Object-Oriented Damage Information Modeling Concepts and Implementation for Bridge Inspection

Mathias Artus[1] and Christian Koch[2]

Abstract: Bridges are designed to last for more than 50 years and consume up to 50% of their life-cycle costs during their operation phase. Several inspections and assessment actions are executed during this period. Bridge and damage information must be gathered, digitized, and exchanged between different stakeholders. Currently, the inspection and assessment practices rely on paper-based data collection and exchange, which is time-consuming and error-prone, and leads to loss of information. Storing and exchanging damage and building information in a digital format may lower costs and errors during inspection and assessment and support future needs, for example, immediate simulations regarding performance assessment, automated maintenance planning, and mixed reality inspections. This study focused on the concept for modeling damage information to support bridge reviews and structural analysis. Starting from the definition of multiple use cases and related requirements, the data model for damage information is defined independently from the subsequent implementation. In the next step, the implementation via an established standard is explained. Functional tests aim to identify problems in the concept and implementation. To show the capability of the final model, two example use cases are illustrated: the inspection review of the entire bridge and a finite-element analysis of a single component. Main results are the definition of necessary damage data, an object-oriented damage model, which supports multiple use cases, and the implementation of the model in a standard. Furthermore, the tests have shown that the standard is suitable to deliver damage information; however, several software programs lack proper implementation of the standard. **DOI: 10.1061/(ASCE)CP.1943-5487.0001030.**

Introduction

Bridges are designed to last for more than 50 years. During this period they consume up to 50% of their life-cycle costs (Schach et al. 2006; Nagel et al. 2016). Conventional inspection processes, which are essential to ensure the safety and serviceability of bridges throughout their lifetime, include manual visual inspections for the damage data acquisition. An inspector or an engineer performs the visual inspection on-site. All defects and related data are recorded in paper-based reports and the inspector subsequently digitizes the data in the office (Hearn 2007; Hurt and Schrock 2016). Other stakeholders, such as structural engineers, retrieve these data as paper-based reports, as exports of databases, or in other proprietary data formats. Again, engineers integrate or import these data into their digital models. Such repeating manual data digitization leads to information loss (Eastman 2011; Borrmann et al. 2018). The concept of building information modeling (BIM) has been designed to overcome this issue.

BIM has been designed to cover the entire life cycle of structures including the operation phase with inspection and maintenance (Borrmann et al. 2018). Defects and related information are vital for the inspection process. However, Sacks et al. (2018a, p. 144) remarked that "There is currently no accepted, consistent or thorough way to represent the defects that may occur in bridges." This leads to the conclusion that BIM requires an extension to support the inspection process.

Hereinafter, the concept of modeling defects and related information will be called damage information modeling (DIM) and the associated model as the damage information model. The aim of this study was to develop, implement, and test a DIM model to exchange damage data between inspectors, owners, and structural engineers. For illustration, two example use cases have been chosen:

1. Transfer inspection data from the inspector to the engineer for three-dimensional (3D) structural simulation; and
2. Transfer inspection data from the inspector to the engineers and owners for assessment.

Fig. 1 illustrates the conceptual idea of DIM. Photos of a bridge with its damaged components are depicted in Figs. 1(a–c). Fig. 1(d) depicts an object-oriented model of this bridge with components and defects. Fig. 1(b) shows a photo of corrosion at the railing. This corrosion is modeled in the right branch of Fig. 1(d). The left branch shows the crack at the pier, which is shown in Fig. 1(c). The vision for future inspections would be to automate data transfer from data acquisition to simulation and define the bridge condition in accordance with simulation results. This is anticipated to make assessments more objective and precise.

The remainder of this paper is organized as follows: The "Background" section discusses the extant literature concerning inspection practices and scientific achievements on damage information modeling. According to the workflow of object-oriented analysis and design by Booch (2007), the requirement analysis for damage information is performed in the "Requirement Analysis" section. Next, the "Object-Oriented Design" section explains the design of the object-oriented data model, followed by the explanation of the implementation in the "Object-Oriented Implementation" section. The "Testing of the Implementation" section explains and analyzes functional tests and their results. The "Use Cases and

[1]Faculty of Civil Engineering, Bauhaus-Univ. Weimar, Marienstraße 13a, 99423 Weimar, Germany (corresponding author). ORCID: https://orcid.org/0000-0001-9995-8670. Email: mathias.artus@uni-weimar.de

[2]Faculty of Civil Engineering, Bauhaus-Univ. Weimar, Marienstraße 13a, 99423 Weimar, Germany. ORCID: https://orcid.org/0000-0002-6170-7611. Email: c.koch@uni-weimar.de

Note. This manuscript was submitted on June 24, 2021; approved on February 27, 2022; published online on July 27, 2022. Discussion period open until December 27, 2022; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, © ASCE, ISSN 0887-3801.

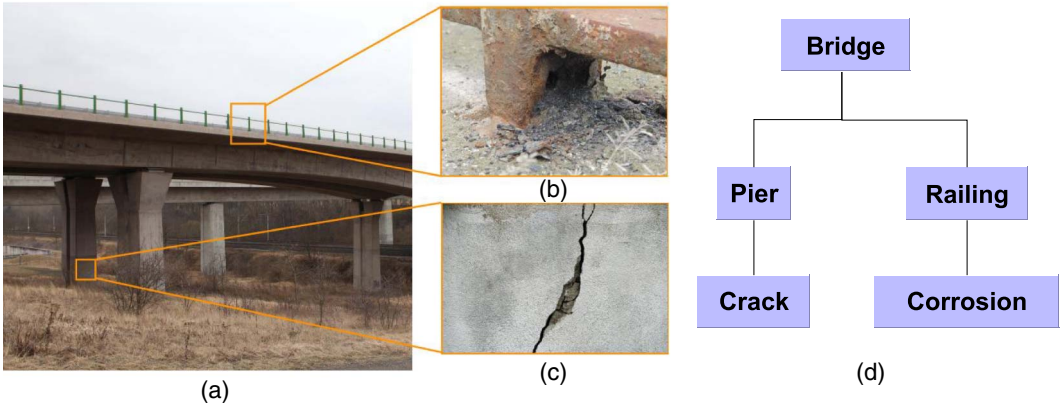


Fig. 1. Conceptual graphic: (a) a bridge with defects and affected bridge elements (image courtesy of Wikimedia Commons/Störfix); (b) an instance of corrosion (image by authors); (c) a crack (image courtesy of Pixabay/struppi0601); and (d) an object-oriented model of the bridge, its elements, and its related defects.

Examples" section illustrates the potentials of the data model in the inspection process. A succinct summary of the work is discussed in the "Summary" section, and finally a "Conclusion and Outlook" are provided.

Background

Hitherto, a number of studies concerning digitizing inspection and maintenance information have been published. This section discusses the state of practice, theoretical background of damage data models, and gaps in knowledge.

BIM and Damage Information

Inspection Process

Fig. 2 summarizes the workflow for bridge inspections and assessments in accordance with the studies of Hearn (2007), Hurt and Schrock (2016), and Sacks et al. (2018b). The raw data acquisition, which encompasses visual inspection and additional inspections, is the first step in the process. Visual inspection refers to the inspection of a bridge by one or more engineers. During this inspection, the engineers look at the bridge, tap critical spots, or measure defects and components. The collected information is stored as text, alphanumeric values, enumerations, and photos. Additional inspections are included if defects below the surface are assumed. Examples of such surveys are ground-penetrating radar, X-ray, or ultrasonic surveys. These surveys deliver reports, photos, and 3D point clouds. All acquired data are stored and subsequently used for simulations and the final assessment.

Simulations and calculations follow the raw data acquisition. This includes simulations for structural analysis, for durability analyses, or for the safety of the transport system. Therefore, defect information must be integrated into these simulation models. All outputs of the raw data acquisition and simulation results are fed back to the DIM for later assessment. Finally, the assessment of the structure uses the bridge data and the data from the DIM and assesses the bridge.

BIM for Infrastructure

The entire bridge inspection process is a manual paper-based procedure. To digitize this process, digital bridge models are necessary. BIM is a well-developed concept for digital planning, design, and construction management for buildings and construction processes (Borrmann et al. 2018). One of the key points is the digital data exchange. buildingSMART has developed the Industry Foundation Classes (IFC) as an open, neutral, and standardized data-exchange format (ISO 2018) to address the problem of data exchange in the BIM process. Unfortunately, IFC 4, the latest standardized version, and below do not contain all necessary entities to properly exchange information about bridges. Working on this issue, buildingSMART finished the IFC Bridge project in June 2018 (IFC Infra 2019a). As a result, the draft of IFC 4 × 2, which includes IFC Bridge, has been available since 2018 (buildingSMART International Ltd. 2018a) but has not reached the status of a standard. IFC 4 × 2 has been withdrawn and the work regarding the infrastructure extension of IFC is still ongoing (buildingSMART International Ltd. 2020).

Although there is the prospect of exchanging bridge data between the different stakeholders with the help of IFC, 78% of bridges were built before 1990. For these bridges, no BIM model exists, which is a prerequisite for the application of DIM. A number of studies have been conducted in the domain of 3D model generation of existing buildings via point clouds (Volk et al. 2014; Barazzetti et al. 2016; Kropp et al. 2018). In accordance with such as-built models, damage information could be added to extend the use of BIM for inspection and assessment.

Damage Information Modeling

To introduce a standardized damage data exchange, Sacks et al. (2018a) published an Information Delivery Manual (IDM) for data exchange during inspection and assessment. The manual provides an overview of the designed inspection process called SeeBridge. The SeeBridge inspection process extends the conservative inspection process, which was described previously. The following tasks are part of the process: generation of the bridge model, damage acquisition, point cloud generation, defect detection, and the calculation procedure for performance indexes.

Based on processes and tasks, an IDM must define the data that have to be exchanged. A basic requirement is that a damaged component must be identifiable as a damaged component. This can be achieved by linking defects and related components. The aforementioned IDM (Sacks et al. 2018a) includes this information. Further details about damage relationships, for instance, aggregation of defects, are not included in the study of Sacks et al (2018a). Hamdan and Scherer (2018) used damage element and individual damage to aggregate defects or defect parts. Causes and effects of relationships are required in the model (Lee et al. 2016). If a defect over time is modeled using individual defect entities, a relationship between these entities is necessary to represent their unity (Tanaka et al. 2018). Relationships of defects were part of numerous prior studies. The present study aims to include all of the mentioned relationships.

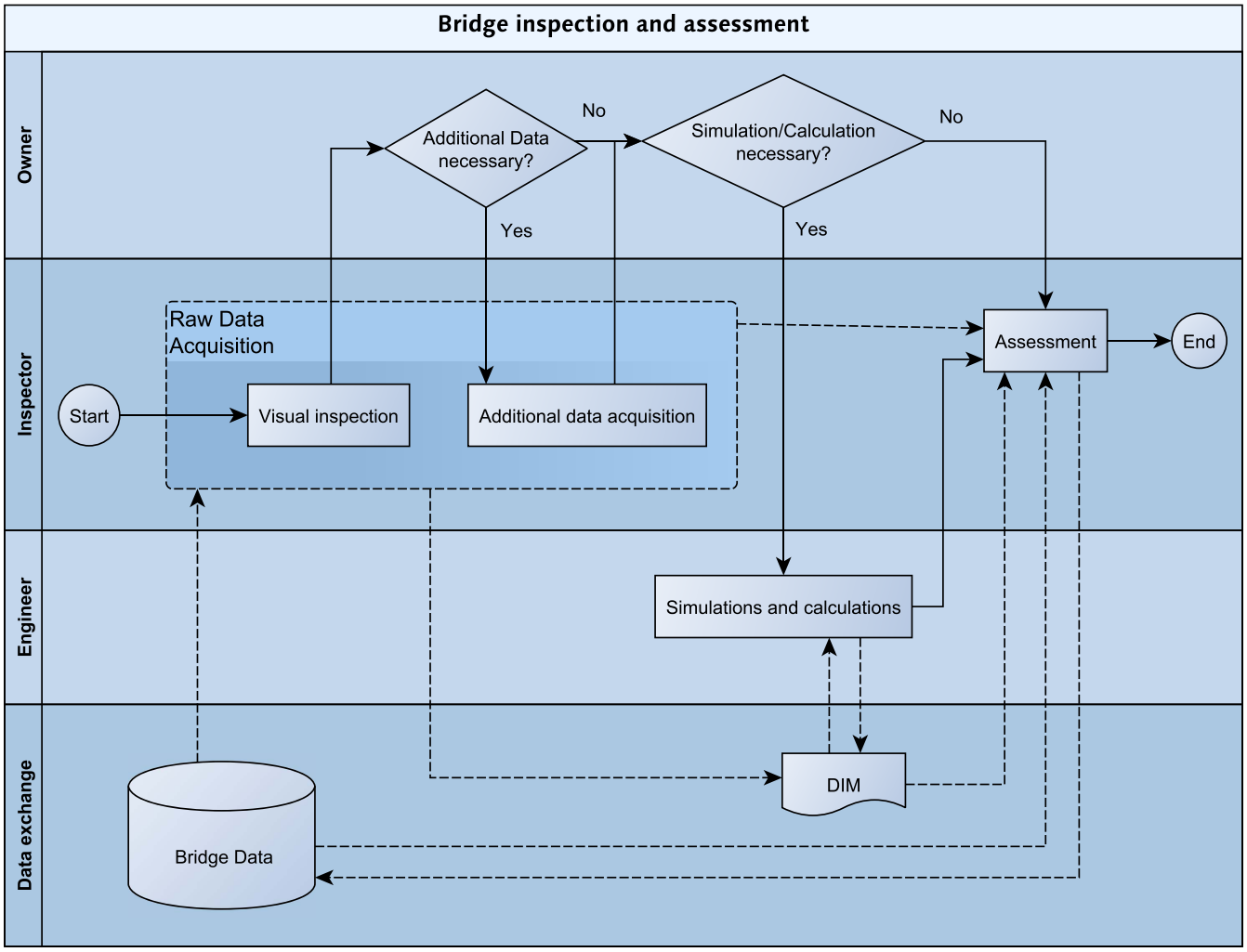


Fig. 2. Entire workflow from inspection to condition rating. Dashed lines show data exchange. Consecutive processes are connected by solid lines.

A single defect could be of different types, such as spalling, crack, or corrosion. Sacks et al. (2018a) included spalling, crack, rust staining, efflorescence, scaling, and abrasion or wear in their data model. As reported by Artus and Koch (2020), there are more defects, such as divergences from specification or design, further material changes, and joint damages. This shows that not all damage types have been covered heretofore. However, Artus and Koch (2020) also asserted that cracks are the most important. Hence, the present study focused on physical defects, namely, cracks and spalling.

