BrepPMIOnt: Ontology for Semantic Validation of Computer-aided Design Models

Julian Gebhard[1,2,*], Ratan Bahadur Thapa[2], Yübo Wang[1] and Steffen Staab[2,3]

[1]Mercedes-Benz Group AG, Stuttgart, Germany
[2]Analytic Computing, KI, University of Stuttgart, Stuttgart, Germany
[3]WAIS, University of Southampton, Southampton, United Kingdom

Abstract

Boundary representation (B-Rep) models solids in computer-aided design (CAD). It describes a solid through a graph of vertices, edges, and faces that represents its topology, with each element assigned a geometric realization (a point, a curve, a surface). Product manufacturing information (PMI) annotates topological entities with information for the manufacturer, such as geometric tolerances. CAD models require early validation to determine whether they are correct, combining topology, geometry, annotations, and numerical checks. Existing CAD standards expose this data through format-specific vocabularies and do not provide a reusable semantic vocabulary for validation constraints. Therefore, a constraint that holds for one format does not necessarily hold for another without reimplementing the same reasoning. We present BrepPMIOnt, an OWL 2 EL ontology for a bounded B-Rep and PMI fragment, together with a mapping from CAD instances to RDF and a SHACL-based validation pipeline. OWL 2 EL axioms classify topology–geometry combinations, reducing the need for redundant reasoning in SHACL shapes, while SPARQL extension functions call external CAD libraries for numerical geometric checks. We implement the approach for one standardized B-Rep and PMI format and test it on 11 public, open-source models, showing the approach's feasibility.

Keywords

B-Rep, PMI, SHACL, OWL 2 EL, Knowledge Graphs

CEUR-WS.org/Vol-4313/SPARK-paper2.pdf

1. Introduction

Computer-aided design (CAD) models are artifacts in engineering and manufacturing that replaced handwritten drawings in construction [1]. Boundary representation (B-Rep) is a common technique to model solids in mechanical CAD. A B-Rep model describes a solid through its topology, a graph of vertices, edges, loops, and faces that records their incidence relations, and its geometric realization, assigning a point to each vertex, a curve to each edge, and a surface to each face [2]. CAD systems delineate topological and geometric information because they change independently. A topological entity's incidence relations to other entities can remain fixed while its assigned geometric realization is edited, and the same geometric realization can be assigned to differently bounded topological entities. Topological entities can be annotated with product manufacturing information (PMI), which specifies the information a manufacturer needs to ensure a part is manufactured correctly. This includes dimensions and tolerances, for example a radius of 3.0 ± 0.5 [3]. In recent years, PMI has become available not only as graphical annotations, but also as machine-readable data, called *semantic PMI* [4]. Various standards support B-Rep and semantic PMI, including STEP [5] and JT [6]. Constraints determine whether a CAD model represents a valid solid and whether it meets the quality standards defined by experts. These constraints involve topology, geometry, and PMI, for example a radial tolerance defined for a face must be compatible with its geometry. It would hold for a cylinder but would not be compatible with a plane.

CAD models must be validated against such constraints during construction, as an undetected error at an early stage becomes more costly to correct once the model reaches the prototyping or production

First International Workshop on Spatial Intelligence and Reasoning Enabled by Knowledge Graphs and Foundation Models (SPARK 2026), in ISWC 2026 Workshops Joint Proceedings, October 25–26, 2026, Bari, Italy
*Corresponding author.
julian.gebhard@mercedes-benz.com (J. Gebhard)
0009-0002-4973-549X (J. Gebhard)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-05

phase [7].

How can constraints on B-Rep and PMI data be specified and validated to ensure consistent results across CAD formats?

This is important because inconsistent constraint evaluation across CAD formats requires the same validation logic to be reimplemented for each format. Such redundancy hinders automation and increases the maintenance effort for CAD model validation. Native CAD systems do not provide the capability to infer information or define constraints on the data [8, 9, 10]. An RDF encoding of a B-Rep and PMI file preserves facts; its vocabulary depends on the CAD file format. As a result, checking constraints can only give consistent results if their vocabulary is the same as that of the RDF instance data.

We address this gap by an ontology-based approach, transforming the data to RDF and using SHACL 1.2 for validation. First, we define BrepPMIOnt, an OWL 2 EL ontology that provides a fixed vocabulary for B-Rep topology, geometry, topology–geometry associations, PMI elements, and attributes. OWL 2 EL axioms in the ontology entail classification facts that SHACL 1.2 constraints rely on. Second, we define a mapping from B-Rep+PMI instances to RDF graphs over this vocabulary. The mapping can be implemented across different CAD file formats, while the ontology and constraints remain unchanged. Third, we extend the validation pipeline with a B-Rep literal datatype and SPARQL extension functions that invoke CAD libraries for numerical geometric checks. We implement the pipeline for STEP AP242 models in RDF4J and evaluate it on 11 public models.

2. Preliminaries

Table 1 summarizes the notation introduced in this section. Let IRI be the set of IRIs, BNode the set of blank nodes, and Lit the set of literals, where IRI, BNode, and Lit are pairwise disjoint.

Definition 1. *The set of RDF terms is* $T := \mathsf{IRI} \cup \mathsf{BNode} \cup \mathsf{Lit}$. *An RDF triple has the form*

$$t \in (\mathsf{IRI} \cup \mathsf{BNode}) \times \mathsf{IRI} \times T$$

, where the first component is the subject, the second the predicate, and the third the object. An RDF graph G is a finite set of RDF triples [11, 12, 13].

We assume concept names, object-property names, and datatype-property names are elements of IRI. An OWL 2 EL ontology $\mathcal{O}$ is a finite set of OWL 2 EL axioms over these names [14, 15]. We write $\mathcal{O} \cup D$ for the ontology together with the RDF instance graph D; the notation denotes the input to entailment, query answering, and validation.

We study B-Rep and PMI, denoted B-Rep+PMI. A B-Rep+PMI *signature* names the types and relation symbols. Let $\mathsf{T}_{\mathsf{top}}$, $\mathsf{T}_{\mathsf{geo}}$, $\mathsf{T}_{\mathsf{pmi}}$, and $\mathsf{T}_{\mathsf{dat}}$ be pairwise disjoint finite sets of objects, called *topological types*, *geometric types*, *PMI types*, and *datatypes*, respectively. Let R, A, and P be finite sets of *B-Rep relation symbols*, *attribute symbols*, and *PMI relation symbols*. We assume total functions

$$\begin{aligned} \mathsf{dom}_{\mathsf{R}}, \mathsf{rng}_{\mathsf{R}} &: \mathsf{R} \to \mathsf{T}_{\mathsf{top}} \cup \mathsf{T}_{\mathsf{geo}}, \\ \mathsf{dom}_{\mathsf{A}} &: \mathsf{A} \to \mathsf{T}_{\mathsf{top}} \cup \mathsf{T}_{\mathsf{geo}} \cup \mathsf{T}_{\mathsf{pmi}}, \\ \mathsf{rng}_{\mathsf{A}} &: \mathsf{A} \to \mathsf{T}_{\mathsf{dat}}, \\ \mathsf{dom}_{P} &: P \to \mathsf{T}_{\mathsf{pmi}}, \\ \mathsf{rng}_{P} &: P \to \mathsf{T}_{\mathsf{top}} \cup \mathsf{T}_{\mathsf{pmi}}. \end{aligned}$$

Definition 2. *A* B-Rep+PMI signature *is a tuple*

$$\Sigma_B = (\mathsf{T}_{\mathsf{top}}, \mathsf{T}_{\mathsf{geo}}, \mathsf{T}_{\mathsf{pmi}}, \mathsf{T}_{\mathsf{dat}}, \mathsf{R}, \mathsf{A}, P, \mathsf{dom}_{\mathsf{R}}, \mathsf{rng}_{\mathsf{R}}, \mathsf{dom}_{\mathsf{A}}, \mathsf{rng}_{\mathsf{A}}, \mathsf{dom}_{P}, \mathsf{rng}_{P}).$$

Table 1
Notation used throughout the paper.

