Deriving Semantic Validation Rules from Industrial Standards: an OPC UA Study

Yashoda Saisree Bareedu [c,*], Thomas Frühwirth [b], Christoph Niedermeier [c], Marta Sabou [a], Gernot Steindl [b], Aparna Saisree Thuluva [c], Stefani Tsaneva [a] and Nilay Tufek Ozkaya [c]

[a] *Institute of Information Systems Engineering, Technical University of Vienna, Vienna, Austria*
E-mails: marta.sabou@ifs.tuwien.ac.at, stefani.tsaneva@tuwien.ac.at
[b] *Institute of Computer Engineering, Technical University of Vienna,Vienna, Austria*
E-mails: thomas.fruehwirth@tuwien.ac.at, gernot.steindl@tuwien.ac.at
[c] *Corporate Technology, Siemens AG, Munich, Germany*
E-mails: yashoda.bareedu.ext@siemens.com, christoph.niedermeier@siemens.com, aparna.thuluva@siemens.com, nilay.tuefek-oezkaya@siemens.com

Abstract. Industrial standards provide guidelines for data modeling to ensure interoperability between stakeholders of an industry branch (e.g., robotics). Most frequently, such guidelines are provided in an unstructured format (e.g., pdf documents) which hampers the automated validations of information objects (e.g., data models) that rely on such standards in terms of their compliance with the prescribed guidelines. This increases the risk of costly interoperability errors induced by the incorrect use of the standards. There is therefore an increased interest in automatic semantic validation of information objects based on industrial standards. In this paper we focus on an approach to semantic validation by formally representing the modeling constraints from unstructured documents as explicit rules (to be then used for semantic validation) and (semi-)automatically extracting such rules from pdf documents. We exemplify an adaptation of this approach in the context of the OPC UA industrial standard and conclude that (i) it is feasible to represent modeling constraints from the standard specifications as rules, which can be organized in a taxonomy and represented using Semantic Web technologies such as OWL and SPARQL; (ii) we could automatically identify constraints in the specification documents by inspecting the tables (P=87%) and text of these documents (F1 up to 94%); (iii) the translation of the modeling constraints into rules could be fully automated when constraints were extracted from tables and required a Human-in-the-loop approach for constraints extracted from text.

Keywords: Semantic Validation, Information extraction, Natural Language Processing, SPARQL, Human-in-the-loop, OPC UA

1. Introduction

Interoperability in the industry has been a research topic since the 1970s [1] and became even more relevant during the fourth industrial revolution, as Cyber-Physical Production Systems (CPPS) rely on networked manufacturing equipment that needs to be seamlessly integrated, often in run-time, dynamic workflows [2]. To ensure such interoperability, industrial standards are core to many industries and represent vital elements in the ecosystem of that industrial domain by providing shared guidelines for data representation and exchange. For example, the International Electrotechnical Commission's (IEC) Common Information Model (CIM) provides a data model for standardised information exchange of energy grid descriptions [3]. Another example is OPC Unified Architecture (OPC UA) - an industrial standard that ensures interoperability between industrial devices on the technical, syntactical, and semantic level: while the transport protocols and payload formats are defined on the technical and syntactical levels, semantic interoperability is achieved through extensible information models, which capture domain-specific knowledge.

[*] All authors have contributed with equal effort to the work and therefore are listed in alphabetical order.

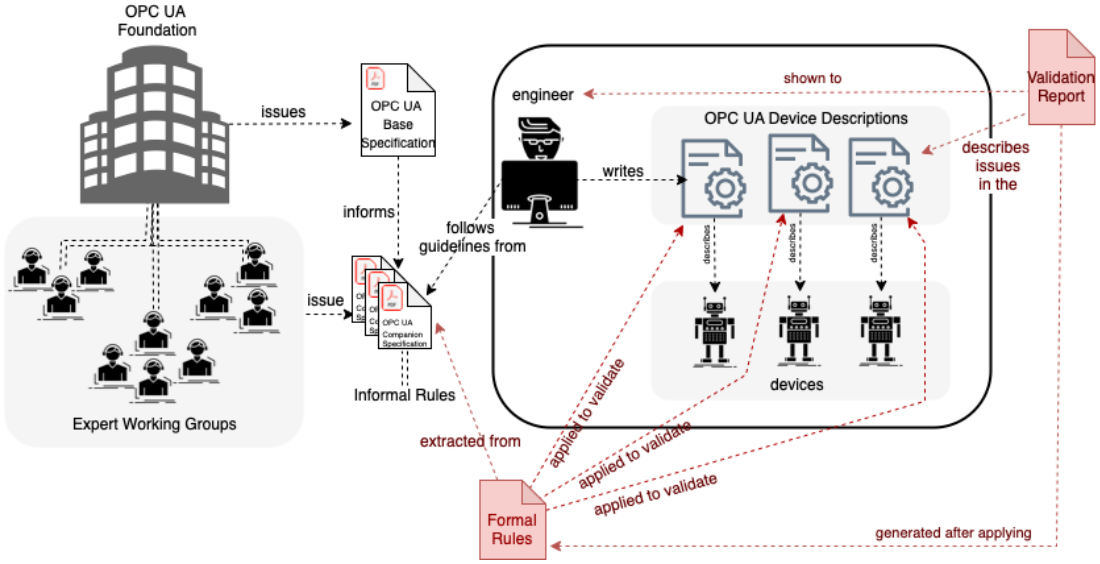


Fig. 1. Standard development, standard usage and the need for semantic validation (shown in red) in the case of OPC-UA.

We distinguish two core stages in the life-cycle of an industrial standard, which we depict in Fig. 1 and discuss in the context of OPC UA. First, *standard development* is undertaken by an industrial standardization body. In the case of OPC UA, standard development is performed by the OPC Foundation which involves experts from the industry and academia around the world (left part of see Fig. 1). The OPC Foundation has defined rich *domain independent* semantics for OPC UA information models captured in one of the *OPC UA base specifications*. Several domain specific working groups are part of the OPC Foundations. Their role is to issue domain specific information models, compliant with the meta-model defined in the base specification, in the form of *OPC UA companion specifications*, e.g., in the domains of Robotics, Machinery etc.

A second key stage is the *standard usage*, when stakeholders in the domain pertinent to the standard create information objects that follow the guidelines proposed by the standard, thus achieving interoperability in that industrial domain. In the case of OPC UA, several manufacturers around the world rely on this standard to ensure interoperability of their devices with the devices from other manufacturers. To that end, *engineers* employed by these manufacturers need to write OPC UA information models of such devices. These models are serialized as structured, XML-based documents (also referred to as NodeSet files) capturing the structure and functionality of the device by following the OPC UA standard laid out in the base and (relevant) companion specifications (right side of Fig. 1).

Provided that the information objects fully comply to the guidelines of the standard, interoperability across stakeholders subscribing to the standard is achieved. However, the semantics defined by (most of) the standards (and by OPC UA in particular) are only available as unstructured (pdf) documents. Currently the assumption is that the *engineer* creating information models has a thorough understanding of the base/companion specifications and has correctly applied all pertinent guidelines described in several hundred pages. In practice, this is unrealistic and often leads to the incorrect application of the standard which triggers costly interoperability errors. There is a need for *automatic semantic validation*, in order to easily and reliably check the compliance of an (OPC UA) information model with relevant standard specifications.

In this work, we address this common pattern in the landscape of industrial standards, with a special focus on OPC UA. In particular, we propose an approach to semantic validation in which modeling guidelines available in non-structured specifications are translated into formally represented rules that can be used to automatically validate the correctness of the information models in terms of their compliance with the specifications (red elements in Fig. 1). To that end we investigate the following research questions:

- *RQ1: To what extent can informal modeling guidelines be captured into formal rules?* Is it possible to identify such rules? Can they be organized in a taxonomy? What is the best way to rep-

resent them? If capturing informal guidelines into formal rules is possible, it is unfeasible to expect that this process will be performed manually by interested stakeholders, such as the specification authors. Therefore, this process should be automated as much as possible, as addressed by the next two RQs.
- *RQ2: To what extent can modeling guidelines be automatically identified in the specification documents?* How complex is the task of identifying constraints in specifications? What information extraction methods are amenable for this task?
- *RQ3: To what extent can informal modeling guidelines be automatically mapped to formal rules?* What methods can be used to map between unstructured constraints and formal rules?

We answer these research questions through the following methodology leading to several contributions:

- We propose a *high-level approach* for the extraction of modeling constraints from specifications and their formal representation, e.g., as SPARQL constraints. The proposed technical solution has several core components: (1) a catalogue of major rule types and their corresponding representations as SPARQL query (templates); (2) a component to automatically identify constraints in specification documents; (3) methods for generating rules from textual constraints identified in stage (2) according to templates proposed in stage (1).
- We apply the steps of this approach in the concrete case of OPC UA specifications, resulting in contributions such as: (a) an *OPC UA specific rule catalog and classification* including the corresponding SPARQL representations; (b) *OPC UA specific semi-automatic methods* for extracting rules from pdf documentations.
- We evaluate the performance of the technical components individually. Then we provide a feasibility evaluation of the proposed method for one concrete OPC UA specification in the Machinery domain and perform an evaluation campaign involving Machinery working group experts.

We continue by providing further motivation for our work and background related to OPC UA in Sect. 2. Then the proposed high-level approach is described in Sect. 3, and the following sections investigate one of the research questions within the context of OPC UA, including the rule taxonomy (Sect. 4), methods for automated constraint extraction (Sect. 5) and methods for generating rules from constraints (Sect. 6). These individual components are evaluated in Sect. 7. We discuss related work and concluding remarks in sections 8 and 9 respectively.

2. Background and motivation

2.1. Background: OPC UA

Basic OPC UA Notions. OPC UA is a framework for industrial communication that additionally provides information modeling capability. The communication of OPC UA is based on the client/server principle, but part 14 of the specification also introduces a publish/subscribe communication paradigm.

OPC UA information modeling provides structure and context to the data of a production facility. It allows the description of devices, such as sensors, actuators, as well as whole production machines, in an object-oriented and semantically meaningful way [4].

The basic elements of the information model are: (1) nodes that represent objects, variables, methods, etc. and (2) references which are used to model relations between nodes. Eight different node classes are defined by OPC UA: *ObjectType*, *Object*, *DataType*, *VariableType*, *Variable*, *ReferenceType*, *Method*, and *View*. Depending on the node class, a set of attributes is defined for each node. One of the most important attributes, which is supported by every node class, is the *NodeId*. The *NodeId* is used to unambiguously identify each node by a so-called *NamespaceIndex* and an *Identifier*. Some other attributes are the *NodeClass* itself, the *DisplayName* (a human-readable name for the node), *Description* (a human-readable description of the purpose of the node), maybe a *Value*, and many more. Nodes can be instantiated based on the defined *OjectTypes* or *VariableTypes*, similarly to object-oriented programming. These objects, variables, and methods are called *instance Nodes*.

The OPC UA information model is extensible, and it is used by domain experts for certain domains (e.g., machinery) to create Companion Specifications. The experts agree upon the modeling and release it as an industry-standard model, which can be used free of charge by anyone, e.g., other machine vendors. Thus, these OPC UA Companion Specifications facilitate interoperability at the semantic level.

Fig. 2 shows an example of an information model from the Machinery Companion Specification using the graphical notation defined in the OPC UA base

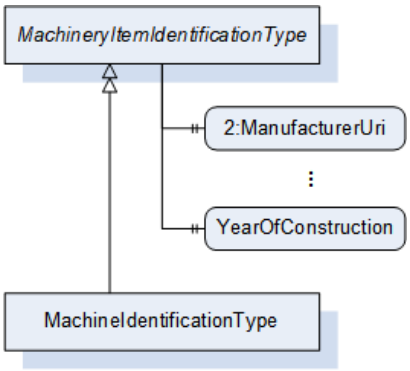


Fig. 2. Part of the OPC UA information model as it is defined in the Machinery Companion Specification.

specification part 3. The abstract *ObjectType MachineryItemIdentificationType* has two variables *ManufacturerUri* and *YearOfConstruction*. These variables are related to the *ObjectType* via a *HasProperty* reference. The *HasSubType* reference is used to specify another *ObjectType* called *MachineIdentificationType* which inherits the properties from *MachineryItemIdentificationType*. As this *ObjectType* is not defined as abstract, instances of this type (Objects) can be created. Even if no instance is shown in this example, and not every aspect of the model is explained, it provides some insight into the OPC UA information modeling concepts. More details can be found in [5].

