

# An ontological model for the representation of damage to constructions

Al-Hakam Hamdan<sup>1</sup>, Mathias Bonduel<sup>2</sup>, and Raimar J. Scherer<sup>1</sup>

<sup>1</sup> Institute of Construction Informatics,  
Technische Universität Dresden, Dresden, Germany  
Al-Hakam.Hamdan@tu-dresden.de

<sup>2</sup> Dept. of Civil Engineering, Technology Cluster Construction,  
KU Leuven, Ghent, Belgium

**Abstract.** The Damage Topology Ontology (DOT) is presented, a web ontology that provides terminology to represent construction-related damages and their topology as well as relations to affected construction elements and spatial zones. Besides the topology, classes and properties for documentation management and a minimal structural assessment have been proposed in DOT. In this regard, DOT provides all classes and properties needed for practical use in construction inspections and damage assessment. The ontology is developed to be used with the modular Linked Building Data ontologies structure, where DOT works as core damage ontology which can be extended with multiple modules related to detailed damage classification, damage assessment, mechanical degradation and other application scenarios. Geometrical damage representations are separated from the topology, so that it is possible to initially record damages during the inspection without any geometrical properties and link it later with a corresponding representation using terminology from geometry-related ontologies. In conclusion, DOT can be applied as a stand-alone web ontology to represent damages in a machine-interpretable format and replace conventional record approaches. Therefore, a generic terminology is used that enables the inclusion of various types of damage, which can be extended with domain-specific information.

**Keywords:** Damage Recording · Building Information Modeling · Ontologies · Linked Data.

## 1 Introduction

Building Information Modeling (BIM) offers new ways for modeling constructions in a digital and collaborative environment. However, existing commercial BIM applications are mainly intended to support the design phase and thus are limitedly applied for already existing constructions [16]. For instance, in 2017, 69% of all construction work in Germany was carried out on existing buildings [5]. Consequently, the status of BIM is in contradiction with a large part of the current development of construction services. In order to create rich digital representations of existing constructions it must be possible to connect them to other data, e.g. related to damages, project phasing or deconstruction processes. In this regard, multiple stakeholders, e.g. damage assessment teams, architects or structural engineers can benefit in particular from a digital representation of damages and its data exchange, since degradation occurs during the entire life cycle of a construction and has an impact on the usability as well as the structural capability.

In this research a model for damage representations has been developed by utilizing web technologies, more specifically Linked Data that allow to easily define new structured terminology based on web standards such as the Resource Description Framework (RDF)<sup>1</sup>, RDF-Schema(RDFS)<sup>2</sup> and Web Ontology Language (OWL)<sup>3</sup>. Therefore, the Damage Topology Ontology (DOT) was developed as a modular web ontology for defining damage objects and their topology. Due to its generality, DOT can be utilized as part of the BIMification process [14], where an in depth survey of damages is an important process step. Besides defining the damage topology, DOT also serves as a core ontology for the digital representation of degradation. Thereby, DOT can be extended by additional ontologies that allow to add more detailed damage classifications and the structural mechanics of damages as well as ontologies for damage assessment according to specific standards. In this regard, one primary objective is the compatibility with existing web ontologies created within the W3C Linked Building Data Community Group (LDB CG), such as the Building Topology Ontology (BOT) [12] or ifcOWL [10]. The aim of this paper is the description of DOT, as well as introducing extension ontologies for damage classification, damage assessment and structural mechanics. Furthermore, the web ontologies are tested on a use case where inspected damages are modeled on ontological models which represent existing constructions. Lst. 1 contains an overview of the used prefixes throughout this research paper.

**Listing 1:** Used RDF prefixes in this paper.

```

@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix bot: <https://w3id.org/bot#> .
@prefix brot:
  <http://www.semanticweb.org/taras/ontologies/2018/7/BROT#> .
@prefix dot: <https://w3id.org/dot#> .
@prefix cdo: <https://w3id.org/damagemodels/cdo#> .
@prefix dasb: <https://w3id.org/damagemodels/d-asb-ing#> .
@prefix dmo: <https://w3id.org/damagemodels/dmo#> .
@prefix omg: <https://w3id.org/omg#> .
@prefix fog: <https://w3id.org/fog#> .
# namespace for example node instances
@prefix inst: <http://ex.org/alhak/data/D1#> .

```

## 2 Related Work

### 2.1 Geometry dependent BIM for Damage Representation

In order to digitize defects and damages of constructions, various approaches have been developed that are dependent on a BIM model with geometry, including several ones

<sup>1</sup> <https://www.w3.org/RDF/>

<sup>2</sup> <https://www.w3.org/TR/rdf-schema/>

<sup>3</sup> <https://www.w3.org/OWL/>

that propose to extend the Industry Foundation Classes (IFC)<sup>4</sup>, a standardized data exchange format for BIM.