RDF and OWL	
IRI, BNode, Lit	Sets of IRIs, blank nodes, and literals
T	Set of RDF terms, $T := \mathsf{IRI} \cup \mathsf{BNode} \cup \mathsf{Lit}$
$\mathcal{O}$	An OWL 2 EL ontology (a finite set of axioms)
G	An RDF graph (a finite set of triples)
D	An RDF instance graph
B-Rep+PMI signature	
$\mathsf{T}_{\mathsf{top}}$, $\mathsf{T}_{\mathsf{geo}}$, $\mathsf{T}_{\mathsf{pmi}}$, $\mathsf{T}_{\mathsf{dat}}$	Topological, geometric and PMI types, and datatype
R, A, P	B-Rep relation, attribute, and PMI relation symbols
$\mathsf{dom}_{\mathsf{R}}$, $\mathsf{rng}_{\mathsf{R}}$	Domain/range types of a relation symbol in R
$\mathsf{dom}_{\mathsf{A}}$, $\mathsf{rng}_{\mathsf{A}}$	Domain type and value type of an attribute in A
dom_P, rng_P	Domain/range types of a PMI relation symbol in P
Σ_B	A B-Rep+PMI signature
B-Rep+PMI Instance	
B	An instance over a signature Σ_B
Δ_B	Domain of an instance B
τ^B, r^B, p^B, a^B	Interpretation of a type, relation, PMI relation, or attribute in B
$\mathsf{adom}(B)$	Active domain of B

The signature separates entity types from datatypes: incidence and topology–geometry associations are captured by relation symbols in R, measurements and coordinates by attribute symbols in A, and PMI-to-B-Rep links by P.

An *instance* over Σ_B interprets each sort as a finite set of domain elements and each symbol as a relation or partial function over those sets.

Definition 3. *Let Σ_B be a B-Rep+PMI signature. An* instance *over Σ_B is a finite structure*

$$B = (\Delta_B, (\tau^B)_{\tau \in \mathsf{T}_{\mathsf{top}} \cup \mathsf{T}_{\mathsf{geo}} \cup \mathsf{T}_{\mathsf{pmi}}}, (r^B)_{r \in \mathsf{R}}, (p^B)_{p \in P}, (a^B)_{a \in \mathsf{A}})$$

such that the sets $\tau^B \subseteq \Delta_B$ are pairwise disjoint, every $r^B \subseteq \mathsf{dom}_{\mathsf{R}}(r)^B \times \mathsf{rng}_{\mathsf{R}}(r)^B$, every $p^B \subseteq \mathsf{dom}_P(p)^B \times \mathsf{rng}_P(p)^B$, and every a^B is a partial function from $\mathsf{dom}_{\mathsf{A}}(a)^B$ to the value domain of $\mathsf{rng}_{\mathsf{A}}(a)$. The active domain *of B, written $\mathsf{adom}(B)$, is $\bigcup_{\tau \in \mathsf{T}_{\mathsf{top}} \cup \mathsf{T}_{\mathsf{geo}} \cup \mathsf{T}_{\mathsf{pmi}}} \tau^B$.*

Attributes are partial functions because exported CAD data may omit values required by validation constraints; the mapping preserves these omissions, which SHACL can detect and report as violations.

Example 1 (Planar face with flatness tolerance)**.** *Let* $\mathsf{Face} \in \mathsf{T}_{\mathsf{top}}$, $\mathsf{Plane} \in \mathsf{T}_{\mathsf{geo}}$, $\mathsf{FlatnessTolerance} \in \mathsf{T}_{\mathsf{pmi}}$, *and* $\mathsf{Float} \in \mathsf{T}_{\mathsf{dat}}$. *Let* $\mathsf{hasSurface} \in \mathsf{R}$ *link a face to its surface,* $\mathsf{hasTarget} \in P$ *link a flatness tolerance to its target face, and* $\mathsf{hasToleranceMagnitude} \in \mathsf{A}$ *map a tolerance to its magnitude with* $\mathsf{dom}_{\mathsf{A}}(\mathsf{hasToleranceMagnitude}) \in \mathsf{T}_{\mathsf{pmi}}$ *and* $\mathsf{rng}_{\mathsf{A}}(\mathsf{hasToleranceMagnitude}) = \mathsf{Float}$. *An instance B contains $f \in \mathsf{Face}^B$, $s \in \mathsf{Plane}^B$, $t \in \mathsf{FlatnessTolerance}^B$, with $(f, s) \in \mathsf{hasSurface}^B$, $(t, f) \in \mathsf{hasTarget}^B$, and* $\mathsf{hasToleranceMagnitude}^B(t) =$ `"0.005"^^xsd:decimal`.

3. Validation Requirements

We motivate the ontology design around four representative validation requirements (R1–R4) that form the basis of the qualitative evaluation.

Requirement 1: Topological integrity. A linear edge must have exactly one start vertex and one end vertex. A missing endpoint invalidates the model for visualization and import into a CAD system.

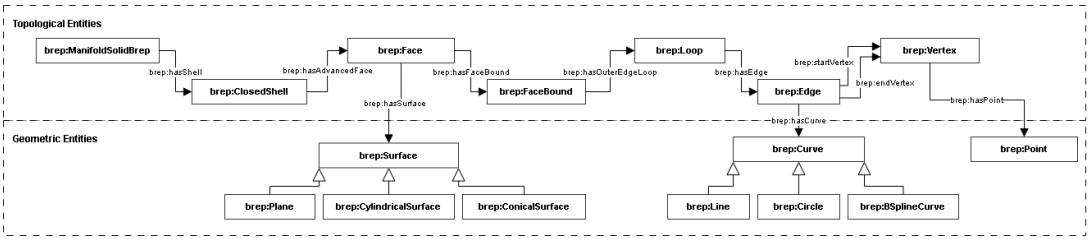


Figure 1: Core B-Rep ontology: topological entities (top) vs. geometric entities (bottom), connected by topology–geometry associations, with exemplary subclasses of curves and surfaces shown.

Requirement 2: PMI integrity. A flatness tolerance and a surface roughness value assigned to the same face are mutually inconsistent when the roughness exceeds the flatness bound, since both specifications cannot hold simultaneously.

Requirement 3: PMI-geometry compatibility. A radial dimension can be attached only to a face whose underlying surface is cylindrical, toroidal, or spherical; attaching it to a planar face is invalid because such a face has no radius attribute.

Requirement 4: Minimum wall thickness. For stability and manufacturability, a minimum wall thickness is essential. In this work, we consider a simplified wall object, which is defined by two parallel, non-touching faces that are fully connected through the solid.

4. BrepPMIOnt

We define BrepPMIOnt as an OWL 2 EL ontology that serves as the basis for mapping B-Rep+PMI instances to RDF. Our design of classes, subsumption axioms, and object/data properties draws on two reference works [2, 3] and three ISO standards: ISO 10303 (STEP, [5]), ISO 1101 (geometrical tolerancing [16]), and ISO 1302 (surface texture, [17]). BrepPMIOnt is available online [18], and its metrics are reported in Table 4.

4.1. B-Rep ontology

For the core B-Rep ontology, T_{top} contains entity types Vertex, Edge, Loop, Face and Shell; T_{geo} contains Point, Curve, Line, Circle, Surface, Plane, CylindricalSurface, ConicalSurface, SphericalSurface, BSplineCurve, and BSplineSurface. The ontology contains subsumption axioms such as

$$\mathsf{Line} \sqsubseteq \mathsf{Curve}, \quad \mathsf{BSplineCurve} \sqsubseteq \mathsf{Curve}, \quad \mathsf{Plane} \sqsubseteq \mathsf{Surface}, \quad \mathsf{CylindricalSurface} \sqsubseteq \mathsf{Surface}.$$

Object properties such as startVertex, endVertex, hasOuterBound, hasEdgeCurve, and hasSurface expose topological incidence and topology–geometry associations; datatype properties expose coordinates, radii, diameters, and lengths. For B-Rep incidence relations, we define both mutual directions explicitly, e.g., has-style and partOf-style properties, following the bidirectional terminology used by Stroud [2]. We define TopologicalEntityGroup (a named finite set of topological entities) and Feature (a classified group), enabling PMI assignment to named features such as holes or walls. Figure 1 illustrates the topological and geometric entities and their relationships, and the full B-Rep ontologies are visualized in Figure 6.

4.2. PMI ontology

The PMI core ontology contains dimensioning as DimensionalSize (e.g., RadialDimension, DiameterDimension, LinearDimension, AngularDimension) and DimensionalLocation, with size tolerances as PlusMinusTolerance (BilateralTolerance, UnilateralTolerance) and LimitsTolerance; for example, a face

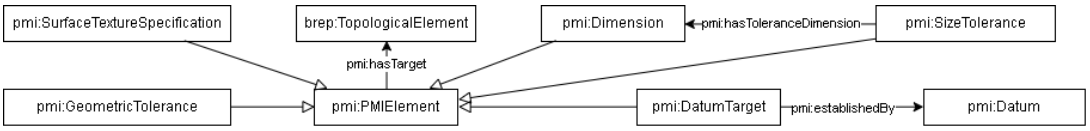


Figure 2: Core PMI ontology, comprising Dimension, Size Tolerance, Geometric Tolerance, Datums, and Surface Textures.

has a RadialDimension of 5 cm associated with a BilateralTolerance of ±0.5 cm. We capture GeometricTolerance subclasses including FormTolerance (e.g., FlatnessTolerance, StraightnessTolerance) and OrientationTolerance (e.g., ParallelismTolerance, PerpendicularityTolerance), as well as ProfileTolerance, LocationTolerance, and RunoutTolerance. We capture a datum system via (DatumReferenceFrame, DatumTarget) and the abstract Datum element. The ontology does not cover features such as holes, slots, or pockets, welding symbols, materials, or PMI qualifier hierarchies. The PMI ontology contains entities with hierarchically structured subclasses, for example

$$\mathsf{FlatnessTolerance} \sqsubseteq \mathsf{FormTolerance} \sqsubseteq \mathsf{GeometricTolerance} \sqsubseteq \mathsf{PMIElement}.$$

The object property hasTarget connects PMI elements to topological entities.

