV) graph holds information about the current state of the system. Therefore, it gives information about the volumes and flows at dedicated equipment or configurations of regulation equipment.

The Topology (TP) graph provides information about the interconnection of equipment. It summarizes the topology in a graph-theoretical meaning and is the connecting graph between EQ and SV.

For EQ and TP, there are additional Boundary (suffix: BD) graphs. BDs are the standard interface between different EQ and TP files from different sources. The instances are usually virtual and placed in between two real nodes.

Beside the described graphs, CGMES also defines graphs that map equipment to geographical positions and give properties for representation. These graphs are part of the standard but not of interest for modelling applications.

The three core graphs EQ, TP and SV are crucial for developing quality metrics. BD graphs are also of interest because they are part of the merging processes. Graphs with representing nature are only informative and do not need to be part of the dataset.

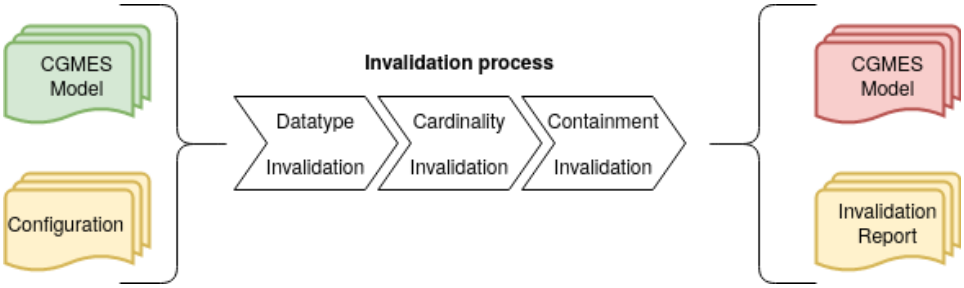


Figure 3.2: Process of invalidating a Power System Model in CGMES serialization. The process takes a CGMES model and a configuration, invalidates the model in multiple ways and writes a report about the invalidation.

3.4 Software for Metric Development and Evaluation

The Apache Jena Framework handles RDF data and executes SPARQL queries. Apache Jena has a great feature-rich API for working with RDF data and is compatible with the most W3C recommendations in the RDF context. The framework supports native SHACL and graph validation against shape graphs. Because Jena is compliant to the semantic web standards form a good base for development.

3.5 Testdata

The *CIM Conformity and Interoperability Assessment Framework* defines five datasets for different purposes but none is present for quality metric testing. For tests, the predefined grids need a configurable amount of invalid data.

Figure 3.2 shows the process of invalidating a CGME model. The invalidation process takes a grid model and a configuration as input. The output is a grid model that violates the quality requirements as configuration together with a list of invalidated instances for evaluation.

3.5.1 Datatype Invalidation

The datatype rule from Equation (4.2) validates that a property has the correct datatype. All literals within a power system model in CGMES serialization need a datatype annotation for this step. The datatype rule holds if the datatype annotation is the same as in the definition.

The datatype invalidation step chooses an arbitrary triple and changes the datatype annotation to violate the datatype rule. If the object is of type

xsd:string, the step flips the type to *xsd:double* and to *xsd:string* otherwise. If the object is a reference, the step set the datatype to *xsd:string*. These steps are applied repeatedly until enough triples are invalid.

3.5.2 Cardinality Invalidation

This step produces violation of the cardinality rule from Equation (4.1). In practice, only 0..1 and 1..1 are used to indicate mandatory fields. If more than two properties with the same label exist, the triple is invalid in terms of cardinality. By duplicating an arbitrary triple, the triples violate the cardinality rule.

3.5.3 Containment Invalidation

This step produces violation of the containment rule form Equation (4.3). A triple violates the containment rule if the type of the container grouping the instances is not of the expected type.

This step introduces a dummy container instance of a dummy type. It is crucial to notice that this type is not part of the CGMES standard and therefore can never be the correct container. The step repeatedly selects instances that are a member of a container and reassigns the container to the invalid dummy container until the number of violations is high enough.

3.5.4 Contextual and Conditional Invalidation

To produce a violation of the Contextual rule from Definition 4.2.3 and of the Conditional rule from Equation (4.4) is not possible without knowledge about the concrete rule. Therefore a manual data manipulation is required.

The contextual rules are easy to test because there are limited combination and therefore limited possible contexts. Conditional rules are not that numerous, editing a few triples within the dataset by hand is sufficient.

Chapter 4

Analysis

This chapter revisits the requirements in the standard and from industry. The first section identifies the requirements from ENTSO-E and the industry based on an analysis of the standards. The second section gives a formal definition of the requirements found.

4.1 Requirement Analysis

4.1.1 Quality Rules of CGMES

The original CGMES specification [4] does not define quality rules for the modelled data, only one RDF schema for each profile.

ENTSO-E released the *Quality of CGMES datasets and calculations for system operations* in 2016 [5] defining data quality in context of CGMES. The quality of CGMES can only be determined with the whole dataset at hand. The ENTSO-E's Quality of CGMES definition divides 106 rules into eight levels:

1. File Names and Packaging: forms the naming and directory structure conventions. For example, rule *1_1 - The ISO 8601:2004 standard will be used for designating dates and times.* This level defines 30 rules on merged CGM.

2. XML Structure assures that the files do not miss any tag and all attributes have correct values. For example, rule *2_2 - XML documents must contain one root element that is the parent of all other elements.* This level defines seven rules on parts of IGMs.

3. RDF Specification checks whether headers or namespaces are missing. For example, rule *3_3 - The mandatory RDF namespace (attribute xmlns:rdf) must be declared as ....* This level defines 12 rules on parts of IGMs and CGMs.

4. Object Constraints: dealing with min/max values and field length. For example, rule *4_1 - All mandatory attributes for an exchanged CGMES class must be provided for the profile(s) declared in the file header.* This level defines four rules on IGMs and CGMs.

5. Consistency: checks whether the references within and across graph boundaries are correct. For example rule *5_3 - The SV instance file must contain a dependency to specify which SSH data was used in the Power Flow calculation.* This level defines 16 rules for IGMs and CGMs.

6. Robustness: validates that entities are correctly tied together and fulfil conditional requirements. For example rule three of subsection 6.3 *Q limits shall be respected (also for slack node/swing bus).* This level defines 26 rules for IGMs and CGMs.

7. Cross IGM Consistency: checks whether references are correct across different IGM and assures that the merge to a CGM is possible. For example, rule *7_2 - The value of CurrentLimit.value is expected to be the same on both sides of a tie-line..* This level defines two rules for IGMs and CGMs.

8. Plausibility: assures that the values do not violate limits. For example, rule *8_1 - It shall be possible to calculate the power flow of an IGM 47 with the power flow settings provided in 6.3.* This level defines nine rules on CGMs.

This work deals with the levels of RDF-Specification and Object Constraints. A full summary of the levels of RDF-Specification and Object Constraints is attached in Section A.2. The grouping shows requirements that are covered by the RFD framework and XML schema, these are not part of the work. The other requirements groups center around dataytypes, cardinalities, conditionals and contexts.