OPC UA specifies an XML Schema, which defines the information model as an XML serialization. Such XML files are called *OPC UA NodeSet files* and can be generated with both open-source and commercial tools. The NodeSet file can be loaded and instantiated by an OPC UA server to make the information available to other clients. This exposed information model is the server's *address space*.

Need for semantic validation. Even if OPC UA provides sophisticated information modeling possibilities, it lacks the ability of defining restrictions on the model. Such restrictions could be beneficial to enforce semantic interoperability when applying the information model in a certain use case. Currently, constraints on the model have to be documented in text, using natural language. Thus, currently they cannot be checked automatically. As an example, the Machinery companion specification states, that the property variable *YearOfConstruction*, as shown in Fig. 2, shall be a four-digit number, such as "2022" or "2023". However, this constraint is only specified in the companion specification (textual) and not in the information model. Thus, nothing prevents a machine manufacturer from *incorrectly* using the information model as defined in the companion specification and assigning the value "22" instead of "2022". The problem occurs when a system in the factory tries to automatically schedule maintenance actions based on the age of the machine. The system will not be able to interpret "22" as the year "2022" and will fail. This shows that the semantic validation of machine information models against the standard is crucial to ensure interoperability between machines and thus avoid costly errors and malfunctioning due to the incorrect use the standard.

Such errors could be avoided by ensuring that the NodeSet correctly follows the specification guideline. This "validation" process is currently the task of the engineer that creates such NodeSets and it is unrealistic since the OPC UA base specification itself contains about one thousand pages and the number of companion specifications is rapidly increasing yearly to address demand for ensuring interoperability between machines from different domains.

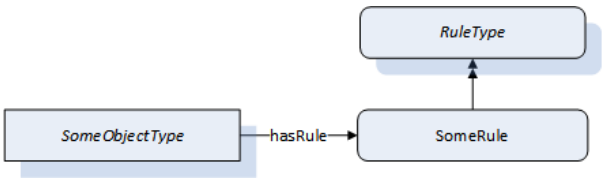


Fig. 3. Rules defined in the OPC UA information model.

There is therefore a need of a *paradigm shift* towards automated, semantic validation of OPC UA information models. To that end, the OPC Foundation already investigates the possibility of incorporating rules in the OPC UA NodeSet files in the future. Figure 3 depicts how this could be modeled in OPC UA in a very abstract way using thee graphical OPC UA notation. The exact implementation is still subject to debate and not the focus of this paper. However, the figure illustrates the basic idea, which is specifying rules for individual nodes within the OPC UA information model. Generally speaking, these rules express restrictions on the node or instances of the node.

To enable such a paradigm shift within OPC UA motivated our work towards a (semi-)automated approach for extracting the constraints from the specifications, organizing them into a rule taxonomy and formalizing them into Semantic Web form such as OWL and SPARQL queries that can achieve automated semantic validation (a technical realisation of such a validation process is reported in Sect. 7.2).

2.2. Motivation: stakeholders use case

Several stakeholder groups can benefit from our work that support this paradigm shift towards semantic validation. Firstly, *users of the Companion Specifications, e.g., a machine vendor* will be able to check the conformance of the created address space of, e.g., a machine with the applied Companion Specification. As a result, *plant operators* can rely on conformance with companion specifications and simplify integration procedures.

Secondly, *Companion Specification creators* (i.e., members of the OPC UA working groups) will be able to check consistency between their companion specification and NodeSet files created in terms of this specification. They will also be able to provide a mechanism to enforce semantic interoperability (by providing formal rules in addition to the pdf-based companion specification). Capturing the informal specification of standards in formal rules will make sure that the standards are correctly applied, thus supporting interoperability. This stakeholder group requires automated support for the process of creating formal rules.

On a long term, work on formal verification of compliance with standard specifications addresses the needs of all stakeholder groups. This paper primarily focuses on supporting *companion specification creators* in turning their specifications into formal rules.

3. Overall approach: Deriving rules from industrial standard specifications

We propose a high-level approach for (semi-)automatically translating modeling guidelines expressed in specification documents into formal rules (that can be used for semantic validation). To that end, we introduce the following terminology:

- *Constraints* represent concrete snippets in the standard specifications that express modeling constraints which are candidates for being formalized into rules. Constraints can be specified either in an unstructured (textual) or semi-structured (tabular, list) form.
- *Rules* are formally represented constraints formulated in a Semi-Formal Notation (SFN), as SPARQL queries, or in any other suitable language. They are applied on the information models created by end users in order to verify that they comply to the constraints expressed in the specification. More complex constraints might require representation through multiple rules, that we refer to as *Rule Sets*.
- *Rule Templates* are generalized versions of rules that use variables. Rules are derived from templates by replacing the variables with specific values. Templates are an important mechanism for supporting automatic rule extraction by being populated with automatically extracted values.

The overall approach for rule extraction encompasses several key stages:

1. *Stage 1: Rule taxonomy creation*: a first stage is understanding whether constraints described in a standard's specification are amenable to be represented as formal rules. If this is feasible, rules that can capture such constraints need to be identified, represented in a formal language (e.g., SPARQL) and organized in a taxonomy.
2. *Stage 2: Constraint extraction* from specification documents, identifies those snippets in the specifications that contain information which should be represented as formal rules. Such information could be present through multiple modalities including textual descriptions, tables or images. In principle, these different modalities can be used individually or in tandem to increase the performance of the extraction process.
3. *Stage 3: Rule generation* represents making the transition from the constraints extracted from the specification document (in Stage 2) into formally represented rules as identified in Stage 1. Rule templates can be used to automate this process when a clear mapping can be established between automatically extracted information and the template variables. In other cases, human intervention might be needed.
4. *Stage 4: Constraint and rule validation.* Although the aim is to automate the constraint extraction and rule generation, automated methods rarely provide perfect results. Therefore, a concluding stage in this pipeline involves domain experts who validate (and if needed correct) the outputs of stages 2 and 3 above. With this process, ground truth data is collected from experts to inform the further development and extension of the existing methods.

The concrete techniques used to implement the stage of our approach highly depend on the characteristics of the concrete standard. Therefore, in the

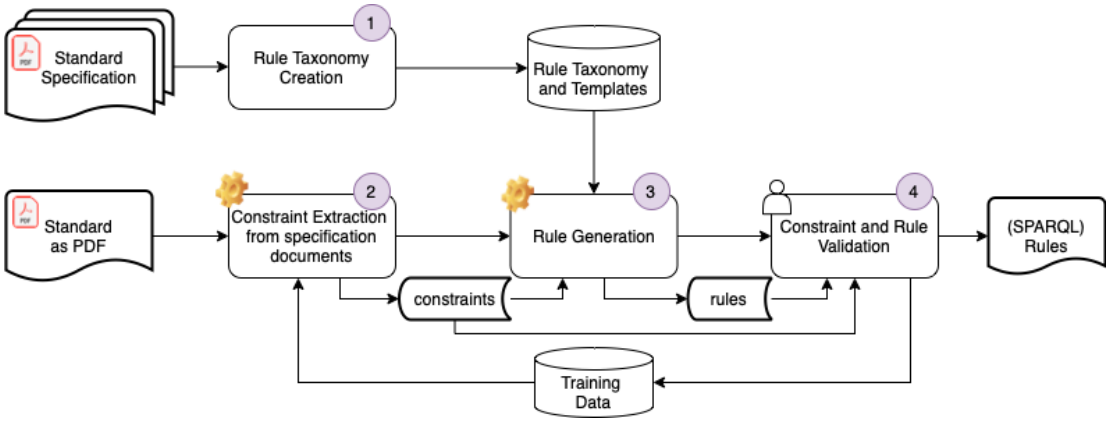


Fig. 4. Overall approach for deriving rules from specifications of industrial standards.

next sections, we describe how we implemented these stages in the context of the OPC UA specification.

4. Stage 1: Rule taxonomy creation

The creation of a rule taxonomy and corresponding rule templates addresses RQ1. The process involved the following four steps (see Fig. 5):

- *Step 1: Identify constraint types.* First, we evaluated whether the tables present in the OPC UA companion specifications represent constraints with regard to the OPC UA information model. The core specification was not considered, because a preliminary analysis showed that it contains unique rules and, therefore, provide little potential for identifying rule templates. The output of this step was the identification of various constraint types, e.g., *ObjectTypeDefinition* constraint. Examples of identified constraints can be found in Sect. 5.
- *Step 2: Structure rule taxonomy.* As stated above, constraints can be verified by checking one or often multiple rules. The different rules were structured in a taxonomy (see Sect. 4.1) but not yet formulated.
- *Step 3: Formulate rules in SFN/SPARQL.* An example for each rule within the taxonomy was formulated in SFN. In addition, some rules were also expressed in SPARQL, which allows them to be verified against existing OPC UA companion specifications (see Sect. 4.2).
- *Step 4: Create rule templates.* Finally, the constraint-specific information of each rule was replaced by template variables, resulting in *rule templates*. Finally, the rule templates were also organized in a taxonomy (see Sect. 4.2).

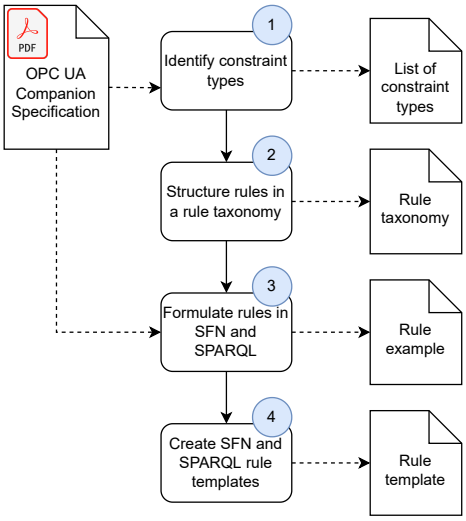


Fig. 5. Process for creating the rule taxonomy and templates.

Additional information about each step and intermediate results are provided in the following sections.

4.1. Step 2: Structure rule taxonomy

Based on the constraint types identified in Step 1, a number of individual rules were derived and subsequently classified into a rule taxonomy. The top-level hierarchy of this taxonomy (Fig. 6) distinguishes between (1) *Global Rules*, (2) *Node Rules*, and (3) *Type Rules* based on the scope of the rule application.

- *Global Rule* - A *Global Rule* is not associated with a specific node but expresses some gen-

eral rule to be fulfilled within the NodeSet (e.g., within the entire device description). For example, the following rule expressed in SFN checks that the node MotionDeviceSystemType exists in the information model (see also Fig. 7).
Example in SFN: *The node MotionDeviceSystemType exists.*

- *Node Rule* - A Node Rule only applies to the node that the rule is associated with.
  Example in SFN: *The node MotionDeviceSystemType references a node MotionDevices. This node has ModellingRule Mandatory.*
- *Type Rule* - A Type Rule applies to all nodes derived from the type node that the rule is associated with.
  Example in SFN: *For all instances of MotionDeviceSystemType: The instance references exactly 1 MotionDevices node.*

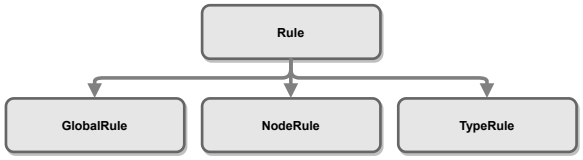


Fig. 6. Top-level rule classes.

These three basic top-level categories are refined, as shown in Fig. 25 - 27 (Appendix A), resulting in 3 *Globale Rules*, 24 *Node Rules* and 18 *Type Rules*.

Currently, there exist only three different types of Global Rules (Fig. 25). The most important one is to check if a specific node exists in the information model or not. The other two are concerned with the initialization of read-only and write variables. As they cannot be associated with a specific node, such rules are defined globally on the entire NodeSet file.

Restrictions regarding the attributes of a node, the references between them, and a referenced node have to be checked at the type level as well as on node instances of an OPC UA information model. Thus, these classes can be found in the taxonomy for *NodeRules* and *TypeRules*.