Figure 2 illustrates the high-level hierarchy of the PMI ontology. The full PMI ontologies are visualized in Figures 7, 2, and 8.

4.3. Classification

In addition to the B-Rep and PMI core ontologies, OWL 2 EL concept inclusion axioms provide classification. Essential for our approach is that the classification and validation depend only on the fixed vocabulary. For instance,

$$\mathsf{Face} \sqcap \exists \mathsf{hasSurface}.\mathsf{CylindricalSurface} \sqsubseteq \mathsf{CylindricalFace}$$

classifies faces supported by cylindrical surfaces, and

$$\mathsf{Edge} \sqcap \exists \mathsf{hasEdgeCurve}.\mathsf{Circle} \sqsubseteq \mathsf{CircularEdge}$$

classifies circular edges. These axioms classify individuals from explicitly represented topology–geometry associations and do not perform numerical geometry. Property chain axioms propagate attributes to inferred classes; for example,

$$\mathsf{hasSurface} \circ \mathsf{hasRadius} \sqsubseteq \mathsf{hasRadius}$$

connects the radius property directly to cylindrical faces. Classifying a CylindricalFace allows the SHACL shapes to check PMI-geometry compatibility directly (a radial dimension may target a CylindricalFace but not a PlanarFace) without referring to CAD-internal names. Inferred classes such as CylindricalFace could in principle be replaced by triple patterns in SHACL-SPARQL that navigate the underlying topology–geometry path directly, without OWL 2 EL reasoning. The benefit of the classification is therefore not to extend the logical expressiveness available to SHACL, but to provide separation of concerns, make constraints reusable, and decouple the structure: each SHACL shape targets a stable class name rather than embedding a triple pattern that has to be repeated and updated in every constraint.

Example 2. *Let* `:ps1` *and* `:f1` *be instances of* `brep:Plane` *and* `brep:Face`*, respectively, and let* `:f1` *be connected to* `:ps1` *by* `brep:hasSurface`*. From these assertions, OWL 2 EL classification infers that* `:f1` *is also an instance of* `brep:PlanarFace`*. Let* `:flat1` *be an instance of* `pmi:FlatnessTolerance`*. It has the tolerance value* `"5"^^xsd:float` *with the unit* µm*. The object-property assertion* `pmi:hasTarget(:flat1, :f1)` *specifies that the flatness tolerance targets this particular face. Figure 3 illustrates the example.*

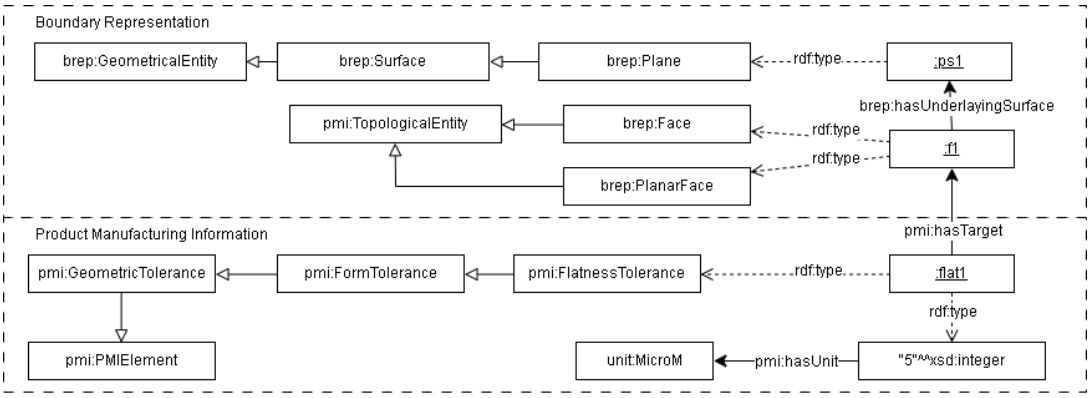


Figure 3: Chowlk diagram of the combined B-Rep+PMI ontology: B-Rep and PMI core ontology (left) vs. instance RDF data (right), inferred classes such as PlanarFace.

5. Mapping

We map B-Rep+PMI to RDF with respect to the BrepPMIOnt ontology. Therefore, we define an interface, based on the vocabulary defined in the ontology, that maps the entities, datatypes, relation symbols, attribute symbols, and PMI relation symbols of a B-Rep+PMI signature to OWL class names and property names. Let cls, obj, and dt be total functions mapping entities to class names, relation and PMI relation symbols to object-property names, and attribute symbols to datatype-property names or structured RDF patterns, respectively.

Definition 4. *An* ontology interface *for a signature* Σ_B *is a tuple* $I = (\mathsf{cls}, \mathsf{obj}, \mathsf{dt})$ *where* cls *maps every* $\tau \in \mathsf{T}_{\mathsf{top}} \cup \mathsf{T}_{\mathsf{geo}} \cup \mathsf{T}_{\mathsf{pmi}}$ *to an OWL class name,* obj *maps every* $r \in \mathsf{R} \cup P$ *to an OWL object-property name, and* dt *maps every* $a \in \mathsf{A}$ *to an OWL datatype-property name or a finite RDF pattern for a value node.*

The mapping constructs the RDF data from the B-Rep+PMI instance using the $\mathcal{O}_{\mathsf{BRep+PMI}}$ vocabulary. The construction follows the pattern of source-to-target mappings in ontology-based data access (OBDA) [19], with the source being a B-Rep+PMI instance rather than a relational database. Let $\mathsf{iri}_B : \mathsf{adom}(B) \to \mathsf{IRI} \cup \mathsf{BNode}$ be a total injective function that assigns an IRI or blank node to each domain element of B.

Definition 5. *Let* $I = (\mathsf{cls}, \mathsf{obj}, \mathsf{dt})$ *be an ontology interface, and let* rev *be a partial function assigning selected relation and PMI relation symbols to OWL object-property names used for the reverse direction. A* mapping *for* I *assigns to every B-Rep+PMI instance* B *a naming function* iri_B *and a finite RDF instance graph* D_B *satisfying:*

1. *for every* $x \in \tau^B$, D_B *contains* $(\mathsf{iri}_B(x),\ \mathsf{rdf:type},\ \mathsf{cls}(\tau))$*;*
2. *for every* $s \in \mathsf{R} \cup P$ *and every* $(x, y) \in s^B$, D_B *contains* $(\mathsf{iri}_B(x),\ \mathsf{obj}(s),\ \mathsf{iri}_B(y))$*; if* $\mathsf{rev}(s)$ *is defined, then* D_B *contains* $(\mathsf{iri}_B(y),\ \mathsf{rev}(s),\ \mathsf{iri}_B(x))$*;*
3. *for every defined* $a^B(x) = v$, D_B *contains the datatype-property assertion or RDF pattern prescribed by* $\mathsf{dt}(a)$ *for* $\mathsf{iri}_B(x)$ *and* v*.*

Let D_B be produced by a mapping for a B-Rep+PMI instance B. Every source sort membership, relation instance, and defined attribute value of B has a corresponding RDF assertion or finite RDF pattern in D_B according to the ontology and the B-Rep+PMI ontology interface, because the mapping follows three clauses. The first clause adds a type assertion for each source element in each source sort. The second clause adds an object-property assertion for each source relation tuple. The third clause adds the RDF representation of each defined attribute value. These clauses show that the assertions are explicit in D_B, so they do not require OWL entailment. This requires the mapping to be correct and to preserve the source facts exposed by the chosen B-Rep+PMI abstraction.

6. Validation

We use OWL 2 EL entailment for classification and SHACL and SPARQL for constraint checking. SHACL produces validation reports over RDF graphs [20, 21, 22], while SPARQL provides explicit graph-pattern queries and arithmetic filters [23, 24]. We target SHACL 1.2 throughout, using SHACL Core [25] and SHACL-SPARQL [26] where noted. Validation of requirements 2 and 4 requires SHACL-SPARQL with `sh:SPARQLConstraint` to traverse the graph via triple patterns, while the remaining requirements can be validated using only SHACL 1.2 Core constructs.

Let $\models_{\mathsf{val}}$ denote the satisfaction relation induced by the selected validation language.