Processing damage data requires semantic and geometric data. First, there are textual descriptions of the defect itself or environmental conditions during inspection. Such information is part of all national bridge management systems (Hearn 2007) and has been focused on in the SeeBridge IDM (Sacks et al. 2018a). This includes measurements, for example, crack width and orientation, and textual information, such as alphanumeric material descriptions. Wan et al. (2019) also included parameters for crack length, direction, and width in their data model. Liu and El-Gohary (2016) included bridge deficiency to represent measurements during a bridge inspection. A common approach is to save measurements in designated objects and link those objects to the defect (Hamdan and Scherer 2018; Hüthwohl et al. 2018). This has the advantage that new measurements may be added easily either if a consecutive inspection has taken place or if newer guidelines require additional measurements.

National bridge assessment practices rely on visual data, such as photos and sketches (Hearn 2007). In most cases, several pictures are taken and stored. Hüthwohl et al. (2018) included images as textures in their data model to provide additional visual information. Future inspection practice may utilize unmanned aerial systems (UASs), as shown by Morgenthal et al. (2019), so multiple photos could be available for a single defect. All relevant photos of a single defect should be included and/or referenced in the as-damaged model. However, multiple photos cannot be used as a single texture patch in parallel. Hence, a new approach is necessary that combines storing several photos and providing the visual information of a texture.

Three-dimensional geometries of defects and damaged components are necessary for the visualization and calculations and simulations. Visualizing the geometry of defects or damaged components enables engineers to observe crack width, length, orientation, and position at a glance. Simulations, for example, finite-element analyses (FEAs), benefit from 3D geometries for the generation of the damaged structure. For conventional FEA with beams, slabs, and plates, the 3D geometry of the damaged building is imported into the simulation environment and the structural model may be generated with the 3D geometry in the background. This eases definition of positions and scaling correctly. In case of FEA with volume elements, the geometry of the bridge or a single bridge component is imported and meshed for the subsequent calculations. McGuire et al. (2016)

used defect bounding boxes to generate the two-dimensional (2D) structural model of a damaged girder. A similar approach was followed by Fröhlich (2020). Barazzetti et al. (2015) performed a 3D FEA by generating a BIM model from the point cloud along with additional information. Further information, for example, material information and component connections, was transferred manually using the BIM model. Summarizing 3D geometries of defects is beneficial for visualization, calculations, and simulations.

Hamdan and Scherer (2018) included the 3D geometry in their data model by linking different models together. A noteworthy drawback of this model is the distribution of the data between different models. The entire data need to be summed up into one exchangeable format to send it to or share it with other stakeholders. This requirement could be omitted by combining building and damage information in the same model. Tanaka et al. (2018) extended the IFC with eight classes to included defect geometry with regard to inspection processes. The existing IFC standard implements a diversified tool set for modeling buildings and structures and offering possibilities for special uses. To keep the complexity of IFC as low as possible, new entities should be added only if necessary and economical. This leads to the question if the goal of exchanging damage data could be done with only a few modifications in an existing standard. Isailovic et al. (2020) proposed a framework for damage acquisition and assessment that calculates defect geometries on the basis of photos and generates an IFC file with damage information. Their approach omitted IFS extensions; however, this model lacks time relations and visual data to properly support the assessment task.

Damage Model Implementation Strategy

Five implementation methods to extend the BIM concept are described in the literature. First, the BIM Collaboration Format (BCF) was designed for sharing change requirements of building models during design, planning, and construction (buildingSMART International Ltd. 2009). BCF files are additional files besides the building information and contain one or more issues with a camera position, photos, and descriptions. These files do not contain building information and hence are not capable of providing damage data, such as geometries or properties of a defect.

Another possibility is to use proprietary and custom data formats, e.g., Autodesk Revit or Microsoft Excel files (McGuire et al. 2016). The most noteworthy advantages of proprietary data formats are the availability of existing tools and hence less effort for software implementation. However, those concepts are not capable of storing all information related to defects, for example, Excel is not made for storing geometry data.

In addition to the mentioned proprietary data formats, open data formats are available, for example, the IFC (Borrmann et al. 2018). IFC is a comprehensive data format for BIM and provides numerous possibilities for modeling semantics and geometries. Additionally, several authoring tools allow exports to IFC and numerous IFC viewers exist. Hence, less implementation efforts are required if a damage-loaded BIM could be modeled by means of IFC. Although the IFC standard is extensive, it lacks specific relationships and entities for defects and damaged components. To overcome this gap, extensions of the IFC have been developed, for example, by Tanaka et al. (2018). The biggest disadvantage of such extensions is that extending the IFC requires further implementation work to read, write, visualize, and edit the resultant data model.

Linked data models incorporate IFC data and associate data via links to other models or model types. One example is the linking of IFC data with semantic web ontologies. All building information is stored in an IFC file and damage information is stored in a separate damage model. Both are linked via a link model (Hamdan and Scherer 2018; Hamdan et al. 2019). On the one hand, a linked data or multimodel concept is highly flexible and semantic web allows defining custom relationships (W3C 2012). On the other hand, distributing data over two models leads to data dispersion and again the necessity of implementing your own viewers and authoring tools arises.

Several open-source IFC viewers exist, such as the Java IFC Toolkit or the xBIM Xplorer. These viewers and related libraries cope with reading, writing, visualizing, and editing IFC files. Extensions to these software tools require less effort compared to developing software from scratch. This leads to the decision to implement the conceptual model by using the IFC 4 standard (buildingSMART International Ltd. 2016).

Problem Statement and Objective

Current and future inspection practices require a suitable DIM with geometric, visual, and semantic damage data. Heretofore, several DIMs with regard to visualization and semantic information have been developed. However, each focuses on a specific aspect, such as visualization or simulation. Some models include semantic data and some include visual data while others include geometries. This study focuses on a comprehensive DIM that incorporates all three aspects and is usable for multiple use cases. Simply combining existing approaches would not be possible because this would lead to redundant information in the model, for example, the different approaches of modeling defect geometries by Isailovic et al. (2020) and Hamdan and Scherer (2018). Instead, a requirement analysis under the consideration of the existing work is necessary to identify the model requirements and address them in the data model. This incorporates synergizing existing models and concepts and adding new aspects, such as multiple defect representations and mapping information.

The objectives of this study were to address the following research questions:

- Which data are necessary to deliver damage information for assessment and simulation?
- How can an object-oriented model independent of software tools or data formats be designed?
- What changes and extensions are required to existing models?
- How can this object-oriented model be implemented using an established architecture engineering and construction (AEC) data format?
- Can the object-oriented model and its implementation be verified using available AEC software tools?

Requirement Analysis

The effectiveness of a model depends on the fulfillment of model requirements. Hence, the requirements for a DIM are analyzed in this section. According to Borrmann et al. (2018), a BIM model consists of semantic and geometric data. This definition has been extended for damage data because visual data, such as photos, play a prime role in the inspection process. In accordance with the definition of the data categories, the different requirements are explained in the following subsections. Table 1 encapsulates all data categories and the content of these data.

Semantic Data

Semantic data refer to data that describe the meaning or content of a defect. Table 1 gives the content of the semantic data. The first semantic information is that there is a defect or damaged component.

Table 1. Data categories and inherited data

Data type	Data category	Data content
Semantic data	Relationships	Affected component(s), defect cause, defect-related documents, parts of a defect
	Classification	Damage type
	Defect properties	Textual descriptions, condition rating, measurements, implications
Visual data	—	Photos, sketches, videos
Geometry data	—	One-dimensional geometry (e.g., defect position coordinates), 2D geometry (e.g., polylines), 3D geometry (e.g., point clouds, meshes, constructive solid geometry, BREP)

Note: BREP = boundary representation.

Hearn (2007) showed in his comparison of national inspection practices that nations acquire either defect- or component-centered data for bridge assessment. Using a defect entity enables supporting both assessment practices. A component with a related defect may be easily abstracted to a damaged component. The other way around would be considerably more difficult. Hence, a class for an individual defect is necessary. Such a defect object requires an identifier, name, and description to cope with textual information.

Bridge assessment relies on defect relationships. The first relationship is with the damaged component. Second, a defect may consist of several part defects, for example, a crack map consists of several cracks. Third, a defect can have another defect as a reason; for example, a spalling is the reason behind an exposed and corroded rebar. Last, a defect has associated documents, like reports.

The entire data of a defect require a relation to time because inspections are snapshots in time. The defect entity should stay the same because only the parameters of the defect vary with time and not the defect itself. A special case could be if two defects become one, such as two cracks growing and becoming one crack. It can be concluded that the defect entity persists over time. In contrast, the associated data represent a snapshot.

Different defect types have different effects, which gives rise to a variation in assessment results. Artus and Koch (2020) defined 12 damage types. In general, different damage types, which may occur at bridges, need to be supported. Some examples are cracks, spalling, and corrosion. A damage of a specific type is represented by a damage type class and a relationship between the defect and the damage type. Finally, semantic data contain defect properties. Such properties may contain textual descriptions, implications, measurements, or condition ratings. These properties may be grouped together on the basis of their context, meaning, or time. This leads to using one class to group properties and one class for each property itself. A property must have a name and value. A description and unit are optional. All aforementioned semantic information may be used to automatically check for specified rules later (Ren et al. 2019). Furthermore, probabilistic simulations use semantic data for predicting bridge conditions (Calvert et al. 2020).

Visual Data

Hearn (2007) emphasized the importance of photos and sketches during inspection, which has necessitated their inclusion in the data model. Photos or sketches are stored in files, which are linked to the defect. Findings of state-of-the-art research have confirmed the benefit of using UAS for visual inspection (Morgenthal et al. 2019). Using video-capturing drones leads to a large number of photos or even video streams. Hence, a DIM must offer storing several photos of a defect or a video stream. This approach is similar to the aforementioned reports and hence the same approach is applicable. Finally, photos may be used as textures for depicting the defect's image at the correct model position. For this purpose, rectified photos are necessary. Furthermore, a texture-mapping algorithm or a mapping table is necessary to properly attach the image. Both need to be stored in case of using textures. If the texture should be depicted only at a specific part of a component geometry, this part needs to be marked.

Geometry Data

Geometry data consist of one-dimensional (1D), 2D, and 3D geometry data. One-dimensional geometry denotes the defect position. The position is important for subsequent assessment; for example, cracks in the deck above the expansion joint could indicate a too small or missing expansion joint.

Two-dimensional geometry data could be poly lines or plane sketches. Instead of adding photos of sketches, inspectors could add those sketches to the model, which would be machine-readable. Sketches are relevant in case of bad illumination because inspectors create sketches to provide additional geometric information for the assessment afterward. Furthermore, on-site inspections primarily rely on paper-based plans. To support this use case, a model of a damaged bridge should contain plans in addition to the 3D model. This leads to the point where a defect has multiple geometries with different contexts.

Automated damage detection systems generate 3D damage geometry data. UASs capture photos or videos during their flight around the bridge. Structure-from-motion (SfM) algorithms are employed to generate point clouds and meshes. Within point clouds, further annotations are possible to accentuate damaged regions (Morgenthal et al. 2019). The points of a point cloud are used to generate meshes of a defect to retrieve solid geometries (Isailovic et al. 2020). Three-dimensional geometries provide damage-related information for structural bridge simulations and bridge assessments. Refer to the "Damage Information Modeling" section for explanations on how to use 3D geometry data for structural bridge simulations.

Requirement Summary

In summary, the following essential requirements must be addressed:

- An entity for a defect;
- Relationships that can represent damaged components, defect groups, defect causes and effects, and defect-related documents;
- Classifications for defects;
- Defect properties to store textual descriptions, condition ratings, measurements, and implications;
- Multiple photos, sketches, and videos need to be stored;
- 1D, 2D, and 3D defect geometry;
- Depict a photo as texture on the 3D model; and
- Multiple geometries have to be stored because a defect may have a 2D and 3D representation.

Object-Oriented Design

As per the damage information summarized in the "Requirement Analysis" section, this section describes two DIM variants independently from implementation strategies to model the information required. In the first step, the semantic data are modeled. In the following steps, the geometry and visual data are added. All class diagrams contain building elements visualizing the relationships between those building elements and affecting defects.

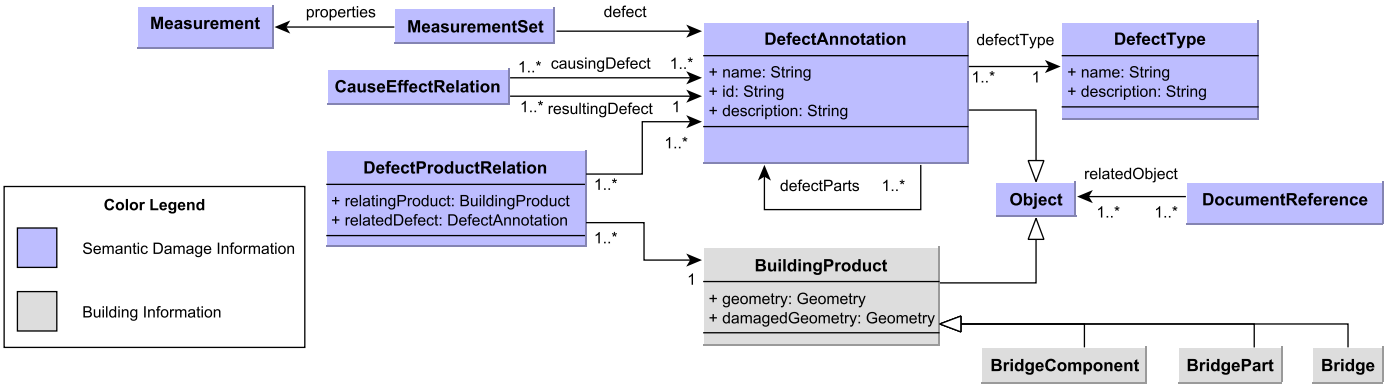


Fig. 3. UML class diagram for the model of semantic data.