4.1.2 The Shape of a Power System

In the definition of CGMES the shape of a power system model is defined as a UML class diagram. All requirements formulated in the UML model are attached in Section A.1. The summary shows the diversity of the requirements. The class diagram in Figure 4.1 allows the abstract modelling and combination of lines. The class hierarchy defines that the composition of elements in their container (in Figure 4.1 Limits and ACLineSegments) has a cardinality. The CGMES standard extends the RDF Schema standard with *cims:cardinality* to denote the multiplicity information and with *cim:datatype* to distinguish between references and literals. The schema property *cim:datatype* shows that the range of a property is literal.

Table 4.1: Size comparison of the different CGMES compliance test datasets and their purpose in this work.

Category	Nodes	Triples	Description	Purpose
Micro	100	2,200	simplified BE and NL with cross border lines	Cover manipulation use-case
Full	5,000	6,500	Full model of a small subset of Europe	Simulate single IGM
Real	33,000	1,005,000	Simplified version of European power system	Simulate small CGM

FromFigure 4.1 following requirements to be covered by the SHACL rule are derived:

- All instances of type *cim:IdentifiedObject* must have a name of type *xsd:string*
- All instances of type *cim:Limit* must have a value of type *xsd:double*
- All instances of type *cim:Terminal* can have one or no limit of type *cim:Limit*
- All instances of type *cim:Terminal* can have zero to two *cim:ACLineSegments* conducted to

From the inheritance principle, all instances of *cim:Limit*, *cim:Terminal* and *cim:ACLineSegment* must also fulfill the requirements for *cim:IdentifiedObjects*.

4.1.3 Constraints from Industry

Applications based on top of the CGMES technique are usually part of a workflow supporting decision making. For decision-making processes performance is crucial. So all applications need to produce correct answers in a short amount time.

The ENTSO-E releases a set of so-called compliance assessment datasets, which differ in size and purpose. Table 4.1 shows the proximate size of the assessment datasets compared to a productive one.

To be able to make use of the distribution, the solution should scale arbitrary and should provide a way to populate the results. This leads to the following industrial constraints:

P erformance: the including workflow should not be blocked by the evaluation.

S calability: the solution needs to be able to evaluate arbitrary large grids with arbitrary rule sets.

C ompliance: the solution needs to be able to pass the compliance assessment of ENTSO-E.

A gile: with the solution, it needs to be easy to change the rule-set.

4.1.4 Conclusion

The rules in the levels of Section 4.1.1 are abstract and each rule applies for multiple classes at once. Section 4.1.2 defines some requirements in more detail and these rules have their root in the underlying UML model. The technical requirements from Section 4.1.3 suggest a flexible definition for the rules.

The Section 4.2 gives a formal definition for cardinalities, datatypes and containments based on the CGMES model. To cover the whole area of level three and four of the Quality of CGMES this section also defines context-based and conditional rules. The quality rules are part of the schema completeness, schema conciseness and consistency of Zaveri et al. [27].

4.2 Formal Definition

This section describes the problem of data quality in the context of CGMES. The formalization is an extension of the formal definition of RDF graphs.

4.2.1 RDF Graph Definition

The basic concepts of RDF are Resources $\mathcal{R}$, Blank nodes $\mathcal{B}$ and Literals $\mathcal{L}$.

Definition 4.2.1. A RDF Graph G consists of triples $(s, p, o) \in \mathcal{T} := (\mathcal{R} \cup \mathcal{B}) \times \mathcal{R} \times (\mathcal{R} \cup \mathcal{B} \cup \mathcal{L})$. The triples form a directed labeled graph :

$G_{RDF} := (V, E)$ with:

$$V := \mathcal{R} \cup \mathcal{B} \cup \mathcal{L}$$
$$E := (\mathcal{R} \cup \mathcal{B}) \times \mathcal{R} \times (\mathcal{R} \cup \mathcal{B} \cup \mathcal{L})$$

4.2.2 CGMES Model Definition

This section groups the quality requirements of the Specification of CGMES and the Quality of CGMES [4, 5] based on appendix Section A.1. Each group are transformed into a formal definition.

$\mathcal{R}$ denotes the set of resources identified by IRIs, $\mathcal{B}$ is the set of blank nodes, and $\mathcal{L}$ is the set of literals. $\mathcal{P} \subseteq \mathcal{R}$ is the set of all properties, $\mathcal{I} \subseteq \mathcal{R}$ the set of all instances and $\mathcal{C} \subseteq \mathcal{R}$ the set of all class identifiers. In the following, $s \xrightarrow{p} o$ means that there is an edge from $s \in \{\mathcal{R} \cup \mathcal{B}\}$ to $o \in \{\mathcal{R} \cup \mathcal{B} \cup \mathcal{L}\}$ with label p.

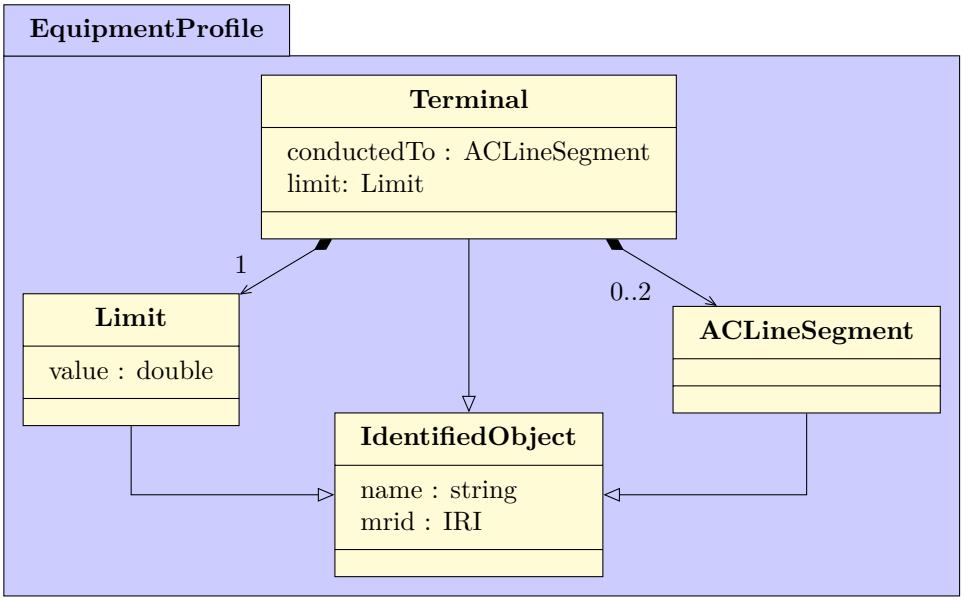


Figure 4.1: UML model of the connections between ACLineSegements, Terminals and Limits. From this model the RDF Schema and shape rules are derived.

Definition 4.2.2. The CGMES model defines a finite set of Classes $\mathcal{C} \subseteq \mathcal{R}$.
Each class $c \in \mathcal{C}$ has a finite set of members $\mathcal{M}_c \subset \mathcal{R}$.
Each member $m \in \mathcal{M}_c$:

1. has a pre-defined multiplicity
2. has a pre-defined datatype

In extraction of the power system model in Figure 4.1 shows the classes $\{Terminal, Limit, ACLineSegment, IdentifiedObject\} \subset \mathcal{C}$. The class : $Terminal$ has the members $\mathcal{M}_{Terminal} := \{conductedTo, limit, name, mrid\}$.
The member $conductedTo$ has the multiplicity 0..2 and the datatype $ACLineSegment \in \mathcal{C}$.

