
SeeBridge
Semantic Enrichment Engine for Bridges
Start date: 01/10/2015
Duration: 21 months

Deliverable 4.1

Semantic enrichment engine for BIM – SeeBIM

| | |
|----------------------------|-----------|
| Main Editor(s) | Technion |
| Due Date | 6 |
| Delivery Date | 30/6/2016 |
| Task number | 4.1 |
| Dissemination level | PU |

Current edition: 2nd Edition, updated 24th May, 2017

Acknowledgment

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 31109806.0007. SeeBridge is co-funded by Funding Partners of the ERA-NET Plus Infravation and the European Commission. The Funding Partners of the Infravation 2014 Call are: Ministerie van Infrastructuur en Milieu, Rijkswaterstaat, Bundesministerium für Verkehr, Bau und Stadtentwicklung, Danish Road Directorate, Statens Vegvesen Vegdirektoratet, Trafikverket – Trv, Vegagerðin, Ministère de L'écologie, du Développement Durable et de L'énergie, Centro para el Desarrollo Tecnológico Industrial, Anas S.P.A., Netivei Israel – National Transport Infrastructure Company Ltd, Federal Highway Administration USDOT.

Contents

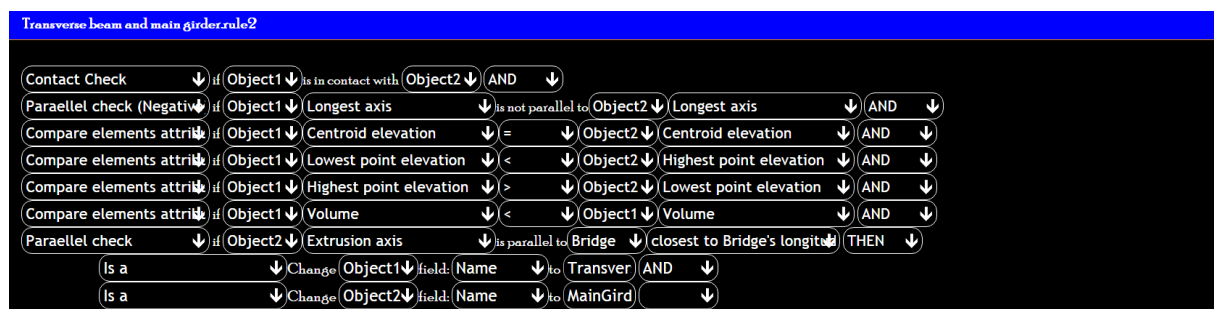
| | |
|--|-----------|
| Acknowledgment | 2 |
| 1 Introduction | 4 |
| 2 SeeBIM 2.0 Operators Reference | 5 |
| 2.1 IF clause operators | 5 |
| 2.1.1 Field Check | 6 |
| 2.1.2 Field Check (Negative)..... | 6 |
| 2.1.3 Adjacency check | 8 |
| 2.1.4 Compare element attribute | 9 |
| 2.1.5 Contact check | 10 |
| 2.1.6 Contained Check..... | 11 |
| 2.1.7 Overlapping check | 12 |
| 2.1.8 Parallel Check..... | 13 |
| 2.1.9 Parallel Check (negative) | 13 |
| 2.1.10 Material Check | 14 |
| 2.2 THEN Clause operators | 15 |
| 2.2.1 Create Axis | 16 |
| 2.2.2 Is a..... | 17 |
| 2.2.3 Create element..... | 19 |
| 2.2.4 Lengthen Occluded Girder | 20 |
| 3 Development of SeeBIM 2.0 | 21 |
| Introduction | 22 |
| Previous Work | 23 |
| Methodology | 25 |
| Rule-based inference | 25 |
| Merging BIM model data with information from external sources | 33 |
| Enhanced Geometric and Topological Operators | 35 |
| Shape representation..... | 35 |
| Spatial and Topological Relationships and Operators | 37 |
| Support for Tolerances in Spatial Operators | 39 |
| Full-scale bridge test | 41 |
| Discussion | 45 |
| Conclusions | 46 |
| References | 47 |