One of the first approaches for representing damages has been developed by Hammad et al. [6], where damage information has been linked with other inspection relevant information about the bridge life cycle stages in a 4D model that uses definitions from IFC for the representation of the 3D bridge model and combines them with time-dependent data. In the project SeeBridge [13] the IFC extension ‘Inspection BIM Model’ has been defined for bridge constructions. It consists of damages modeled in 3D as boundary shape representations combined with meaningful damage parameters (e.g. damage type or size measurements) and attached texture images. The defect elements are linked with construction elements by using the aggregation relationship ‘IfcRelAggregates’. Based on this research, Tanaka et al. [15] developed a bridge information model, using degradation elements, as well as measured and repaired regions as primary elements. Additionally, a new subtype of ‘IfcRelConnects’, named ‘IfcRelConnectsToTimeVariations’ has been created to link equivalent degradation elements from different inspection dates with each other, so that the degradation progress could be represented.

Despite the aforementioned methods for modeling damage representations in IFC or other geometrical BIM formats, they are still not supported by current BIM modeling tools and formats. One of the reasons is related to the process of creating 3D geometrical construction representations, which is more complicated for existing constructions. To get a 3D geometry model, geometric information reflecting the as-is state of the existing construction (e.g. via a laser scan or from existing plans) is necessary. Furthermore, a 3D solid geometry of each construction component has to be created - either manually or semi-automatically - before these objects can be classified, get properties and can be related to damages that affect them. Both steps can have a serious impact on the price and time to complete a BIM model. The first step can be expensive if accurate as-built documentation is lacking because of the survey equipment or expertise while the second step, depending on the aimed accuracy, requires a relatively long modeling time. Another problem, that concerns the IFC extensions for damage representation is the monolithic structure of IFC which leads to a slower approval process of new proposed extensions in the main schema. This problem could be avoided when using a more modular data structure that can be extended over time. Consequently, no such data structure exists that supports linking a damage representation with geometry, assessment, documentation and digital construction elements. However, this is necessary to enable a collaborative data environment between multiple stakeholders. Therefore, in the following section, the usage of Linked Data for defining damages is studied more in detail.

## 2.2 Linked Data Approach for Damage Representation

To use BIM without the need to solve the problem of modeling a 3D geometry of an existing construction, Bonduel et al. [2] suggests a more flexible workflow utilizing Linked Data, by starting to describe the building topology without geometry. As a result, it is directly possible to classify building components and to connect them with other data. In a later phase, when the need arises, geometrical data can be connected to the earlier created building topology. By utilizing this approach, it would be possible to connect

<sup>4</sup> <http://www.buildingsmart-tech.org/specifications/ifc-overview>

damages to an existing construction even before a 3D geometry is available and use them together with a construction topology ontology, like BOT in case of buildings or with other future infrastructure web ontologies. The use of Linked Data principles is based on open web standards published by the W3C such as RDF, RDFS, OWL and SPARQL. Linked Data offers several advantages when it comes to representing damages and the affected construction elements, such as the usage of HTTP URIs to identify objects which allow us to connect different disparate datasets, a standardized querying language (SPARQL), generic reasoning engines and the use of a shared data terminology (TBox) that provides definitions of classes and relations. Furthermore, RDF data represents a directed graph of connected nodes that allows for more flexible data structures. Linked Data technologies also allow the partitioning of one data model into a modular structure, consisting of multiple smaller interlinked data schemas or web ontologies. Smaller and more modular web ontologies are considered easier to understand for software developers and software applications can select which ontologies they support for their case. Several web ontologies have been developed for representing damages, damage classification and inferring implicit knowledge about them. Lee et al. [9] developed a web ontology utilized in a management system for querying specific defect cases based on construction conditions derived from a BIM model. Therefore, multiple instances, named as 'cases' are predefined in the ABox of the web ontology. Each of these instances belongs to a certain amount of classes that define defects as well as construction properties (e.g. component type or material) so that defect patterns could be concluded. However, the web ontology aims to use its instances mainly as categorization templates and does neither support modeling the topology of damage representations nor linking damages to construction components. To the best of our knowledge, the ontology of Lee et al. [9] is not publicly available online, which hinders its application. One of the most extensive approaches towards representing damages in an ontological model has been developed in the project MONDIS, which focuses on damage to cultural heritage and monuments [4]. The publicly available web ontology utilized in MONDIS<sup>5</sup> allows modeling damage representations, as well as linking them to certain objects such as construction elements. Furthermore, the damage causation can be defined as well as additional properties for damages and affected components. Despite the possibility to define topological relations between damages and construction components, additional terminology to represent a detailed damage topology is missing. Examples of such terminology could be related to methods to define relations between adjacent damages or to group collections of smaller damages in a damaged area. Moreover, the web ontology contains concepts not related to damages such as classes to represent construction elements and therefore is not modular, which leads to a lesser compatibility with existing web ontologies from the LBD CG, such as BOT. To support the already existing Linked Data ontologies developed by the LBD group, a publicly available web ontology for a wide variety of damage representations related to different types of constructions has been developed in this research. It possesses a modular structure that is compatible with the LBD ontologies and will be described in detail in the following sections.