Definition 6. *A* validation instance *is a tuple* $V = (\mathcal{O}_{\mathsf{BRep+PMI}}, D_B, \Phi, \models_{\mathsf{val}})$*, where* $\mathcal{O}_{\mathsf{BRep+PMI}}$ *is a B-Rep+PMI ontology,* D_B *is an RDF instance graph produced from a B-Rep+PMI instance, and* Φ *is a finite set of constraints. The* validation graph *is* $G_B = \mathcal{O}_{\mathsf{BRep+PMI}} \cup D_B$*, and the* violation set *is*

$$V_B = \{\varphi \in \Phi \mid G_B \not\models_{\mathsf{val}} \varphi\}.$$

A constraint may express integrity and quality requirements, such as topological structures or PMI specifications. A validation report records the violated constraint, the offending entity, the observed value when available, and the expected value or admissible range.

With BrepPMIOnt, constraints can be written against a stable conceptual vocabulary. Assume that D_B is produced by a mapping for B, and that every constraint in Φ refers only to class and property names in the ontology. Then every atomic graph pattern evaluated by a constraint is interpreted over a vocabulary whose source-level meaning is specified by the ontology and populated from B, because each class name and property name used by a constraint belongs to the ontology by assumption. OWL reasoning may add derived class memberships or property consequences according to $\mathcal{O}_{\mathsf{BRep+PMI}}$, but those consequences use the same ontology-level vocabulary. Hence, the validation engine evaluates constraints against a vocabulary whose connection to the source instance is specified by the ontology and the mapping.

Example 3. *Consider requirement R1 (topological integrity). Let* `:e1` *be an instance of* `brep:LinearEdge`*, inferred by OWL 2 EL classification from an asserted* `brep:hasEdgeCurve` *link to an instance of* `brep:Line`*. Suppose the source B-Rep+PMI instance B omits the reference to the starting vertex; the mapping then asserts* `:e1 a brep:Edge` *together with the topology–geometry association to the line, but does not assert a* `brep:startVertex` *triple for* `:e1`*. The constraint targets* `brep:LinearEdge` *and requires exactly one* `brep:startVertex` *value. Evaluating* $\models_{\mathsf{val}}$ *over* $G_B = \mathcal{O}_{\mathsf{BRep+PMI}} \cup D_B$ *shows that the constraint for* `:e1` *is not satisfied and therefore belongs to the violation set* V_B*.*

7. Expressiveness

OWL 2 EL classifies topological elements on the basis of explicitly asserted topology–geometry associations. For instance, an OWL 2 EL reasoner infers that a face is a CylindricalFace when it is associated with an individual of type CylindricalSurface via an object-property assertion. However, OWL 2 EL entailment is subject to expressiveness limitations that constrain the scope of the classification of more complex features. OWL 2 EL reasoners operate on symbolic assertions and do not support operations on continuous geometric representations [27]. Geometric relationships such as parallelism, perpendicularity, tangency, and coaxiality cannot be derived within the ontology: determining whether two surface normals are parallel requires vector arithmetic outside the inference procedures of any OWL 2 EL reasoner [28]. Such relationships must be computed externally and supplied as explicit assertions to the RDF graph. Similarly, OWL 2 EL does not support arithmetic or the evaluation of algebraic expressions [29, 30]; metric quantities such as distances and angles must be computed and asserted prior to validation. Consider a wall-thickness validation requirement, which refers to a *wall* as a named feature, whereas the exported model contains only faces and edges. Detecting a wall requires establishing that two planar faces are parallel, connected through solid material, and satisfy an angular

constraint relative to a reference direction, each condition depending on geometric computation over continuous representations and numerical constraint checking [31].

Similar limitations appear in validation with SHACL. Constraints can contain metric and geometric relationships, such as distances or parallelism. SHACL-SPARQL integrates SPARQL 1.1 into constraints, enabling algebraic operations but not geometric reasoning [32, 33]. Therefore, the ontology and SHACL alone do not enable full CAD validation; geometric computation must be supported by external means.

8. B-Rep Literal Datatype

The expressiveness limitations can be bridged by SPARQL extension functions that invoke external CAD libraries [32, 33]. However, the modularization of B-Rep information in the RDF graph complicates passing sufficient context to such functions. For example, a function `brepf:minDistance` computing the minimum distance between two faces requires not just their IRIs but the full hierarchy of geometric and topological entities associated with each face. Preserving the source serialization as a literal in the RDF graph avoids this problem by providing a self-contained input to extension functions, at the cost of storing redundant information alongside the decomposed RDF triples. Let L be a lexical space of serializations, V a value space of typed B-Rep+PMI entities, and ID a finite set of source identifiers occurring in a serialization.

Definition 7. *A* B-Rep literal datatype *is a triple $d_B = (L, V, \mu)$, where $\mu : L \times \mathsf{ID} \rightharpoonup V$ is a partial function that maps a serialization and a source identifier to the corresponding typed entity when the identifier occurs in the serialization.*

Every topological entity in the RDF graph must be connected to the serialization; we keep the ID of the origin format under the predicate brep:hasStepID, a subProperty of brep:hasID. The ID of each B-Rep+PMI element must equal its identifier in the serialization. We defined the class brep:Serialization and the relationship brep:hasSerialization with the domain brep:Serialization to enable property chains to the literal data. A datatype property specific to the type of input format, e.g. brep:hasStepSerialization, points to the actual lexical value.

Example 4 (Accessing value space)**.** *Let L be a specific lexical space, extracted from a CAD file and stored as literal data: "...#253=CIRCLE("",#2122,3.);#102=CYLINDRICAL_SURFACE...". Let* "253" *be the string stored under the dataProperty brep:hasStepID of an RDF object: ":IRI a brep:Circle . brep:hasStepID 253^^xsd:string". Then the mapping μ accesses the values of the key "253" and returns them in a structured format including all relevant information to describe them: "CIRCLE","Radius":3.,"EDGE":"Start-Point":...".*

We use this mapping inside custom SPARQL functions to forward the B-Rep data to the external CAD libraries. The mapping asserts mutual object properties; thereby, we define property chains in the OWL 2 EL profile that connect the serialization to each topological entity, for example

$$\texttt{inShell} \circ \texttt{inSolid} \circ \texttt{hasSerialization} \sqsubseteq \texttt{hasSerialization},$$

connects Face and brep:Serialization.

Proposition 1. *Let B be a B-Rep+PMI instance, D_B its RDF instance graph, and $\ell \in L$ a serialization of B attached to the solid representation node r_B via `brep:hasSerialization`. For every topological entity $x \in \mathsf{adom}(B)$, the B-Rep information of x is accessible in ℓ via the identifier* $\mathsf{iri}_B(x)$.

Proof. The mapping assigns to every $x \in \mathsf{adom}(B)$ the IRI $\mathsf{iri}_B(x)$ derived from the *ID* of x in ℓ. Since the *ID* is preserved by construction, the entity of x in ℓ is located by resolving $\mathsf{iri}_B(x)$ within the lexical form. The mapping directly asserts `brep:inShell`, `brep:inSolid`, and `brep:hasSerialization` triples along the containment path from x to r_B, and the OWL 2 EL property chain axioms above entail that $\mathsf{iri}_B(x)$ is linked to r_B via `brep:hasSerialization` for every x, while r_B carries ℓ via `brep:hasSerialization`. A SPARQL extension function evaluating ℓ therefore retrieves the full B-Rep information of x by its identifier. Since D_B is produced from the same instance B as ℓ, the information in D_B and in ℓ is consistent. □

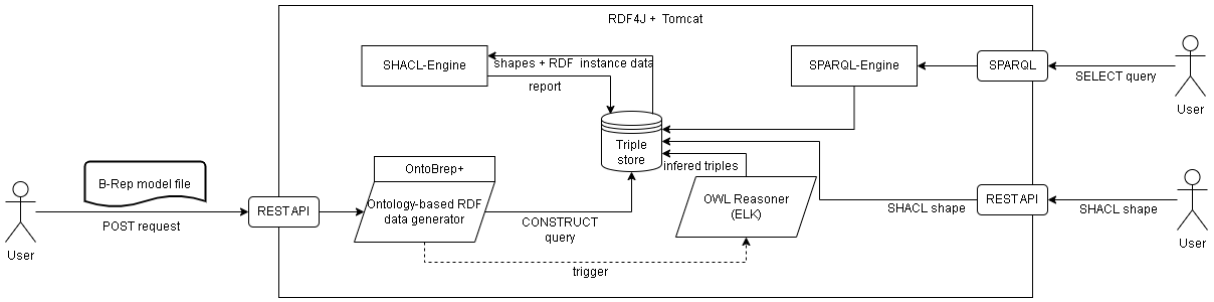


Figure 4: Pipeline for semantic CAD validation, comprising the module for graph generation, OWL 2 EL reasoner, triple store, SHACL, and SPARQL engine.