1 Introduction

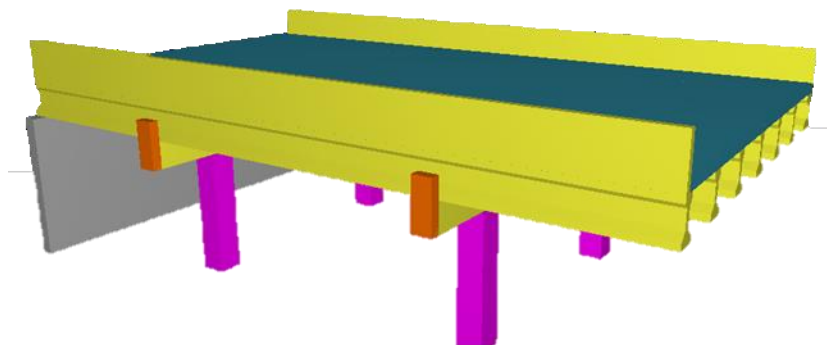
The Technion and TUM teams have developed the beta version of the semantic enrichment software. Semantic enrichment refers to the automatic or semi-automatic addition of meaningful information to a digital model of a building or other structure by software that can deduce new information by processing rules. The software developed uses a run time forward chaining rule-processing engine to classify, label, aggregate and infer obscured objects in a BIM model based on the geometry and topological relationships of the components. It is provided through a web interface and is available at:

<http://vclab.technion.ac.il> and from the SeeBridge project web site.

The software includes functions for parsing the existing bridge model, testing for geometrical and topological relationships, and for creating new objects, properties and relationships and adding them to the model. The rule-processing inference in the software is defined as *IF-THEN* rules using a predefined set of objects, relational and logical operators, expressed in a format comprehensible to domain experts who are not programmers. The figure below shows an example of the interface with a rule for classification of elements of a girder bridge. The rule checks for a set of conditions concerning two objects, and if they evaluate as TRUE, then the objects are classified as a Transverse Beam and a Main Girder.



A synthetic bridge model was prepared for testing the system. The model contains 59 objects of eight typical types of bridge elements. All of the objects in the model were correctly identified using the rules compiled.




Complete details of the preparation of the SEEBIM software can be found in the journal paper provided in the appendix to this deliverable.

2 SeeBIM 2.0 Operators Reference

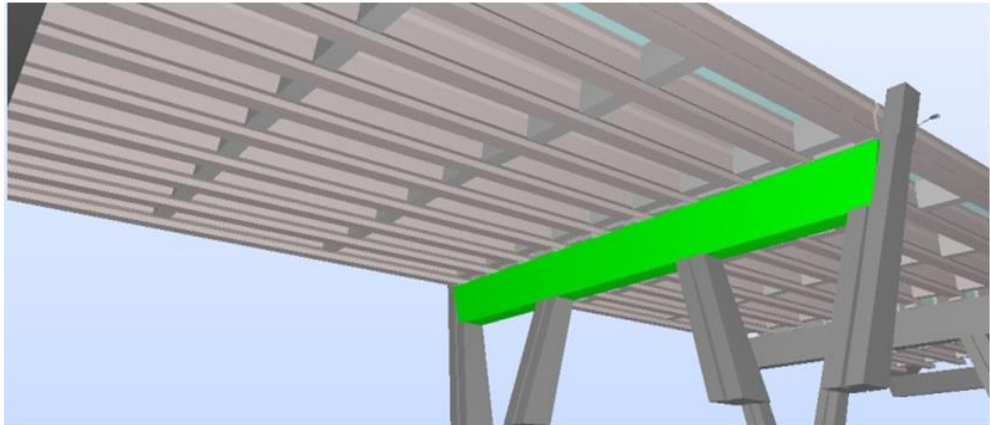
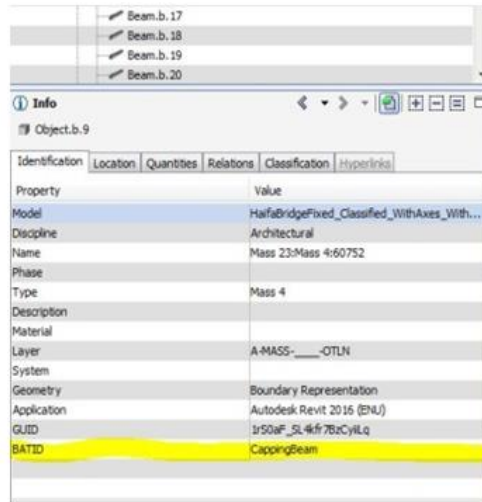
All Tier 2 operators return Boolean constants: *true* or *false*. The operators can be called on one or two objects. There are also operators that are called just by their name, for example, *are_equal_to* operator. The full list of implemented Tier 2 operators is given below.