Additionally, Node Rules have restrictions on general *Data Type Structures* and *Enumerations*. These rules and the further refinement of these rule classes are depicted in Fig. 27.

For *TypeRules*, an addional rule is needed to check that an abstract ObjectType is not instantiated as an Object (Fig. 26).

4.2. Step 3-4: Rule examples and templates.

The rules present in the taxonomy were formulated in SFN and SPARQL using examples from the companion specifications. These rules were then generalized into rule templates by replacing constraint-specific information with template variables.

The rules and rule templates are not presented as a whole here, but the following examples should provide an impression of general concepts and the general form of rules formulated in SFN and as SPARQL queries. In general, SPARQL rules produce an error message if the rule is violated, and otherwise, they return no result. The rule, presented in Fig. 7, checks the existence of a node *MotionDeviceSystemType* on the global, NodeSet file level. The SPARQL query results in a message if no such node could be found and returns no results otherwise.

SFN:

```
The node MotionDeviceSystemType exists.
```

SPARQL:

```
PREFIX ta: <http://opcfoundation.org/UA/Meta/TA/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?msg
WHERE {
  FILTER NOT EXISTS { ?s ta:browseName
"http://opcfoundation.org/UA/Robotics/MotionDeviceSystemType"^^xsd:anyURI }
  BIND(STR('Node http://opcfoundation.org/UA/Robotics/MotionDeviceSystemType
does not exist') as ?msg)
}
```

Fig. 7. SFN and SPARQL rule examples.

The SFN and SPARQL *rule templates*, depicted in Fig. 8, are generalized versions of the rule which is depicted in Fig. 7. They verify that a node with a specific browse name, provided via the template variable @@BrowseName@@, exists in the ontology representing the NodeSet file. Such templates are important because they can be used as basis for automatic rule extraction processes that can generate concrete rules by replacing the template variables with concrete values.

SFN:

```
The node @@BrowseName@@ exists.
```

SPARQL:

```
PREFIX ta: <http://opcfoundation.org/UA/Meta/TA/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?msg
WHERE {
  FILTER NOT EXISTS { ?s ta:browseName "@@BrowseName@@"^^xsd:anyURI }
  BIND(STR('Node @@BrowseName@@ does not exist') as ?msg)
}
```

Fig. 8. SFN and SPARQL rule template examples.

5. Stage 2: Constraint extraction from specifications

To address RQ2, we investigated various methods to support the automatic detection of constraints in OPC UA specifications. The OPC UA industrial standards, published in PDF format, convey information by relying on three modalities: (i) tables capturing in a structured way modeling guidelines pertinent for the corresponding domain; (ii) graphical charts depicting the components of the information model (see example in Fig. 2); (iii) textual descriptions explaining the information represented in tables and graphical charts. We investigate to which extent information related to modeling constraints can be extracted automatically from tables (Sect. 5.1) and text (Sect. 5.2) in OPC-UA specification documents.

5.1. Constraint extraction from tables

We report an initial study to clarify whether tables are a good source for constraint extraction (Sect. 5.1.1), the creation of a data set as a basis for this process (Sect. 5.1.2), as well as an approach for extracting constraints from tables (Sect. 5.1.3).

5.1.1. Preparatory Study: which tables are useful for constraint extraction?

To extract all relevant constraints from specification documents an understanding of the information structured in tables is needed and whether tables contain information that is relevant for the constraint extraction. Three of the authors performed an analysis of 543 tables from 10 different companion specifications and concluded that:

- tables are a rich and structured source of constraints;
- there are several *table types*, and each table type conveys specific OPC UA constraint types. For instance, a *Reference TypeDefinition* table can include a symmetric constraint and an inverse name constraint while an *Object TypeDefinition* table contains among others cardinality and existence constraints.

As part of the conducted table analysis, 42 table types are identified, 9 of which (e.g., *Example*, *Commonterms* table types, etc.) are considered irrelevant for the constraint extraction task. Based on the number of instances (e.g., concrete tables) a table type has and the number of specification documents that used this table type, the top most frequent 11 table types are selected and used as input to the automated extraction process described in Sect. 5.1.3. These table types are: *Object TypeDefinition*, *Enumeration*, *Method Parameters Definition*, *Method AddressSpace Definition*, *Method ResultCode Definition*, *DataType Structure*, *Reference TypeDefiniton*, *NamespaceURI*, *ProfileURI*, *NamespaceMetadata*, *Transition*.

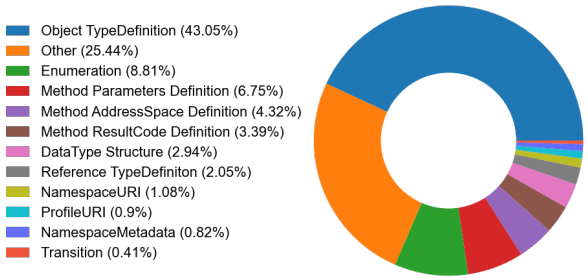


Fig. 9. Distribution of table types in 28 preselected OPC UA companion specifications.

5.1.2. Data set creation

For the task of automatically extracting constraints from tables, 28 companion specification PDF documents are considered. Fig. 9 visualizes the distribution of the selected table types in those 28 documents. It shows that *Object TypeDefinition* type tables appear most frequently and account for 43% of all tables in the inspected specifications. *Enumeration* type tables make up 9% percent of all the tables in these specifications. The tables from the 11 table types selected as most frequent during the analysis phase (Sect. 5.1.1) cover almost 75% of all occurring tables. The rest of 25.5% tables (shown as *Other* in Fig. 9) belong to table types that are very infrequent or are specific to only a single document.

5.1.3. Approach for constraint extraction from tables

In this section we focus on the process of automatically extracting OPC UA constraints from tables. Figure 10 shows an overview of the approach and each step of the process is explained next.

Step 1 - Table detection and extraction. This step takes as input a PDF companion specification document and returns a set of tables. It relies on *Camelot* (https://camelot-py.readthedocs.io), an open-source python library, specialised in the automatic detection and individual extraction of different tables from PDF documents.

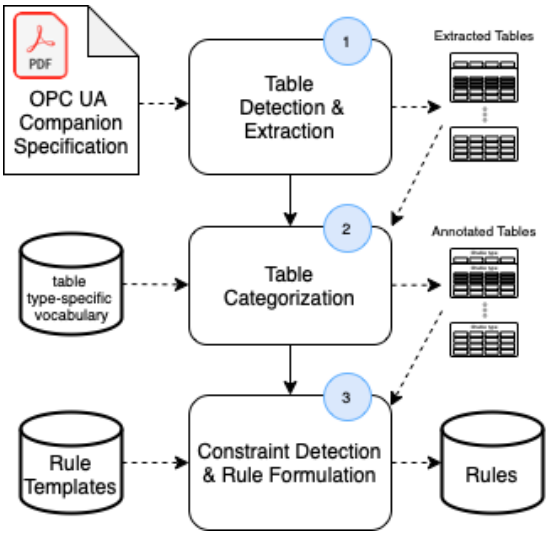


Fig. 10. Process of extracting constraints from tables.

Step 2 - Table categorization. In this step, the set of tables from the previous step is processed and each table is categorized into its corresponding table type.

We apply a heuristics based approach which relies on making use of two types of manually specified lists of terms: (i) the first list contains terms that indicate that a table is of a certain type; and (ii) the second list contains terms that are not specific to that table type.

Fig. 11 shows an example of a filter used to identify whether a table is of type *Object TypeDefinition*. Strings such as "TypeDefinition" and "isAbstract False True" indicate that the table is of type *Object TypeDefinition*, while the keywords "ToState" and "FromState" refer to a *Transition* table and would thus classify a table as not-*ObjectTypeDefinition*. Using this technique to further filter out the table types results in the extraction of their exact types.

The main challenge in this step remains the inconsistencies in the appearance of a specific table type in the companion specification documents. Since not all tables are compliant with the defined OPC UA companion specification template, a table could sometimes have different structural formats with new or varying column names, new positioning of the columns in the table, missing or misaligned rows and columns, as well as typographical errors. Therefore, the classification of the table types had to be further customized to increase the correctness in table extraction. Since only known inconsistencies among the same table type were considered, the results of the algorithm might vary when applied to new specification documents.

```
objtypdef = ["IsAbstract False True"," HasComponent
HasProperty Requires", "NodeClass Object  Variable",
"TypeDefinition", "ModellingRule MandatoryPlaceholder
OptionalPlaceholder", "OrganizedBy", "Organized by",
"Powerlink Attributes", "Details", "DisplayName", "Access
level", "ValueRank"]
nonobjtypedef = ["InverseName","Symmetric", "Subtype of
HierarchialReferences defined", "Argument[]",
"InputArguments","OutputArguments","Namespace","Conf
ormanceUnit","Conformance Unit","ToState","FromState",
"HasEffect","Notes –  Notes referencing footnotes of the
table content.", "NOTE  Notes referencing footnotes of the
table content.","SourceBrowsePath","Source Path"]

for table in tables:
if any(s in table.df.to_string() for s in objtypdef):
if not any(s in table.df.to_string() for s in nonobjtypedef):
```

Fig. 11. Filtering *Objct TypeDefinition* table types by using lists of indicative/non-indicative strings.

Step 3 - Constraint detection & rule formulation. Once the types of the extracted tables are known, for each table type a set of its respective type-specific constraints are extracted. Constraints are detected in the tables by iterating through the respective rows and columns. Subsequently, rules that verify each constraint can be automatically extracted by filling in the values extracted from the tables into predefined rule templates (rule templates are explained in detail in Sect. 4). This is feasible because the Rule Taxonomy contains a constraint taxonomy which maps constraint types to individual rule templates. An example of the constraint extraction is illustrated in Fig. 15. In the example, SPARQL templates are populated with the table values. An evaluation of tabular constraint extraction is provided in Sect. 7.3.

5.2. Constraint extraction from text

OPC UA companion specification documents include large amounts of tables and figures. Nevertheless, it can be observed that the text surrounding the tables can extend the constraints defined in the tables or introduce new constraints. Therefore, the extraction of constraints from text is an important task. While extracting constraints from tables is a straightforward process, extractions from text require a well-trained algorithm and a human-in-the-loop for verification.

We investigate two different approaches - a machine learning based classification and a lexical pattern-based extraction, introduced in the next sections.

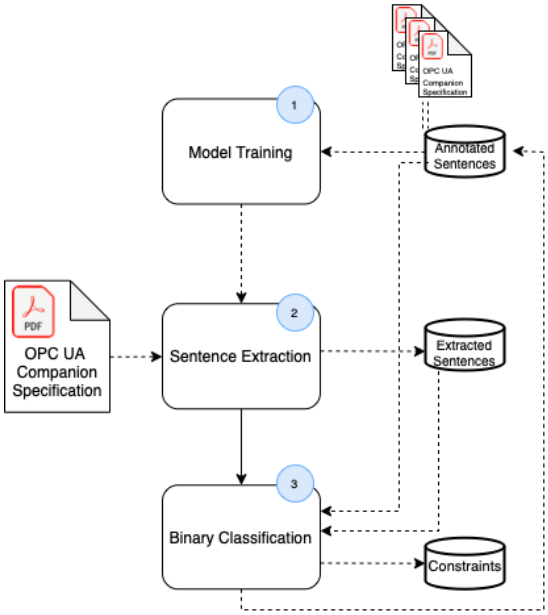


Fig. 12. Process of extracting constraints from text.

5.2.1. Machine learning based approach

The first approach frames the constraint identification as a binary classification problem. It involves training a classification model using labeled sentences and then classifying new text inputs as constraints or not-constraints.

Training set. A prerequisite for this approach is the availability of an annotated data set for the training of the classification algorithm. For this purpose one companion specification (*PackML*) was analyzed by the authors of this paper (who are also OPC UA members) and constraints were manually identified. The resulting training data set includes 198 text snippets which consist of either a single sentence or several sequential sentences extracted from the specification and their binary classification as constraint (approx. 20% of the data) or not-constraint (80%). A sample of the data can be seen in Table 1.

Fig. 12 shows an overview of the extraction process as discussed next.