9. Implementation

We designed a prototype based on RDF4J [34] (Figure 4) that comprises a mapping module based on the ontology interface, an OWL 2 EL reasoner, and a SPARQL engine supporting the B-Rep literal datatype. We chose STEP AP242 [5] as the B-Rep+PMI source format. STEP provides a formal schema, and models are stored in human-readable Part 21 files. The standard is widely used in industry and is supported by CAD systems including Siemens NX, SolidWorks, and CATIA, all of which support the export of B-Rep data and semantic PMI. The RDF mapping module parses each Part 21 entity, for example "`#101=CIRCLE("",#2122,3.);`", into a hashmap, then generates triples in Turtle syntax according to the BrepPMIOnt ontology interface, for example "`:UID1 a brep:Circle ; brep:hasPlacement :UID2 ; brep:radius "3.0"^^xsd:float .`". We implemented the extraction of the information from the Part 21 file and the generation of the RDF data in Java as extensions of RDF4J, directly linked to the RDF4J triple store. We use the ELK reasoner [35] for OWL 2 EL classification. ELK materializes classification and property chain consequences. The triples are stored in an RDF4J triple store. To demonstrate the B-Rep literal datatype, we defined custom SPARQL extension functions `brepf:minDistance`, `brepf:volume`, and `brepf:parallel` as Java wrappers over the OpenCASCADE (OCC) libraries [36]. During SPARQL execution, the literal datatype is passed as input to OCC operations, and the result is returned to the SPARQL engine. RDF4J provides an SHACL engine that checks SHACL-SPARQL shapes.

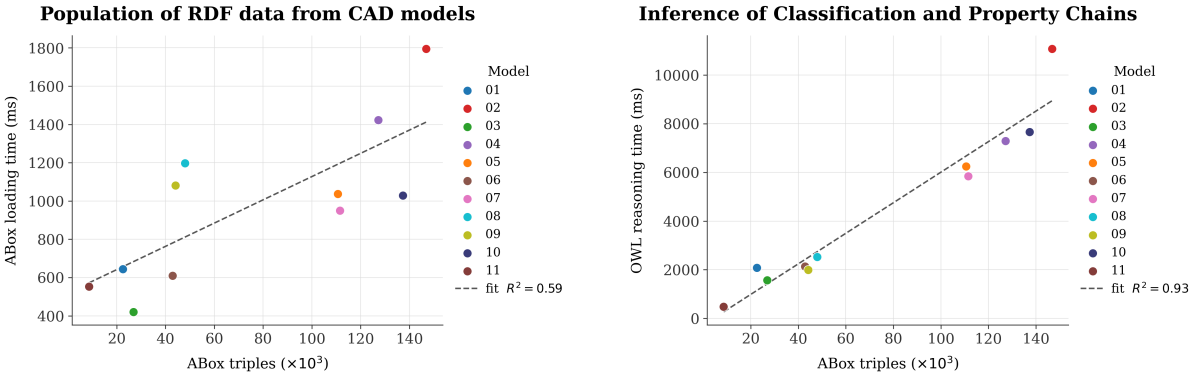
10. Evaluation

In this chapter, we show the feasibility of the approach by implementing it and applying it to the four defined requirements. We then capture the performance of the SHACL validation and compare validation with and without OWL 2 EL inference. We used 11 open-source mechanical-part models from the National Institute of Standards and Technology (NIST), stored as STEP AP242 Part 21 files, that combine B-Rep with semantic and visual PMI for testing STEP implementations [37].

10.1. Feasibility

The time required to initialize the graph and the time required for OWL 2 EL reasoning both scale linearly with the number of ABox triples, as shown in Figures 5a and 5b. A NIST model with more than 100,000 RDF triples takes up to 1.8 seconds to initialize. OWL 2 EL reasoning for classification and property chains takes between 0.5 and 11.1 seconds across the models. Reasoning is more expensive than graph initialization.

We modified the 11 NIST models and inserted 5 errors per requirement for R1–R3 in every model (165 errors in total), directly removing or adding triples in the RDF dataset. We did not insert errors for R4; instead, we identified existing wall-thickness violations in 5 of the 11 models (1 in model 5, 41 in model 7, 91 in model 8, 8 in model 9, and 8 in model 11, totaling 149 violations). We confirmed the detection of every error. The four SHACL 1.2 shapes are in the appendix (Listings 1, 2, 3,



(a) Graph initialization time vs. ABox triple count

(b) OWL 2 EL reasoning time vs. ABox triple count

Figure 5: Graph initialization time (left) and OWL 2 EL reasoning time (right) vs. ABox triple count, measured on the 11 NIST STEP AP242 models.

and 4). We check requirement 1 (a linear edge must have exactly one start vertex) using the classified entity `LinearEdge` and a SHACL 1.2 shape that checks the cardinality of its `startVertex` property (`LinearEdgeIntegrityShape`, Listing 1). We check requirement 2 (whether the roughness value of a face exceeds its flatness-tolerance bound) using SHACL-SPARQL to compare the roughness parameter of a SurfaceTextureSpecification with the magnitude of a FlatnessTolerance that targets the same face and to detect a violation (`RoughnessNotExceedsFlatnessShape`, Listing 2). For requirement 3 (validating that no radial dimension is attached to a planar face), we use SHACL 1.2 to check whether the classified object PlanarFace has a compatible PMI element (`RadialDimensionOnPlanarFaceShape`, Listing 3). For requirement 4 (validating a minimum thickness of walls), we use the B-Rep literal datatype to define custom SPARQL functions for SHACL-SPARQL. First, the simplified wall is detected by checking pairs of PlanarFace for parallelism and checking if they are fully connected through the solid without a gap in between. The custom SPARQL extension functions `brepf:parallel`, `brepf:minDistance`, and `brepf:connected` retrieve the B-Rep literal via the `hasSerialization` path and the faces within the ClosedShell, delegate the computation to the OCC library, and flag violations of a too small distance (`WallThicknessShape`, Listing 4). All four shapes detected every inserted error.

10.2. SHACL runtime

Our approach takes an RDF graph as input and uses the RDF4J SHACL engine with the registered SPARQL functions to calculate minimum distances. We evaluate the validation time and do not include graph loading and inference. The validation runtime for SHACL is shown in Table 2. Validation of requirements 1–3 was completed within a maximum of 36 ms per model and scaled linearly with the ABox size. Validation of requirement 4 does not exhibit linear scaling: it took 1.8 seconds for model 2 (146,800 triples) and 3.6 s for model 5 (110,599 triples). The execution time depends on the complexity of the custom SPARQL extension functions, the initialization of the CAD model and how often a specific feature is occurring in the model. Analysis and optimization of the SPARQL queries is left for future work.

Table 2
Average validation runtime for SHACL pre and post OWL 2 EL inference, averaged over the 11 NIST models and 100 iterations.

Approach	R1	R2	R3	R4
SHACL	20.4 ms	35.5 ms	229.8 ms	5977.5 ms
SHACL+OWL 2 EL axioms	9.9 ms	35.6 ms	7.7 ms	2902.5 ms

The precomputed information through OWL 2 EL inference leads to shorter paths and less redundant

Table 3
Comparison of related work.

Work	B-Rep	Classification	PMI	Mapping to RDF	Validation	B-Rep literal
OntoBREP	Yes	No	No	No	No	No
OntoSTEP	Yes	No	No	Yes	No	No
3DPMIOnto	Yes	No	Yes	No	Yes	No
ifcOWL	Yes	No	No	Yes	Yes	No
BrepPMIOnt	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

computations. Only R2 is not affected by the classification and uses the same SHACL shape twice. In R1 the classification allowed us to use LinearEdge as the target node and reduced the runtime by about 51%. In R3 the classification allowed us to simply use `sh:not` onto the classified element `brep:PlanarFace`, reducing the runtime by about 97%. The property chain for `brep:hasSerialization` and the classification lead to a runtime reduction of about 51%.

11. Related Work and Discussion

OntoBREP [38, 39] defines an ontology for B-Rep data covering topology, geometry, cardinalities, and constraints, bridging product models with semantic robot task descriptions. Mittel and Perzylo extend this by translating natural language queries into SPARQL via large language models, enabling non-expert access to B-Rep information through knowledge graphs. OntoSTEP [40] and its reimplementation STP2OWL [41] translate the EXPRESS schema of the STEP standard into OWL. 3DPMIOnto [42] defines an ontology for PMI in CAD models, connecting dimensions, tolerances, and other annotations to simple geometric primitives, and specifies SWRL rules to check PMI correctness. Industry Foundation Classes (IFC) [43] is a standard for the representation of Building Information Models (BIM). Besides B-Rep, it expresses other concepts of Architecture Engineering and Construction (AEC), such as IfcWindow, IfcDoor, IfcConstraint, and IfcMetric. ifcOWL is a translation of the IFC EXPRESS schema into OWL [44]. BimSPARQL [33] provides spatial SPARQL extension functions for validating BIM data using ifcOWL and SPIN [45]. As Table 3 shows, these approaches cover central parts of semantic mechanical CAD data representation and validation, but none provides the combination required for validation of mechanical B-Rep CAD models with PMI annotations.