---

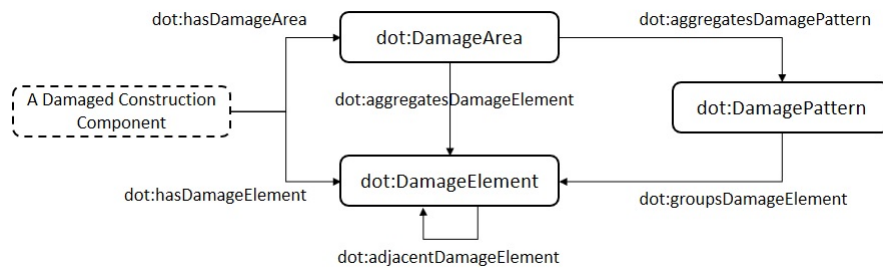
<sup>5</sup> <https://kbss.felk.cvut.cz/ontologies/2011/monument-damage-core.owl>

### 3 Structure of the Damage Topology Ontology (DOT)

The Damage Topology Ontology or DOT provides the necessary terminology and definitions (TBox) defined in OWL (Web Ontology Language) to support the modeling of generic damage representations and their topology, where documents and other information that is assembled during the different phases of inspection can be connected to each damage instance. Furthermore, a simple classification of each generic damage regarding the influence on the structural capability is supported. Thus, DOT works as a core ontology, which can be used for any type of damage or construction. The ontology is published on Github<sup>6</sup> and has a public HTML documentation that can be accessed via the [base URI](#)<sup>7</sup>.

#### 3.1 Topological Components

DOT is designed for defining and linking damage representations to instances classified by other web ontologies that define a construction (e.g. instances of BOT classes for buildings). Thereby, the instances for damage representations are topologically structured based on damage definitions of different detail levels. The OWL classes and object properties used for this purpose are shown in Figure 1.



**Fig. 1:** Overview of the topological classes and object properties defined by DOT.

Damages can be modeled and assigned to affected components by using one of two subclasses of the class `dot:Damage`. Thereby, individual damage representations can be modeled by using the class `dot:DamageElement` and be linked to any instance of a construction component, e.g. `bot:Element` by using the object property `dot:hasDamageElement`. A detailed damage geometry could then be linked to the instance of `dot:DamageElement` at a later point in time by utilizing additional ontologies such as the [Ontology for Managing Geometry \(OMG\)](#)<sup>8</sup> and [File Ontology for Geometry formats \(FOG\)](#)<sup>9</sup> [17,3], or the [Information Container for Data Drop \(ICDD\)](#) [7]. The ICDD stores all used geometry files as well as other documents in a data container and links

<sup>6</sup> <https://github.com/Alhakam/dot>

<sup>7</sup> <https://w3id.org/dot#>

<sup>8</sup> <https://w3id.org/omg#>

<sup>9</sup> <https://w3id.org/fog#>

them with DOT instances via RDF linksets. However, modeling damages as detailed as possible is not always advisable, e.g. in the case of structural analysis, since using this discrete approach for a huge amount of damages could lead to intolerable computation times. Furthermore, modeling geometry of complex damage patterns manually, such as a collection of hairline cracks, is a time consuming activity. Therefore, a damaged area that represents a conceptual bounding box for multiple smaller damages can be modeled by using the class `dot:DamageArea` which can be linked to a more simplified geometry. Instances of `dot:DamageArea` can be linked to damaged construction components via the object property `dot:hasDamageArea`. To assign instances of `dot:DamageElement` to a `dot:DamageArea` instance, the object property `dot:aggregatesDamageElement` is used. However, it is also possible to model a `dot:DamageArea` instance without defining the individual `dot:DamageElement` instances it aggregates, if it is only used for a simplified damage representation. To define damages that have a physical connection with other damages, such as adjacent cracks, multiple instances of `dot:DamageElement` can be linked together with the object property `dot:adjacentDamageElement`. By utilizing this property, complex patterns of damages can be modeled. To refer to these patterns, e.g. for documentation or addition of metadata, a pattern of physically connected damages is represented by an instance of the class `dot:DamagePattern` in which instances of `dot:DamageElement` can be grouped via the object property `dot:groupsDamageElement`. The pattern is then connected to an instance of `dot:DamageArea` through the `dot:aggregatesDamagePattern` object property. In addition, by applying a reasoning engine to the ontology, instances of `dot:DamageElement` that are linked via `dot:adjacentDamageElement` can be derived through a property chain axiom as grouped elements in a `dot:DamagePattern` instance. Thereby, at least one of the connected instances of `dot:DamageElement` must be linked to the `dot:DamagePattern` instance. The same is achieved via a property chain axiom for adjacent instances of `dot:DamageElement` in a `dot:DamageArea` instance.