Semantic Data

Hamdan and Scherer (2018) and Sacks et al. (2018a) covered semantic data in their studies in the form of defect relations and measurements. The concepts presented in this paper are based on their work in a sense that relationships and measurements are included. Fig. 3 shows the unified modeling language (UML) diagram for semantic defect data. Building products and subclasses represent bridge parts and semantic damage data are the annotation and relation. To address the target of a defect entity, a *DefectAnnotation* is defined, which consists of a name, ID, and description. Relationships are the next point in the list of requirements. The relationship between the defect and the building product is designed via an objectified relationship *DefectProductRelation*. This relationship has two attributes: relatingProduct points to the affected bridge component and relatedDefect points to the affecting defect. Such an objectified relationship has the advantage that it can contain additional information, for example, the relation type. The *CauseEffectRelation* offers the possibility to represent causing and resulting defects. The causingDefect points to the defect that is the reason for the resultingDefect. This relationship is an *m* to *n* relationship because a defect may have multiple causing defects and one defect may have several resulting defects. An objectified relationship has been used to cope with this requirement of an *m* to *n* relationship. Related documents may be referenced via the *DocumentReference* class. This points to a general object because the documents may be related to a specific defect, such as photos of a defect, or to a building product, such as an ultrasonic survey of a bridge component. Furthermore, the class contains an identifier, name, description, and uniform resource identifier (URI) about the referenced document.

To address the requirement of a defect type, the *DefectAnnotation* is provided with the defectType attribute, which is of type *DefectType*. The *DefectType* class has a name and description. Possible names could be crack, spalling, corrosion, or similar. The defect properties are added by the class *Measurement* and grouped via a *MeasurementSet*. A measurement contains a name, value, and an optional unit. In summary, this part of the model covers the requirements of modeling semantic data as shown by Table 1.

Visual Data

Hüthwohl et al. (2018) proposed defect representations using textures; however, their approach misses the requirement of including multiple photos and necessary texturing information. Addressing the requirement of storing multiple photos, they can be referenced via *DocumentReference*. Fig. 4 shows the use of a photo as texture. The *DefectAnnotation* has a *Texturing* with the texture area, mapping information, and URI. A texture is applied to the texture area by mapping the image from the URI with the given texture mapping. *TextureMappingAlgorithm* is a subclass of *TextureMapping* and represents generic algorithms to calculate texture mappings, like spherical texture mapping. *TextureGeometryMap* represents a point-based texture map also as subclass of *TextureMapping*. A *TextureArea* defines the location of a texture.

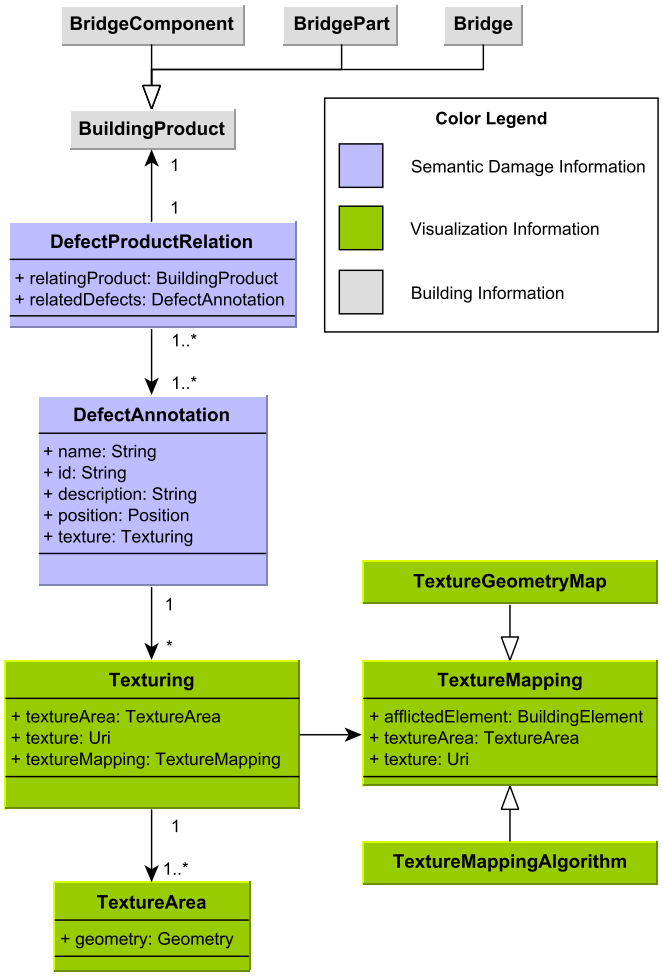


Fig. 4. UML class diagram of the information model with additional textures.

This may be the entire geometry of a defect or component or it is a part of those geometries.

Defect Geometries

Physical defects lead to cutouts in component geometries. Either this is derived from the relationship between the damage and the component, or the relationship between the component and the damage is modeled separately from the geometry of the damaged component. The advantage of interpreting the geometry on the basis of the relationship between the component and its defect is that fewer entities are necessary, which entails less risk for incoherence. However, including multiple geometries is defined as a requirement, but calculating the geometry on the basis of the relationship does not offer involving multiple geometries for a defect or damaged component. A distinction of relationships and geometries enables including multiple geometries of defects and damaged components. For the sake of completeness, both methods are modeled, implemented, and tested. These methods have in common that the defect and component have individual geometries; furthermore, the concept of constructive solid geometry (CSG) s used to calculate the geometry of a damaged component for both approaches.

Relationship-Based Geometry

Isailovic et al. (2020) showed the integration of damage geometry into BIM models. However, their implementation has the shortcoming of using a surface feature for subtraction, which contradicts with the definition of IFC 4. The concept proposed in this paper is inspired by their work. Fewer relationships and entities are required if relationships between defects and components lead to geometric interpretations. Fig. 5 illustrates this approach. *DefectProductRelation* has been replaced with a new relationship, *DamagedGeometryCutOut*, combines the relation between a defect and the affected component and involves geometric data. buildingElement points to the affected component and defect points to the affecting defect. To model a crack in a wall, buildingElement would point to that wall and defect would point to the crack. Both objects, i.e., the undamaged wall and the crack, have a geometry. The geometry of the crack is subtracted from the geometry of the wall to create the geometry of the damaged wall.

Independent Relationship and Geometry

The data model of Isailovic et al. (2020) has a second drawback: it does not allow multiple geometries for a single defect, which is required as explained in the "Requirement Analysis" section. Using the following concept, this requirement is fulfilled. Fig. 6(a) illustrates how to independently model the geometry and the relationship between defects and products. This approach uses the *DefectProductRelation* from Fig. 3 to represent the relationship between a defect and the affected component. The component and the defect have their individual 3D geometries. Fig. 6(b) shows the object diagram for this method. The defect has a geometry called *defectGeometry* and the damaged component has two geometries: the *componentGeometry* and the *damagedGeometry*. Cutting out the damage geometry from the component geometry results in the *damagedGeometry* of the component. Each geometry has a representation context to allow selecting the desired geometric representation.

Synergized Damage Model

Fig. 7 depicts the entire UML model including both geometry modeling approaches. The geometry may be included by the *DamagedGeometryCutout* relation or by the *DefectProductRelation* in combination with CSG geometries.

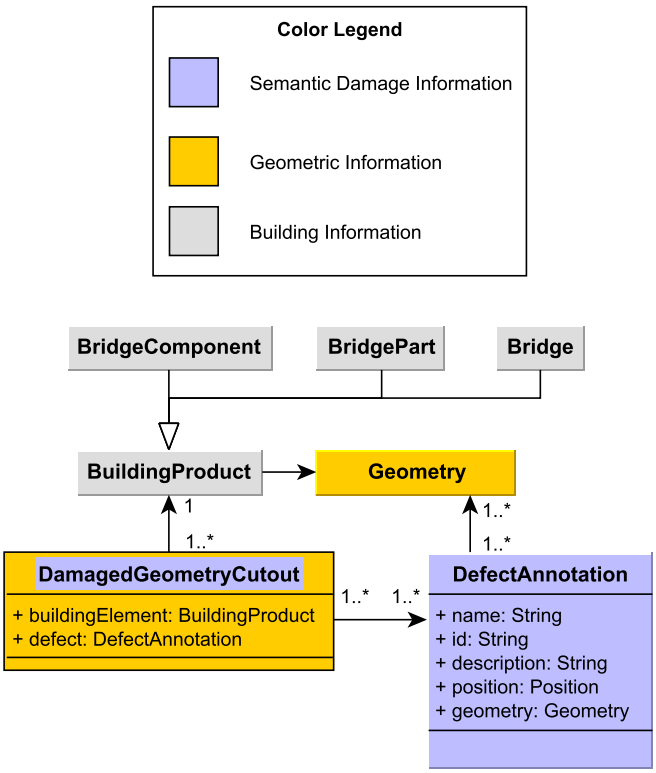


Fig. 5. Class diagram for modeling the damaged component geometry with the component geometry and the defect geometry.

Object-Oriented Implementation

This section explains the implementation of the object-oriented damage model and extensions to xBIM Xplorer. The "Damage Model Implementation Strategy" section summarizes five possible implementation strategies. IFC is a comprehensive standard that supports several domains during planning, design, and construction. Additionally, IFC is flexible because of generic classes and custom classifications. Together with the broad support of IFC by several software tools, those arguments led to the decision to implement the object-oriented model in IFC.

IFC Classes for Semantic Data

Up to now, there have been no specific defect entities implemented in the IFC. However, the IFC offers several alternatives: *IfcProxy*, *IfcAnnotation*, *IfcSurfaceFeature*, and *IfcVoidingFeature*. Table 2 compares the advantages and disadvantages of these IFC entities. *IfcProxy* is a generic entity, but IFC 4 lists the proxy as deprecated and recommends using *IfcBuildingElementProxy* instead. A look at newer versions of the IFC reveals the proxy is marked deprecated no longer. The DIM should be usable in future standards and hence the proxy is taken as an option instead of *BuildingElementProxy*. A proxy is very flexible and thus suitable for every defect type. However, a proxy is treated as an individual element or building element, which conflicts with the nature of a defect. A defect cannot exist without the affected component. An *IfcAnnotation* may be used to add further annotations to a component. However, *IfcAnnotations* are only meant to have zero-dimensional, 1D, or

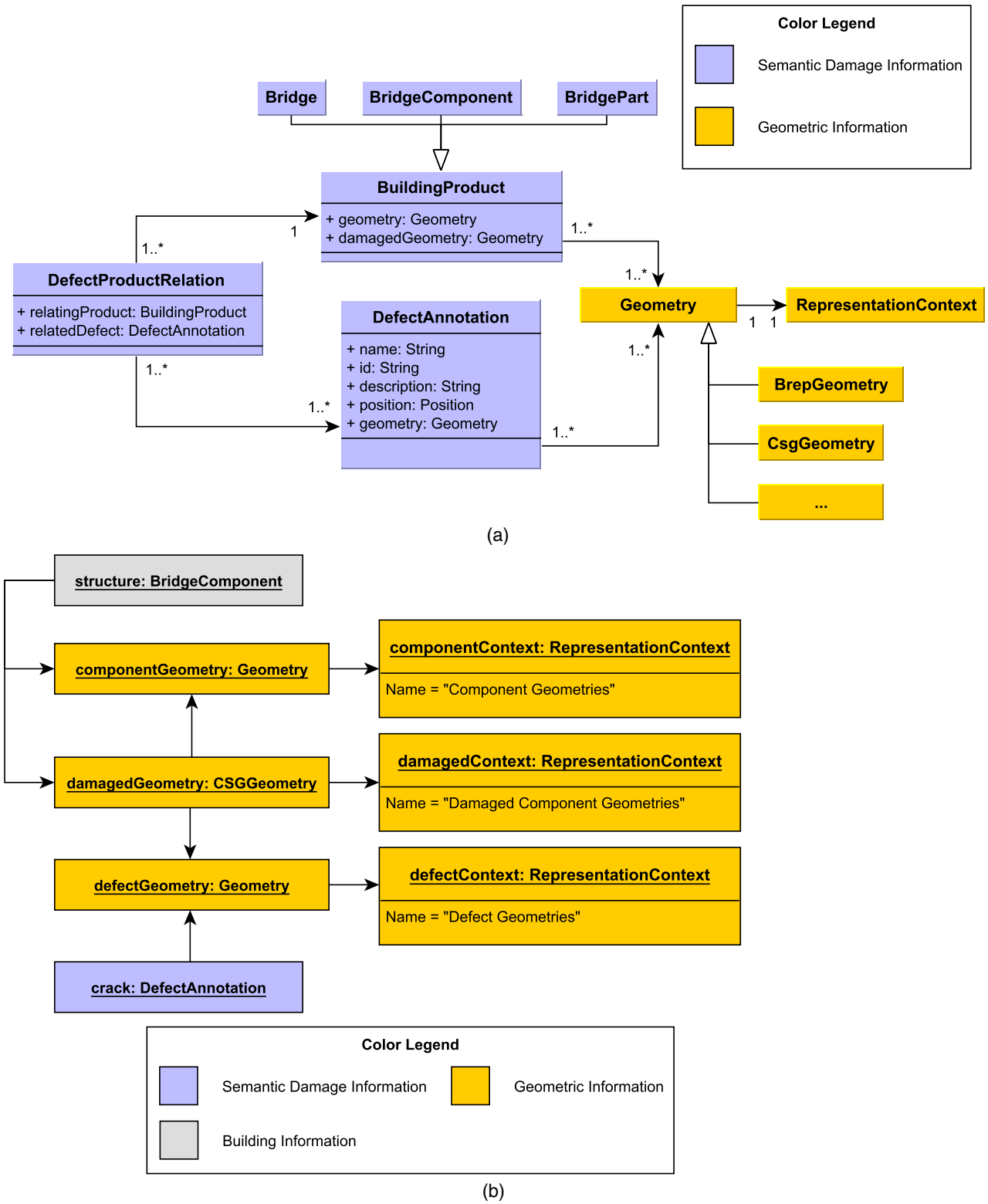


Fig. 6. (a) UML class diagram for modeling the threefold geometry: intact component geometry, defect geometry, and damaged component geometry; and (b) object diagram for the threefold geometry model.

2D geometries. *IfcSurfaceFeatures* and *IfcVoidingFeatures* are suitable for specific defects. Surface features are suitable for modeling corrosion or other defects that only affect the surface of a component. Voiding features are suitable for defects like cracks or spalling, which add a void to a component. However, they are not suitable for modeling further damage types, e.g., material changes below the surface, divergences from specifications, or washouts. On the whole, depending on the defect, a suitable IFC entity must be chosen. Corrosion or other surface changes that affect surface features. Cracks and spalling are represented by voiding features at best. Other defects could use either annotations or proxies.