OntoBREP, OntoSTEP and ifcOWL define OWL-based representations of B-Rep, but do not provide an integrated vocabulary for B-Rep and PMI validation. We chose not to extend OntoBREP or ifcOWL and instead developed a new ontology for two reasons: (i) we required a lightweight OWL 2 EL ontology that places validation in the SHACL constraints rather than in OWL entailment; and (ii) we introduced additional object properties between topological and geometric entities, enabling attributes to be assigned to inferred classes via property chain axioms. BimSPARQL provides geometric functions for validation of ifcOWL instances, but not for mechanical CAD models. OntoSTEP is tied to specific STEP application protocols and does not support STEP AP242 or PMI entities. 3DPMIOnto covers PMI concepts but attaches PMI elements to geometric primitives rather than to topological entities and follows different design decisions (e.g., it has no profile-tolerance class or measurement-value object and uses a less expressive datum system). Neither OntoBREP nor 3DPMIOnto defines a mapping for RDF instance data generation from CAD files. None of the compared approaches preserves the original B-Rep information as a literal datatype, enabling geometric computation via SPARQL extension functions.

We investigated OWL 2 EL as a representation language for B-Rep+PMI, the construction of RDF instance data via a B-Rep+PMI mapping, and the definition of OWL 2 EL classification axioms for entailment. In contrast to related work, we fix the ontology vocabulary, thereby decoupling the constraint set from CAD-internal names and improving maintainability. OWL 2 EL classification runs in polynomial time [35] and proves to be feasible for all 11 models. By preserving the CAD serialization

as a literal datatype, we combine semantic validation with third-party CAD libraries, overcoming the expressiveness limitations of OWL 2 EL. Current limitations include evaluation on a single B-Rep+PMI implementation (STEP AP242), the absence of a detailed scalability analysis of SPARQL extension functions over the B-Rep literal datatype, and the use of models with manually inserted errors rather than real-world industrial datasets. The assumption about how to detect a wall is too simple; the detection depends on the source CAD tool and the type of wall. In future work, we plan to investigate extensions of the SPARQL algebra for a fixed vocabulary for geometric computation, evaluate their scalability, and validate the approach on industrial-scale graphs and real-world CAD datasets obtained from production environments. We plan to extend the research to define a SPARQL fragment for B-Rep data and to detect features using SHACL 1.2 rules [46].

12. Conclusion

This work shows the potential of combining ontologies, RDF, and SHACL for semantic CAD validation. The central design decision was to define the ontology before the mapping. BrepPMIOnt provides a fixed OWL 2 EL vocabulary for mapping B-Rep+PMI data to RDF and expressing stable SHACL constraints. Classification axioms and property chains derive information used by these constraints and reduce the logic required for validation. The B-Rep+PMI literal datatype addresses the expressiveness and reasoning limitations of OWL 2 EL and SHACL by enabling an external CAD kernel to perform operations on B-Rep data. The evaluation showed that our approach is feasible and the OWL 2 EL axioms lead to a SHACL runtime reduction of more than 50% on R1, R3, and R4. However, further work is required to evaluate the approach on industrial datasets and analyze the scalability of the geometric extension functions.

Acknowledgments

This research is conducted in cooperation with Mercedes-Benz Group AG and the Institute for Artificial Intelligence at the University of Stuttgart. This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1574 - Project number 471687386.

Declaration of use of Generative AI

During the preparation of this work, the author used Claude Code (Anthropic), running the Claude Sonnet 5.0, Claude Opus 4.7, and Opus 5.0 models, in order to: Grammar and spelling check, Paraphrase and reword, Peer review simulation. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] K. L. Narayan, K. M. Rao, M. Sarcar, Computer aided design and manufacturing, PHI Learning Private Limited, 2008.

[2] I. Stroud, Boundary Representation Modelling Techniques, Springer, 2006. doi:https://doi.org/10.1007/978-1-84628-616-2.

[3] G. Henzold, Geometrical dimensioning and tolerancing for design, manufacturing and inspection: a handbook for geometrical product specification using ISO and ASME standards, Elsevier, 2020. doi:https://doi.org/10.1016/C2020-0-01081-9.

[4] S. K. Mohammed, M. H. Arbo, L. Tingelstad, Using semantic geometric dimensioning and tolerancing (gd &t) information from step ap242 neutral exchange files for robotic applications, International Journal on Interactive Design and Manufacturing 18 (2024) 6587–6603. doi:https://doi.org/10.1007/s12008-023-01242-7.

[5] M. J. Pratt, Introduction to ISO 10303 —- the STEP standard for product data exchange, J. Comput. Info. Sci. Eng. 1 (2001) 102–103. doi:https://doi.org/10.1115/1.1354995.

[6] S. Handschuh, R. Dotzauer, A. Fröhlich, Standardized formats for visualization: Application and development of jt, in: Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment: Proceedings of the 19th ISPE International Conference on Concurrent Engineering, Springer, 2012, pp. 741–752. doi:https://doi.org/10.1007/978-1-4471-4426-7_63.

[7] H. Lee, C. Yeo, S. Lim, B. C. Kim, K. C. Bae, D. Mun, Rule-based design verification for mechanical parts with dynamic rule subset selection, Scientific reports 15 (2025) 17153. doi:https://doi.org/10.1038/s41598-025-01815-4.

[8] D. Elshani, A. Nahkaee, A. A. Arrascue, H. Isakovic, N. Hedayati, J. Karlapudi, T. Wortmann, Comparative analysis of approaches for geometric data representation in RDF, in: Proceedings of the 13th Linked Data in Architecture and Construction Workshop, LDAC, 2025, pp. 92–107.

[9] J. Gebhard, Y. Wang, Semantic interoperability for digital twin-driven product development: a maturity model approach, Digital Twin (2026) 2618294. doi:https://doi.org/10.1080/27525783.2026.2618294.

[10] J. Gebhard, A. Emrich, Y. Wang, P. Loos, Ai-powered knowledge graphs: Accelerating simultaneous engineering in the automotive digital era, Digital Engineering (2025) 100070. doi:https://doi.org/10.1016/j.dte.2025.100070.

[11] F. Manola, E. Miller, B. McBride, et al., Rdf primer, W3C recommendation 10 (2004) 6. URL: https://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[12] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, B. McBride, RDF 1.1 concepts and abstract syntax, Technical Report, W3C, 2014. URL: https://www.w3.org/TR/rdf11-concepts/.

[13] G. Schreiber, Y. Raimond, RDF 1.1 Primer, Technical Report, 2014. URL: https://www.w3.org/TR/rdf11-primer/, accessed: 2026-07-17.

[14] W3C Owl Working Group, OWL 2 web ontology language document overview, Technical Report, 2012. URL: http://www.w3.org/TR/owl2-overview/, accessed: 2026-07-17.

[15] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An introduction to description logic, Cambridge University Press, 2017. doi:https://doi.org/10.1017/9781139025355.

[16] I. 1101, Geometrical product specifications (GPS)–geometrical tolerancing—-tolerances of form, orientation, location and run-out, International Organization for Standardization, Geneva, Switzerland, Standard (2017). URL: https://www.iso.org/obp/ui/#iso:std:iso:1101:ed-4:v1:en.

[17] DIN Deutsches Institut für Normung, DIN EN ISO 1302:2002-06: Geometrische Produktspezifikation (GPS) – Angabe der Oberflächenbeschaffenheit in der technischen Produktdokumentation (ISO 1302:2002), Technical Report, Beuth Verlag, Berlin, 2002.

[18] J. V. P. Gebhard, BrepPMIOnt, 2026. URL: https://github.com/juli0505/BrepPMIOnt.

[19] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, International Joint Conferences on Artificial Intelligence, 2018. URL: https://www.ijcai.org/proceedings/2018/0777.pdf.

[20] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), Technical Report, 2017. URL: https://www.w3.org/TR/shacl/, accessed: 2026-07-17.

[21] J. Corman, J. L. Reutter, O. Savković, Semantics and validation of recursive SHACL, in: International Semantic Web Conference, Springer, 2018, pp. 318–336. doi:https://doi.org/10.1007/978-3-030-00671-6_19.

[22] J. Corman, F. Florenzano, J. L. Reutter, O. Savkovic, SHACL2SPARQL: Validating a SPARQL endpoint against recursive SHACL constraints., in: ISWC (Satellites), 2019, pp. 165–168. doi:https://doi.org/10.22028/D291-24817.