2.1 IF clause operators

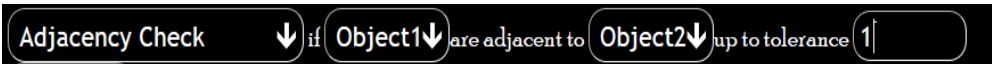
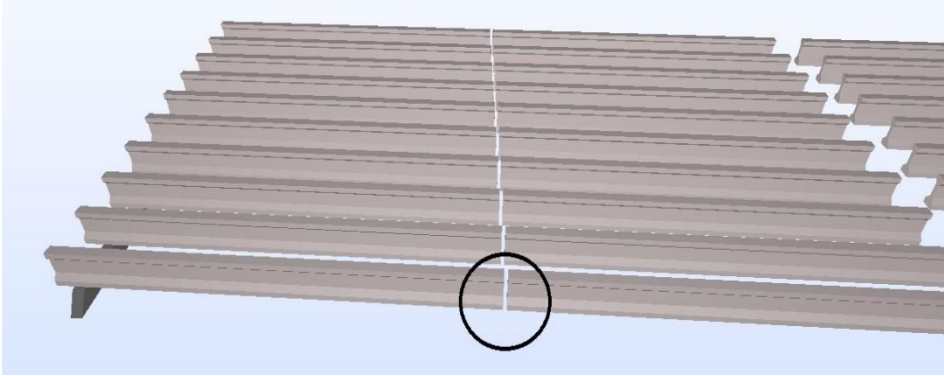
The IF operators execute a statement if a specified condition is found to be true. These operators are generally a part of the first half of the rule, where a condition check is required to progress to the execution of the rest of the rule. If the condition that follows the operator is valid, then the operator is executed. Several rules in SeeBIM 2.0 contain multiple IF statements to be able to accurately obtain the desired outcome. An IF clause is generally followed by a THEN clause, which encompasses the second half of the rule.