Step 1 - Model training. By using the pre-annotated text snippets 8 scikit-learn based machine learning models (*Nearest Neighbors, Linear SVM, RBF SVM, SGD, Decision Tree, Random Forest* and *Neural Network, AdaBoost*) are trained to recognize different types of output - in our context, to differentiate between constraints and not-constraints. The annotated sentences from the PackML training data were split into Train and Test sets with a ratio of 4:1. When trained with enough example data, the models can then predict the type of new text inputs.

Step 2 - Sentence extraction. The next step is to extract all sentences from the PDF specification document from which the constraints should be derived.

Since the first chapters of each companion specification include general information about OPC UA, terms explanations as well as introductory examples, they are not considered for this task. For the extraction of the rest of the sections, *Spacy* (https://spacy.io) and *PyPDF2* (https://pypi.org/project/pyPdf/), Python libraries supporting various Natural Language Processing tasks such as information extraction, are used.

Step 3 - Binary classification. After a set of sentences is extracted, each sentence needs to be categorized as constraint or not-constraint. First, the extracted sentences, as well as the training data, are preprocessed using stop word removing, lemmatization and tokenization. Second, each of the trained models is applied to the extracted sentences and their results are compared. The results are discussed in Sect. 7.4.1.

5.2.2. Lexical pattern-based approach

In the English language, a sentence can be considered a rule if it conveys a semantic meaning of some function or some property or something that "has to be present" in some entity or needs to be necessarily followed. Such sentences have a syntactical structure that contain representative words such as *must, have to, should, will,* etc. Such words are called auxiliary verbs and play a key role in identifying if a statement is a "compulsory condition" and thus a constraint. This observation prompted us to experiment with a lexical-pattern based constraint extraction, depicted in Fig. 13 and explained next.

Step 1 - Linguistic patterns identification. The first step of the approach is to analyze the structure of constraints. For instance, a constraint could be expressed with a auxiliary verb in a lexical pattern structure such as *pronoun/noun/(proper noun) + auxiliary verb + verb*. An example is: "At least one instance of a MotionDeviceSystemType must be instantiated in the DeviceSet.". The word *MotionDeviceSystemType* is a proper noun and vocabulary belonging to the OPC UA context, *must be* is a auxiliary verb and these words are followed by the verb *instantiated*. Subsequently, the following lexical patterns are formulated:

Table 1
A sample of the gold standard data for the *PackML* companion specification.

Text Snippet	isConstraint
UnitCurrentMode - is used to display the current mode of the instance of this type. The DataType is Enumeration which is abstract, but an instance shall be assigned a concrete enumeration, which corresponds to the enumeration listed in SupportedModes.	yes
EquipmentInterlock.Blocked - If TRUE, then processing is suspended because downstream equipment is unable to receive material (e.g. downstream buffer is full).	no

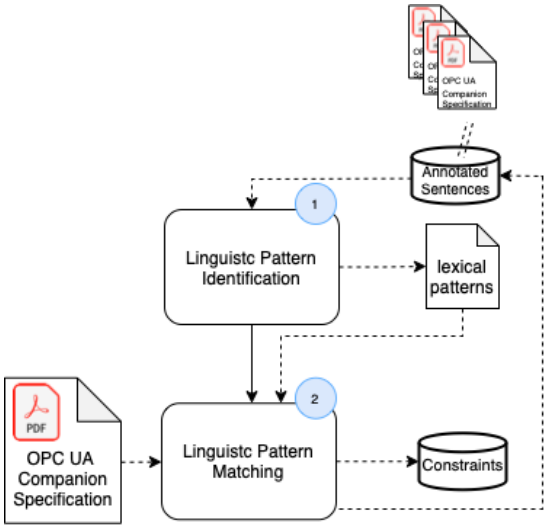


Fig. 13. Rule-based approach for extracting constraints from text.

1. pattern1 = ['POS':'SCONJ','OP': '?', 'POS':'DET', 'POS': 'NOUN','OP': '?', 'POS': 'PROPN']
2. pattern2 = ['POS': 'PROPN', 'POS': 'NOUN','OP': '?', 'POS': 'PUNCT','OP': '?', 'POS': 'AUX', 'POS': 'PUNCT', 'OP': '?', 'POS': 'DET','OP': '?', 'POS': 'ADV', 'OP': '?', 'POS': 'VERB']

Here POS is a linguistic attribute that defines the parts of speech of the word in our pattern. Operators and quantifiers define how often the token must be matched. In the above pattern the operator OP: '?' makes the Parts of speech or POS token with the AUX or auxiliary verb matching, optional, by allowing to match 0 or 1 times. Similarly, PROPN - represents a proper noun, PUNCT - represents punctuation, DET - represents determiner, ADV - represents adverb, SCONJ - represents subordinating conjunction (e.g. *if, while, that*).

Step 2 - Linguistic patterns matching. For this task, we use *Spacy* and its *Matcher* object, which allows for the specification of linguistic patterns and leads to extracting only sentences structured according to the preassigned pattern, which are likely to be constraints. We evaluate this method in Sect. 7.4.2.

6. Stage 3: Rule generation and validation

Our third research question focuses on the feasibility of mapping constraints specified in the tables and text of specification documents to rules. In the case of OPC UA, there is a strong link between table types and rule types, and rules can be generated automatically from tables as we describe in Sect. 6.1. As it is more difficult to connect textual constraints to rules, we rely on a Human-in-the-loop approach to identify correct rules for constraints extracted from text (Sect. 6.2).

6.1. Generating rules from tables

All tables found in the companion specifications are classified as NonCheckableConstraint, RuntimeConstraint, or OfflineCheckableConstraint. The constraint classes are depicted in Fig. 14, which shows a snippet of the created ontology that explicitly defines the constraint and rule types. NonCheckableConstraints are tables with information that can not be checked on the information model. An example would be the description of parameters. RuntimeConstraints can only be checked on a running instance of an OPC UA server because the information is not directly available in a NodeSet file. Examples would be Server Profiles or Namespace URIs. Our work focuses on the so-called OfflineChechableConstraints, which are concerned with information about the structure of the information model.

To *automate the generation of rules from tabular data*, we relate each one of an OfflineChechableCon-

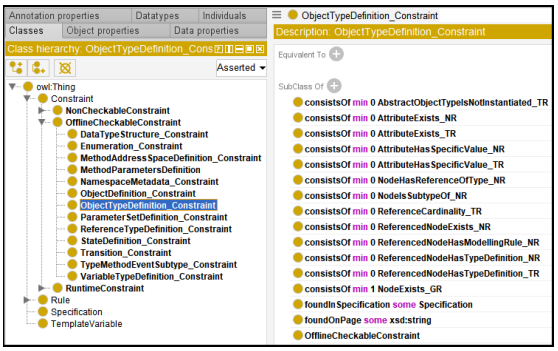


Fig. 14. Snippet of the Rule Taxonomy, which is an ontology describing the types of constraints, types of rules as well as the mappings from constraint types to rules.

straint to a set of possible rules via an OWL class definition. If a table is found in a companion specification, the first step is to look up if the table represents an OfflineChechableConstraint. If so, the information from the table is parsed (using techniques described in Sect. 5.1), and the applicable rule (based on the constraint to rule mappings provided) is retrieved from the set. Afterward, the rule template is filled with the retrieved information from the table. This procedure is repeated untill all necessary rules are created for the particular constraint.

As an example, the ObjectTypeDefinition constraint is related to a set of 12 rules (Fig. 14). The needed information, e.g., that a node has a certain attribute value or the reference is of a certain type, is extracted from the tables and used to fill in the variables in the related SPARQL rule templates. These completed rule templates can now be executed on the SPARQL endpoint, which has loaded the OPC UA information model. Violations of the rules are reported as error messages.

Fig. 15 shows an example of this procedure. In the upper part of Fig. 15, a snippet of the ObjectTypeDefiniton table is shown for the WidgetType, as defined in the Machinery companion specification. It shows the part of the table where it is stated that the WidgetType is not abstract. This is done by setting the value of the IsAbstract attribute to False.

Among others, the ObjectTypeDefinition constraint is related to the *AttributeHasSpecificVale_NR* rule template, which checks if a certain attribute has a specific value. In this case, it checks if the attribute IsAbstract has the value “False”. The template of the *AttributeHasSpecificVale_NR* rule is depicted in the lower part of Fig. 15. The template variables are now assigned with the values from the table to make the SPARQL rule executable. The SPARQL rule will lead to the error message if the WidgetType is defined as abstract in the OPC UA information model.

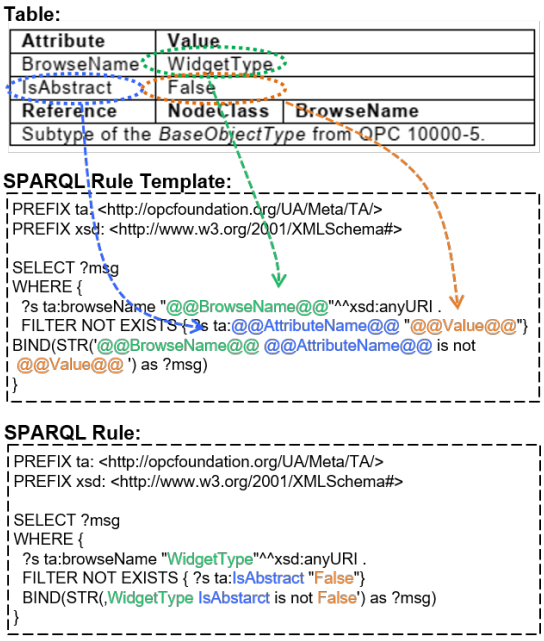


Fig. 15. Procedure of generating SPARQL rules from tables.

6.2. *Generating rules from text with Human-in-the-loop*

While the types of constraints expressed in certain table types can be mapped to suitable rule templates thus enabling the automated generation of rules from tabular data (Sect. 6.1), generating rules from text is a much harder process. As currently there is insufficient training data to train automatic classifiers for this task, we propose a *Human-in-the-loop* approach to map textual constraints to rules. The approach involves three tasks in order to (1) verify the correctness of textual constraints to remove noise introduced by the automated extraction modules; (2) classify constraints into categories linked to rules; (3) validate the resulting rules. As such, these tasks also serve as an approach to implement *Stage 4: Constraint and Rule Validation* of the overall approach presented in this paper (see Sect. 3, Fig. 4).

We rely on Human Computation (HC) techniques for the Human-in-the-loop approach. HC implies outsourcing specific tasks of a system, which cannot be fully automated, to human participants and leveraging the human processing power to solve those tasks. HC has been already successfully applied in a variety

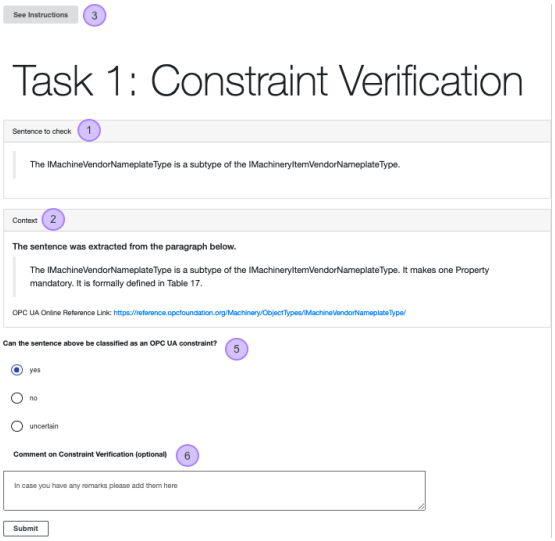


Fig. 16. HC task designed for constraint verification.

of domains for verification tasks [6] and we believe it is an important building block of a semi-automated method to support the steps needed for the derivation of formal rules from PDF-based specifications. Next, the designed HC approach is explained and later in Sect. 7.5 a discussion of the results of an expert-sourcing validation campaign are presented.