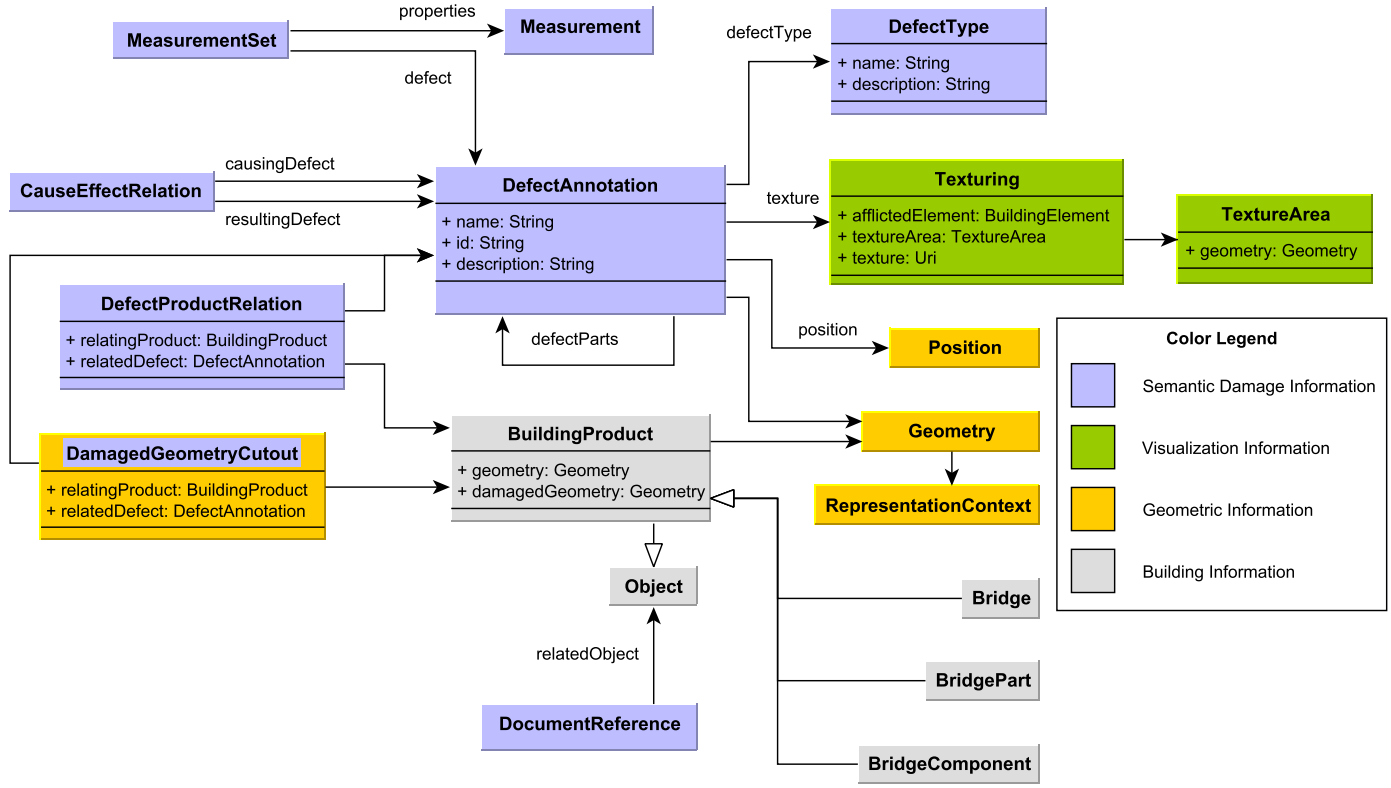


Fig. 7. UML model including both Variants A and B with the relationship-based geometry and the distinct geometry and relationship.

Table 2. Overview of the possible IFC entities with advantages and disadvantages

+/−	*IfcProxy*	*IfcAnnotation*	*IfcSurfaceFeature*	*IfcVoidingFeature*
+	Interpretable by most applications	Add (textual) information about defect to component	Suitable for specific defects Geometry is not visualized like geometries of components	Suitable for specific defects Geometry is not visualized like geometries of components
−	Generic container Independent object in contrast to a dependent defect	Less supported by applications Limited representations Modeling defects as annotations conflicts with the original meaning of annotations	Only designed for modifications at surface Less supported by applications Modeling defects as surface features may conflict with original meaning of surface feature	Only designed to reduce volume of element Less supported by applications Modeling defects as voiding features may conflict with original meaning of voiding feature

Note: +/− = advantages and disadvantages.

In the next step, the analysis of suitable relationships from the IFC standard is presented. *IfcRelAssigns* "is a generalization of 'link' relationships among instances of IfcObjects and its various 1st level subtypes" (buildingSMART International Ltd. 2018b). A specific identification of the relationship is stored as the name of the relationship. In case of the *DefectProductRelation*, the name of *IfcRelAssignsToProdcut* would be defect product relation. However, a defect is part of a component and if the component is destroyed the defect no longer exists. Hence, a composition is more precise. Strict compositions are modeled with *IfcRelAggregates*. Construction and design practice understands aggregations as a sum of different products. This would imply a defect is a product if an aggregation is used, which is questionable. Altogether, aggregations seem to be the most precise relationship for physical defects. Both relationships of the aggregation and the assignment may be used for other defects. In case of using *IfcVoidingFeature* to represent defects, the decomposition relationship *IfcRelVoidsElement* is suitable. "IfcRelVoidsElement is an objectified relationship between a building element and one opening element that creates a void in the element" (buildingSMART International Ltd. 2018b). As stated previously, the voiding feature and the voids relationship are only applicable to cracks or spalling and not in case of material changes or other damage types. An example is depicted in Fig. 8. *IfcRelAssignsToProdcut* may be used for effect-cause relations with the name "cause" or "reason." Additional information about the relation might be given by the description of the relationship. Table 3 provides an overview of the existing relationships and related advantages and disadvantages.

To model damage types, IFC type objects are suitable entities. Furthermore, property sets or templates are suitable for defining type-related properties or necessary properties for entities of a type, for example, measurements, the condition rating, dates, or further descriptions. Document references link to existing inspection reports, results of additional surveys, or photos. Fig. 9 depicts an IFC file that includes a damage type, some measurements, and a document reference.

```
/*Building element*/
#244= IFCBEAM('2BvIrDXR15IBPk7viSE8Ne',#42,
    'Concrete - Rectangular Beam:300 x 600mm:322549',
    $,'300 x 600mm',#242,#233,'322549',.BEAM.);

/* Defect Spalling */
#9002= IFCVOIDINGFEATURE('0Zmm6ecPUESkXa2wa9ySyQ',#42,
    'Spalling','Spalling at beam','Defect - Spalling',
    #8556,$,'331134',.CUTOUT.);
#9004= IFCRELVOIDSELEMENT('0uTuF9cIxk6g3O8pJhXrRg',
    #42,$,$,#244,#9002);
```

Fig. 8. Extract of an IFC file modeling a damaged beam (#244) as damaged building element. The defect is represented by a voiding feature (#9002) and the voids relationship (#9004) models the relationship to the beam.

IFC Classes for Geometry Data

This subsection discusses the implementation of geometries of the DIM by using the IFC. In addition to modeling the geometry of the damaged component, the method of modeling a defect geometry and the use of geometric representation contexts are illustrated.

Relationship-Based Geometry

This paragraph illustrates the implementation of the geometry model described previously and is related to Fig. 5. Fig. 10 shows an extract of an IFC file that contains an *IfcVoidingFeature* (#9002) as defect entity and a related component. The *IfcRelVoidsElement* (#9004) represents the relationship between the defect and component and implies cutting the defect geometry out of the component geometry.

Independent Relationship and Geometry

This paragraph illustrates the implementation of the geometry model described previously and is related to Fig. 6. In case of storing the geometry of the damaged component in the IFC, representation contexts are chosen to distinguish the geometries of intact and damaged components. A product might have multiple representations and every representation has a different representation context. Fig. 11 illustrates the use of multiple geometries and representation contexts. The defect and the beam have their own geometries as shown by Entities #9003 and #233. In this context, the damaged geometry of the beam, Entity #9100, is a CSG geometry with a subtraction of the undamaged beam and the defect geometry.

IFC Classes for Visual Data

Visual data may be stored as document references or as textures, which are depicted on a 3D surface. Document references were discussed in the "IFC Classes for Semantic Data" section. Coming to the implementation of textures, Fig. 12 illustrates how to include a texture in an IFC file. To position an image, for example, a png file, within the 3D model, a geometry is necessary. This geometry is represented by the *IfcRepresentationItem*. Such a geometry could be a plane, which carries the texture slightly above the related position of the affected component. A listing example can be found in the study by Hüthwohl et al. (2018). In addition to the texture itself, the mapping is necessary. The IFC offers the class *IfcTextureCoordinate* to add texture-mapping information and subclasses, such as *IfcTextureCoordinateGenerator* and *IfcIndexedTriangleTextureMap*, to either define an algorithmic or point-based texture mapping.

xBIM Extensions

During the tests, which are explained in the next section, two problems could be observed:

- None of the software tools supports selection of *IfcGeometricRepresentationContexts*; and
- None of the software tools interprets textures correctly.

To verify the entire model, one software tool had to be extended; xBIM was chosen for the same.

Representation Context Selection

The selection of *IfcGeometricRepresentationContexts* is required to illustrate the visualization of multiple geometries and switch between different geometries. Fig. 11 illustrates the use of *IfcGeometricRepresentationContext* and *IfcGeometricRepresentationSubContext* in an IFC file. Entity #9050 is the subcontext for the final visualization. xBIM should be able to select contexts or subcontexts. In case of selecting a context, all subcontexts should be selected as well. A new menu item called *Select visualization context* has been added in the view menu of xBIM. This opens a dialog, which displays all available geometric representation contexts and subcontexts. Fig. 13 depicts the view menu and the opened dialog. The user can select multiple contexts or subcontexts there. According to the selection the new scene is generated and visualized.

Texture Visualization

To use textures, the image of the texture must be loaded and applied correctly according to the given texture mapping. The path to the image is stored as the absolute path or as the path relative to the IFC file. After loading the given image, texture mapping is required to determine the texture position. A spherical texture mapping was implemented as a first approach. The spherical texture mapping can be imagined as a sphere around the 3D geometry. Rays start from a calculated or given midpoint of the 3D geometry, go through one vertex of the geometry, and land on the outer texture. The vertex of the geometry and the pixel coordinate of the image are mapped together. Another method is that the user manually defines a mapping between texture coordinates and the related vertices or triangles (Heckbert 1989). Both algorithms were implemented and screenshots of the results are part of the "Testing of the Implementation" section.

Testing of the Implementation

Testing the IFC implementation concept was performed by checking the compliance of the test files according to IFC 4 and through evaluation of the visualization. Future tests will include the transfer to calculations in simulation environments. First, solitary test files are defined as per the implementation. Next, all solitary test files are checked for compliance against IFC 4 and used to validate the single requirements. The solitary test files consist of damaged beams

Table 3. Overview of the possible IFC entities with advantages and disadvantages

+/−	*IfcRelAssignsToProduct*	*IfcRelAggregates*	*IfcRelVoidsElement*
+	Interpretable by most applications	Interpretable by most applications	Avoids additional data for geometry
−	Not usable for defects, which are part of a component (e.g., cracks, spalling)	Some defects are not part of component (e.g., vegetation) parts Representation results from geometry of parts	Designed for voids only Less supported by applications

Note: +/− = advantages and disadvantages.

```
/* Defect Spalling */
#9000= IFCPROXY('17nNSsgA50etMCCiVyNexg',#42,
    'Spalling',$,'Defect',#242,$,.NOTDEFINED.,$);

/*Damage type*/
#9010= IFCRELDEFINESBYTYPE('13WjkWDd7UaP4U8kWb24XA',
    #42,'Damage type','Typification of a defect',(#9000),
    #9011);
#9011= IFCTYPEOBJECT('1sEnsXHLyU2mkrxPrKPC8g',#42,
    'Damage type Spalling',$,'IfcProxy/Defect',$);

/* Measurements */
#9020= IFCRELDEFINESBYPROPERTIES('1ruRh1w1u0Wc8DOTLQjxHA',#42,
    'Defect Measurements','Diameter and depth of the
    spalling',(#9000),#9021);
#9021= IFCPROPERTYSET('1rTM5CXyX0GSQW5bsLwvmQ',#42,
    'Diameter and Depth',$,(#9022,#9023));
#9022= IFCPROPERTYSINGLEVALUE('Diameter',$,IFCREAL(151.0),#43);
#9023= IFCPROPERTYSINGLEVALUE('depth',$,IFCREAL(12.0),#43);
#43= IFCSIUNIT(*,.LENGTHUNIT.,.MILLI.,.METRE.);

/* Document */
#9030= IFCRELASSOCIATESDOCUMENT('1zBLkQ0LMEyVhpcTpqsrvw
    1240',#42,'Report of ultra sonic survey',$,(#9000),#9031);
#9031= IFCDOCUMENTREFERENCE('http://standards.buildingsmart.org/
    IFC/RELEASE/IFC4_1/FINAL/EXPRESS/IFC4x1.exp',
    'U_S_16092020-42','Ultra Sonic report 16092020-42 from
    the 16th September 2020',$);
```

Fig. 9. Part of an IFC file modeling a proxy for the defect (#9000) and a type object (#9011) to define a damage type, namely, damage type spalling. Additionally, some measurements (#9021) and a reference to an external document (#9031) are included.

```
/*Building element*/
#244= IFCBEAM('2BvIrDXR15IBPk7viSE8Ne',#42,'Concrete -
    Rectangular Beam: 300 x 600mm:322549',$,'300 x 600mm',
    #242,#233,'322549',.BEAM.);
#233= IFCPRODUCTDEFINITIONSHAPE($,$,(#227,#231));
#231= IFCSHAPEREPRESENTATION(#103,'Axis','MappedRepresentation',
    (#229));

/* Defect Spalling */
#9002= IFCVOIDINGFEATURE('0rxG1zyy7UqaiUoneVATvg',#42,
    'Volumetric: Box:331134',$,'Element Defect Spalling'
    ,#8556,#9003,'331134',.CUTOUT.);
#9004= IFCRELVOIDSELEMENT('26SmEnuw1Ey1zjxe7txNJA',
    #42,$,$,#244,#9002);

/* Defect Geometry */
#9003= IFCPRODUCTDEFINITIONSHAPE($,$,(#8548));
#8548= IFCSHAPEREPRESENTATION(#105,'Body',
    'MappedRepresentation',(#8546));
#8546= IFCMAPPEDITEM(#8542,#232);
#8542= IFCREPRESENTATIONMAP(#8541,#8539);
#8539= IFCSHAPEREPRESENTATION(#105,'Body','SweptSolid',(#8538));
#8538= IFCEXTRUDEDAREASOLID(#8534,#8537,#20,250.);
#8526= IFCCARTESIANPOINTLIST2D(((-125.,-30.),(125.,-30.),
    (125.,30.),(-125.,30.),(-125.,-30.)));
#8533= IFCINDEXEDPOLYCURVE(#8526,$,.F.);
#8534= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'Box',#8533);
```

Fig. 10. Extract of an IFC file modeling a beam (#244) as damaged building element. The defect is represented by a voiding feature (#9002) and the voids relationship (#9004) represents the relationship between the beam and the defect.

without reinforcement. Revit was used for designing the beam model and exporting it to IFC 4. The damage information was added to the beam manually by using a plain text editor.