4.2.3 Quality of CGMES Model Problem

The Quality of CGMES [5] describes quality rules for the classes $\{Terminal, Limit, ACLineSegment,$ $\mathcal{C}$. The definition gives rules for the cardinality and datatype of each property, but some additional more complex rules are present as well. From here $s \xrightarrow{p} o$ means that their is a edge from $s \in \{\mathcal{R} \cup \mathcal{B}\}$ to $o \in \{\mathcal{R} \cup \mathcal{B} \cup \mathcal{L}\}$ with label p.

Cardinality Rule

CGMES requires instances to have a fixed number of properties. The cardinality rule checks whether an instance of a class fulfil the requirements on the number of its properties. The requirements lead to the formal rule in Equation (4.1).

$$\begin{aligned} r_{cardinality} &: \mathcal{I} \times \mathcal{P} \times \{=, \leq\} \times \mathbb{N} \to \mathbb{B}, \\ &(s, p, op, n) \mapsto op(|\{s \xrightarrow{p} o\}|, n) \end{aligned} \tag{4.1}$$

The set $R_C^{cardinality}, C \in \mathcal{C}$ combines all cardinality rules for the type C.

In Figure 4.1, the associations between *Terminal* and *ACLineSegment* or between *Terminal* and *Limit* provide examples of such requirements. A *Terminal* must have a connection to one *Limit* and can have zero to two connections to a *ACLineSegment*. Each instance of the *Terminal* class must comply with this.

Datatype Rule

CGMES requires instances to use correct datatypes for property values. The datatype rule checks whether each property value of an instance has the expected type. The requirements lead to the formal rule in Equation (4.2).

$$\begin{aligned} r_{datatype} &: \mathcal{I} \times \mathcal{P} \times (\mathcal{C} \cup \texttt{XSDDatatype}) \to \mathbb{B}, \\ &(s, p, t) \mapsto \forall o \colon (s \xrightarrow{p} o) \implies (o \xrightarrow{rdf:type} t \vee datatype(o) = t) \end{aligned} \tag{4.2}$$

The set $R_C^{datatype}, C \in \mathcal{C}$ combines all datatype rules for the type C.

In Figure 4.1, each class formulates an instance of this rule. For *Terminals* it defines the object's type of the *conductedTo* property to be *ACLineSegment*. For *IdentifiedObjects* the object of the *name* property must be a string.

Containment Rule

Most CGMES classes have requirements on a so-called container. A container is an instance of a particular type that groups other instances. This leads to the formal containment rule in Equation (4.3).

$$\begin{aligned} r_{\in} &: \mathcal{I} \times \mathcal{P} \times \mathcal{C} \to \mathbb{B}, \\ &(s, p, t) \mapsto \forall o \colon (s \xrightarrow{p} o) \implies (o \xrightarrow{rdf:type/rdfs:subClassOf*} t) \end{aligned} \tag{4.3}$$

The set $R_C^{\in}, C \in \mathcal{C}$ combines all containment rules for the type C.

In Figure 4.1, the class *ACLineSegment* forms a container for up to two instances of the *Terminal* class with the property *conductedTo*. The type of instances *conductedTo* refers to must be a subclass of *Terminal*. For example, the class *ACTerminal* is a sub class of *Terminal* and thus fulfils the requirement.

Conditional Rule

In CGMES, some requirements on values of properties are dependent on conditions. *If* the instance satisfies a certain condition, *then* a requirement applies. For example, if the instance has a specific property, some other property must meet a requirement. Conditional rules enable a combination of rules. This leads to the formal conditional rule in Equation (4.4).

$$\begin{aligned} r_{IF} &\colon \mathcal{K} \times \mathcal{Q} \to \mathbb{B}, \\ &(con, rule) \mapsto con \implies rule \equiv \neg con \vee rule \end{aligned} \tag{4.4}$$

Where $\mathcal{K}$ defines the set of conditions and $\mathcal{Q}$ defined the set of rules. The set $R_C^{IF}, C \in \mathcal{C}$ combines all conditional rules for the type C.

Contextual Rule

In CGMES, some requirements on the model are dependent an external context.

Definition 4.2.3. The set of model types is defined as

$$\mathcal{MT} := \{IGM, CGM, BD\}$$

The set of process types is a finite but arbitrary set of strings

$$\mathcal{PT} \subset \Sigma^*$$

The Context of a model is the tuple

$$c := (t_m, t_p), t_m \in \mathcal{MT}, t_p \in \mathcal{PT}$$

A contextual rule is a conditional rule whose condition refers to the context. The set $R_c^{context}, c \in \mathcal{C}$ combines all contextual rules the the type c. These rules are in the informal passages of the CGMES standard.

Evaluation of CGMES Quality Rules

In the Quality of CGMES definition, each rule has one of the severity levels $Warning$ or $Error$. Each rule for type $c \in \mathcal{C}$ must hold for each c' which is subclass of c.

Definition 4.2.4. A Quality Rule Evaluation for instance i of type $c \in \mathcal{C}$ is the set:

$$RE_c(i) = \{(\varsigma(r), r(i)) \colon r \in RS_c\}$$

Where $\varsigma \colon R \to \{Warning, Error\}$ maps each rule to its severity. The Quality Result is the mapping

$$\begin{aligned} \rho \colon RE &\to \{Valid, Warning, Error\}, \\ r &\mapsto \begin{cases} Warning & , Warning \in r \wedge Error \notin r \\ Error & , Error \in r \\ Valid & , otherwise \end{cases} \end{aligned} \tag{4.5}$$

Chapter 5

Implementation

This chapter develops a basic approach for automatic execution of the rules as defined in Section 4.1.2. SPARQL producible SHACL shapes are the outcome of the chapter for RDFS based requirements. A parser implemented in Java transforms the OCL requirements to SHACL shapes. The contextual requirement shapes are written by hand.

5.1 Basic Framework

A SHACL property shape must be part of a SHACL node shape defining the target and general information. The focus node defines this context and can be selected in different ways. The formal definitions from Section 4.1.2 apply to instances of CGMES classes. Defining the target for SHACL via the CGMES class is sufficient.

The statement in Listing 2 defines the target of the constraints by setting the *sh:targetClass* to the respective CGMES class. The quality assessment focuses on statements about instances, so the type of shape should be *sh:NodeShape*. The class-based definition of CGMES implies no additional properties. Thus, *sh:closed* can be set to true unless the use case requires otherwise.

Listing 2: SPARQL construct query to generate the basic SHACL shape for each class of CGMES

```
CONSTRUCT { [] rdf:type sh:NodeShape; sh:targetClass
    ?class; sh:closed true; sh:ignoreProperties
    (rdf:type).
} WHERE { ?class rdf:type rdfs:Class; rdfs:subClassOf
    ?superClass. ?property rdfs:domain ?subclass; }
```

The different error levels in CGMES are warning and error. Warnings inform the end-user that something may not be correct. Error disqualify the dataset for further consideration and indicate the need for adjustment. The

levels map to SHACL's *sh:severity* property, where warning corresponds to *sh:Warning* and error to *sh:Violation*.