Human Computation task design. The HC solution consists of three tasks. In a first task, as shown in Fig. 16, the evaluators are asked to verify whether an extracted sentence (1 in Fig. 16) represents a constraint. To provide enough background information for the decision the task includes additional context (2) such as the paragraph from which the sentence is extracted as well as relevant materials from the OPC UA Online Reference tool (https://reference.opcfoundation.org). To further support the experts, Instructions (3) are available in which nomenclature is explained and detailed task explanations with examples are provided. To allow for easy aggregation of responses provided by multiple experts, the task is designed as closed-ended (4). In case the evaluator is unsure whether the sentence refers to a constraint they can select the *Uncertain* option. A comment field (5) is also available for remarks that the experts would like to share.

When the evaluator selects that the sentence is a constraint, Task 1b, shown in Fig. 17, is displayed. The goal is to classify the constraint (1 in Fig. 17) into the class to which it refers to (2). For instance, the constraint can be related to an *Object TypeDefinition* or an *Enumeration*. To support the generation of rules

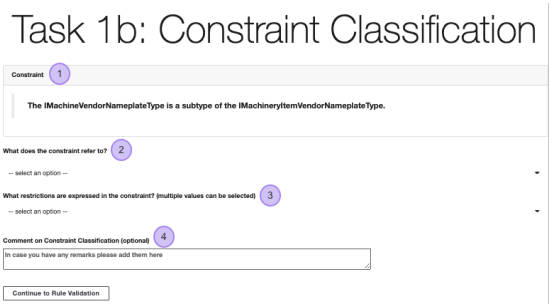


Fig. 17. HC task designed for constraint classification.

the user is also asked what restrictions (e.g., cardinality)(3) are defined in the constraint. As with a previous task, a comment field (4) is available as well.

Once the constraint is classified, Task 2, shown in Fig. 18, follows. Here the participant's role is to validate whether the shown rule (set) (2 in Fig. 18) correctly and completely reflects the constraint expressed in the extracted sentence (1). The form in which the rules are shown is a variable and should be decided based on the evaluation population. In the shown example in Fig. 18 a formal natural language was selected, however, a rule language like SPARQL could be used as well. As with Task 1 Instructions (3) are available and a *Yes/No/Uncertain* answer (4) is expected. In case the person does not agree with the proposed rule (set) they are asked to select whether the rule (set) is *incorrect, incomplete, incorrect and incomplete* or *superfluous*. A comment filed (5) is included in case the evaluator wants to share some further insights regarding the rule.

7. Evaluation

Evaluation primarily focused on assessing the performance and quality of individual elements of the proposed approach including: (i) the rule taxonomy and rules defined in Stage 1 (Sect. 7.1 and 7.2); (ii) the methods for extracting constraints from tables and text (Sect. 7.3 and 7.4) and (iii) the feasibility of HC tasks for constraint and rule validation as envisioned in Stage 4 by performing an evaluation campaign with OPC-UA experts from the Machinery domain (Sect. 7.5).

7.1. Evaluation of rule taxonomy

As described in Section 5.1, the rule taxonomy was created based on the analysis of the constraints re-

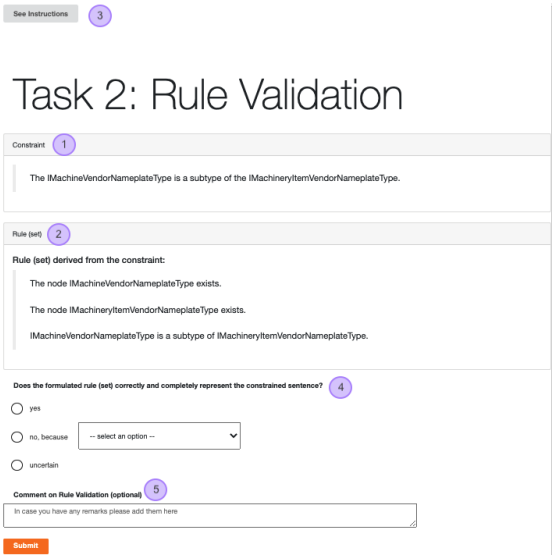


Fig. 18. HC task designed for rule validation.

lated to a type of *table* in a companion specification. Therefore, the question arose whether and to what extent this rule taxonomy would enable expressing constraints present in the *textual* part of specifications.

To validate the coverage of the rule taxonomy, we manually identified the textual constraints in the Machinery companion specification. Afterward, the textual constraints were analyzed to see if the already identified rules could also be used to express these textual constraints. We found that the rule taxonomy can already cover the vast majority of these textual constraints. Only two global rules had to be added, which are concerned with instantiating variables. This seems reasonable as this kind of information can hardly be expressed in a table and needs additional context.

We conclude that, although the rule taxonomy was created based on constraints expressed in tables, it is sufficiently complete to also express constraints described in the textual part of the specifications. Having said that, the rule taxonomy is by no means a final collection of rules, but a core set of rules that can (and should) be extended as required.

7.2. Evaluation of SPARQL rules

Another important question to clarify was whether the SPARQL rules created as part of the rule taxonomy could be applied successfully on a concrete information model.

To that end, in conformance with the Robotics companion specification, an information model was created, including instance nodes. These instance nodes are needed for evaluating some of the rules. Figure 19 illustrates the applied process. As SPARQL cannot be executed directly on the OPC UA NodeSet file, a transformation from an OPC UA NodeSet to an OWL ontology was required. More details about the OPC UA to OWL transformation can be found in [7]. Afterward, the resulting ontology was loaded into a triplestore (Apache Jena Fuseki) to execute 14 specified SPARQL rules. These rules were evaluated by checking the rules' outputs against the expected results. As the information model provides only a limited amount of possible ObjectTypes, Objects, Variables, etc., only a subset of the defined rules was implemented in SPARQL. The other rules were only formulated in SNF. However, the approach's feasibility and the correctness of the implemented SPARQL rules are demonstrated.

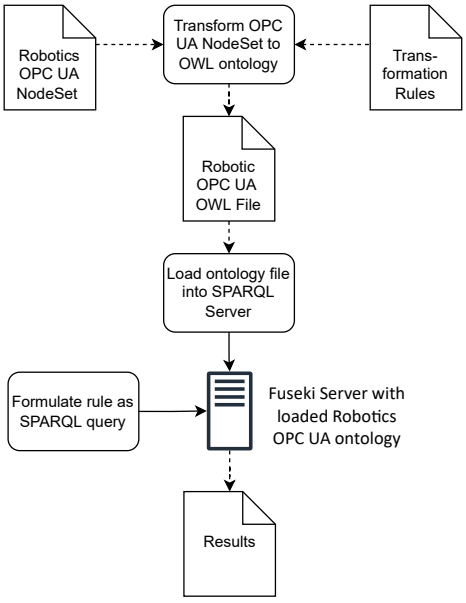


Fig. 19. Process for evaluating SPARQL rules.

7.3. Evaluation of constraint extraction from tables

To assess the performance of the constraint extraction from tables (described in Sect. 5.1), we compute: (i) True Positives (TP) which occurs when a table of a certain type (e.g., *Enumeration*) is extracted and classified correctly as its type; (ii) False Positives (FP) for tables of a specific type (e.g., *Reference TypeDefinition*) that are extracted and classified as a table of a differ-

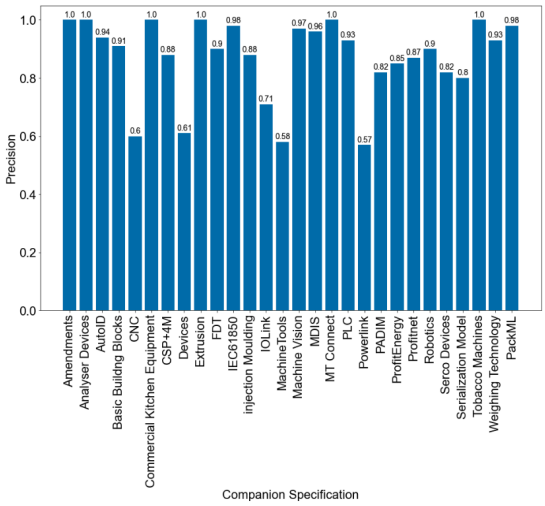


Fig. 20. Precision of table extraction per companion specification.

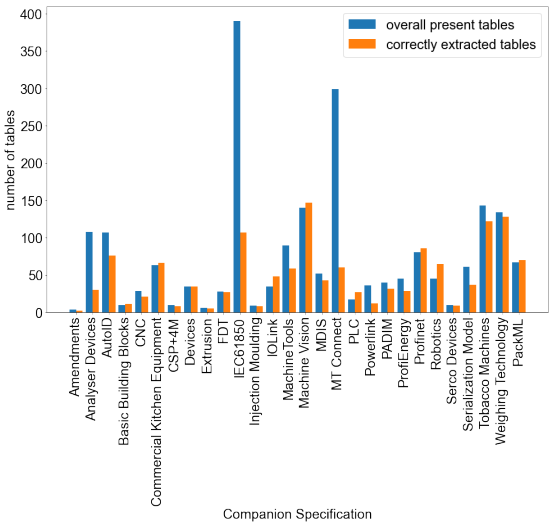


Fig. 21. Comparison between the total number of tables and the correctly extracted tables for each companion specification.

ent, incorrect type (e.g., *Enumeration*); (iii) Precision as formalized in Eq. (1). Due to the nature of the input sources and the implemented solution False Negative and True Negative situations arise very rarely and are not considered the purpose of this task.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Since the constraints are formulated using the values obtained from the tables, it can be stated that once a table is extracted correctly, the constraints associated with it will also be properly extracted. In the majority of cases, the constraint extraction is directly proportional to the correctness of the table extraction as constraints are derived from the table values. Therefore, the main focus of the evaluation is on table extraction and categorization, also as a proxy for the correctness of constraint extraction.

7.3.1. Table extraction and categorization

Fig. 20 shows the precision of extraction from each of the 28 selected companion specifications, thereby resulting in an average precision of 0.87 and standard deviation of 0.134. As mentioned in Step 2 some customization of the algorithm is needed for documents that do not follow the guidance and template provided by OPC UA exactly. The more inconsistencies in the layout the lower the precision scores are. In section 7.3.2 a more detailed error analysis is provided.

Fig. 21 illustrates the Recall analysis. There are a total of 1998 target tables (tables of the types selected for the extraction) in the considered companion specification documents. With the used algorithms 1357 True Positive table extractions were possible thereby also leading to proper constraint extractions for those tables. This results in a recall of approx. 68% for the automatic extraction of tables and constraints. However, True Positive extractions are in some cases higher in numbers than the actual number of tables present. This can occur when a large table split on several pages is perceived as multiple tables in the extraction algorithm. Another case where a single table is extracted as multiple smaller tables is when there is irregular spacing or positioning of columns in the table. In some other scenarios a single table is extracted twice because it fits to the conditions of more than one table type. Such special scenarios that lead to some challenges or complexities in extraction are discussed in detail in the following section.

7.3.2. Error analysis

We hereby discuss the difficulties in the extraction of tables (and constraints) from PDF-documents. Overall six different error types were identified which led to challenges or imperfections in extraction. Fig. 22 shows an overview of the occurrence of those errors in the processing of each companion specification.

- *Error 1: Error due to fixed line-scale value.* The line-scale value is a feature in Camelot and plays a key role in the proper extraction of data in tabular format from a PDF document. It has the purpose of a line-size scaling factor. After testing dif-

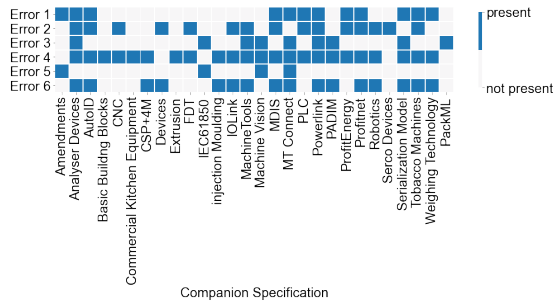


Fig. 22. Error types encountered during the processing of each companion specification.

ferent values on the companion specification, we empirically determined a line-scale value of 80. Nevertheless, the fixed value becomes an issue for tables on which the algorithm has not been trained and which are not defined in the OPC UA specification templates.