Multiple software tools were used for verifying the broad usability of the concept. Table 4 gives an overview of all examined software applications. This study focused on modeling data and not on the usability of authoring software. Hence, only Revit was tested as representative of authoring tools. Future research should investigate editing possibilities as well.

Testing Semantic Data

In the first step, the functionality of visualizing semenatic data was tested. All four IFC entities, i.e., *IfcAnnotation*, *IfcProxy*,

```
/* Building Element */
#244= IFCBEAM('091e0b9f052a49d2b9af35c7fe711a91',#42,'Concrete -
    Rectangular Beam: 300 x 600mm:322549',$,'300 x 600mm',
    #242,#233,'322549',.BEAM.);

/* Undamaged Geometry */
#105= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','Model',*,*,
    *,*,#99,$,.MODEL_VIEW.,$);
#155= IFCEXTRUDEDAREASOLID(#149,#154,#20,12125.4);
#187= IFCREPRESENTATIONMAP(#186,#165);
#225= IFCMAPPEDITEM(#187,#224);
#227= IFCSHAPEREPRESENTATION(#105,'Body','MappedRepresentation',
    (#225));
#233= IFCPRODUCTDEFINITIONSHAPE($,$,(#227,#231,#9100));

/* Defect Spalling */
#9000= IFCPROXY('2IPLat5bzU2P8UCfR1KnJw',#42,
    'Spalling',$,'Defect',#242,#9003,.NOTDEFINED.,$);

/* Defect Geometry */
#8526= IFCCARTESIANPOINTLIST2D(((-125.,-30.),(125.,-30.),
    (125.,30.),(-125.,30.),(-125.,-30.)));
#8533= IFCINDEXEDPOLYCURVE(#8526,$,.F.);
#8534= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'Box',#8533);
#8538= IFCEXTRUDEDAREASOLID(#8534,#8537,#20,250.);
#8539= IFCSHAPEREPRESENTATION(#9050,'Body','SweptSolid',(#8538));
#9003= IFCPRODUCTDEFINITIONSHAPE($,$,(#8539));
#9050= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Defect Geometry',
    'Defect Geometry',*,*,*,*,#9051,$,.MODEL_VIEW.,$);
#9051= IFCGEOMETRICREPRESENTATIONCONTEXT('Defect',
    'Model' ,3,0.01,#96,#97);

/* Damaged Component Geometry*/
#9100= IFCSHAPEREPRESENTATION(#9150,'Body','CSG',(#9101));
#9101= IFCCSGSOLID(#9102);
#9102= IFCBOOLEANRESULT(.DIFFERENCE.,#155,#8538);
#9150= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Damaged Components',
    'Damage Model',*,*,*,*,#9151,$,.MODEL_VIEW.,$);
#9151= IFCGEOMETRICREPRESENTATIONCONTEXT('Damaged-geometry',
    'Model',3,0.01,#96,#97);
```

Fig. 11. Extract of an IFC file modeling a distinct geometry and relationship. An assignment (#9001) represents the relationship between the beam (#244) and the defect (#9000). The beam has two geometries: a damaged geometry (#9100) and an undamaged geometry (#227).

IfSurfaceFeature, and *IfcVoidingFeature*, were tested. The expectation is that the software provides a geometric view, a hierarchical tree view, and a view for the properties. Table 5 presents an overview of the test results. Revit, Desite BIM, and Solibri Model Viewer lack the hierarchical view of the model and hence the defects without geometries could not be selected. Furthermore, none of the three includes a hierarchical view of the model. All other software visualizes the test files properly.

Next, the visualization of the relationships was tested. For this purpose, typification, external references, and defect relationships were added. Classification could be visualized via a property view or by using the correct product type. Table 6 summarizes the test results. IFC viewers do not access product catalogs and hence the type is shown as property in the view. Revit uses its internal type catalog to select the corresponding type of an entity. However, this is only possible if the typification is stored with correct Revit family names. The same problem arises with measurements or properties in Revit. External references should be shown at least in the property view with their URI. The apstex IFC viewer and xBIM show external references in such a way. None of the other software tools showed the external document references. Last, defect relationships, i.e., aggregation, association, or voids element, should be shown in the hierarchical view or as properties. xBIM and apstex show aggregations in the hierarchical view and associations as properties. BIM Vision was able to show aggregations but not the associations.

Testing Texture Implementation

Textures are the second requirement in the data model. To test texturing, an image was attached to an additional plane, which is at the

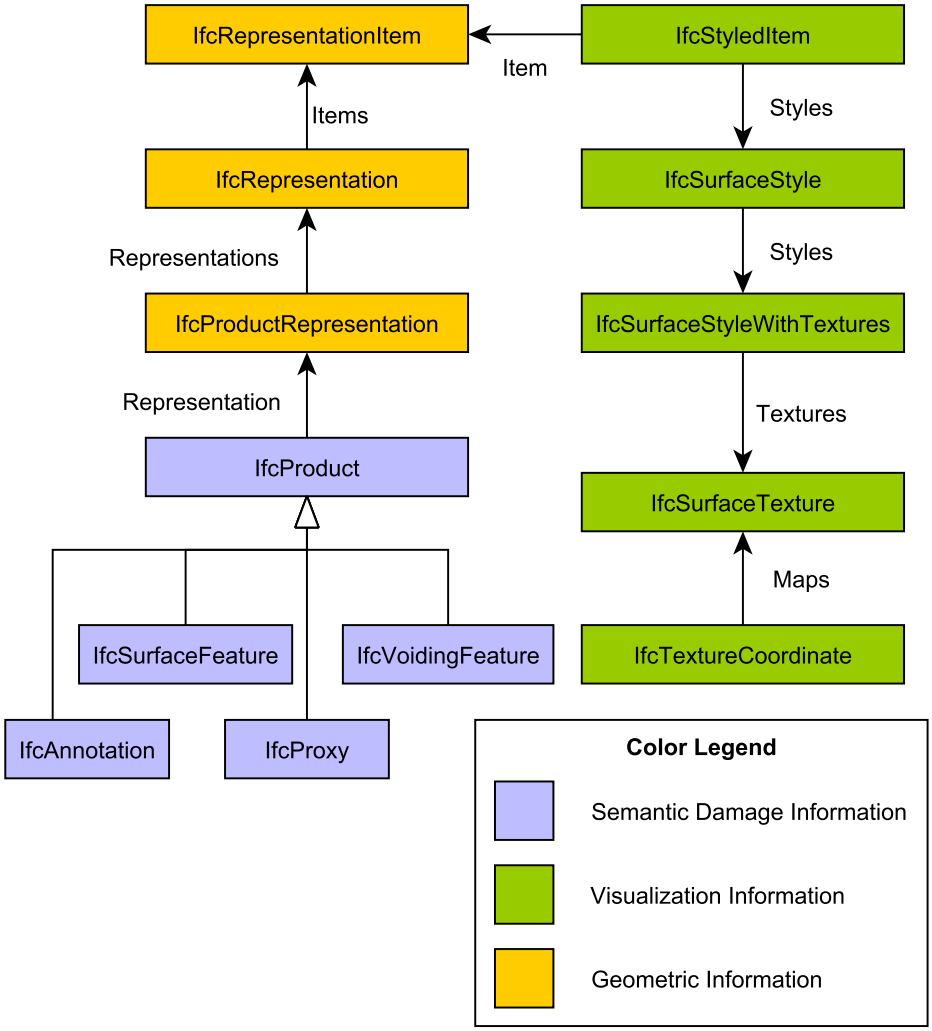


Fig. 12. Damage model with texture using *IfcSurfaceFeature* and related elements. Multiple superclasses, subclasses, and selects are omitted for simplicity.

defect position. Other geometries may be used instead of a plane. As depicted by the last row in Table 7, none of the available software was able to properly visualize the texture. Most ignored the texture parameter. usBIM only shows the plane where the texture should be depicted. xBIM Xplorer has received an extension to visualize textures on a geometries with a given texture mapping. Fig. 14 shows an example of a beam with a texture representing a defect.

Testing of Geometry Data

Geometric representations are very common in the AEC sector. However, the software programs support the geometric representations in different quality, which is evidenced in Table 7. The visualization of CSG geometries was done properly by all IFC viewers except Desite BIM and the Solibri Model Viewer. None of the viewers that are available by the software vendors offer a selection of representation context. This requirement was achieved only by Revit. Revit includes 2D plans and 3D views for its building models; however, multiple 3D geometries are not possible in Revit. The custom extended version of xBIM was able to offer the context selection with multiple 3D geometries.

The next step tested the visualization of an *IfcVoidingFeature* with an *IfcRelVoidsElement* relationship in accordance with the relationship-based cutout. The voiding feature is correctly supported by apstex's IFC Viewer and xBIM Xplorer. Other programs do not respect an *IfcVoidingFeature* with an *IfcRelVoidsElement* relationship. Many viewers are able to handle an opening in conjunction with an *IfcRelVoidsElement*. However, defining a defect as an opening is semantically wrong. Fig. 15 shows the visualization of an *IfcVoidingFeature* with an *IfcRelVoidsElement* relationship in the original xBIM Xplorer. Fig. 15(a) shows a beam with typical spalling. Fig. 15(b) depicts a close-up screenshot of the cutout of the defect in the beam. Finally, in Fig. 15(c) one can see the highlighted defect geometry of the spalling. A similar result was achieved with the apstex IFC viewer.

Fig. 16 shows the selection and output of different visualization contexts in the extended xBIM Xplorer. Figs. 16(a–c) show the selected representation context, Figs. 16(d–f) present an overview of the model, and Figs. 16(g–i) depict a close-up view of the damaged section. Figs. 16(d and g) show the visualization of the undamaged beam, Figs. 16(e and h) show the defect geometry, and Figs. 16(f and i) show the damaged beam after subtracting the defect geometry. If the defect geometry and the geometry of the damaged component are activated, the used defect element, which is a proxy in this case, is shown as filling in the damaged beam. This is disadvantageous because the defect geometry

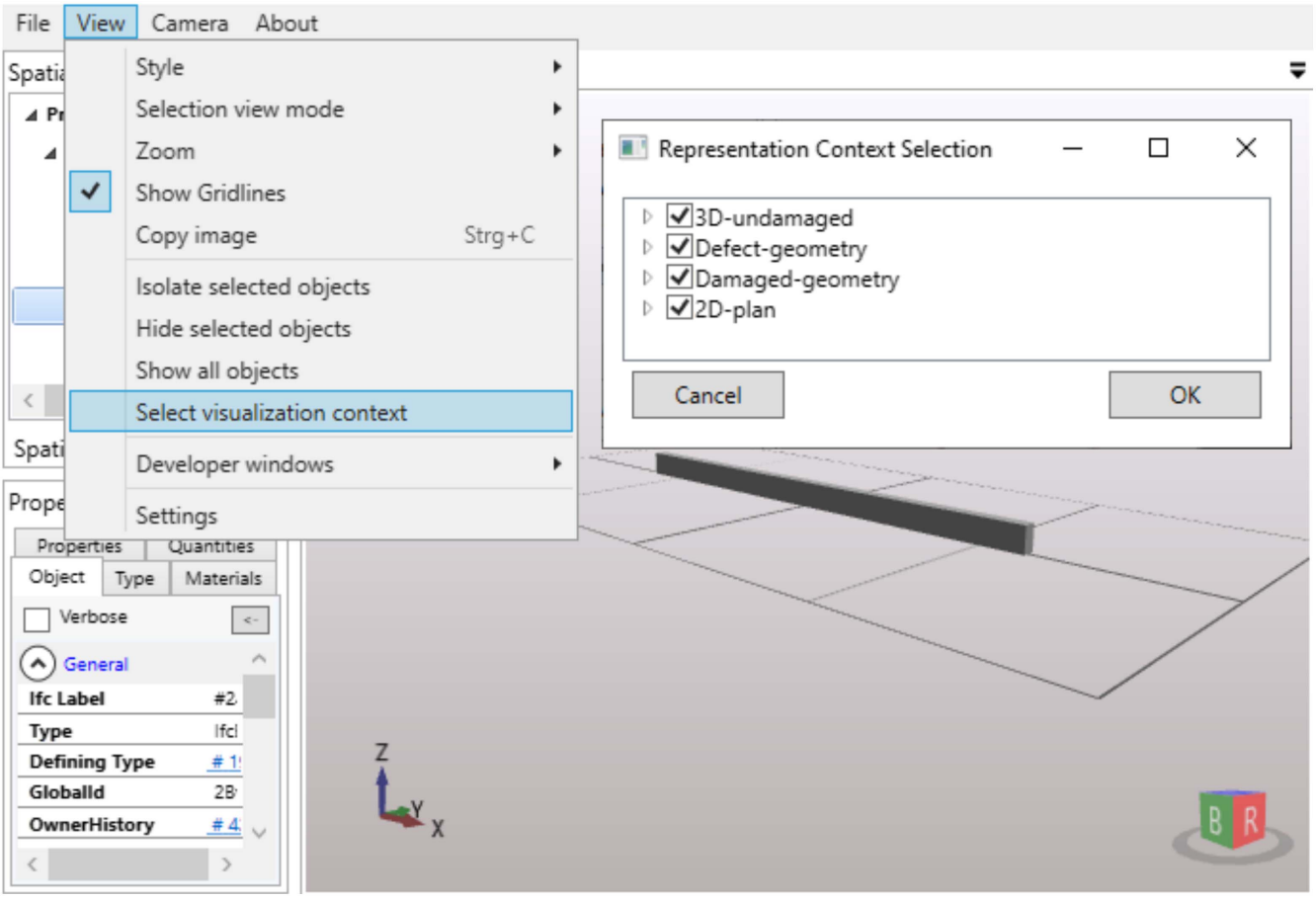


Fig. 13. The view menu of xBIM with the opened context selection dialog.