### 3.2 Documentation Data

An important part of each inspection is a detailed documentation of all detected damages. Therefore, multiple documents e.g. photos, sketches, reports or test protocols are created during the various inspection and damage assessment phases. These documents can be stored in an ICDD and linked with the damage instances from DOT through RDF linksets. To ensure, that the model can be used without being dependent on other data models, the class `dot:ExternalResource` is used for representing documents that describe damage related information. By using the data property `dot:filePath` the location in a file system can be referred. Instances of `dot:ExternalResource` can be linked to any instance node that represents a damage, a construction component or the whole construction via the object property `dot:hasDocumentation`. For non-damage related documents, classes and properties from other ontologies, like `schema:DigitalDocument` could be used. Additionally, small textual descriptions for damage information can be connected to the same instances by using the class `dot:Description`. Instances of `dot:Description` can be linked with various metadata as well as with the actual textual content of the description. For the latter the datatype property `dot:descriptionContent` should be used. For an enhanced management of multiple inspections that are processed through the construction lifecycle, the class `dot:Inspection` should be used for assigning detected instances of `dot:Damage` via the object property `dot:belongsToInspection`. In addition,

the corresponding inspector can be represented through an instance of `dot:Inspector`. Furthermore, the damage causation can be defined through the class `dot:Causation` and linked with an instance of `dot:Damage`. However, it is recommended to specify further subclasses of `dot:Causation` in domain specific extensions for practical application.

### 3.3 Classes for Structural Assessment

In order to differentiate damages by their impact on structural capability, the two classes `dot:StructuralDamage` and `dot:Defect` have been implemented. While `dot:StructuralDamage` classifies every damage that influences the structural capability, `dot:Defect` works as an opposite class and marks a damage as harmless to the structural properties of the construction. However, defects can still have an impact on the durability or traffic safety of the construction. It is recommended to use these classes always for instances of `dot:DamageElement`, to infer the structural impact of `dot:DamageArea` or `dot:DamagePattern` through aggregated instances.

### 3.4 Supported Extensions of DOT

One main objective of the presented damage ontology is its modularity and extensibility. Therefore, DOT contains only generic damage concepts that are always applicable for describing any type of damage related to any type of structure. Other classes for classifying damages further or additional properties, e.g. for mechanical behavior or damage assessment, should be implemented in DOT ontology extensions. Therefore, three additional ontologies have been developed as [sample extensions](#)<sup>10</sup> for DOT. The Concrete Damage Ontology (CDO) classifies damages in reinforced concrete and thus can be considered a specialized damage taxonomy. The DASB Ontology<sup>11</sup> adds assessment parameters for damaged bridges based on DIN 1076 [1]. Lastly, the Damage Mechanics Ontology (DMO) defines reduced material parameters to describe the mechanical influence of damages during a structural analysis. In the current version only the smeared crack structural analysis approach is supported and thus the classes and properties of DMO are only applicable for instances of `dot:DamageArea`. In the following section, a demo data set is introduced applying the aforementioned DOT extensions.

## 4 Application of DOT

To demonstrate the developed ontology and its extensions, two sample (ABox) datasets have been published in a [part 1](#)<sup>12</sup> and a [part 2](#)<sup>13</sup>, together with example queries which

<sup>10</sup> <https://github.com/Alhakam/dot/tree/master/Extension>

<sup>11</sup> DASB is a German acronym and stands for: "Deutsche Anweisung Straßeninformationsbank für (Ingenieur)Bauten"

<sup>12</sup> [https://raw.githubusercontent.com/Alhakam/dot/master/ABox-Examples/dotSampleData\\_pt1.ttl](https://raw.githubusercontent.com/Alhakam/dot/master/ABox-Examples/dotSampleData_pt1.ttl)

<sup>13</sup> [https://raw.githubusercontent.com/Alhakam/dot/master/ABox-Examples/dotSampleData\\_pt2.ttl](https://raw.githubusercontent.com/Alhakam/dot/master/ABox-Examples/dotSampleData_pt2.ttl)

were applied in a [demo](#)<sup>14</sup> utilizing the [SPARQL-Visualizer](#)<sup>15</sup>. In this regard, DOT is applied on two different ABox graphs. The first ABox dataset represents a structural frame that is modeled with BOT. The triples of the second ABox graph, which is used in section 5.2 for query application, represent a damaged bridge construction and therefore use terminology from DOT, CDO, DASB and DMO. In addition, BROT [8], an ontology that is currently under development, has been used for defining the topology and classification of bridges and their components. In this section, methods for creating and querying RDF data (ABox) using terminology from DOT and its extensions (TBox) are described.