- *Error 2: Error due to intersection of words in table-type-specific vocabulary lists.* In Step 2 from Sect. 5.1.3 the usage of table-type-specific vocabulary lists for table categorization is discussed. While in most cases the keywords inside the table are specific to a single table type, there are few cases where the same identification words are included in several table types. This causes some tables to be classified incorrectly (False Positive).
- *Error 3: Error due to page-breaks*. Page-breaks are observed in many companion specifications documents and become an issue for the extraction when a table is split into multiple pages. In cases where the table continuation does not include headings, it does not get extracted and a loss of constraints can be observed. On the other hand, when the split table includes a heading on each page, this can result in the extraction of several tables instead of one, which is not an issue for the correct extraction of constraints, however, the performance metrics get affected.
- *Error 4: Error due to improper alignment of columns and table structures.* Improper alignment of columns inside the tables also leads to incorrect or missing constraint extractions even when the table extraction is correct. Such problems are mainly observed in *NamespaceMetadata* and *NamespaceURI* tables.
- *Error 5: Typographical errors or unexpected special characters.* Other factors that can negatively influence table extractions are typographical errors or misprints. Newline characters in unexpected positions in tables or incorrect column names also lead to errors in the extraction of constraints.
- *Error 6: Error due to some unique structure of a table type that is not covered within the algorithm.* In a few cases no particular error in the algorithm was found causing the incorrect or missing extractions. A possible explanation could be the nonconformity of some tables to the standard companion specification template structures.

To conclude, the approach for the automatic extraction of tables and constraints from OPC UA PDF specification documents achieves a Recall of 68% and an average Precision of 87%. Customizing the algorithm to the unique structures in different documents could result in better performance, however, it would also reduce the dynamic nature of the algorithm and would not improve extractions from future documents if they use incompatible structures. Moreover, the currently used software Camelot has some limitations as discussed above for extracting information from tables in textual documents. This software should be improved in order to improve the overall performance of information extraction from tables. On the other hand, the structuring of tables in the OPC UA documents and the correct usage of table templates provided by the OPC Foundation while creating the companion specification documents could further contribute to increased quality of the information extraction algorithms.

7.4. Evaluation of constraint extraction from text

For the evaluation of the constraint extraction from text, one further companion specification document (*Machinery*) was selected, for which a gold standard was manually created. The data set was retrieved semi-automatically by using an automatic extraction method and the expertise of the authors for corrections and extensions of the extracted data. In total the gold standard data set includes 44 constraints and 229 not-constraints.

7.4.1. Machine learning based approach

The machine learning based approach was first tested on the 20% of *PackML* gold-standard which was not used for the training of the algorithms. The 8 supervised machine-learning-based models (Nearest Neighbors, Linear SVM, RBF SVM, SGD, Decision Tree, Random Forest, Neural Network, Ad-

aBoost) prevously trained with 80% of the *PackML* gold-standard data are evaluated by testing the performance of each classifier in terms of precision, recall, F1 score, and accuracy. The results are shown in Table 2. Due to the imbalance of the data weighted average is used for the evaluation metrics. Results indicate that Neural Network (Multi-layer Perceptron) outperforms the rest of the models and produces the best results with a recall of 0.95.

Table 2
PackML test results in weighted average.

Classifier	Precision	Recall	F1-score	Accuracy
Nearest Neighbors	0.93	0.93	0.93	0.93
Linear SVM	0.76	0.87	0.81	0.87
RBF SVM	0.93	0.92	0.93	0.92
SGD	0.90	0.90	0.90	0.90
Decision Tree	0.92	0.92	0.92	0.92
Random Forest	0.76	0.87	0.81	0.87
Neural Network	**0.95**	**0.95**	**0.94**	**0.95**
AdaBoost	0.90	0.90	0.90	0.90

Recall indicates how many relevant items are retrieved. In other words, how many constraints are detected correctly by the algorithms. It can be formalized in terms of True Positive and False Negative as expressed in Eq. (2).

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Among the evaluation metrics shown in Table 2, we chose to focus on recall due to the unbalanced data set consisting of only a small number of constraints, each of which is highly valuable for the extraction. Additionally, we consider False Positives as less important than False Negatives because for the validation approach, it is important that no constraints are missed.

To further asses the constraint extraction approach, the *PackML* trained models are tested on the *Machinery* specification document as well. The results are shown in Table 3 and indicate that the best performance is achieved using Linear SVM and Random Forest. The recall values reach 86% which is sufficiently good for such an experiment (training with one document and testing on another one).

In the light of these experiments, we conclude that, when the training and test data are similar (from the same set), Neural Networks work very well. However, Linear SVM and Random Forest achieve more consistent results, even if there is a considerable difference between the training and test set. Therefore, we can interpret that, Neural Networks might have the problem of overfitting while Linear SVM and Random Forest can be generalized to new data more easily.

Table 3
Machinery test results on the entire data.

Classifier	Precision	Recall	F1-score	Accuracy
Nearest Neighbors	0.79	0.83	0.81	0.83
Linear SVM	**0.74**	**0.86**	**0.80**	**0.86**
RBF SVM	0.80	0.77	0.78	0.77
SGD	0.79	0.76	0.77	0.77
Decision Tree	0.80	0.81	0.80	0.81
Random Forest	**0.74**	**0.86**	**0.80**	**0.86**
Neural Network	0.79	0.83	0.81	0.83
AdaBoost	0.79	0.79	0.79	0.79

7.4.2. Lexical pattern-based approach

The evaluation of the lexical pattern-based approach consists in comparing the performance of the linguistic pattern matching to a manually curated gold standard. Using the *Machinery* companion specification as an input document, the linguistic patterns defined in Sect. 5.2.2 are applied. As a result all sentences are extracted which fit the constraint patterns, which leads to (i) 26 True Positives (extracted sentences labeled as constraints in the gold standard); (ii) 131 False Positives (extracted sentences labeled as not-constraints in the gold standard); (iii) 18 False Negatives (sentences that did not get extracted but are labeled as constraints in the gold standard). Using Eq. 1 and Eq. 2 we calculate a precision score of approximately 0.17 and a recall of 0.59.

In comparison to the ML-based approach evaluated in the previous section, the constraints extracted based on linguistic patterns do not achieve good results. Since the OPC UA data is large, natural-language-based and diverse, OPC UA constraints can be expressed in different structures. Nevertheless, formulating a higher number of linguistic patterns would results in a higher number of False Positives. To conclude, the ML-based extraction proved to be more flexible and thus outperformed the lexical pattern-based approach. However, training an ML-model that offers high accuracy predictions is a more complex and expensive approach. Therefore, based on the requirements and desired results one should make a trade-off between high accuracy and low resource extraction.

7.5. Feasibility evaluation with domain experts

To evaluate the usefulness and clarity of the designed HC Tasks for the verification of automatically extracted constraints and generated rules (Stage 4 of our approach) an expert-sourcing validation campaign was conducted in the concrete context of the OPC UA Machinery specification.

7.5.1. Expert validation campaign of the Machinery specification

From the gold-standard *Machinery* data created by the authors we selected: (i) 60 sentences, 70% of which were considered by authors to represent modeling constraints; and (ii) the rules that could be formulated for the constraints expressed by these sentences. Six OPC UA experts involved in the creation of the chosen companion specification document (*Machinery*) were asked to perform the validation tasks. The rules in Task 2 (Fig. 18) were shown in a structured natural language, rather than SPARQL, so that they are understood also by experts not familiar with the SPARQL.

The validation campaign was run on Amazon Mechanical Turk (https://www.mturk.com) - a crowdsourcing platform that offers the possibility to implement HC processes. The platform allows for easy results aggregation and makes it possible to avoid sequence bias by showing the tasks in a random order to each of the evaluators.

For the campaign, the extracted sentences were split into two batches of 30 sentences (and their corresponding rules) and 3 experts were assigned to each batch. The evaluators completed the tasks within two weeks and each of them submitted 25-30 responses. This setup allowed for the collection of 3 responses on average per sentence and provided 168 judgments in total.

Since there was no variety in the classes to which the selected sentences referred to (e.g. *Object TypeDefinition*), Task 1b (Constraint Classification) was not included in the validation campaign. In the next subsections the results of both HC tasks are analyzed to identify areas for improvement of the approach.

7.5.2. Constraint extraction from text (Task 1)

Fig. 23 shows an overview of the results of the constraint classification task. Based on the experts judgments a majority vote could be calculated for almost 90% of the sentences, however, a complete agreement among the experts was not achieved on over 50% of the data items as it is seen in Fig. 23a. Since for each task a variable number of evaluation were collected, Krippendorff's alpha coefficient was used to calculate the inter-rater agreement among the experts. The alpha score of 0.19 indicates very low agreement and shows that the identification of constraints is a difficult task.

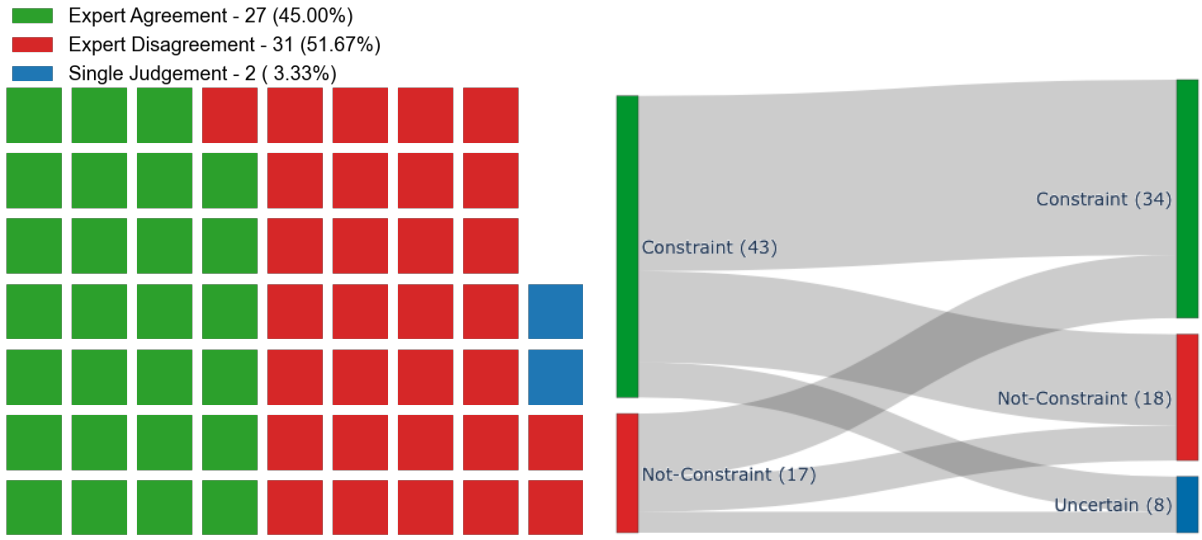
In Fig. 23b a comparison between the classification proposed by the authors and the classification resulting from the experts validations is shown. Sentences, for which a majority vote could not be established, have been added to the category "Uncertain". When comparing the majority vote of the experts against the authors' classification there is an overlap on less than 60% of the sentences. These results add to the statement that identifying constraints in OPC UA documents is a complex problem and textual definitions are open to interpretation. Therefore identifying textual constraints and defining formal rules to verify them is an important task for ensuring the conformity of OPC UA Nodeset files to the OPC UA base/companion specifications, which can be enabled by the proposed validation approach.

7.5.3. Rule validation (Task 2)

In this section, the ability of the proposed rules to completely and correctly capture textual constraints is examined. Because of the low agreement among the experts on the constraint verification task, rules were not validated by all participants and a majority voting was not feasible for all judgments. Moreover 45% of the rule validations were evaluated only by a single expert and for 67% of the rules defects were identified by one evaluator only. Since an inter-rater agreement score is not meaningful considering the gathered data, we compute for each expert to what percentage they agree with the proposed rules.

The expert scores vary from 33% to 100% and result in an average of 68% acceptance of the proposed rules. The frequency of found defects for the 18 rules, for which a defect was selected by at least one evaluator, can be seen in Fig. 24. While no rule set was found to be superfluous, 8 rules were judged as incomplete and 6 as incorrect. In 4 of the rule sets both defects (incompleteness and incorrectness) were found, where each have been selected by at least one expert. In the comments added by the experts for this task, exact mistakes and proposals for improvements were identified.

To conclude the campaign results show that (i) the tasks of constraint identification is difficult even for experts and (ii) the Human-in-the-loop approach enables the successful identification and classification of invalid constraints and rules which is essential for improving the automatic algorithms.