Table 4. Tested BIM authoring software and IFC viewers

Authoring software	IFC viewers
Autodesk Revit 2019	apstex IFC viewer BIM Vision Desite BIM (Thinkproject 2019) Solibri Model Viewer usBIM xBIM Xplorer

should not be a filling. If the relationship-based cutout is used, i.e., *IfcVoidingFeature* with *IfcRelVoidsElement*, only the damaged component geometry is visible; however, the cutout is never shown as filling.

The test with the CSG representation revealed there is another feature missing in xBIM Xplorer. xBIM Xplorer is not able to use *IfcBooleanResults* in combination with *IfcTriangulatedFaceSets*. Because of this, the example in Fig. 16 uses a simplified defect geometry.

Table 5. Software used for visualizing the defect information in a hierarchical or properties view

Defect type	Autodesk Revit	apstex IFC viewer	BIM vision	Desite BIM	Solibri model viewer	usBIM	xBIM Xplorer
Annotation	—	✓	(✓)	—	—	✓	✓
Proxy	—	✓	✓	—	—	✓	✓
Surface feature	—	✓	✓	—	—	✓	✓
Voiding feature	—	✓	✓	—	—	✓	✓

Table 6. Performance of the software regarding relationships

Defect information	Autodesk Revit	apstex IFC viewer	BIM vision	Desite BIM	Solibri model viewer	usBIM	xBIM Xplorer
Classification	(✓)	✓	—	—	—	✓	✓
External references	—	✓	—	—	—	—	✓
Measurements	(✓)	✓	—	—	—	—	✓
Defect relationship	—	✓	(✓)	—	—	—	✓

Table 7. Performance of the software regarding different geometric representations and texture

Model aspects	Modeling approaches	Autodesk Revit	apstex IFC viewer	BIM vision	Desite BIM	Solibri model viewer	usBIM	xBIM Xplorer (original)	xBIM Xplorer (extended)
CSG + contexts	Context selectable	✓	—	—	—	—	—	—	✓
	Show different represen-tations	—	—	—	—	—	—	—	✓
	Show defect geometry	✓	✓	—	—	—	✓	✓	✓
Voiding feature	Subtract geometry	✓	✓	—	—	—	✓	✓	✓
	Show defect geometry	✓	✓	—	—	—	✓	✓	✓
Texture	Visualize texture	—	—	—	—	—	—	—	✓

Comparing Test Results to Requirements

Altogether, with the use of IFC and an extension of xBIM Xplorer, it was possible to address all requirements stated in the "Requirement Analysis" section. Table 8 shows an overview of the requirements and finally used entities of the standardized IFC 4. All implementations could be verified using an extended version of xBIM Xplorer.

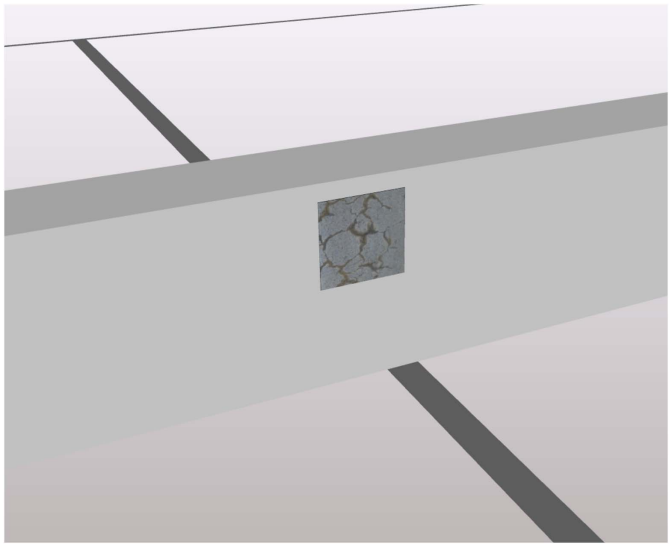


Fig. 14. Screenshot of xBIM Xplorer with a textured proxy as defect at a beam.

Use Cases and Examples

For a better illustration of use cases and examples, a bridge model of IFC Infra was used (IFC Infra 2019b). The bridge model was originally written according to the IFC 4x2 standard and was transferred to IFC 4. Some entities, classes, and enumerations are not available in IFC 4 and hence were replaced by proxies or other suitable entities. Defects and damage data were added manually by using a plain text editor. Hence, damage geometries are kept simple. Furthermore, four geometric visualization contexts were defined in the model: body, defect geometry, damaged components, and defect photos.

Fig. 17 depicts the bridge with the positions of the defects marked. Starting from the left, there are two test drills at the abutment. The defect at the midtop represents some cracks in the pavement. Third, the railing in the lower midsection is corroded. Last, there is a spalling on the lower-right at the abutment wall.

Model-Based Inspection Review

On the basis of the damaged bridge model, an inspection review may be performed. Fig. 18 shows a scenario of discussing defects using the as-damaged bridge model. All defects, their properties, and related documents may be reviewed by a team of engineers. Instead of using drawings and textual descriptions only, a 3D model can be examined by moving around, selecting images, showing related data, and discussing defect geometries and their impact on the condition assessment.

Corrosion

Fig. 19(a) shows a photo of a corroded railing. Right next to this photo is the selected railing in the model. Fig. 19(b) shows the

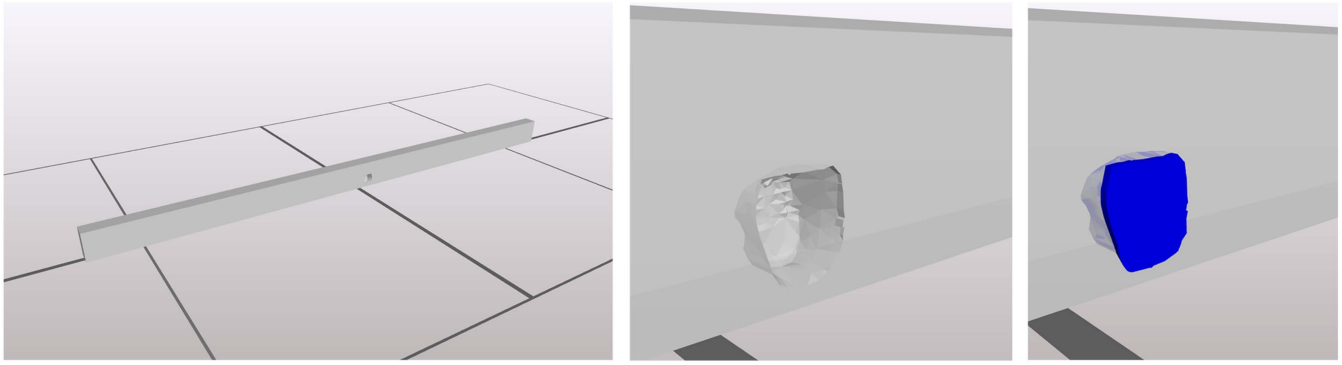


Fig. 15. (a) Visualization in xBIM Xplorer of a beam with spalling modeled by using a voiding feature and (b) a close view at the spalling at the beam; and (c) typical spalling geometry. The transparency has been increased to improve the visibility of the cutout.

J. Comput. Civ. Eng., 2022, 36(6): 04022029

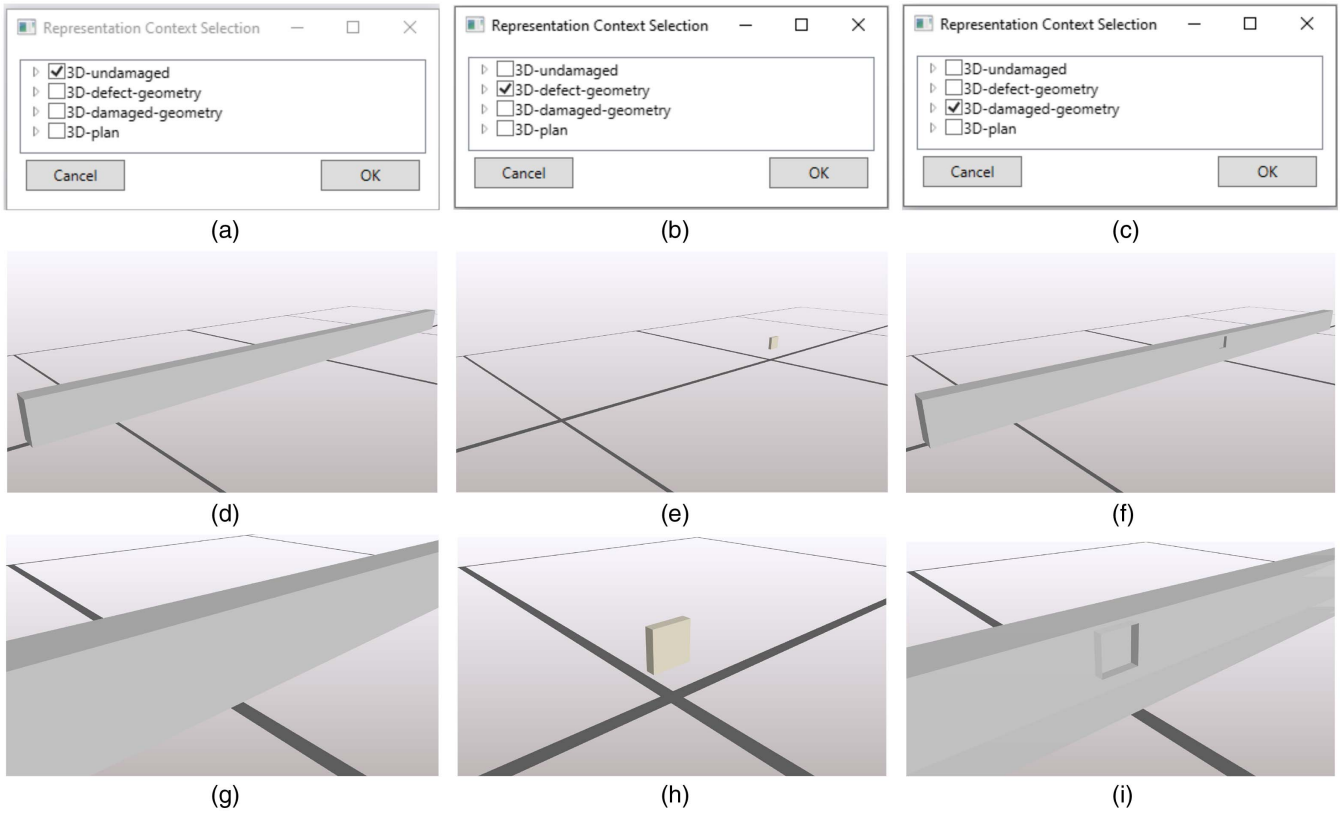


Fig. 16. Model of a defect using CSG and different visualization contexts in the 3D view: (a, d, and g) undamaged beam; (b, e, and h) defect geometry; and (c, f, and i) damaged beam.

properties of the railing in the model. Taking adequate photos to represent the corrosion of the entire railing is cumbersome and time-consuming. Therefore, a simple property is used to represent this defect. The railing has only a body geometry and hence, if the damaged component geometric representation context is selected, the railing is not shown anymore in the 3D view. This defect revealed that it is not sufficient to use geometric representation contexts for selection. Specialized views are necessary, such as a view with highlighted components or damage textures.

Table 8. Summary of test requirements and test results

Requirement	Successfully tested solutions
Defect entity	*IfcAnnotation*, *IfcProxy*, *IfcVoidingFeature*, *IfcSurfaceFeature*
Relationship for damaged components	*IfcRelAssociatesProduct*, *IfcRelAggregates*, *IfcRelVoidsElement*
Relationship for defect groups	*IfcRelAggregates*
Relationship for cause and effect	*IfcRelAssociates*
Relationship for related documents	*IfcRelAssociatesDocument*
Classification	*IfcTypeObject* and *IfcRelDefinesByType*
Defect properties	*IfcPropertySet* and *IfcProperty*
Multiple photos, images, or videos	See relationships for documents
Textures	*IfcImageTexture* and *IfcTextureCoordinate*
1D, 2D, and 3D defect geometry	*IfcProductDefinitionShape* and subclasses
Multiple geometries and selection	*IfcGeometricRepresentationContext*

Cracks

Fig. 20 shows some cracks on the pavement of the bridge. Furthermore, there is a bump in the pavement and the sidewalk. By using this photo as a texture, the inspector or engineer can get a quick impression of the defect. Aligning the texture to the 3D model can help the user gather additional information about related or near elements faster compared to studying 2D plans. However, this defect shows the problem of image rectification for textures. The image, which has been used for the texture, has not been rectified. Hence, the texture shows the slope of the bridge at the position of the sidewalk, which could be misunderstood.

Spalling

Fig. 21 shows an example of a geometric representation of a defect. Fig. 21(a) shows a photo of the spalling, and Fig. 21(b) shows the defect in the final model. The geometry of the spalling was generated manually within the IFC file. This leads to visual inaccuracies, such as the different paths of the lower part of the spalling. By using SfM, damage geometries can be modeled with higher accuracy (Isailovic et al. 2020). However, the example shows that the principal concept provides geometry information of a defect.

Holes from Drilling Samples

Fig. 22 represents two holes from drilling samples in the abutment. Figs. 22(b and c) show the model after selecting the context of damaged components only. Hence, the abutment with the drill holes is shown without near components. Fig. 22(b) shows the abutment with a texture at the position of the drill holes. Fig. 22(c) depicts the visualization of the drill holes by cutouts. The user can

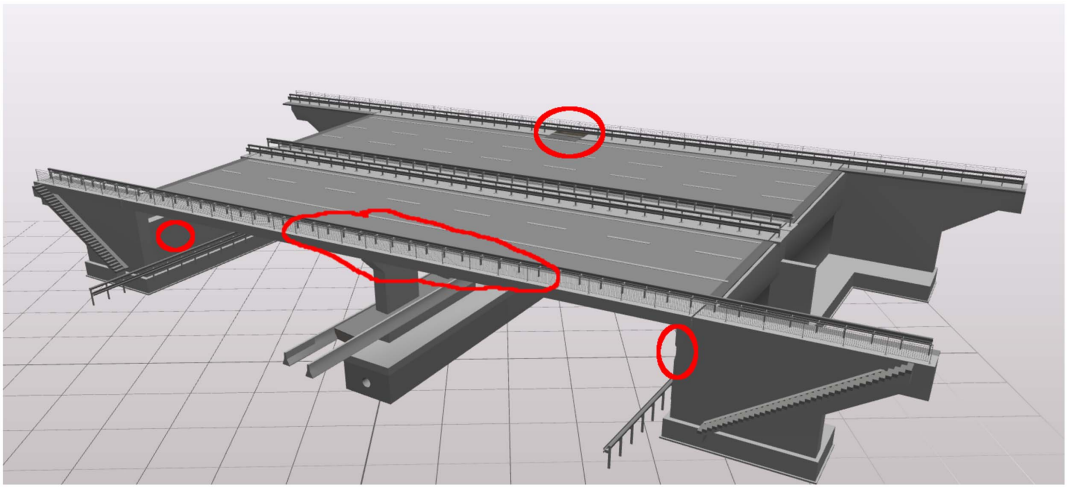


Fig. 17. Bridge with four defects. The places of the defects are marked.