#### 4.1 Methods for creating damage representations with DOT

DOT allows the digital representation of damages and assigning them to construction components. Therefore, several methods are possible for linking damages to such components and defining a damage topology. The methods, presented in this section are applied to an example structural frame named `inst:structuralFrameA` that aggregates two columns and one beam using the BOT terminology (see Listing 2).

**Listing 2:** Definition of the structural frame and its aggregated components

```
inst:structuralFrameA a bot:Element ;
  bot:aggregates inst:frameA-beam1 , inst:frameA-column1 ,
  inst:frameA-column2 .
inst:frameA-beam1 a bot:Element .
inst:frameA-column1 a bot:Element .
inst:frameA-column2 a bot:Element .
```

*Method 1:* Listing 3 illustrates the most intuitive method for modeling damages, where an instance of `dot:DamageElement` is directly linked to a construction component. However this method should only be used for representing significant individual damages, such as big cracks or spallings and not for groups of smaller damages.

**Listing 3:** Assigning a single damage element to a component

```
# asserted triple
inst:frame1-column1 dot:hasDamageElement inst:damageElement1 .
# inferred triples
inst:damageElement1 a dot:DamageElement , dot:Damage .
inst:frame1-column1 dot:hasDamage inst:damageElement1 .
```

*Method 2:* An alternative and often more time-efficient method for modeling collections of small damages in a damaged area, is using an instance of `dot:DamageArea` as damage representation and linking it to a component (See Listing 4).

**Listing 4:** Assigning a damaged area to a component

```
# asserted triple
inst:frameA-column2 dot:hasDamageArea inst:damageArea2 .
```

<sup>14</sup> <https://madsholten.github.io/sparql-visualizer/?file=https://raw.githubusercontent.com/Alhakam/dot/master/ABox-Examples/dot-demo.json>

<sup>15</sup> <https://github.com/MadsHolten/sparql-visualizer>



```
# inferred triples
inst:damageArea2 a dot:DamageArea , dot:Damage .
inst:frameA-column2 dot:hasDamage inst:damageArea2 .
```

*Method 3:* While it is allowed to define an instance of `dot:DamageArea` without aggregated instances of `dot:DamageElement` for a simpler damage recording, the method as shown in Listing 5 should be preferred. This method operates on two detail levels and links the simplified representation of a damaged area with its more detailed aggregated single damage representations.

**Listing 5:** Assigning a damaged area with aggregated damages to a component

```
# asserted triples
inst:frame1-beam1 dot:hasDamageArea inst:damageArea3 .
inst:damageArea3 dot:aggregatesDamageElement inst:damageElement3-1,
                                              inst:damageElement3-2 .

# inferred triples
inst:damageArea3 a dot:DamageArea , dot:Damage .
inst:damageElement3-1 a dot:DamageElement , dot:Damage .
inst:damageElement3-2 a dot:DamageElement , dot:Damage .
inst:frame1-beam1 dot:hasDamageElement inst:damageElement3-1 ,
                                       inst:damageElement3-2 .
inst:frame1-beam1 dot:hasDamage inst:damageElement3-1 ,
                               inst:damageElement3-2 .
```

*Method 4:* Instances of `dot:DamageArea` define damaged areas on the lowest detail level. Therefore, instances of `dot:DamageArea` should not be aggregated. To group damages at a higher detail level, an instance of `dot:DamagePattern` should be aggregated in a `dot:DamageArea` instance that defines a group of physically connected damages (see Listing 6). Instances of `dot:DamagePattern` should only be aggregated in instances of `dot:DamageArea`. Thus, they should not be directly linked to affected components.

**Listing 6:** Assigning a damaged area with aggregated damage pattern to a component

```
# asserted triples
inst:frame1-beam1 dot:hasDamageArea inst:damageArea4 .
inst:damageArea4 dot:aggregatesDamagePattern inst:damagePattern4 .

# inferred triples
inst:damageArea4 a dot:DamageArea , dot:Damage .
inst:damagePattern4 a dot:DamagePattern .
inst:frame1-beam1 dot:hasDamage inst:damageArea4 .
```