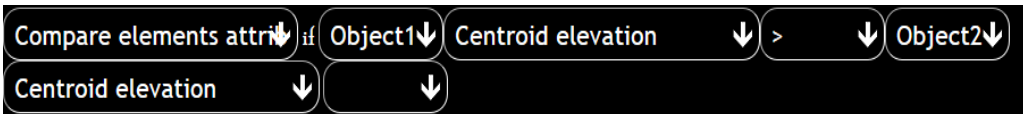
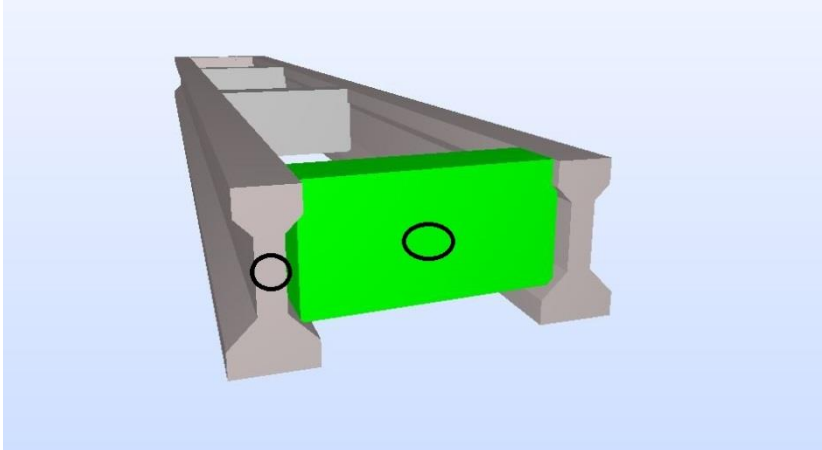
| | |
|-------------------------------------|---|
| 2.1.1 Field Check | |
| 2.1.2 Field Check (Negative) | |
| Number of Objects | 1 |
| Description | The operator evaluates value of different geometric and non-geometric attributes of an object to find out if they are greater than, smaller than or equal to (or not equal to) the designated value. It allows a check for the following sub-domains: Element Type, Object Type, Tag, Domain Type, Description, Extrusion Axis Length, Centroid Elevation, Max/Min X/Y coordinates, X/Y/Z Dimensional ratios, Centroid lateral/longitudinal coordinates, Volume, Lowest/Highest point elevations and P21line. |
| Structure | <p><Field Check> if <Object1> <sub-Domain> equals [<i>element</i>]</p> <p>Where: Element: is any relevant element in the bridge that the rule can be applied on</p> <p>The sub-domain list includes:</p> <p>Element Type, Object Type, Tag, Domain Type, Description, Extrusion Axis Length, Centroid Elevation, Max/Min X/Y coordinates, X/Y/Z Dimensional ratios, Centroid lateral/longitudinal coordinates, Volume, Lowest/Highest point elevations and P21line.</p> |
| Interface |  |