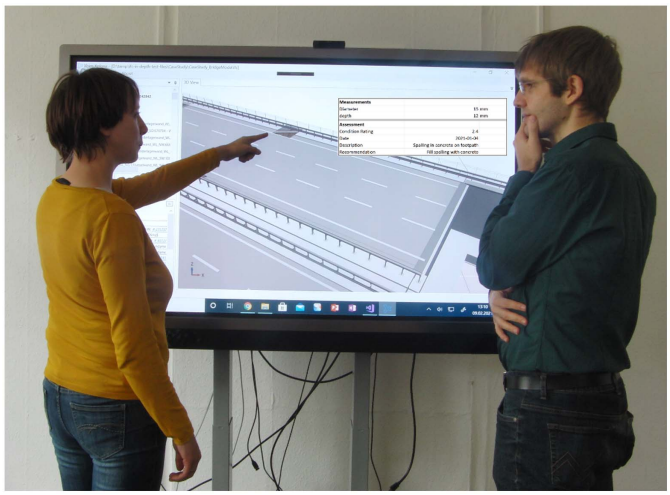


Fig. 18. Model-based inspection review by engineers and discussion of detailed defect geometries.

switch between these visualizations by selecting the representation context. Those multiple visualization approaches would provide information about color changes or geometrical information by using the defect photos' context and the damaged geometry context, respectively.

Detailed Investigation

After the overall review of the bridge, some components may need further investigation. For this step, a geometry-based structural analysis, e.g., FEA, is applicable to determine the impact of the defect on internal forces and stresses. Fig. 23 illustrates an FEA in ANSYS with an individual beam. As an example, the equivalent von Mises stresses were calculated. Fig. 23(a) presents the 3D model views of the beam and the spalling. Fig. 23(b) shows the colored beam in ANSYS and a close look at the beam. The color legend is shown in Fig. 23(b). For the FEA, the IFC file that contains the beam is converted into a step file using IfcConvert (IfcOpenShell 2015). The engineer can add load conditions, bearings, and simulation parameters. With this workflow, the geometry of the beam can be imported directly instead of redrawing it. Subsequently, the FEA can be performed. This FEA model is used as an example, not to perform an in-depth analysis but to show the capability of the information model.

Summary

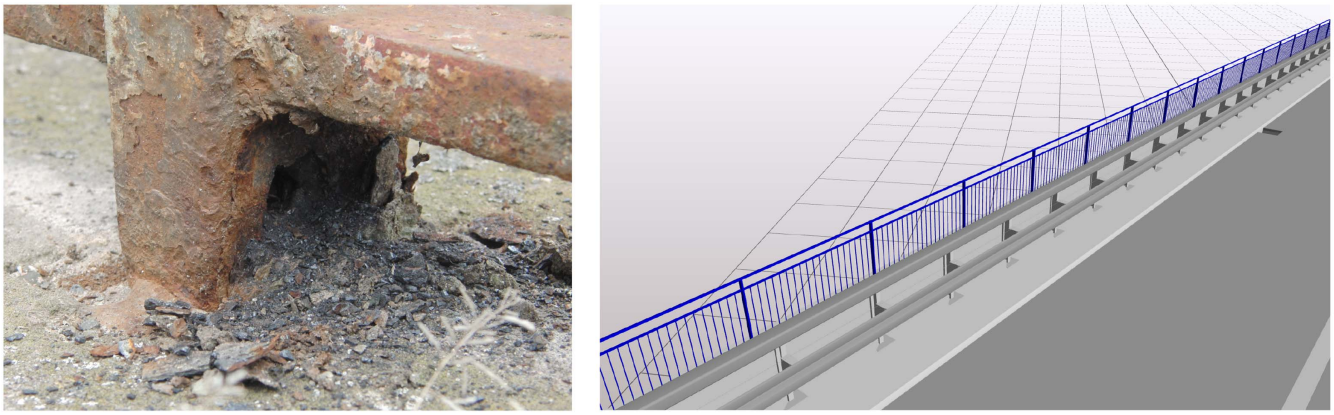
Current inspection and assessment practices are paper-based, and data exchange leads to information loss. To overcome this issue, a DIM is required for data exchange. This study focused on developing a DIM to deliver necessary data of physical defects for inspection review and a simplified example for a detailed investigation via FEA. According to the requirements, an object-oriented DIM was designed. For the implementation, several options, such as proprietary formats, IFC, linked data, and BCF, were discussed in this paper. IFC was chosen for implementation because several viewers exist and hence less effort in writing source code is necessary. Using multiple IFC entities, the object-oriented DIM was implemented. The IFC viewer xBIM Xplorer was extended to be capable of selecting representation contexts and visualize textures. Capabilities and limitations of the model and multiple software programs were evaluated by several tests. Finally, two use cases showed the capabilities and limitations of the proposed DIM. The research questions can be answered as follows.

Which Data Are Necessary to Deliver Damage Information for Assessment and Simulation?

A 3D visualization of the damaged bridge allows model-based assessment reviews. For this purpose, a 3D BIM model of the bridge with all related components forms the basis. Necessary data are photos, measurements, geometry, textures, documents, and typification. Furthermore, the data model must respect that a defect may have multiple measurements, photos, or geometries because of several consecutive inspections. When using a photo as texture, a rectified photo and a texture map are necessary. Structural simulations benefit from an automatic exchange of geometry data. This needs the 3D BIM model, semantic data, and defect geometries.

How Can an Object-Oriented Model Independent of Software Tools or Data Formats Be Designed?

The defect should be represented by an object because it groups all related data, for example, photos, measurements, and documents.



(a)

Properties

Object | Type | Materials | Properties | Quantities

Verbose

General

HasProperties

HasProperties[0]	#330007=IFCPROPERTYSINGLEVALUE('Condition Rating',$,IFCREAL(2.),$);	# 330007
HasProperties[1]	#330008=IFCPROPERTYSINGLEVALUE('Assessment Date',$,IFCDATE('2021-01-26'),$);	# 330008
HasProperties[2]	#330011=IFCPROPERTYSINGLEVALUE('Corrosion state',$,IFCINTEGER(2),$);	# 330011
HasProperties[3]	#330012=IFCPROPERTYSINGLEVALUE('recommended maintenance action',$,IFCLABEL('replace railing'),$);	# 330012

(b)

Fig. 19. (a) Corroded railing on-site (the selected railing is right next to the photo in the model visualized by xBIM Xplorer); and (b) the related property set with a condition rating, an assessment date, and further information.

Fig. 20. Representation of some cracks at the pavement as a texture depicted on a plane.

Classifications should be defined once and used several times. Hence, a typification object, which can be used for all defects of the same type, is necessary. The defect has a geometry that influences the geometry of the related component. To calculate the geometry of damaged components, the geometry of the component and the defect geometry are required. This leads to the fact that a component has two geometries: an intact and a damaged geometry. Hence, a selection of the visualized geometry is necessary, which is respected by using a representation context. Finally, a defect photo may be used as texture. This texture must be mapped to the geometry either by an algorithm, for example, spherical mapping, or by pointwise mapping.

What Changes and Extensions Are Required to Existing Models?

The work of Hamdan and Scherer (2018) and Sacks et al. (2018a) included alphanumeric damage properties in their model but omitted geometry data, visual data, and relationships to additional documents. Geometric data were incorporated by Isailovic et al. (2020)

Fig. 21. Representation of a spalling at the abutment.

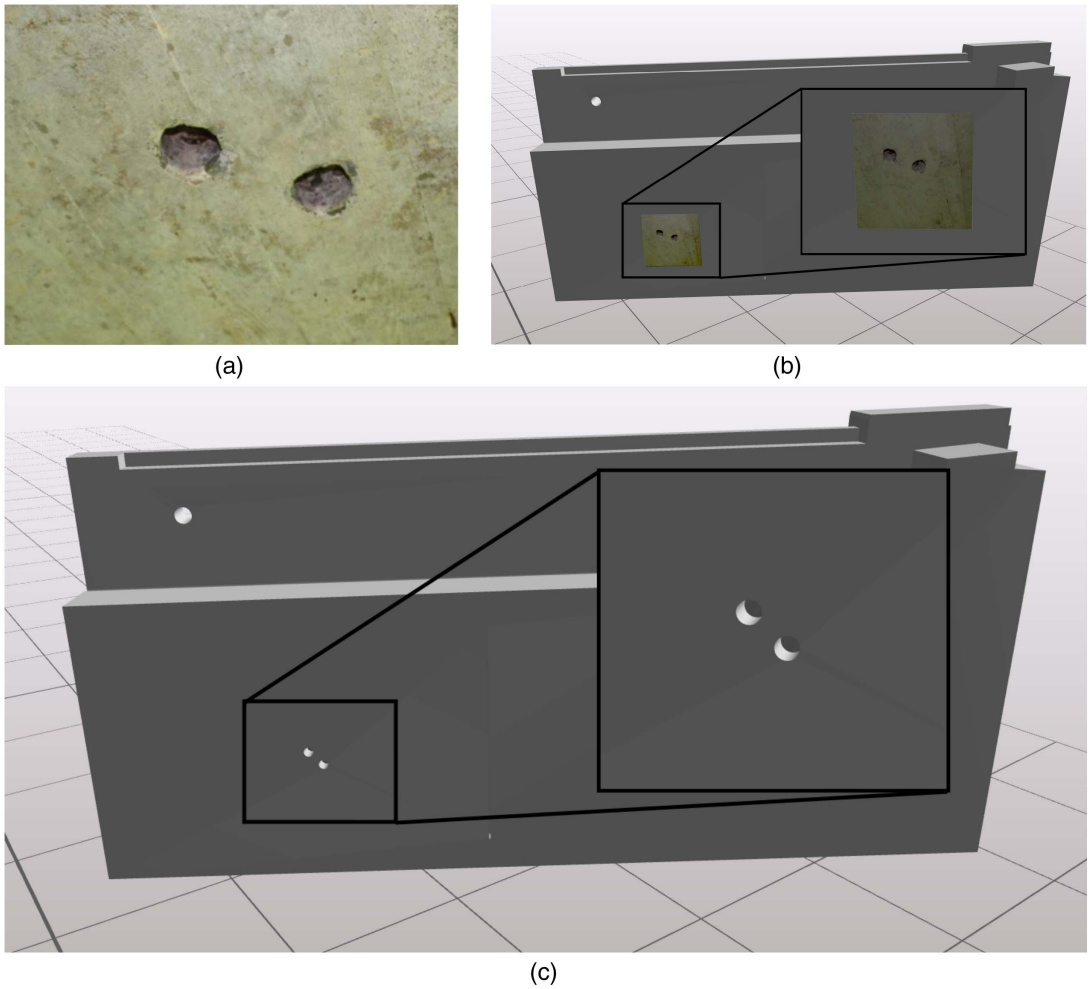


Fig. 22. (a) Some test drills in the abutment; (b) photo as texture on the position of the test drills in the model; and (c) geometry of the defect in the building model.

and McGuire et al. (2016). However, these studies did not incorporate multiple defect representations in parallel, for example, a defect representation as texture and geometry. Although the approach of Hüthwohl et al. (2018) showed how to visualize a defect as a texture, it does not explain how the required texture-mapping information is provided to the visualization software. The proposed concept synergizes semantic, geometric, and image data as well as adding the possibility of multiple representations and multiple images.

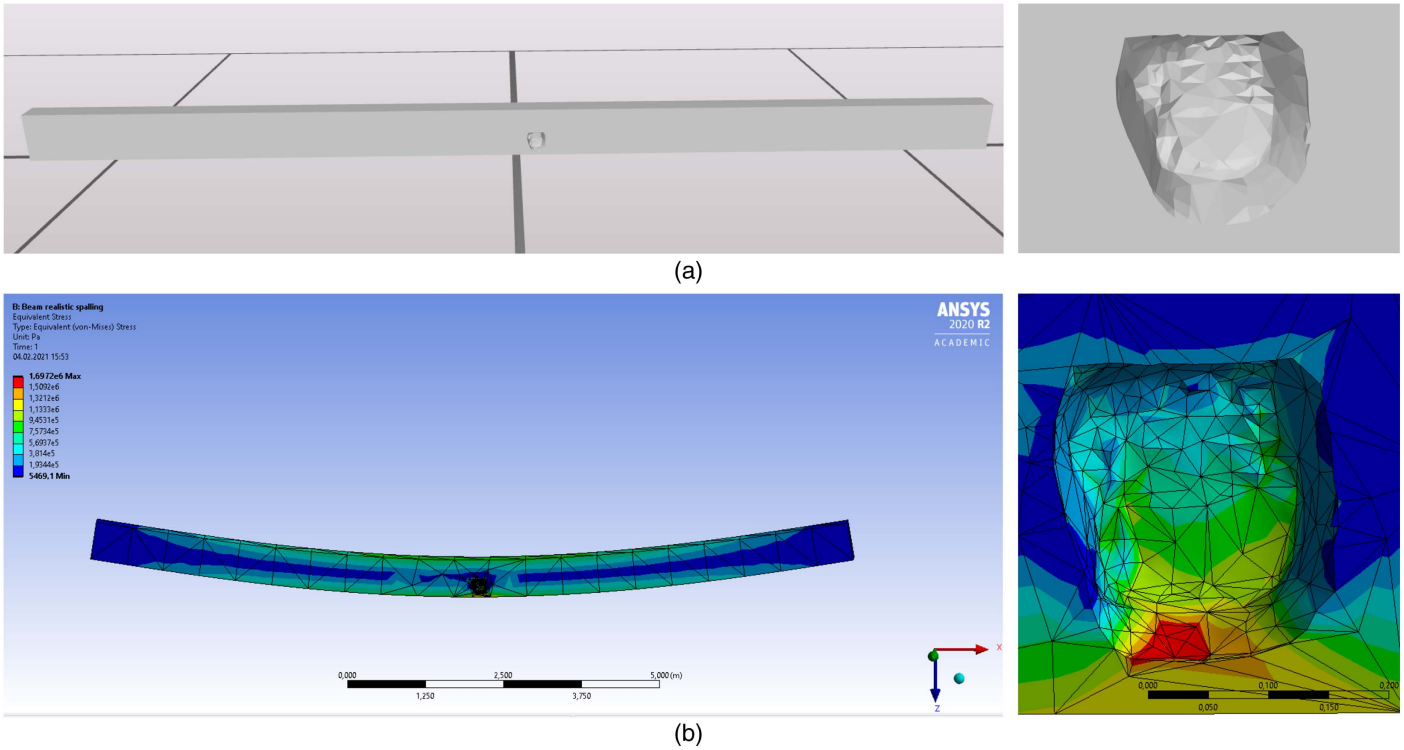


Fig. 23. (a) BIM model of the damaged beam; and (b) FEA model in ANSYS Mechanical showing the equivalent von Mises stress.