5.2 Schema Based Rules

The next sections give an example rule for each blueprint. The Figure 4.1 is the base for the exemplary data in Example 5.2.1. A *cim:ACLineSegment* must have a literal property *cim:IdentifiedObject.name* of type *xsd:string*, this property is inherited from *cim:IdentifiedObject*. Additional a *cim:ACLineSegment* can have a reference property *cim:ConductingEquipment.Terminal* which refers to an instance of *cim:Terminal*. This property is inherited from *cim:ConductingEquipment*.

Example 5.2.1. The RDFS provide to following information on instances of the type *cim::ACLineSegment*:

?property	?range	?dataType	?multiplicity
cim:IdentifiedObject.name	-	xsd:string	1..1
cim:ConductingEquipment .Terminal	cim:Terminal	-	0..1

5.2.1 Cardinality Rule Shape

The statement in Listing 3 produces cardinality checks for all properties. The *sh:path* defines the property to check. The min and max variables store the values for the minimum and maximum cardinalities. It is essential to notice that *sh:minCount* and *sh:maxCount* are inclusive.

Listing 3: SPARQL construct statement to generate SHACL shapes from RDFS to execute the cardinality rule

```
CONSTRUCT {  # omit general shape blueprint
 sh:property [sh:path ?property; sh:maxCount
    xsd:integer(?max); sh:minCount
    xsd:integer(?min);].
} WHERE { ?property cims:multiplicity _:card.
 BIND(STRBEFORE(STRAFTER(STR(?multiplicity), "M:"),
    "..") AS ?min ).
 BIND(STRAFTER(STR(?multiplicity), "..") AS ?max). }
```

Example 5.2.2. The execution of the statement in Listing 3 against the RDFS files of CGMES produce cardinality checking shapes for all classes. The produced SHACL shape checking the cardinality for *cim:ConductingEquipment* is the following:

```
sh:property [sh:path cim:IdentifiedObject.name;
    sh:maxCount 1; sh:minCount 1;];
sh:property [sh:path
    cim:ConductingEquipment.Terminal; sh:maxCount 1;
    sh:minCount 0;];
```

This section presented a shape that can evaluate the cardinalities of instances in a CGMES model. The generated shapes evaluate an instance concerning the requirement in Equation (4.1).

5.2.2 Datatype Rule Shape

The CGMES model [4] has datatype requirements for each property of a class. In Equation (4.2) the formal definition of this rule is given.

If the object is a literal, the *sh:datatype* property restricts its datatype. Listing 4 below shows the literal case producing SPAQRL construct statement.

Listing 4: SPARQL construct statement to generate SHACL shapes from RDFS to execute the literal datatype

```
CONSTRUCT { # omit general shape blueprint
  sh:property [sh:path ?property; sh:datatype
      ?dataType;].
} WHERE { ?property cims:dataType ?dataType. }
```

In the resource case restrictions on the datatype are not that straightforward. The *sh:nodeKind* property of the property shape needs to be *sh:IRI* to make sure it is a reference. The *sh:class* property indicates the expected type of the resolved target object. The rule is valid if the target node has an rdf:type property with the type. The SPARQL construct statement in Listing 5 generates property shapes to validate the type of a target object.

Listing 5: SPARQL construct statement to generate SHACL shapes from RDFS to execute the resource datatype

```
CONSTRUCT { # omit general shape blueprint
  sh:property [sh:path ?property; sh:nodeKind sh:IRI;
      sh:class ?targetClass;]
} WHERE { ?property rdfs:range ?targetClass. }
```

Example 5.2.3. The produced SHACL shape checking the datatype of *cim:IdentifiedObject.name* and *cim:ConductingEquipment.Terminal* is the following:

```
sh:property [ sh:path cim:IdentifiedObject.name;
    sh:datatype xsd:string;]
```

```
sh:property [ sh:path
    cim:ConductingEquipment.Terminal; sh:nodeKind
    sh:IRI; sh:class cim:Terminal;]
```

The shape for the *cim:IdentifiedObject.name* property is the literal case. This shape produces a violation if the object is of any other type than *xsd:string*. Note that a missing type annotation does not constitute an error because the default datatype of RDF is a string (*rdf:langString*).

The shape for the *cim:ConductingEquipment.Terminal* property is the resource case. This shape produces a violation if the reference does not have an *rdf:type* property with object *cim:Terminal* or if it is not an *sh:IRI*. No violation is present if the interpreter can not find the resource.

This section presented a shape that can check the datatype of instances in a CGMES model. The generated shapes assess an instance concerning the target type of their property object.

5.3 Invariant Based Rules

This section deals with the creation of the rules based on the OCL invariant in CGMES definition and the quality of CGMES [4, 5]. The invariant define the containment and the conditional rules from definitions Equation (4.3) and Equation (4.4).

5.3.1 Containment Rule Shape

For some classes the CGMES model [4] defines container types for grouping. Equation (4.3) gives the formal definition of this rule. The OCL files define these type of requirements.

The Listing 6 shows the blueprint of a shape that checks the containment of an instance. Only an instance of the container class should group this instance.

Listing 6: Blueprint of a SHACL shape that check containment rules where *seq* are sequence to the container and *con* are class of the container of requirement i

```
sh:property [
 sh:path(<seq> rdf:type [sh:zeroOrMorePath
    rdfs:subClassOf]);
 sh:hasValue <con> ; ]
```

The SHACL PathSequence feature defines the path to the property to check. The braces indicate an ordered list of properties to follow. With this notation arbitrary long navigation from the target node is possible. The shape checks whether the container is an instance of a specific class.

Example 5.3.1. For *cim:ACDCConverter* it is required that they are grouped via the *EquipmentContainer* property under instances of type *DCConverterUnit*. The OCL invariant for this fact is the following:

```
context IEC61970::Base::DC::ACDCConverter
inv acDcConverterDCConverterUnit:
    self.EquipmentContainer.oclIsKindOf(
    IEC61970::Base::DC::DCConverterUnit )
```

This OCL invariant transforms into the SHACL shape below. The shape validats the containment rule on *cim:ACDCConverter*:

```
sh:property [
 sh:path(cim:ConductingEquipment.EquipmentContainer
     rdf:type [sh:zeroOrMorePath rdfs:subClassOf]);
 sh:hasValue cim:DCConverterUnit; ]
```

5.3.2 Conditional Rule Shape

For some classes the CGMES model [4] defines conditional if-then rules. In Equation (4.4) the formal definition of this rule is given.

OCL does not have an "if" feature. The condition is compiled into the "then" part by conjunction. The "else" parts start with a conjunction of the negated condition. SHACL does not have an "if" feature either and it is a typical pattern to combine shapes with the condition conjugated disjunctive. Listing 7 shows the blueprint to execute the “then” part of the rule if the condition evaluated to false.

Listing 7: SPARQL CONSTRUCT query to generate SHACL shapes that execute conditional rules, where c_i are conditions and r_i are rules of requirement i

```
CONSTRUCT { # omit general shape blueprint
 sh:or ([sh:not [?con]] [?rule])
} WHERE { VALUES (?con ?rule) { (c_1 r_1) ··· (c_n r_n) } }
```

5.4 Other Shapes

A small subset of the quality requirements for CGMES are textual and therefore are not directly machine-readable.

5.4.1 Contextual Rule Shape

In different contexts the definition of CGMES [4] defines varying requirements. In Definition 4.2.3, the context is a tuple of model and process type.

The context is not part of CGMES and therefore needs to be configured externally. If such a configuration is present, the conditional rule shape from Listing 7 covers the contextual rule.