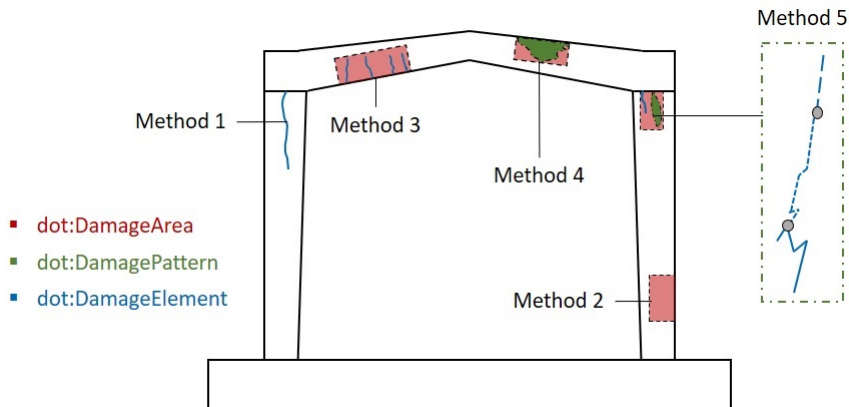
*Method 5:* When defining an instance of `dot:DamagePattern` it is recommended to group instances of `dot:DamageElement` with it as shown in Listing 7. Instances of `dot:DamageElement` that are adjacent to each other can be inferred as aggregated by the same instance of `dot:DamagePattern`. Additionally all grouped instances of `dot:DamageElement` can be inferred as aggregated to the corresponding instance of `dot:DamageArea`.

**Listing 7:** Damage Pattern grouping multiple damages aggregated in a damaged area

```

# asserted triples
inst:frameA-column2 dot:hasDamageArea inst:damageArea5 .
inst:damageArea5 dot:aggregatesDamagePattern inst:damagePattern5 .
inst:damagePattern5 dot:groupsDamageElement inst:damageElement5-1 .
inst:damageElement5-1 dot:adjacentDamageElement inst:damageElement5-2 ,
                                                    inst:damageElement5-3 .

# inferred triples
inst:damageArea5 a dot:DamageArea , dot:Damage .
inst:damagePattern5 a dot:DamagePattern .
inst:damageElement5-2 a dot:DamageElement , dot:Damage .
inst:damageElement5-3 a dot:DamageElement , dot:Damage .
inst:damageElement5-2 dot:adjacentDamageElement inst:damageElement5-1 .
inst:damageElement5-3 dot:adjacentDamageElement inst:damageElement5-1 .
inst:damageArea5 dot:aggregatesDamageElement inst:damageElement5-1 ,
                                                    inst:damageElement5-2 , inst:damageElement5-3 .
inst:frameA-column2 dot:hasDamageElement inst:damageElement5-1 ,
                                                    inst:damageElement5-2 , inst:damageElement5-3 .
inst:frameA-column2 dot:hasDamage inst:damageElement5-1 ,
                                                    inst:damageElement5-2 , inst:damageElement5-3 .
    
```



**Fig. 2:** A structural frame with examples of the five methods to define damages in DOT

#### 4.2 Queries for DOT and extensions

Based on the available properties of DOT and its extensions, various queries can be applied on digital representations of damaged constructions that enhance the inspection and assessment process.

*Query 1:* An essential part for the damage assessment of a construction is obtaining

information about all damaged construction components. Therefore, the query in Listing 8 selects all components that are affected by an instance of `dot:DamageArea` or `dot:DamageElement`. With an active reasoner, additional instances of `dot:DamageElement` that are aggregated in a `dot:DamageArea` will be returned, as they are now inferred to be directly connected to a damaged construction component. If available, the query will also return every instance of `dot:DamagePattern` that is aggregated by the `dot:DamageArea` or that is grouping individual `dot:DamageElement` instances. If this query would be used without reasoning, it would not return `dot:DamageElement` instances that are defined using modeling method 3 and 5.

**Listing 8:** SELECT all damaged construction components (with reasoning)

```
SELECT ?component ?hasDamage ?damage ?damagePattern
WHERE {
  ?component ?hasDamage ?damage .
  FILTER (?hasDamage IN (dot:hasDamageArea , dot:hasDamageElement)) .
  OPTIONAL { ?damagePattern dot:groupsDamageElement ?damage }
  OPTIONAL { ?damage dot:aggregatesDamagePattern ?damagePattern }
}
```

*Query 2:* Defects such as surface damages or damaged coatings have almost no impact on the structural capability of a construction and thus do not need to be considered during a structural analysis. To filter for damages that influence the structural capability, the query in Listing 9 selects all components that are affected by instances of `dot:StructuralDamage`. By using reasoning, aggregated `dot:DamageElement` instances related to instances of `dot:DamageArea` or `dot:DamagePattern`, are also evaluated.