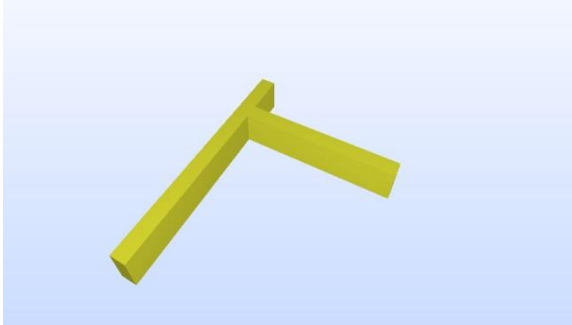
Example

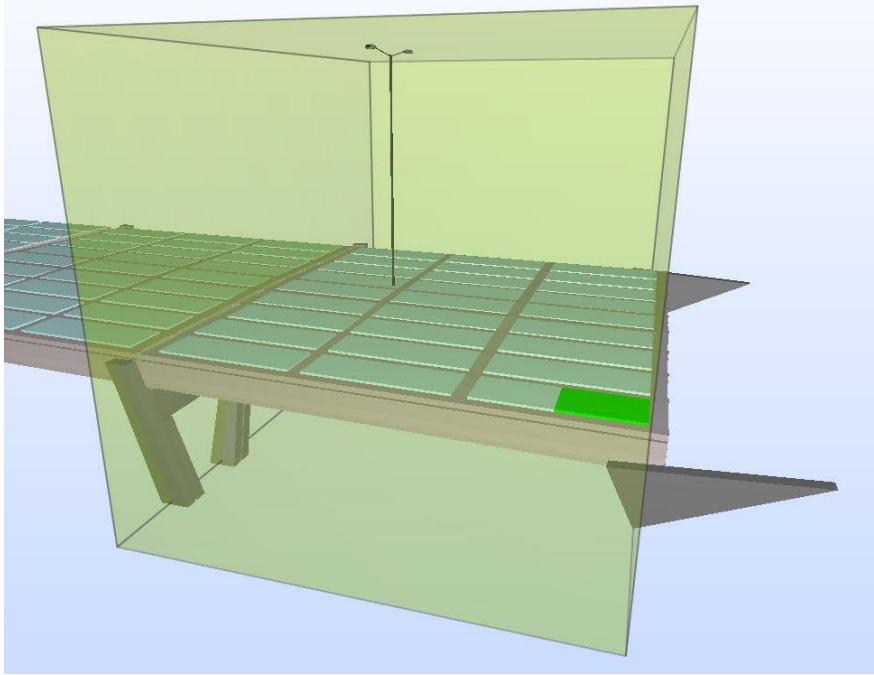


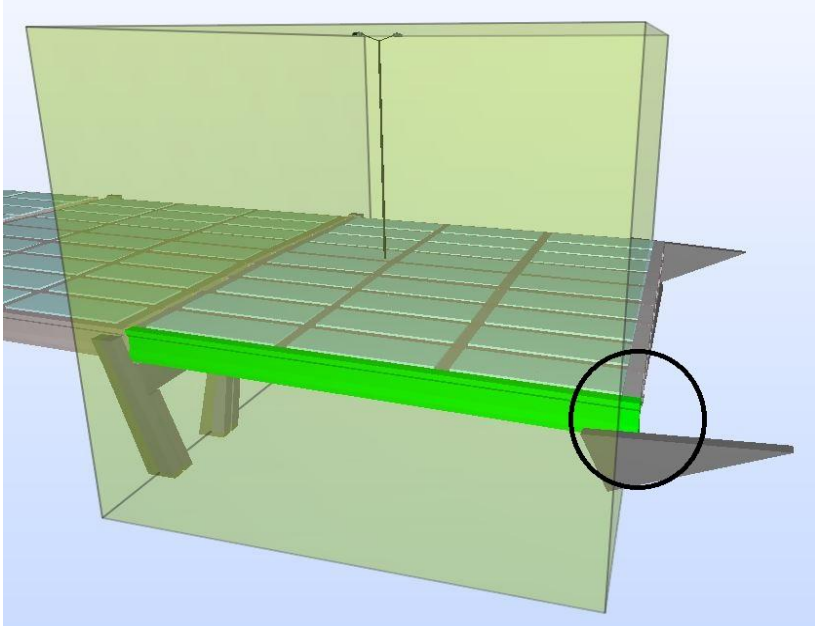
In this example, the operator field checks if the green element is a capping beam.


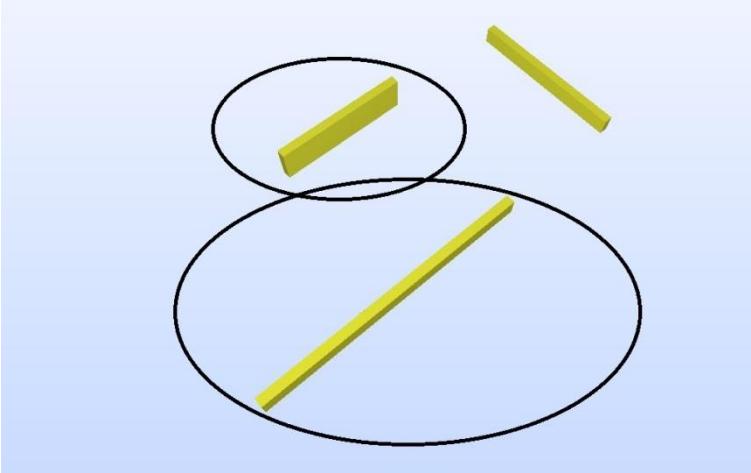
| 2.1.3 Adjacency check | |
|------------------------------|---|
| Number of Objects | 2 |
| Description | The operator checks and evaluates the distances between all sides of any two objects in the bridge. If the distance between any two faces is less than or equal to a given tolerance, the operator returns true. The operator compares vertical narrow faces, vertical wide faces, all horizontal faces, top horizontal faces, or bottom horizontal faces, of any two intended objects. |
| Structure | <p>if <object1> is adjacent to <object2> up to tolerance [<i>tolerance by user</i>]</p> <p>Where:</p> <p><i>Tolerance by user:</i> is defined as appropriate by the user, with a recommended range of no more than a few millimetres.</p> |
| Interface |  |
| Example |  <p>In this example, the operator checks if the two girders are adjacent.</p> |