Example 5.4.1. In CGMES a ControlArea has an optional reference to an EnergyArea. Section 3.12.1 “ControlArea General” of the CGMES specification requires mandatory EnergyArea references if the process is operational [4]. The SHACL shape below assumes an example property *ex:processType* to have the value *ex:operation* if the process is operational.

```
sh:property[sh:or (
 [sh:not [sh:path ex:processType; sh:hasValue
    ex:operation]]
 [sh:path cim:ControlArea.EnergyArea; sh:minCount 1])]
```

SHACL does not directly support external configurations. A context feature is possible in three ways:

1. An additional data graph can hold the context definition. A SPARQL based sub shape can derive this context information.
2. For each combination of process and model type, a graph must be present or generated with slightly different shapes.
3. A custom validation engine can hold the information and provide it via a custom function. An extension mechanism for SHACL is not yet specified.

The first approach requires an extension to CGMES because there is no definition for such information. The second approach requires the generation of shape graphs for each combination of process and model type and external code to determine which shape graph is to be used. The latter approach requires extending the SHACL execution process. Stored procedures are possible, e.g., in Apache Jena’s using a custom "property functions" .[1]

[1] https://jena.apache.org/documentation/query/extension.html

Chapter 6

Evaluation

This chapter show that the concrete shapes resulting from Chapter 5 are capable of detecting the violations. It demonstrates to which degree the solution can determine the data quality defined in CGMES [4] and the definition of quality [5].

6.1 Correctness of the Solution

This section makes use of the test data defined by the ENTSO-E and the invalidation process in Section 3.5. The violations in this section are examples extracted from a test run.

6.1.1 Cardinality Rule Evaluation

Section 5.2.1 shows a shape detecting violation of the cardinality rule. This evaluation sticks with the shapes in Example 5.2.2. The triples in Listing 8 below are a sub-graph after the test data generation. This sub-graph contains violations of the cardinality rules for *cim:ACLineSegments*.

Listing 8: Sub-graph of invalid test data containing cardinality rule violations.

```
ex:inst1 rdf:type cim:ACLineSegment;
    cim:ConductingEquipment.Terminal ex:t1;
    cim:IdentifiedObject.name "Inst1".
ex:inst2 rdf:type cim:ACLineSegment;
    cim:ConductingEquipment.Terminal ex:t1;
    cim:ConductingEquipment.Terminal ex:t2;
    cim:IdentifiedObject.name "Inst2".
ex:inst3 rdf:type cim:ACLineSegment;
    cim:ConductingEquipment.Terminal ex:t1.
```

Table 6.1: Violations detected by the cardinailiy rule shape for the invalid sub-graph.

Instance	Property	Violation
inst2	cim:ConductingEquipment .Terminal	Expected maxCount(1) but was 2
inst3	cim:IdentifiedObject.name	Expected minCount(1) but was 0

Table 6.1 depicts the validation report for the sub-graph. The SHACL shape for cardinality rule evaluation detects two violated and one compliant instance. The instance "inst1" has no violations because it contains the expected number of *cim:ConductingEquipment.Terminal* and *cim:IdentifiedObject.name* properties.

6.1.2 Datatype Rule Evaluation

Section 5.2.2 show two shapes detecting violation of the datatype rule. The triples in Listing 9 are a sub-graph after the test-data generation. This sub-graph contains violations of the datatype rule for *cim:ACLineSegment*.

Listing 9: Sub-graph of invalid test data containing datatype rule violations.

```
ex:inst1 rdf:type cim:ACLineSegment;
    cim:IdentifiedObject.name "42"^^xsd:string.
ex:inst2 rdf:type cim:ACLineSegment;
    cim:IdentifiedObject.name "42";
    cim:ConductingEquipment.Terminal ex:t1.
ex:inst3 rdf:type cim:ACLineSegment;
    cim:IdentifiedObject.name 42;
    cim:ConductingEquipment.Terminal "t1".
ex:inst4 rdf:type cim:ACLineSegment;
    cim:IdentifiedObject.name "42"^^xsd:double;
    cim:ConductingEquipment.Terminal ex:t2.
ex:t1 rdf:type cim:Terminal.
ex:l2 rdf:type cim:ACLineSegment.
```

Table 6.2 shows the validation report for the sub-graph. The SHACL shape for cardinality rule evaluation detects two violated and two compliant instances. The instances "inst1" and "inst2" are no violations because both names are of type *xsd:string* even if they represent integers. The reference *cim:ConductingEquipment.Terminal* in "inst2" is valid as ex:t1 is of type *cim:Terminal*. Instances "inst3" and "inst4" violate the literal datatype rule since the data-types of *cim:IdentifiedObject.name* do not match *xsd:string*. Instances "inst3" violate the resource datatype rule at *cim:ConductingEquipment.Terminal* as the type is not an IRI. Instances "inst4" violate the

Table 6.2: Violations detected by the Datatype Rule Shape for the invalid sub-graph.

Instance	Property	Violation
inst3	cim:IdentifiedObject.name	Expected datatype xsd:string but was xsd:integer
inst3	cim:ConductingEquipment .Terminal	Expected an IRI but was xsd:string
inst4	cim:IdentifiedObject.name	Expected datatype xsd:string but was xsd:double
inst4	cim:ConductingEquipment .Terminal	Expected class cim:Terminal but was cim:ACLineSegment

resource datatype rule at *cim:ConductingEquipment.Terminal* because the type of the target type is not *cim:Terminal*.

6.1.3 Containment Rule Evaluation

Section 5.3.1 defined a shape blueprint to evaluate containment rules. The triples in Listing 10 are a sub-graph after the test-data generation. The sub-graph contains violations of the containment rule for *cim:ACDCConverter*.

Listing 10: Sub-graph of invalid test-data containing containment rule violations.

```
ex:inst1 rdf:type cim:ACDCConverter;
    cim:ConductingEquipment.EquipmentContainer
    ex:cont1.
ex:inst2 rdf:type cim:ACDCConverter;
    cim:ConductingEquipment.EquipmentContainer
    ex:dummy.
ex:cont1 rdf:type cim:DCConverterUnit.
ex:dummy rdf:type ex:DummyContainer.
```

Table 6.3 displays the validation result for the evaluation of the containment rule on the sub-graph. The shape detects the invalid instance ex:inst2 because of the value of the property path *cim:ConductingEquipment.EquipmentContainer/rdf:type* is not the expected *cim:DCConverterUnit*. The other instance is valid sine it is a member of a collection with the expected type. This example asks for the *rdf:type* but this can be any other path as well.

6.1.4 Conditional Rule Evaluation

Section 5.3.2 defined a shape blueprint to evaluate containment rules. The triples in Listing 11 are a sub-graph after the test-data generation. The

Table 6.3: Violations deteced by the Containment Rule Shape for the sub-graph.

Instance	Property	Violation
inst2	cim:ConductingEquipment .EquipmentContainer/rdf:type	Expected value cim:DCConverterUnit but was ex:DummyContainer

sub-graph contains violations of the conditional rule for *cim:LoadResponseCharacteristic*.