**Listing 9:** SELECT all structurally damaged construction components (with reasoning)

```
SELECT ?component ?hasDamage ?structuralDamage
WHERE {
  {
    # dot:DamageElement or dot:DamageArea
    ?component ?hasDamage ?structuralDamage .
    ?structuralDamage rdf:type dot:StructuralDamage .
    FILTER (?hasDamage IN (dot:hasDamageArea , dot:hasDamageElement))
  } UNION {
    # dot:DamagePattern
    ?component dot:hasDamageArea ?damageArea .
    ?damageArea ?hasDamage ?structuralDamage .
    FILTER (?hasDamage = dot:aggregatesDamagePattern)
    ?structuralDamage rdf:type dot:StructuralDamage .
  }
}
```

*Query 3:* During inspections it is possible that a recorded damage could not immediately be classified as a defect or a structural damage. Throughout the subsequent damage assessment these undefined damage representations and their affected components need to be selected for a retrospective classification. A certified engineer can then add the correct class to these undefined damage instances. Therefore, the query in Listing 10 selects all components that are not affected by a known `dot:StructuralDamage`

instance and affected by at least one instance of `dot:DamageElement`, `dot:DamageArea` or `dot:DamagePattern` that is not also an instance of `dot:Defect`.

**Listing 10:** SELECT all components with an undefined damage (with reasoning)

```
SELECT ?component ?hasDamage ?undefinedDamage
WHERE {
  # component should have at least one dot:DamageArea, dot:DamageElement
  # or dot:DamagePattern that is not of class dot:Defect
  {
    ?component ?hasDamage ?undefinedDamage.
    FILTER (?hasDamage IN (dot:hasDamageArea , dot:hasDamageElement))
  } UNION {
    ?component dot:hasDamageArea ?damageArea1 .
    ?damageArea1 ?hasDamage ?undefinedDamage .
    FILTER (?hasDamage = dot:aggregatesDamagePattern)
  }
  FILTER NOT EXISTS { ?undefinedDamage a dot:Defect }
  # component has no damage individual that is of class
  # dot:StructuralDamage
  FILTER NOT EXISTS {
    {
      # dot:DamageElement or dot:DamageArea
      ?component dot:hasDamageArea|dot:hasDamageElement ?structuralDamage.
    } UNION {
      # dot:DamagePattern
      ?component dot:hasDamageArea ?damageArea .
      ?damageArea dot:aggregatesDamagePattern ?structuralDamage .
    }
    ?structuralDamage a dot:StructuralDamage .
  }
}
```

*Query 4:* To use queries referring to the taxonomy of damage, such as cracks in concrete, an extension compatible with DOT is required. Therefore, the CDO extension has been developed that defines a taxonomy for various damage types in reinforced concrete and additional properties to describe them (e.g. crack width or spalling area). By utilizing these extensions, more specific queries could be created, such as the query shown in Listing 11, where all cracks (instances of `cdo:Crack`) are selected that are aggregated in a specific instance of `dot:DamageArea` and that have a crack width of more than 3 mm. The query also returns the specific CDO crack type that was asserted.

**Listing 11:** SELECT all cracks wider than 3 mm (with reasoning)

```
SELECT ?damageElement ?crackType ?width
WHERE {
  BIND(inst:D1_6 AS ?damageArea)
  ?damageArea dot:aggregatesDamageElement ?damageElement .
  ?damageElement rdf:type ?crackType.
  GRAPH <http://ontologies.org/dot/>{
    ?crackType rdfs:subClassOf cdo:Crack .
  }
```

```

}
FILTER (?crackType != cdo:Crack)
?damageElement cdo:crackWidth ?width.
FILTER (?width > 3)
}

```

## 5 Conclusions and Future Work

In this research, a web ontology named DOT for defining digital damage representations and their topology is proposed. Besides topological terminology, DOT contains classes for documentation management and structural assessment. Using DOT requires a separate web ontology for representing the affected construction and its components (e.g. BOT or BROT). Furthermore, it is recommended to use DOT as a modular core ontology in combination with additional web ontologies as extensions when adding more specific information, e.g. taxonomies for various damage types or national assessment standards. In this regard, three example ontologies have been developed that extend DOT with a taxonomy for damages in reinforced concrete (CDO), mechanical parameters for damaged areas (DMO) and properties based on the German inspection standard DIN 1076 [1] (DASB). DOT damage instances can be linked to geometrical representations via specialized ontologies such as OMG-FOG or by using the ICDD approach for linking them to geometrical files and other documents via RDF linksets in a unified data container. Future research will focus, how existing ontologies such as MONDIS for defining damage causes, can be utilized for a combined use with DOT. Another important issue is the versioning of damages and how they evolve over time. Since damage representations from earlier inspections must not be removed to derive the damage progression, an ontology for version control of damages, similar as the Ontology for Property Management (OPM) [11] does for properties and OMG for geometries, should be developed. It should also be possible to link damages with repair and maintenance actions.