(a) Overview of the agreement among experts on the classification of the input sentences in absolute numbers and in percentages.

(b) Overview of the classification of the input sentences by the authors and by the experts in absolute numbers.

Fig. 23. Results of the constraint classification task in terms of (a) agreement among experts on the classifications, and (b) alignment between the classification proposed by the authors (left) and by the experts' majority vote (right).

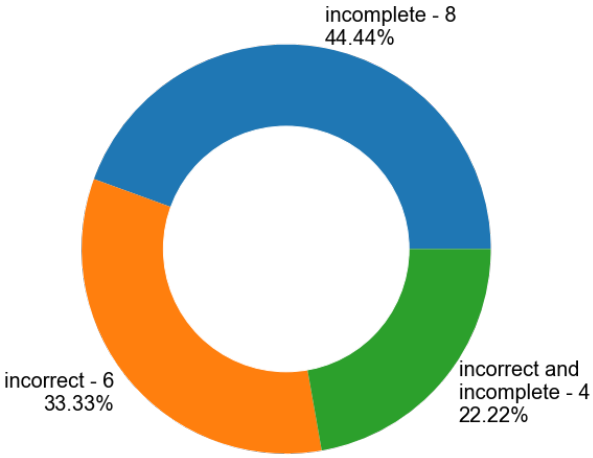


Fig. 24. Frequency of the defects identified by the experts for rules that were marked as invalid in absolute numbers and in percentages.

8. Related work

Application of Semantic Web in industrial settings. Semantic Web technologies have been extensively applied to support various industrial applications in the last decade ranging from combining sensor networks with the Web [8], augmenting products with semantic descriptions [9], or enabling smart city infrastructures [10]. The application of these technologies has been taking place in a variety of (mission-critical) domains, such as manufacturing [11, 12], electric grids [13], or buildings [14] and addressed tasks both during the *engineering* (e.g., engineering model integration, digital twin model consistency management) and *operation&maintenance* (monitoring and anomaly detection, optimization and reconfiguration) phases of cyber-physical systems in such domains [15]. Furthermore, an important task is the verification of technical information objects (e.g., ontologies, semantic data sets) in terms of complying to constraints specified in languages such as SHACL, SWRL or SPARQL.

Against this backdrop, in this paper we investigate a *novel setting* for the application of Semantic Web technologies in the context of industrial standards, in particular as a solution option for automating validation processes of information models based on standards. Our work is in line with the current paradigm shift in such standards towards automatic standard validations. In particular, the OPC Foundation[1] has started an OPC UA Semantic Validation Working Group in July, 2019 with OPC UA experts and Semantic experts as members with the primary goal to provide the foundation to create valid, consistent NodeSets. As a part of this goal, the OPC UA experts identified the constraints in the OPC UA core specifications manually by reading the specification documents and understanding them. Semantic experts are working towards formal repre-

[1] https://opcfoundation.org/

sentation of the constraints as rules in order to validate them on the nodesets. However, we observed that this process of manually collecting the constraints and formalizing them is a tedious and time consuming activity. Therefore, in this study we worked towards a (semi-) automatic approach to validate the OPC UA information models and to the best of our knowledge we are the first to work on such an approach.

Semantic information extraction. A core part of our work is proposing methods to (semi-)automatically extract validation rules from the standard specifications, thus being related to the large body of work on information extraction (IE) in the context of the Semantic Web recently reviewed in [16]. This review identified that IE systems have been applied in numerous domains and have been developed for different types of sources - structured text, semi-structured text, unstructured text, images and tables or mixed multi-modal forms [17]. While many studies focus on text-based sources, there are other modalities in textual inputs like tables and schema used for information extraction as well. In [18] the authors aim to make a classification of information, structured in tables, and define the relations between cell information. In a few studies, multi modalities such as either graph and text together or a combination of table and text are used for information extraction. In those studies, the main goal is to extract semantic and well-formed machine-readable data from huge online or offline data [19].

An important distinction also refers to the extracted information which can be *entities*, *concepts* or *n-ary relations* (e.g., triples). As the focus of our work is on extracting rules, which can be seen as complex n-ary relations, we identified the following examples of works focused on extracting n-ary relations. In [20], semantic relations between concepts are extracted from medical semi-structured text based on belief states. The use of IE for the generation of triples from documents is discussed in [21]. A more complex problem is acquiring rules, which often can contain several triples. Rules have been acquired from web content [22], [23] or textual content, e.g., in the medical domain [24].

There are also approaches where the output of IE results in a complex knowledge structure such as a knowledge graph. For example, in [25] the main goal is the creation of a legal knowledge graph based on the Austrian platform RIS [2]. For the population of the knowledge graph legal entities (such as legal rules, references, contributors, etc.) are extracted from structured resources and from text. For the extraction both machine and deep learning techniques are used as well as a rule-based approach.

Related to this area, our work focuses on technical documents, explores both textual and tabular modalities and aims to extract rules (e.g., complex n-ary relations). To the best of our knowledge, this is the first effort to perform this complex information extraction task in the context of OPC UA specifications.

9. Conclusions and future work

The use of industrial standards is core to industrial engineering across application domains to introduce cannons for digital communication, data exchange, etc. that ensure interoperability across domain stakeholders. However, a commonly taken approach is that standard specifications are provided in non/semi-structured documents which makes it difficult to automate compliance checks of information artifacts relying on them, as demonstrated by the concrete use case of OPC UA. Therefore, a paradigm shift in the area of industrial standards is needed towards more machine-processable, explicitly represented standard specifications, additionally to textual specifications, so that the validation of information artifacts can be automated.

To support such a paradigm shift towards automated semantic validation, we proposed a high-level approach for representing and (semi-)automatically extracting formal rules from unstructured standard documents and then instantiated this approach in the case of OPC UA and reached the following conclusions.

In terms of *RQ1*, we found that it was feasible to represent modeling guidelines from the specifications as formal rules. Furthermore, these rules could be organised in a taxonomy represented by means of an OWL ontology, thus benefiting from the capabilities of explicit semantic representation of the Semantic Web technologies. We used SPARQL to provide rule-templates that can be instantiated into concrete rules and also tested a subset of these rules on the Robotics companion specification. We found that the current taxonomy of rules could express constrains available both in tabular and textual format, and can be easily extended if needed.

Related to *RQ2*, as a first step towards automating the derivation of rules from specifications, in the context of OPC UA, both tabular and textual information

[2] https://www.ris.bka.gv.at

can be processed to identify modeling constraints with a high precision (P=87%) for tables and high performance (F1 up to 94%). In terms of methods, although they require training data, Machine learning methods lead to more promising results that approaches relying on lexico-syntactic patterns, which are hampered by the high variety in the style of the text across domains as well as specification creators. Furthermore, the evaluation campaign with the Machinery experts showed that the way constraints are expressed in text is often highly ambiguous and leads to low agreement even across experts. This further motivates the needs for methods as presented in this paper where a combination of techniques are used to identify, verify and explicitly define such constraints to reduce the ambiguity of the textual specifications.

Related to *RQ3*, generating rules based on automatically identified modeling constraints in the specifications could be solved in two ways in the OPC UA context. Firstly, fully-automated generation was possible based on information available in tables as the table types are already indicators of types of rules that are expressed; pre-requisites were a clear understanding of constraint and rule types, a mapping between these as well as the use of rule templates as defined in the rule taxonomy. Second, to connect textual constraints to rules a Human-in-the-loop approach was proposed, given the lack of training data for automating this task.

Future work includes:

- *Improve rule taxonomy.* It was found that both the taxonomy and rule template definitions heavily depend on the language used to express the rules and, in the case of SPARQL, also on the specific OPC UA NodeSet to OWL Ontology transformation. Thus, if the OPC UA NodeSet to OWL Ontology transformation changes, the SPARQL rules may also require adaptations. Also, a two step approach for rules, separating the navigation to the focus node from the actual rule application would help to reduce the amount of needed rules. Therefore, we will investigate alternative approaches for representing the rule taxonomy in order to reduce the complexity of the taxonomy and provide simpler rule templates.
- *Explore multi-modal constraint extraction.* While constraints could be identified both in tables and text, we observed an overlap of content across the information expressed in these modalities. Therefore, we wish to advance the constraint extraction methods by leveraging this overlap as a signal of which information is a constraint (e.g., strong signal is when mentioned in both modalities).
- *Provide rule explanations through provenance information.* The multi-model extraction will lead towards a more complex rule extraction workflow. Therefore, providing traces on how a rule was derived and what it means (e.g., based on the originating part of a specification) is important for OPC UA Specification authors and end-users of these specifications that want to validate their files. Ideally, if a rule identifies an issue in a file, it is important to provide rule provenance information in terms of tables, text snippets that lead to its derivation as well as the version of specification it was derived from as a first immediate documentation for the end-user to debug/improve his files. To achieve such features, we will enable the auditability of the rule extraction process by providing a provenance tracing and representation middleware.
- *Extend to other OPC UA specifications* beyond those considered as part of current work.
- *Apply methods to other industrial standards* While the points above deepen the work in the context of the OPC UA standard, an orthogonal effort will be investigating to what extent the methods and algorithms developed for OPC UA are applicable in the context of other industrial standards.

With the future work mentioned above, this work has the potential to bring a major contribution towards automatic, semantic validation within the widely used OPC UA standard, as well as other industrial standards, thus leading to improved interoperability in industrial engineering settings and exploring the capabilities of Semantic Web technologies for this novel and important problem.

Acknowledgements

We thank the six OPC UA Machinery working group experts (Sebastian Friedl, Götz Görisch, Tonja Heinemann, Timo Helfrich, Heiko Herden and Wolfgang Mahnke) for taking part in the feasibility evaluation campaign. We also thank the experts from the OPC UA Semantic Validation working group, who helped us pilot the Human Computation Tasks and provided feedback on possible improvements. Parts of this work related to Human Computation were funded through the FWF HOnEst project (V 745-N).

References

[1] Y. Liao, L.F.P. Ramos, M. Saturno, F. Deschamps, E. de Freitas Rocha Loures and A.L. Szejka, The Role of Interoperability in The Fourth Industrial Revolution Era, *IFAC-PapersOnLine* **50**(1) (2017), 12434–12439, 20th IFAC World Congress. doi:https://doi.org/10.1016/j.ifacol.2017.08.1248. https://www.sciencedirect.com/science/article/pii/S2405896317317615.

[2] H. da Rocha, A. Espirito-Santo and R. Abrishambaf, Semantic Interoperability in the Industry 4.0 Using the IEEE 1451 Standard, in: *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 5243–5248. doi:10.1109/IECON43393.2020.9254274.

[3] B. Wollenberg, J. Britton, E. Dobrowolski, R. Podmore, J. Resek, J. Scheidt, J. Russell, T. Saxton and C. Ivanov, A Brief History: The Common Information Model [In My View], *IEEE Power and Energy Magazine* **14**(1) (2016), 128–126. doi:10.1109/MPE.2015.2481787.

[4] W. Mahnke and S.-H. Leitner, OPC Unified Architecture - The future standard for communication and information modeling in automation, *ABB Review* **3** (2009), 2009.

[5] W. Mahnke, S.-H. Leitner and M. Damm, *OPC Unified Architecture*, Springer, Berlin, 2009. ISBN 978-3-540-68898-3. doi:10.1007/978-3-540-68899-0.

[6] M. Sabou, L. Aroyo, K. Bontcheva, A. Bozzon and R.K. Qarout, Semantic Web and Human Computation: The status of an emerging field, *Semantic Web* **9**(3) (2018), 291–302.

[7] R. Schiekofer, S. Grimm, M. Milicic Brandt and M. Weyrich, A formal mapping between OPC UA and the Semantic Web, 2019, pp. 33–40. doi:10.1109/INDIN41052.2019.8972102.

[8] O. Corcho and R. García-Castro, Five challenges for the Semantic Sensor Web, *Semantic Web* **1**(1–2) (2010), 121–125. doi:10.3233/SW-2010-0005.

[9] M. Sabou, Smart objects: Challenges for Semantic Web research, *Semantic Web* **1**(1–2) (2010), 127–130. doi:10.3233/SW-2010-0011.