Listing 11: Sub-graph of invalid test-data containing conditional rule violations.

```
ex:inst1 rdf:type cim:LoadResponseCharacteristic;
    cim:LoadResponseCharacteristic.exponentModel false.
ex:inst2 rdf:type cim:LoadResponseCharacteristic;
    cim:LoadResponseCharacteristic.exponentModel
    false;
    cim:LoadResponseCharacteristic.qConstantCurrent
    "13".
ex:inst3 rdf:type cim:LoadResponseCharacteristic;
    cim:LoadResponseCharacteristic.exponentModel true;
    cim:LoadResponseCharacteristic.qConstantCurrent
    "13".
ex:inst4 rdf:type cim:LoadResponseCharacteristic;
    cim:LoadResponseCharacteristic.exponentModel true.
ex:inst5 rdf:type cim:LoadResponseCharacteristic;
    cim:LoadResponseCharacteristic.qConstantCurrent
    "13".
ex:inst6 rdf:type cim:LoadResponseCharacteristic.
```

Table 6.4 shows the evaluation result for the conditional rule. The first instance ex:inst1 violate the conditional rule because exponentModel is false, but qConstantCurrent is not present. Instance, inst2 is evaluated to valid since qConstantCurrent exists. Instances inst3 and inst4 are valid as exponentModles value is true and therefore the value of qConstantCurrent is not needed. Instances inst5 and inst6 are valid to conditional rule because not hasValue is valid for missing property.

6.2 Performance of the Solution

The performance is one of the primary requirements for power system modelling in an industrial use case. Throughput is the key performance indicator.

Table 6.4: Violations detected by the Containment Rule Shape for the invalid sub-graph.

Instance	Property	Violation
inst1	cim:LoadResponseCharacteristic .qConstantCurrent	Expected minCount(1) but was 0

The throughput is the ratio between data size and time, in this case, triples per millisecond.

6.2.1 Measurement Setup

This subsection gives an overview of the used setup for the performance measurements. The first part explains the cases to measure and the second concentrate on the execution of the measurements.

Measurement Cases

The categories of the dataset correspond to the ENTSO-E terminology mentioned in Table 4.1. The combination of dataset categories from Micro to Real covers the usual size of models in a CGMES process. The differences in size between the three lead to a good diversification of the measurement points.

The invalidation process from Section 3.5 takes the valid ENTSO-E grids and invalidates them. The amount of invalidation is in percentage and set to 10%, 50%, and 80% to cover the realistic areas. This setup covers the use-case UC1 and UC2 from Section 1.3.

Measurement Hardware

Two different machines execute the same measurements.

The first machine, identified with *P*, is a Lenovo Thinkpad T540P with Intel i5 CPU, 16GB DDR3 RAM and 500GB SSD. At execution, Ubuntu 19.10 with OpenJDK 14 64Bit is used at this machine. The second machine, identified with *S*, is a Dell Precision 7540 Mobile Workstation with Intel i9, 32GB DDR3 RAM and 500GB SSD. At execution, Microsoft Windows 10, with OpenJDK 14 64Bit is used at this machine.

The JVM runs with the default configuration, so the Just-In-Time optimization feature is active. A fresh JVM instance executes one measurement for the cold-start situation. So no JIT optimization is possible. The same JVM runs all measurements for the warm-up situation. So the JIT optimization optimizes the execution runs. These sequences are executed in different orders to get stable results.

6.2.2 Measurement Results

This section discusses the performance of the solution for the measurement cases from Section 6.2.1. The analysis splits into two parts. The first subsection looks into the impact of violated triples on the throughput, while the second centres on the impact of dataset size.

Amount of Violations Impact

The amount of violations has an impact on throughput. Each violation triggers a generation of an entry in the validation result. This entry consists of a reference to violating triple, a reference to the violated shape and a message text. If a node marked as valid, it would not be validated the same shape again.

Figure 6.1 shows the measurement results for different amounts of violations. The lower the throughput, the lower is the performance of the solution.

The left diagram shows the values for the cold-start situation. The throughput of the different machines (S, P) is developing similar to each other for different amounts of violations.

The right part shows the throughput values for the warm-up situation. Like in the cold-start cases, the two machines (S, P) are performing similarly on the different amounts of violations for the warm-up situation.

For different amounts of violations, the hardware limitations do not have an impact. The quantitative difference between the cold-start and warm-up is around 100 Triple/ms. The JVM JIT features are the reason for the performance differences. For the varying amount of violations, the solution performs well and scales with the hardware.

Impact of Dataset Size

The measurement results of grouped by the dataset size are in Figure 6.2. The higher the Triples/s, the higher the performance.

The left part shows the values for the cold-start scenario. The bigger the dataset, the better the solution performs.

The right diagram shows the values for the warm-up scenario. It shows the same connection between dataset size and throughput. The combination of dataset size full and more than 10% changes lead to a loss of throughput indicating a high resource consumption.

The reason for this observation is a more efficient parallelization of evaluation on big graphs than on small ones. The JVMs JIT-feature produces the differences between the cold-start and the warm-up situation. The more violations are found in the graph, the lower is the throughput of the solution.

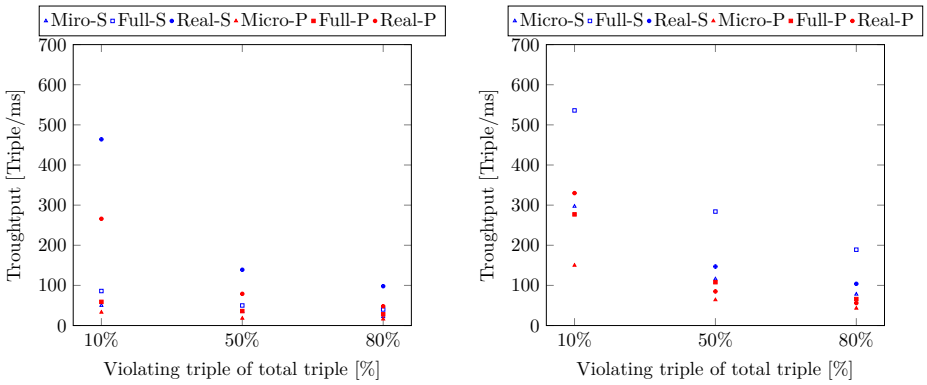


Figure 6.1: Performance as throughput to amount of violations, for cold-start (left) and warm-up situations (right). In two situations both machines (P,S) perform similar and warm-up situation profit from the JVM JIT features. With increasing amount of violations the throughput decreases.

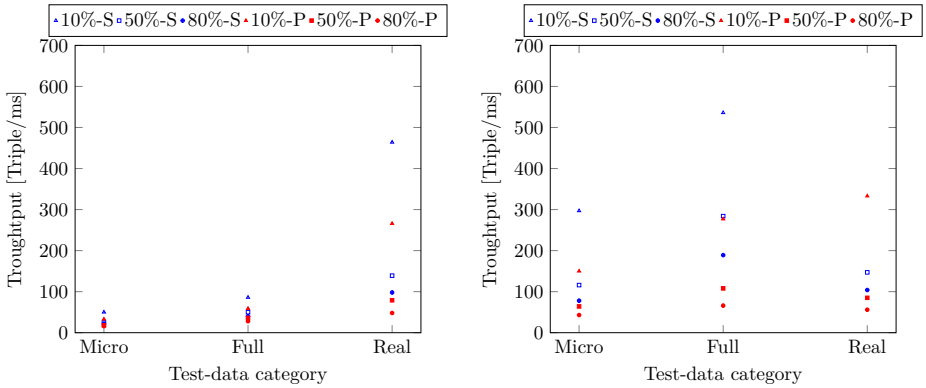


Figure 6.2: Performance as throughput to dataset size, for cold-start (left) and warm-up situations (right). The cold-start situation benefits from the clean heap space the throughput increases with increasing dataset size. With increasing dataset size the performance first raises and then drops with increasing dataset size.