## 6 Acknowledgments

This research work was enabled by the support of the Federal Ministry of Education and Research of Germany through the funding of the projects wiSIB (project number 01—S16031C).

## References

1. 1076:1999-11, D.: Engineering structures in the course of roads and paths - monitoring and testing (1999), [in German, original title: Ingenieurbauwerke im Zuge von Straßen und Wegen - Überwachung und Prüfung]
2. Bonduel, M., Rasmussen, M.H., Pauwels, P., Vergauwen, M., Klein, R.: A novel workflow to combine BIM and Linked Data for existing buildings. In: 12th European Conference on Product and Process Modelling (ECPPM). Copenhagen, Denmark (2018)
3. Bonduel, M., Wagner, A., Pauwels, P., Vergauwen, M., Klein, R.: Including Widespread Geometry Formats in Semantic Graphs Using RDF Literals. In: Proceedings of the European Conference on Computing in Construction (EC3 2019). Chania, Greece (In Press)

4. Cacciotti, R., Blaško, M., Valach, J.: A diagnostic ontological model for damages to historical constructions. *Journal of Cultural Heritage* **16**, 40–48 (2015). <https://doi.org/10.1016/j.culher.2014.02.002>
5. Gornig, M.: Structural data on production and employment in construction - Calculations for 2017 (2018), [in German, original title: Strukturdaten zur Produktion und Beschäftigung im Baugewerbe - Berechnungen für das Jahr 2017]
6. Hammad, A., Zhang, C., Hu, Y., Mozaffari, E.: Mobile Model-Based Bridge Lifecycle Management System. *Computer-Aided Civil and Infrastructure Engineering* **21**(7), 530–547 (oct 2006). <https://doi.org/10.1111/j.1467-8667.2006.00456.x>
7. Iso/np 21597: Information container for data drop (icdd). Standard, International Organization for Standardization (2017)
8. Kozak, T.: Development of a method to transform semantic data of bridge models from IFC format into an ontology. Master's thesis, Technische Universität Dresden, Germany (2019, unpublished)
9. Lee, D.Y., Lin Chi, H., Wang, J., Wang, X., Park, C.S.: A linked data system framework for sharing construction defect information using ontologies and BIM environments. *Automation in Construction* **68**(May), 102–113 (2016). <https://doi.org/10.1016/j.autcon.2016.05.003>
10. Pauwels, P., Terkaj, W.: EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction* **63**, 100–133 (mar 2016). <https://doi.org/10.1016/J.AUTCON.2015.12.003>, <https://www.sciencedirect.com/science/article/pii/S0926580515002435>
11. Rasmussen, M.H., Lefrançois, M., Bonduel, M., Hviid, C.A., Karlshøj, J.: OPM: An ontology for describing properties that evolve over time. In: 6th Linked Data in Architecture and Construction Workshop (LDAC), CEUR Workshop Proceedings. London, UK (2018)
12. Rasmussen, M.H., Pauwels, P., Hviid, C.A., Karlshøj, J.: Proposing a Central AEC Ontology That Allows for Domain Specific Extensions. In: LC3 2017: Volume I - Proceedings of the Joint Conference on Computing in Construction (JC3). pp. 237–244. Heriot-Watt University, Heraklion, Greece (July 2017). <https://doi.org/10.24928/JC3-2017/0153>
13. Sacks, R., Kedar, A., Borrmann, A., Ma, L., Brilakis, I., Hüthwohl, P., Daum, S., Kattel, U., Yosef, R., Liebich, T., Barutcu, B.E., Muhic, S.: SeeBridge as next generation bridge inspection: Overview, Information Delivery Manual and Model View Definition. *Automation in Construction* **90**, 134–145 (jun 2018). <https://doi.org/10.1016/J.AUTCON.2018.02.033>
14. Scherer, R.J., Katranuschkov, P.: BIMification: How to create and use BIM for retrofitting. *Advanced Engineering Informatics* **38**, 54–66 (2018). <https://doi.org/10.1016/j.aei.2018.05.007>
15. Tanaka, F., Tsuchida, M., Onosato, M., Date, H., Kanai, S., Hada, Y., Nakao, M., Kobayashi, H., Hasegawa, E., Sugawara, T., Oyama, T.: Bridge Information Modeling based on IFC for supporting maintenance management of existing bridges. Tampere (2018)
16. Volk, R., Stengel, J., Schultmann, F.: Building Information Modeling (BIM) for existing buildings — Literature review and future needs. *Automation in Construction* **38**, 109–127 (2014). <https://doi.org/10.1016/j.autcon.2013.10.023>
17. Wagner, A., Bonduel, M., Pauwels, P., Rüppel, U.: Relating Geometry Descriptions to its Derivatives on the Web. In: Proceedings of the European Conference on Computing in Construction (EC3 2019). Chania, Greece (In Press)