[23] W3C SPARQL Working Group, SPARQL 1.1 Overview, W3C Recommendation, World Wide Web Consortium, 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[24] D. Hernández, A core sparql fragment (2020). URL: https://users.dcc.uchile.cl/~dhernand/reports/a_core_sparql_fragment.pdf.

[25] H. Knublauch, T. Bergwinkl, Y. Taghzouti, J. Wright, SHACL 1.2 Core, W3C Working Draft, W3C,

2026. URL: https://www.w3.org/TR/shacl12-core/, accessed: 2026-07-17.
[26] H. Knublauch, M. Goldberg, S. Henninger, L. Robaldo, SHACL 1.2 SPARQL Extensions, W3C Working Draft, W3C, 2026. URL: https://www.w3.org/TR/shacl12-sparql/, accessed: 2026-07-17.
[27] F. Baader, P. Hanschke, A scheme for integrating concrete domains into concept languages (1991).
[28] V. Haarslev, R. Möller, Description logic systems with concrete domains: Applications for the semantic web. (2003). URL: https://www.researchgate.net/profile/Ralf-Moeller-2/publication/221113143_Description_Logic_Systems_with_Concrete_Domains_Applications_for_the_Semantic_Web/links/0fcfd50a291541b550000000/Description-Logic-Systems-with-Concrete-Domains-Applications-for-the-Semantic-Web.pdf.
[29] F. Baader, U. Sattler, Description logics with concrete domains and aggregation., in: ECAI, volume 98, 1998, pp. 336–340. doi:https://doi.org/10.1016/S0306-4379(03)00003-6.
[30] L. Carsten, C. Areces, I. Horrocks, U. Sattler, Keys, nominals, and concrete domains, Journal of Artificial Intelligence Research 23 (2005) 667–726. doi:https://doi.org/10.1613/jair.1542.
[31] F. Baader, The description logic handbook: Theory, implementation and applications, Cambridge university press, 2007. doi:https://doi.org/10.1017/CBO9780511711787.
[32] D. Elshani, D. Hernandez, A. Nakhaee, A. A. Arrascue, S. Staab, T. Wortmann, geof3d: Sparql geometric functions for co-designing buildings, Advanced Engineering Informatics 71 (2026) 104261. doi:https://doi.org/10.1016/j.aei.2025.104261.
[33] C. Zhang, J. Beetz, B. de Vries, Bimsparql: Domain-specific functional sparql extensions for querying rdf building data, Semantic Web 9 (2018) 829–855. doi:https://doi.org/10.3233/SW-180297.
[34] The Eclipse Foundation, The eclipse RDF4J framework, 2026. URL: https://rdf4j.org/about/, computer software. Accessed: 2026-07-02.
[35] Y. Kazakov, M. Krötzsch, F. Simančík, The incredible elk: From polynomial procedures to efficient reasoning with ontologies, Journal of automated reasoning 53 (2014) 1–61. doi:https://doi.org/10.1007/s10817-013-9296-3.
[36] Open Cascade SAS, Open CASCADE Technology (OCCT), 2026. URL: https://github.com/Open-Cascade-SAS/OCCT, computer software. Accessed: 2026-07-02.
[37] National Institute of Standards and Technology, Download free CAD models, STEP files, and test results, 2025. URL: https://www.nist.gov/ctl/smart-connected-systems-division/smart-connected-manufacturing-systems-group/mbe-pmi-0, updated December 30, 2025, accessed: 2026-09-10.
[38] A. Perzylo, N. Somani, M. Rickert, A. Knoll, An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions, in: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, 2015, pp. 4197–4203. doi:https://doi.org/10.1109/IROS.2015.7353971.
[39] D. Mittel, A. Perzylo, Ontology-based CAD analysis with LLMs: Natural language querying of boundary representations, in: 2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2025, pp. 1–8. doi:https://doi.org/10.1109/ETFA65518.2025.11205658.
[40] S. Krima, R. Barbau, X. Fiorentini, R. Sudarsan, R. D. Sriram, Ontostep: OWL-DL ontology for step, National Institute of Standards and Technology, NISTIR 7561 (2009). URL: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7561.pdf.
[41] S. Kwon, L. V. Monnier, R. Barbau, W. Z. Bernstein, A new implementation of ontostep: Flexible generation of ontology and knowledge graphs of express-driven data, Journal of Computing and Information Science in Engineering 22 (2022). doi:https://doi.org/10.1115/1.4053079.
[42] Y. Chen, B. Bai, Y. Gan, H. Wang, X. Lin, Y. Chen, Ontology-based methodology for the intelligent detection of product manufacturing information semantic representation errors in the model-based definition, Journal of Advanced Mechanical Design, Systems, and Manufacturing 18 (2024). doi:https://doi.org/10.1299/jamdsm.2024jamdsm0065.
[43] International Organization for Standardization, ISO 16739-1:2024: Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries – Part 1: Data schema,

Technical Report, ISO, Geneva, Switzerland, 2024. URL: https://www.iso.org/standard/84123.html.
[44] J. Beetz, J. Van Leeuwen, B. De Vries, Ifcowl: A case of transforming express schemas into ontologies, Ai Edam 23 (2009) 89–101. doi:https://doi.org/10.1017/S0890060409000122.
[45] C. Fürber, M. Hepp, Using sparql and spin for data quality management on the semantic web, in: International Conference on Business Information Systems, Springer, 2010, pp. 35–46. doi:https://doi.org/10.1007/978-3-642-12814-1_4.
[46] R. David, D. Habgood, A. Seaborne, S. Steyskal, SHACL 1.2 Rules, W3C Working Draft, W3C, 2026. URL: https://www.w3.org/TR/shacl12-rules/, accessed: 2026-07-17.

A. Ontology Metrics

Table 4
Ontology metrics of BrepPMIOnt.

Metric	B-Rep	PMI	Axioms	Total
owl:Class	42	63	26	131
owl:ObjectProperty	47	32	21	100
owl:DatatypeProperty	32	14	0	46
rdfs:subClassOf	41	64	13	118
owl:equivalentClass	0	0	13	13
owl:propertyChainAxiom	0	0	29	29
Total triples	567	453	436	1456

B. PMI Chowlk Diagrams

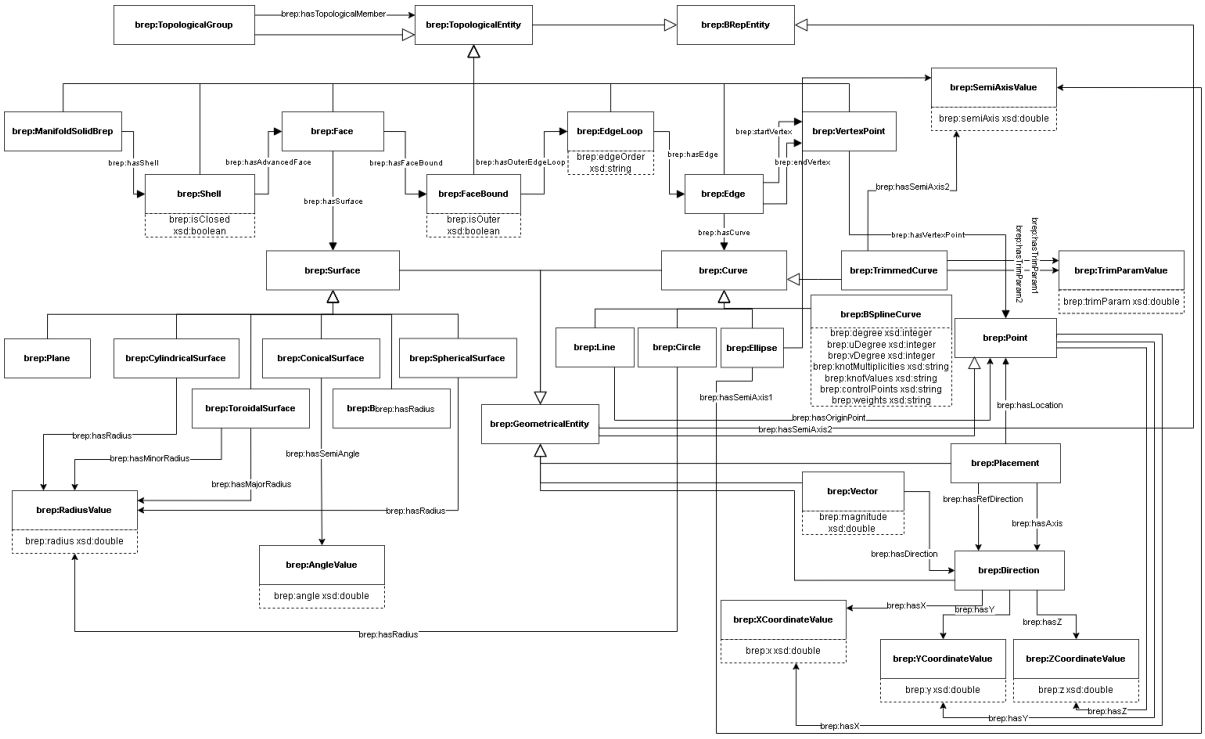


Figure 6: Chowlk diagram of the full B-Rep ontology.