6.2.3 Measurement Analysis

Section 6.2.2 shows the results of the measurements and gives the roots for differences. This section concentrates on the applicability in CGMES driven processes. The problem of model analysis is present in two situations of a CGMES driven process.

1. When an application receives a dataset, the model needs to be evaluated. At this step, the size of a CGM is similar or bigger than the one in the RealGrid category. In the worst case, nearly all data in the model is a violation.

2. The TSO employees make the changes to the model by hand and usually in not more than two IGMs. The size of an IGM is similar to the one in the FullGrid category. There are very few violations, so 10% is a pessimistic upper bound.

Data Import Use Case

The results in Section 6.2.2 cover the UC1-data import case from Section 1.3 with the measurements on models from the RealGrid category. The fact that the import step is usually the first step in the process indicates that the cold-start scenario is the most significant.

Figure 6.2 shows that the solution performs best on the model form the RealGrid category in the cold-start situation. In the worst case, 80% violations the performance drops to 50-100 triple/ms.

Based on the grid sizes from Table 4.1 and assuming a single validation process, the throughput leads to an approximated evaluation time of 100 seconds. In reality, the data split in IGMs and profiles. The profiles of an IGM is usually around 2000-3000 triple that drops the expected evaluations time below 10 seconds per profile.

Data Manipulation Use Case

The results in Section 6.2.2 cover the UC2-data manipulation case from Section 1.3 with the measurements on models from the FullGrid category with 10% violation. Data manipulation occur repeatedly. Therefore, the warm-up case is the most sufficient one.

Figure 6.1 shows that the solution achieve 270 - 530 triple/ms for models of the category FullGrid with 10% violation. Based on the grid sizes from Table 4.1 and assuming a single validation process, the throughput leads to an approximated evaluation time of 40 seconds. In reality, the data split in IGMs and profiles. Only some profiles are affected by the manipulation use case. The profiles of an IGM are usually around 2000-3000 triple that drops the expected evaluations time below one second per profile.

Chapter 7

Conclusion

Chapters 1 and 3 pointed out the need for assessing the quality of CGMES models. They also formulated requirements regarding performance, scalability and maintainability.

Chapter 4 derived a formal definition for each identified type of quality requirements of CGMES. The quality requirements of CGMES cover a wide range.

Chapter 5 developed a SHACL shape for each quality requirement based on the formal definitions. Each shape is executable with a standard-compliant SHACL interpreter. The group of contextual requirements forms an exception: supporting them would require extensions to the SHACL standard. Embedding SHACL into a programming language, as demonstrated with the SHACL JavaScript Extension [15], can be a solution.

A different approach than using SHACL is possible with the Luzzu framework [2, 1]. This framework defines a domain-specific language (LQML) for easy metric implementation. The purpose of the Luzzu Framework is more data orientated than shape orientated. For the requirements covered in this work, the Luzzu framework is a huge overhead and the same holds for SANSA.

Chapter 6 shows that the solution is correct and that the performance is sufficient for industrial application. The joint project *Redispatch-Ermittlungs-Server*[1] (RES) of SOPTIM AG and FGH GmbH will include a quality evaluation based on this solution. From late 2020 we will validate incoming IGM and CGM data from different sources and enrich the datasets with quality information. This quality information strengthens the significance of CGMES process results.

Another use case in the context of TSO planning is the validation of manipulations. Any data manipulation should not decrease the data quality therefore, our solution will provide information to reject manipulations.

[1] `https://www.soptim.de/de/presse/pressemitteilungen/leuchtturmprojekt-der-deutschen-uebertragungsnetzbetreiber/` (announcing the kick-off in 2018)

This solution deals with the level three and four of the CGMES quality definition. They define the requirements regarding RDF specification and object constraints.

For the higher levels like checking the integrity of an IGM, a metric has to query multiple graphs. Such querying is not possible with SHACL; a prior merging would somehow solve the problem. Merging two graphs into one is not an option, because it would increase the size of the graph and therefor increase the execution time. One way would be to use advanced frameworks like Luzzu or SANSA for the metrics. However, Luzzu and SANSA require massive infrastructure. Another way would be to empower SHACL to take multiple graphs into account.

Some rules in the higher levels require numeric values to be in a predefined relation. These types of requirements are not shape-dependent but data-dependent. Numerical checks like *sh:equals* or *sh:lessThan* are already part of the SHACL standard. Injecting the numerical bounds from the data is possible if the dataset to evaluate is available at shape generation. This extra step would add additional complexity to the evaluation process because every evaluation run comes with a shape generation.

Since some surrounding systems in the electrical world require fixed fields length and efficient representation of enumerations, the Quality of CGMES is the length of fields and enum values. The evaluation of field lengths and enum values is efficiently possible with SHACLs *sh:in*, and the string-based Constraint Components feature. The presented solution is not able to deal with these requirements because they are part of a higher level. An efficient mapping from string/enum-based OCL rules to SHACL is a topic for further development in this area.

The group of contextual requirements are also not covered by this solution even though they are part of the covered levels. SHACL does not have a feature which deals with external information or configuration. There are multiple ways to achieve this; one would be to empower SHACL with a context variable similar to $\$this$ or bounded to this. The interpreter binds the value of this variable based on external configuration.

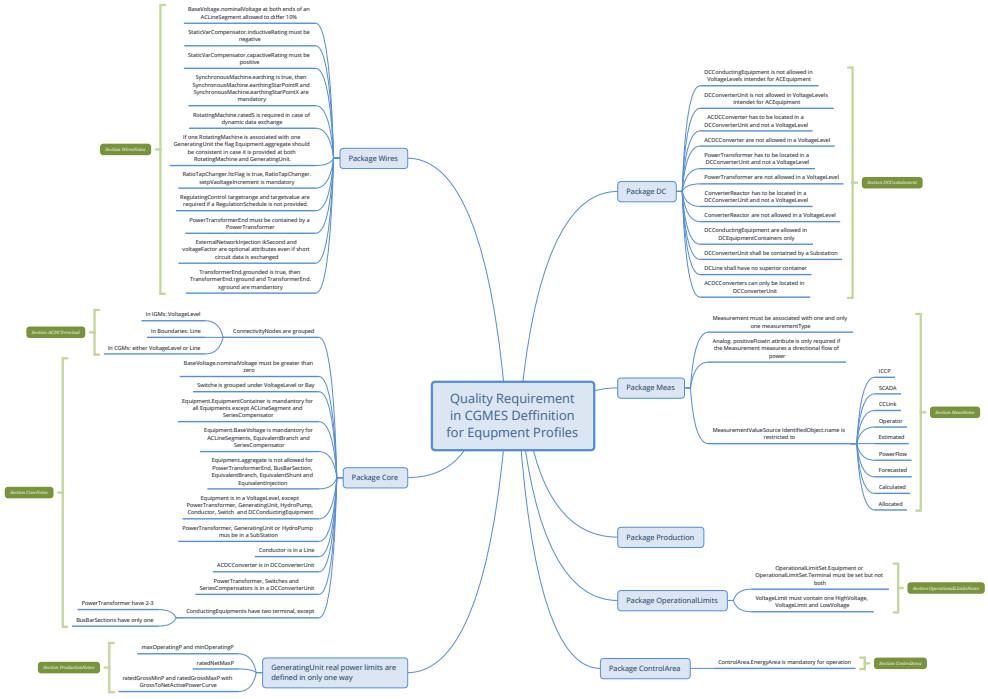
Overall, this work shows how the schema completeness, conciseness and consistency of electrical power system models can be assessed using semantic web technology. The presented approach is part of ongoing software developments towards ensuring a stable and reliable power system in Europe.