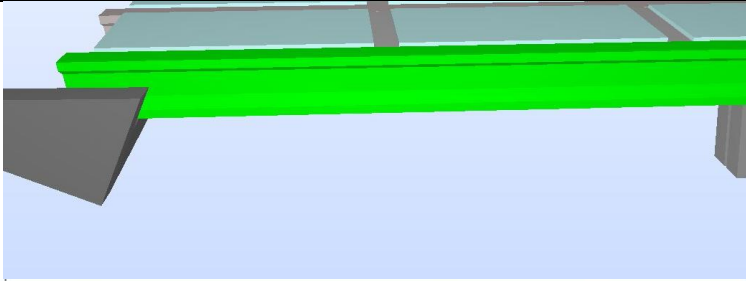
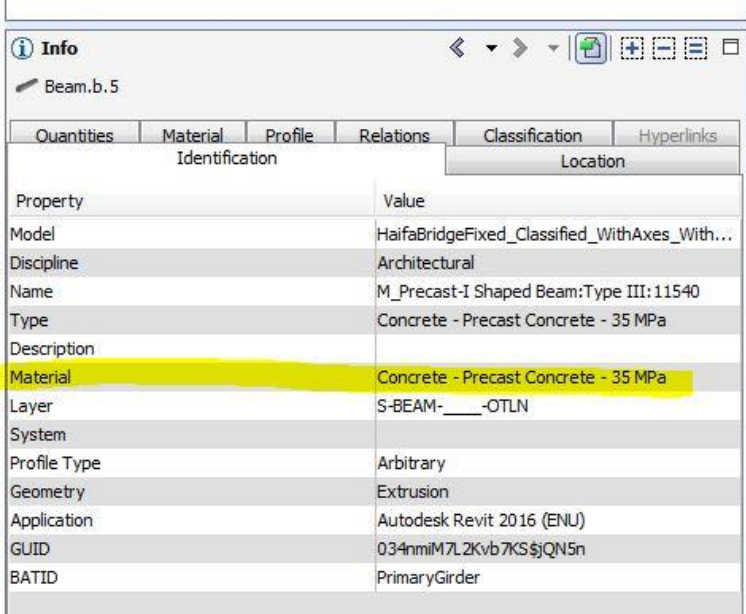
| 2.1.4 Compare element attribute | |
|--|---|
| Number of Objects | 2 |
| Description | The operator compares and evaluates the values of different geometric and non-geometric attributes of two objects and finds out if the given numeric operator is true. The numeric sub-operators include: greater than, smaller than, greater than or equal to, smaller than or equal to, and equal to. |
| Structure | <p>if <object1> <attribute> <numeric operator> <object2> <attribute></p> <p>where:</p> <p>attribute: any property field name of the BIM object</p> <p>numeric operators: one of <, >, <=, >=, = (greater than, smaller than, smaller than or equal, greater than or equal to, equal to)</p> |
| Interface |  |
| Example |  <p>In the example, the operator returns that the centroid elevation of object 1 is higher than that of object 2.</p> |