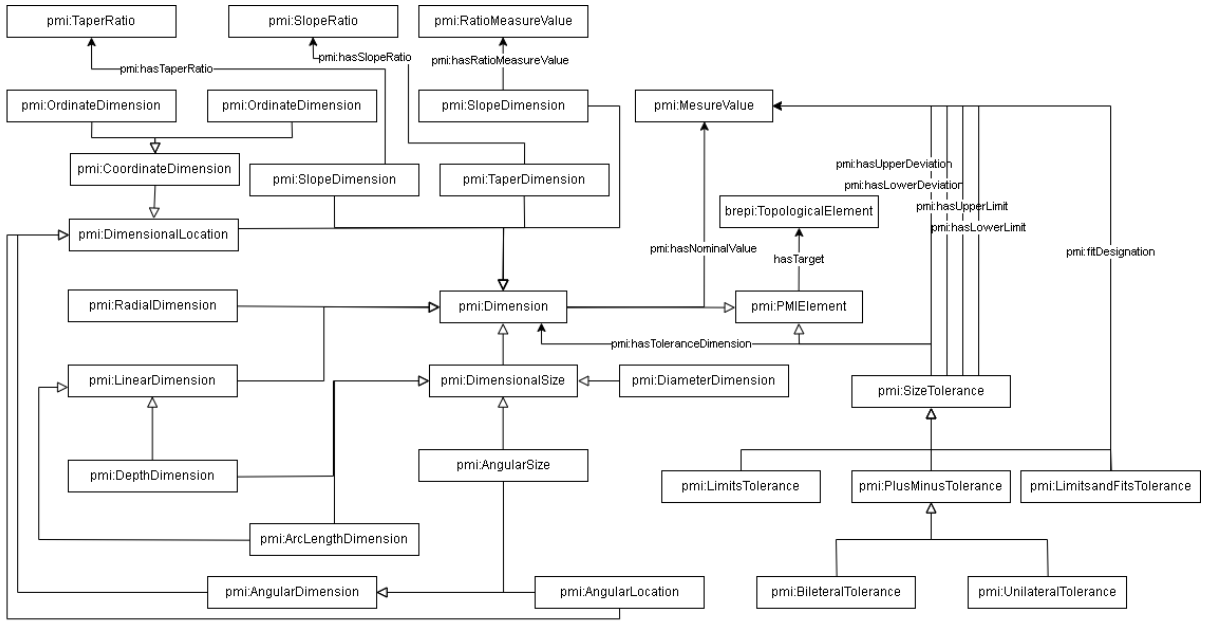


Figure 7: Chowlk diagram of the full PMI ontology of Dimensioning with Tolerances.

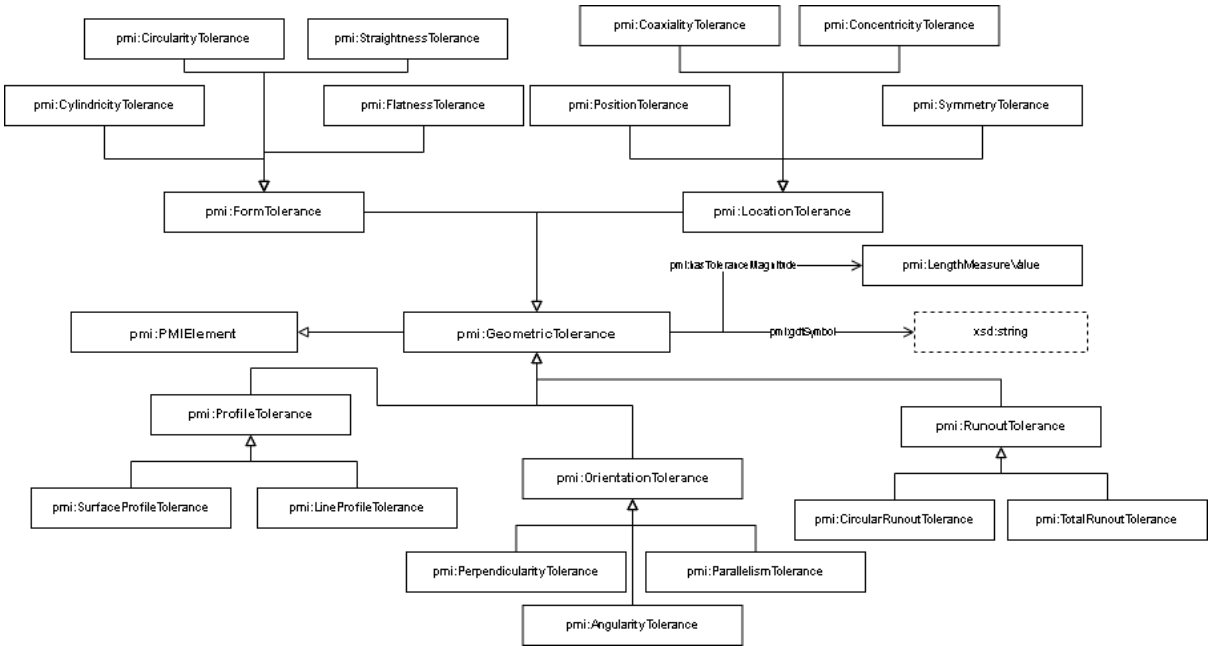


Figure 8: Chowlk diagram of the PMI ontology of Geometric Tolerances.

C. SHACL Shapes

Listing 1: SHACL shape for linear edge integrity.

```
shapes:LinearEdgeIntegrityShape a sh:NodeShape ;
    sh:targetClass brep:LinearEdge ;
    sh:property [
        sh:path brep:startVertex ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] .
```

Listing 2: SHACL shape for flatness–roughness consistency.

```
shapes:RoughnessNotExceedsFlatnessShape a sh:NodeShape ;
    sh:targetClass pmi:SurfaceTextureSpecification ;
    sh:sparql [
        a sh:SPARQLConstraint ;
        sh:select """
            SELECT $this ?roughness ?flatness
            WHERE {
                $this pmi:hasSurfaceTextureParameter ?param .
                ?param pmi:hasParameterValue ?rv .
                ?rv pmi:numericValue ?roughness .
                $this pmi:hasTarget ?face .
                ?flat a pmi:FlatnessTolerance ;
                      pmi:hasToleranceMagnitude ?fmv ;
                      pmi:hasTarget ?face .
                ?fmv pmi:numericValue ?flatness .
                FILTER (?roughness > ?flatness)
            }
        """ ;
    ] .
```

Listing 3: SHACL shape for radial dimensions on non-planar faces.

```
shapes:RadialDimensionOnPlanarFaceShape a sh:NodeShape ;
    sh:targetClass pmi:RadialDimension ;
    sh:property [
        sh:path pmi:hasTarget ;
        sh:not [ sh:class brep:PlanarFace ] ;
    ] .
```

Listing 4: SHACL shape for minimum wall thickness.

```
shapes:WallThicknessShape
    a sh:NodeShape ;
    sh:targetClass brep:PlanarFace ;
    sh:sparql [
        a sh:SPARQLConstraint ;
        sh:prefixes shapes:WallThicknessPrefixes ;
        sh:message "Planar faces {$this} and {?value} form a wall thinner than 4.0 mm: {?
    distance} mm"@en ;
        sh:select """
            SELECT DISTINCT $this ?value ?distance WHERE {
                {
                    SELECT DISTINCT $this ?other ?shell ?geo ?shellId ?id1 ?id2 ?distance
    WHERE {
                        ?shell a brep:ClosedShell ;
```

```
                            brep:hasSerialization ?ser ;
                            brep:serializationID ?shellId ;
                            brep:hasFace $this ;
                            brep:hasFace ?other .
                    ?ser brep:hasGeometry ?geo .
                    FILTER(str($this) < str(?other))

                    $this brep:serializationID ?id1 .
                    ?other a brep:PlanarFace ;
                            brep:serializationID ?id2 .
                    $this brep:hasSurface ?surface1 .
                    ?other brep:hasSurface ?surface2 .
                    ?surface1 a brep:Plane ; brep:hasPlacement ?placement1 .
                    ?surface2 a brep:Plane ; brep:hasPlacement ?placement2 .

                    FILTER(brepf:parallel(?geo, ?id1, ?id2))
                    BIND(brepf:minDistance(?geo, ?id1, ?id2) AS ?distance)
                    FILTER(bound(?distance) && ?distance > 0.0 && ?distance < 4.0)
                }
            }
            BIND(?other AS ?value)
            FILTER(brepf:connected(?geo, ?shellId, ?id1, ?id2))
        }"""
    ] .
```