Appendices

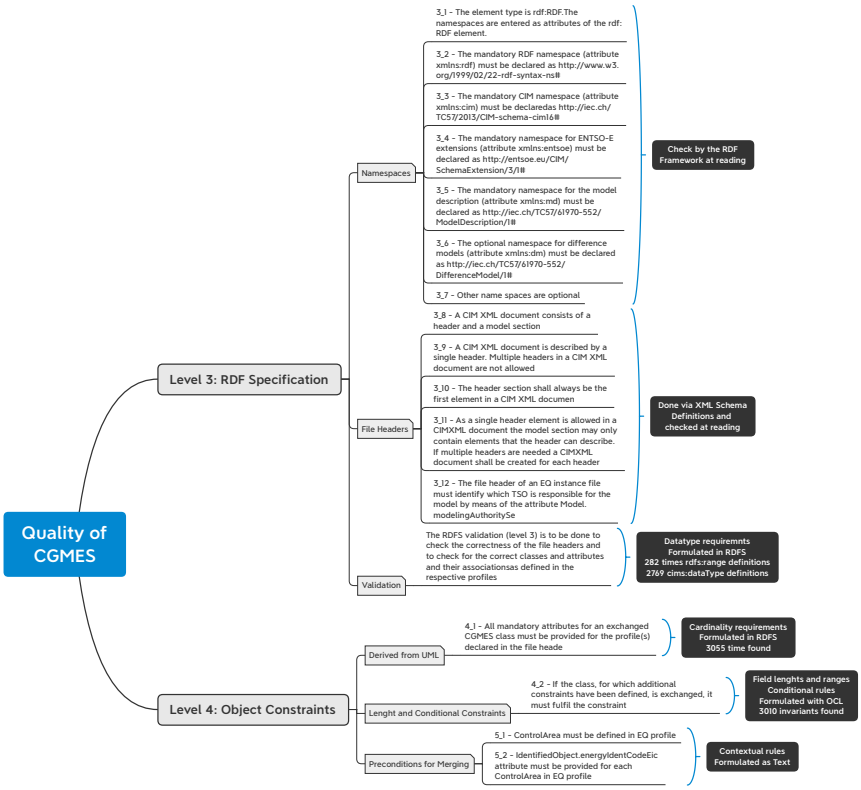
Appendix A

Quality of CGMES

A.1 Requirements from Definition of CGMES



A.2 Requirements from Quality of CGMES



Bibliography

1. J. Debattista, S. Auer, and C. Lange. "Luzzu—a methodology and framework for linked data quality assessment". In: *Journal of Data and Information Quality (JDIQ)* 8.1 (2016), p. 4.
2. J. Debattista, S. Auer, and C. Lange. "Luzzu–A Framework for Linked Data Quality Assessment". In: (2015).
3. J. Debattista, C. Lange, and S. Auer. "Luzzu Quality Metric Language–A DSL for Linked Data Quality Assessment". In: *arXiv preprint 1504.07758* (2015).
4. ENTSO-E. *Common Grid Model Exchange Specification (CGMES) - Structure and rules*. Version 2. ENTSO-E. 2017.
5. ENTSO-E. *Quality of CGMES Datasets and Calculations for System Operations Second Edition*. ENTSO-E. 2016.
6. I. Ermilov et al. "The Tale of Sansa Spark." In: *International Semantic Web Conference (Posters, Demos & Industry Tracks)*. 2017.
7. European Parliament and Council of the European Union. *REGULATION (EC) No 714/2009 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 July 2009 on conditions for access to the network for cross-border exchanges in electricity and repealing Regulation (EC) No 1228/2003*. 2009.
8. European Parliament and Council of the European Union. *Regulation (EU) 2019/943 of the European Parliament and of the Council of 5 June 2019 on the internal market for electricity*. 2019.
9. D. M. T. Force. *Common Information Model*. 1999. URL: `https://www.dmtf.org/standards/cim`.
10. C. Fürber and M. Hepp. "Using semantic web resources for data quality management". In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer. 2010, pp. 211–225.
11. O. M. Group. "Object Constraint Language Specification Version 2.4". In: (2014).
12. ISO/IEC. *ISO/IEC 25012:2008(E) Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model*. 2008.
13. H. Knublauch. *SPIN-Modeling Vocabulary. W3C Member Submission*. 2014.

14. H. Knublauch and D. Kontokostas. *Shapes constraint language (SHACL). W3C recommendation, W3C, July 2017*. 2017.

15. H. Knublauch and P. Maria. *SHACL JavaScript Extensions*. Tech. rep. W3C, 2017.

16. J. Lehmann, G. Sejdiu, and H. Jabeen. "Distributed Knowledge Graph Processing in SANSA". In: *HPI Future SOC Lab: Proceedings 2017* 130 (2017), p. 21.

17. U. S. N. models and forecast tools. *UCTE data exchange format for load flow and three phase short circuit studies*. UCTE Subgroup „Network models and forecast tools. 2007.

18. K. R. Nenadić, M. M. Gavrić, and V. I. Đurđević. "Validation of CIM datasets using SHACL". In: *2017 25th Telecommunication Forum (TELFOR)*. IEEE. 2017, pp. 1–4.

19. Object Management Group. "Unified Modeling Language 2 (OMG UML)". In: (2011).

20. L. L. Pipino, Y. W. Lee, and R. Y. Wang. "Data quality assessment". In: *Communications of the ACM* 45.4 (2002), pp. 211–218.

21. Power Info LLC. *CIMDesk – CIM data engineering solution kit*. 2009. URL: `http://www.powerinfo.us/CIMdesk.html`.

22. Power Info LLC. *CIMSpy – A CIM-based Model Exploratory Tool*. 2007. URL: `https://powerinfo.us/CIMSpy.html`.

23. E. Prud'hommeaux, I. Boneva, et al. *Shape Expressions Language 2.1. Draft Community Group Report*. 2018.

24. H. Pundt. "Field data collection with mobile GIS: Dependencies between semantics and data quality". In: *GeoInformatica* 6.4 (2002), pp. 363–380.

25. C. Stadler et al. "Querying large-scale RDF datasets using the SANSA framework". In: (2019).

26. B. P. Weidema and M. S. Wesnaes. "Data quality management for life cycle inventories—an example of using data quality indicators". In: *Journal of cleaner production* 4.3-4 (1996), pp. 167–174.

27. A. Zaveri et al. "Quality assessment for linked data: A survey". In: *Semantic Web* 7.1 (2016), pp. 63–93.

All links were successfully followed on 10$^{\text{th}}$ June 2020.