How Can This Object-Oriented Model be Implemented Using an Established AEC Data Format?

Out of five possibilities, IFC was chosen for implementating the object-oriented model because it is standardized and supported by numerous software tools. By using *IfcBooleanResult*, *IfcVoidingFeature*, *IfcImageTexture*, *IfcGeometricRepresentationContext*, *IfcTextureCoordinate*, *IfcDocumentReference*, *IfcPropertxSet*, and *IfcProperty*, all designed concepts could be implemented without extensions of the IFC. This has the advantage that all models based on this concept can be used in current and future inspection software.

Can the Object-Oriented Model and Its Implementation Be Verified Using Available AEC Software Tools?

Several software programs lack proper implementation or do not support all concepts of the IFC. Most software can handle proxies, aggregations, and assignments. If a software has a hierarchical view that is not fulfilled by all tested software tools, the semantic data are displayed properly. Some problems occur using classifications because several software tools struggle to properly include classification information. Even fewer software programs supported the concept of cutouts, especially if it is implemented using *IfcRelVoidsElement*. None of the tested tools offered the user the option for selecting a representation context or visualizing a texture. To test textures and geometry selection, xBIM Xplorer was extended. The extended version of xBIM Xplorer was capable of all the required concepts.

In summary, the paper has the following contributions:

1. A DIM was developed that synergizes existing concepts and models, and has been extended as follows:
2. As opposed to existing concepts, this DIM now
   - supports multiple geometries of a defect,
   - allows inclusion of multiple photos of a single defect,
   - incorporates necessary texture-mapping information to deliver required texture-mapping information to the visualization, and
   - allows multiple views of a defect, for example, a view with a texture or with a geometry.
3. Existing software packages and extensions were tested regarding the support of the damage data model.
4. Test cases and a case study were conducted to evaluate the proposed damage information modeling concept.

Conclusion and Outlook

Damaged building models can be visualized for assessments and transferred to structural simulation environments using the presented data model. The consecutive steps of a bridge inspection require different types and details of information. Sacks et al. (2018a) proposed an IDM and model view definition (MVD), which lacks geometric and visualization data of defects. Future work must address the IDM and MVD again with respect to geometric and visual data, additional domains, such as non- or low-destructive testing, and include further damage types, for example, material changes or divergences from specification.

With the use of multiple geometries, multiple components and defect states may be stored. Furthermore, textures provide visual information about the defect. The model stores semantic, geometric, and visual data from visual inspections. Storing data from UAS inspections, i.e., photos or videos, can be included as well. However, to verify the usability of the data model in combination with UASs, further tests are required. The same applies for detailed FEAs because the described FEA was only an example without validation of the outputs.

Applying textures to geometries delivers additional information to the engineer. Textures need rectified photos and the related texture-mapping algorithms. The texture-mapping algorithm depends on the geometry and the aimed visualization. Hence, the

texture-mapping algorithm must be transferred together with the texture image. Both can be stored in the IFC file.

The proposed model aims to support different views of a defect, such as an undamaged component, defect geometry, defect photos, and the damaged component. This was achieved by adding geometric representation contexts to the 3D geometries and allowing the user to select the desired context. The drawback of the proposed approach is that components that do not have a geometry in the selected context are not displayed. Special views need to be defined instead of simple geometric representation contexts to overcome this issue.

All test files were written manually. This is impractical in case of generating buildings or structures with several defects. A proper way for editing defects at structures is necessary to enable the use of the IFC during the operation phase.

Incomplete or invalid implementation remains the biggest issue of the IFC. Neither proprietary software, such as Autodesk Revit nor open source software, for example, xBIM Xplorer, support all concepts, classes, and entities of IFC 4. xBIM Xplorer was extended in this study. However, further improvements are required. By improving documentation, training, examples, and assistance regarding the IFC standard, software developers could be empowered with the knowledge required for better implementation.

Data Availability Statement

Some or all data, models, or code generated or used during the study are available in a repository online in accordance with funder data retention policies:

- Source code: https://github.com/Noranius/XbimWindowsUI
- Test files: https://www.bitbucket.org/damageinformationmodelingteam/ifc-in-depth-test-files.git

Acknowledgments

We thank the "Thüringer Landesamt für Bau und Verkehr" (Thuringian Department for Construction and Transport) for providing us with data and helping us with their practical expertise.

References

Artus, M., and C. Koch. 2020. "State of the art in damage information modeling for RC bridges—A literature review." *Adv. Eng. Inf.* 46 (Oct): 101171. https://doi.org/10.1016/j.aei.2020.101171.

Barazzetti, L., F. Banfi, R. Brumana, G. Gusmeroli, D. Oreni, M. Previtali, F. Roncoroni, and G. Schiantarelli. 2015. "BIM from laser clouds and finite element analysis: Combining structural analysis and geometric complexity." *ISPRS–Int. Archiv. Photogramm., Remote Sens. Spatial Inf. Sci.* XL-5/W4: 345–350. https://doi.org/10.5194/isprsarchives-XL-5-W4-345-2015.

Barazzetti, L., F. Banfi, R. Brumana, M. Previtali, and F. Roncoroni. 2016. "BIM from laser scans. Not just for buildings: Nurbs-based parametric modeling of a medieval bridge." *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.* III-5: 51–56. https://doi.org/10.5194/isprs-annals-III-5-51-2016.

Booch, G. 2007. *Object-oriented analysis and design with applications*. 3rd ed. *The Addison-Wesley Object Technology Series*. Upper Saddle River, NJ: Addison-Wesley.

Borrmann, A., M. König, C. Koch, and J. Beetz. 2018. *Building information modeling: Technology foundations and industry practice*. 2nd ed. Cham, Switzerland: Springer International.

buildingSMART International Ltd. 2009. "BIM collaboration format (BCF)—An introduction." Accessed September 21, 2020. https://technical.buildingsmart.org/standards/bcf/.

buildingSMART International Ltd. 2016. "Industry foundation classes 4.0.2.1: Version 4.0—Addendum 2—Technical corrigendum 1." Accessed June 1, 2018. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/.

buildingSMART International Ltd. 2018a. "IFC infrastructure deployments." Accessed May 17, 2022. https://www.buildingsmart.org/standards/rooms/infrastructure/ifc-bridge/.

buildingSMART International Ltd. 2018b. "Industry foundation classes version 4.2 bSI candidate standard: IFC bridge extension." Accessed April 10, 2019. https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/.

buildingSMART International Ltd. 2020. "IFC specifications database." Accessed February 15, 2021. https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/.

Calvert, G., L. Neves, J. Andrews, and M. Hamer. 2020. "Multi-defect modelling of bridge deterioration using truncated inspection records." *Reliab. Eng. Syst. Saf.* 200 (Aug): 106962. https://doi.org/10.1016/j.ress.2020.106962.

Eastman, C. M. 2011. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors*. 2nd ed. Hoboken, NJ: Wiley.

Fröhlich, J. 2020. "On systematic approaches for interpreted information transfer of inspection data from bridge models to structural analysis." Master's thesis, Faculty of Civil Engineering, Bauhaus Universität Weimar.

Hamdan, A.-H., M. Bonduel, and J. R. Scherer. 2019. "An ontological model for the representation of damage to constructions." In *Proc., 7th Linked Data in Architecture and Construction Workshop*, edited by M. Poveda-Villalón, P. Pauwels, R. de Klerk, and A. Roxin, 64–77. Madrid, Spain: Universidad Politécnica de Madrid.

Hamdan, A.-H., and J. R. Scherer. 2018. "A generic model for the digitalization of structural damage." In *Life cycle analysis and assessment in civil engineering*, edited by R. Caspeele, L. Taerwe, and D. M. Frangopol. London: Chapman and Hall.

Hearn, G. 2007. *Bridge inspection practices*. Washington, DC: Transportation Research Board.

Heckbert, P. S. 1989. "Fundamentals of texture mapping and image warping." Master's thesis, Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley.

Hurt, M., and S. D. Schrock. 2016. "Bridge inspection and evaluation." In *Highway bridge maintenance planning and scheduling*, 99–154. San Diego: Elsevier.

Hüthwohl, P., I. Brilakis, A. Borrmann, and R. Sacks. 2018. "Integrating RC bridge defect information into BIM models." *J. Comput. Civ. Eng.* 32 (3): 04018013. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000744.

IFC Infra. 2019a. "IFC-Bridge Projektabschluss." Accessed June 26, 2019. http://ifcinfra.de/events/ifc-bridge-projektabschluss/.

IFC Infra. 2019b. "ifcbridge-model01.zip." Accessed November 4, 2020. https://ifcinfra.de/wp-content/uploads/2019/07/ifcbridge-model01.zip.

IfcOpenShell. 2015. "IfcConvert." Accessed February 8, 2021. http://ifcopenshell.org/ifcconvert.

Isailovic, D., V. Stojanovic, M. Trapp, R. Richter, R. Hajdin, and J. Döllner. 2020. "Bridge damage: Detection, IFC-based semantic enrichment and visualization." *Autom. Constr.* 112 (Apr): 103088. https://doi.org/10.1016/j.autcon.2020.103088.

ISO. 2018. *Industry foundation classes (IFC) for data sharing in the construction and facility management industries*. ISO 16739:2018. London: ISO.

Kropp, C., C. Koch, and M. König. 2018. "Interior construction state recognition with 4D BIM registered image sequences." *Autom. Constr.* 86 (Feb): 11–32. https://doi.org/10.1016/j.autcon.2017.10.027.

Lee, D.-Y., H.-L. Chi, J. Wang, X. Wang, and C.-S. Park. 2016. "A linked data system framework for sharing construction defect information using ontologies and BIM environments." *Autom. Constr.* 68 (Aug): 102–113. https://doi.org/10.1016/j.autcon.2016.05.003.

Liu, K., and N. El-Gohary. 2016. "Semantic modeling of bridge deterioration knowledge for supporting big bridge data analytics." In *Proc., Construction Research Congress 2016*, edited by J. L. Perdomo-Rivera,

A. Gonzáles-Quevedo, C. L. del Puerto, F. Maldonado-Fortunet, and O. I. Molina-Bas, 930–939. Reston, VA: ASCE.

McGuire, B., R. Atadero, C. Clevenger, and M. Ozbek. 2016. "Bridge information modeling for inspection and evaluation." *J. Bridge Eng.* 21 (4): 04015076. https://doi.org/10.1061/(ASCE)BE.1943-5592.0000850.

Morgenthal, G., N. Hallermann, J. Kersten, J. Taraben, P. Debus, M. Helmrich, and V. Rodehorst. 2019. "Framework for automated UAS-based structural condition assessment of bridges." *Autom. Constr.* 97 (Jan): 77–95. https://doi.org/10.1016/j.autcon.2018.10.006.

Nagel, L.-M., M. Pauly, V. Mucha, J. Setzer, and F. Wilhelm. 2016. "Wettlauf gegen den Verfall." Accessed September 27, 2018. https://www.welt.de/print/welt_kompakt/article157003133/Wettlauf-gegen-den-Verfall.html.

Ren, G., R. Ding, and H. Li. 2019. "Building an ontological knowledgebase for bridge maintenance." *Adv. Eng. Software* 130 (Apr): 24–40. https://doi.org/10.1016/j.advengsoft.2019.02.001.

Sacks, R., et al. 2018a. "SeeBridge as next generation bridge inspection: Overview, information delivery manual and model view definition." *Autom. Constr.* 90 (Jun): 134–145. https://doi.org/10.1016/j.autcon.2018.02.033.

Sacks, R., C. M. Eastman, G. Lee, and P. M. Teicholz. 2018b. *BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers*. 3rd ed. Hoboken, NJ: Wiley.

Schach, R., J. Otto, H. Häupel, and M. Fritzsche. 2006. "Lebenszykluskosten von Brücken." *Bauingenieur* 81 (7–8): 7–14.

Störfix. 2012. "Talbrücke Apfelstädt." Accessed April 4, 2022. https://commons.wikimedia.org/wiki/File:Talbrücke-Apfelstädt.jpg.

struppi0601. 2015. "Crack-wall-concrete." Accessed April 4, 2022. https://pixabay.com/photos/crack-wall-concrete-texture-stone-695010/.

Tanaka, F., et al. 2018. "Bridge information modeling based on IFC for supporting maintenance management of existing bridges." In *Proc., ICCCBE 2018 Conf.*, edited by K. Mela, S. Pajunen, and V. Raasakka, 778–785. London: International Society for Computing in Civil and Building Engineering.

Thinkproject. 2019. "DESITE BIM." Accessed October 14, 2019. https://thinkproject.com/products/desite-bim/.

Volk, R., J. Stengel, and F. Schultmann. 2014. "Building information modeling (BIM) for existing buildings—Literature review and future needs." *Autom. Constr.* 38 (Mar): 109–127. https://doi.org/10.1016/j.autcon.2013.10.023.

W3C. 2012. "OWL 2 web ontology language: Document overview." Accessed September 17, 2019. https://www.w3.org/TR/owl2-overview/.

Wan, C., Z. Zhou, S. Li, Y. Ding, Z. Xu, Z. Yang, Y. Xia, and F. Yin. 2019. "Development of a bridge management system based on the building information modeling technology." *Sustainability* 11 (17): 4583. https://doi.org/10.3390/su11174583.