[10] I. Celino and S. Kotoulas, Smart Cities [Guest editors' introduction], *IEEE Internet Computing* **17**(6) (2013), 8–11. doi:10.1109/MIC.2013.117.

[11] S. Biffl and M. Sabou, *Semantic Web Technologies for Intelligent Engineering Applications*, Springer, 2016, pp. 1–405. doi:10.1007/978-3-319-41490-4.

[12] F. Ocker, C.J.J. Paredis and B. Vogel-Heuser, Applying knowledge bases to make factories smarter, *at - Automatisierungstechnik* **67**(6) (2019), 504–517. doi:10.1515/auto-2018-0138.

[13] S. Howell, Y. Rezgui, J.-L. Hippolyte, B. Jayan and H. Li, Towards the next generation of smart grids: Semantic and holonic multi-agent management of distributed energy resources, *Renewable and Sustainable Energy Reviews* **77** (2017), 193–214. doi:10.1016/j.rser.2017.03.107.

[14] B. Butzin, F. Golatowski and D. Timmermann, A survey on information modeling and ontologies in building automation, in: *Proc. of Annual Conf. of the IEEE Industrial Electronics Society*, IEEE, 2017, pp. 8615–8621. doi:10.1109/IECON.2017.8217514.

[15] M. Sabou, S. Biffl, A. Einfalt, L. Krammer, W. Kastner and F.J. Ekaputra, Semantics for cyber-physical systems: A cross-domain perspective, *Semantic Web* **11**(1) (2020), 115–124.

[16] J.L. Martinez-Rodriguez, A. Hogan and I. Lopez-Arevalo, Information extraction meets the semantic web: a survey, *Semantic Web* **11**(2) (2020), 255–335.

[17] L. Eikvil, Information extraction from world wide web-a survey, Technical Report, Citeseer, 1999.

[18] H. Dong, S. Liu, Z. Fu, S. Han and D. Zhang, Semantic structure extraction for spreadsheet tables with a multi-task learning architecture, in: *Workshop on Document Intelligence at NeurIPS 2019*, 2019.

[19] P. Dolog and W. Nejdl, Challenges and benefits of the semantic web for user modelling, in: *Proceedings of the Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2003) at 12th International World Wide Web Conference, Budapest*, 2003.

[20] T. Goodwin and S.M. Harabagiu, Automatic generation of a qualified medical knowledge graph and its usage for retrieving patient cohorts from electronic medical records, in: *2013 IEEE Seventh International Conference on Semantic Computing*, IEEE, 2013, pp. 363–370.

[21] R. Upadhyay and A. Fujii, Semantic knowledge extraction from research documents, in: *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2016, pp. 439–445.

[22] J. Kang and J.K. Lee, Rule identification from Web pages by the XRML approach, *Decision Support Systems* **41**(1) (2005), 205–227.

[23] S. Schoenmackers, J. Davis, O. Etzioni and D. Weld, Learning first-order horn clauses from web text, in: *Proceedings of the 2010 Conference on Empirical Methods on Natural Language Processing*, 2010, pp. 1088–1098.

[24] A. Boufrida and Z. Boufaida, Rule extraction from scientific texts: Evaluation in the specialty of gynecology, *Journal of King Saud University-Computer and Information Sciences* (2020).

[25] E. Filtz, S. Kirrane and A. Polleres, The linked legal data landscape: linking legal data across different countries, *Artificial Intelligence and Law* **29**(4) (2021), 485–539.

[26] D. Šenkýř, SHACL Shapes Generation from Textual Documents, in: *Workshop on Enterprise and Organizational Modeling and Simulation*, Springer, 2019, pp. 121–130.

[27] J. Dibie-Barthélemy, O. Haemmerlé and E. Salvat, A semantic validation of conceptual graphs, *Knowledge-Based Systems* **19**(7) (2006), 498–510.

[28] O. Lassila, R.R. Swick et al., Resource description framework (RDF) model and syntax specification (1998).

[29] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler and N. Lindström, JSON-LD 1.0, *W3C recommendation* **16** (2014), 41.

[30] D. Beckett and B. McBride, RDF/XML syntax specification (revised), *W3C recommendation* **10**(2.3) (2004).

[31] D.L. McGuinness, F. Van Harmelen et al., OWL web ontology language overview, *W3C recommendation* **10**(10) (2004), 2004.

[32] X. Yao and B. Van Durme, Information extraction over structured data: Question answering with freebase, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 956–966.

Appendix A. Rule Taxonomies

Figures in this appendix depict the created rule taxonomies.

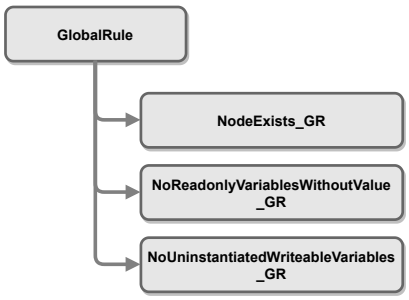


Fig. 25. Identified Global Rules.

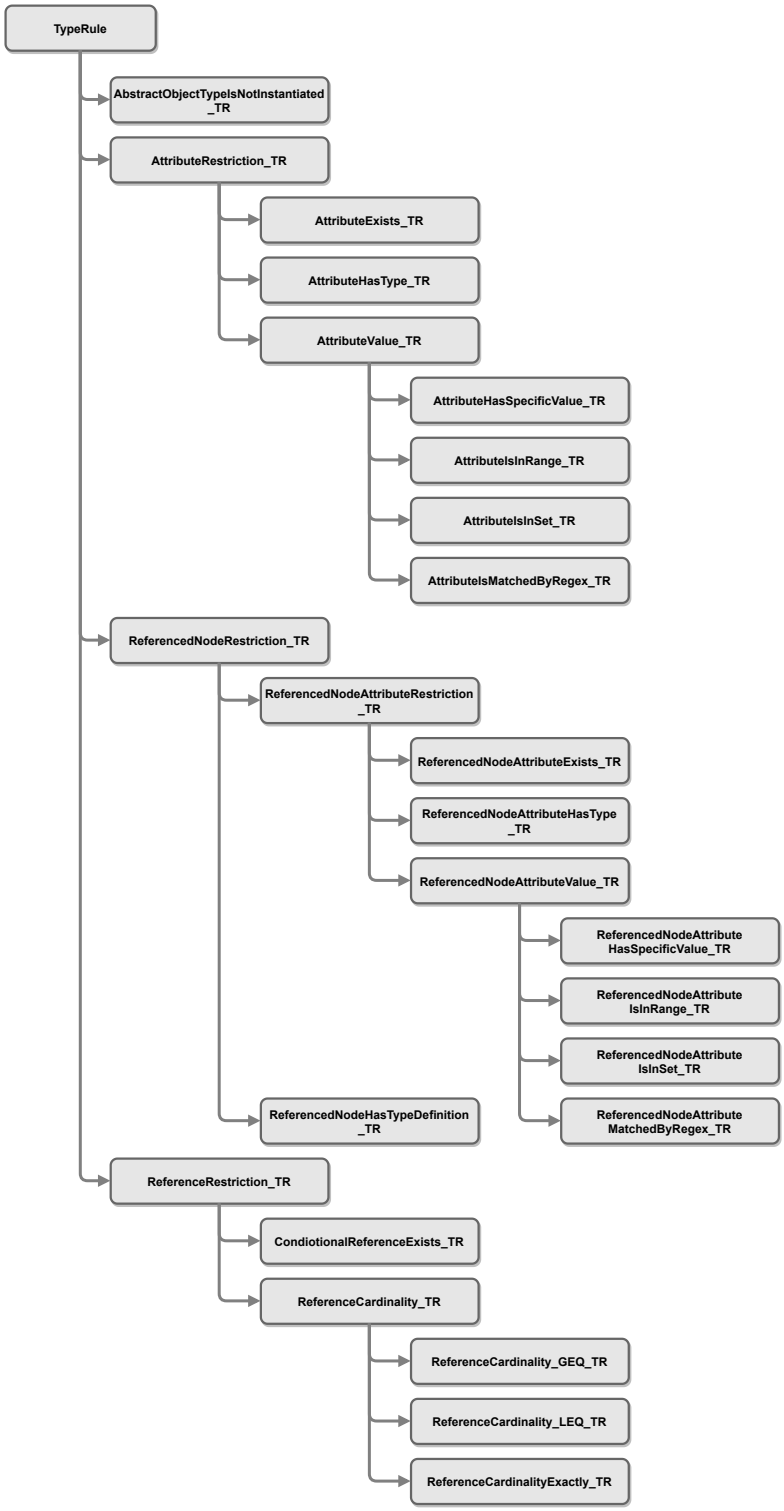


Fig. 26. Identified Type Rules.

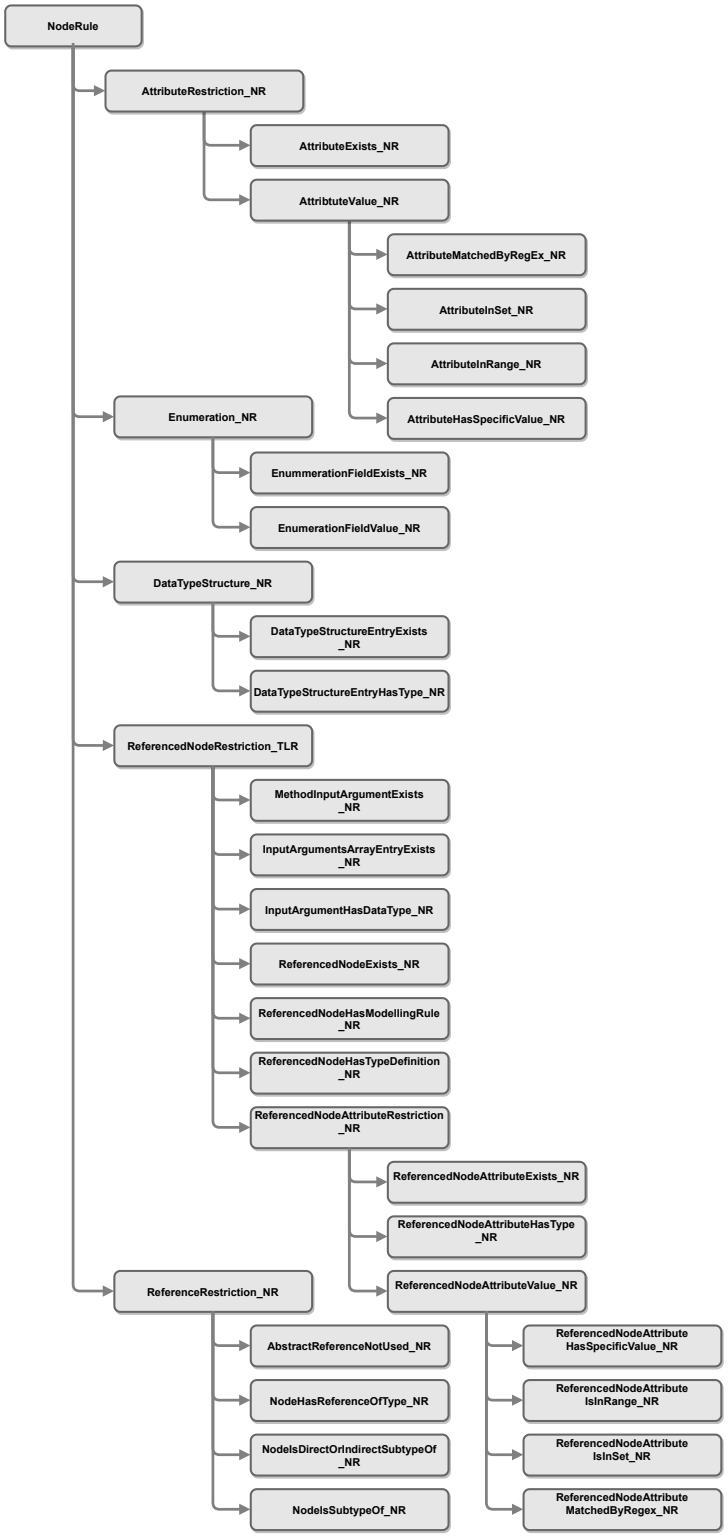


Fig. 27. Identified Node Rules.