| 2.1.5 Contact check | |
|----------------------------|---|
| Number of Objects | 2 |
| Description | The operator checks for physical geometric contact between two BIM objects. It makes use of Query language for BIM (QL4BIM), an engine which facilitates optimized geometric algorithms and spatial indexing, and performs the required checks. The difference between 'Contact Check' and 'Adjacency Check' is that there is a flexible tolerance defined by the user to determine the degree of physical proximity of two given object in 'Adjacency Check', whereas the tolerance is practically zero in 'Contact Check.' Adjacency is based on bounding boxes and contact and If there is found to be geometric contact, the operator returns true. |
| Structure | <Contact Check> if <Object1> is in contact with <Object2> |
| Interface |  |
| Example |  <p>In this example, the operator returns that object 1 and 2 are in contact.</p> |

| 2.1.6 Contained Check | |
|------------------------------|---|
| Number of Objects | 2 |
| Description | The operator checks if one object is fully contained in another. This is done by checking if the object 1 Oriented Bounding Box is fully contained within that of the object 2. Finally, the operator returns true if the aforementioned exists. |
| Structure | if <object1> is contained within <object2> |
| Interface | <div style="border: 1px solid black; padding: 5px; display: inline-block;"> Contained Check ↓ if Object1 ↓ is contained within Object2 ↓ ↓ </div> |
| Example | <div style="text-align: center;">  </div> <p>In this example, the deck is fully contained in the bounding box.</p> |

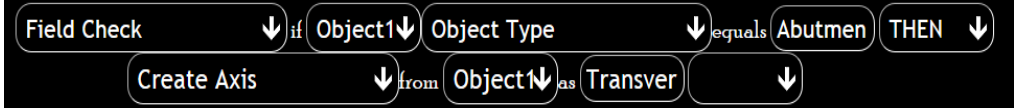
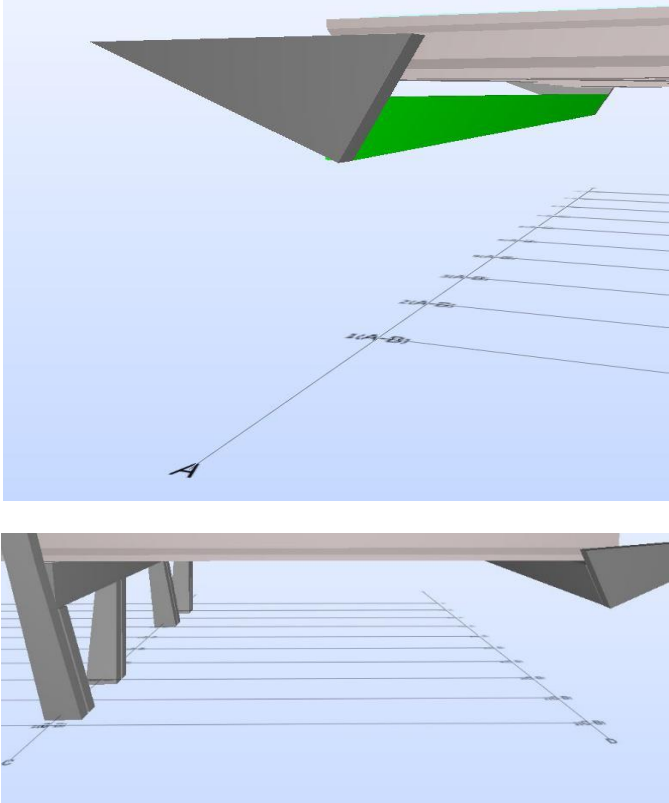
| 2.1.7 Overlapping check | |
|--------------------------------|--|
| Number of Objects | 2 |
| Description | The operator first evaluates and checks if object 1 is not fully contained in object 2, then checks if it is partially contained. |
| Structure | <Overlapping Check> if <object1> is overlapping with <object2> |
| Interface | <div style="border: 1px solid black; padding: 5px; background-color: #f0f0f0;"> Overlapping Check ↓ if Object1 ↓ is overlapping with Object2 ↓ ↓ </div> |
| Example | <div style="text-align: center;">  </div> <p>In this example, the operator returns that the girder is overlapping with the abutment.</p> |


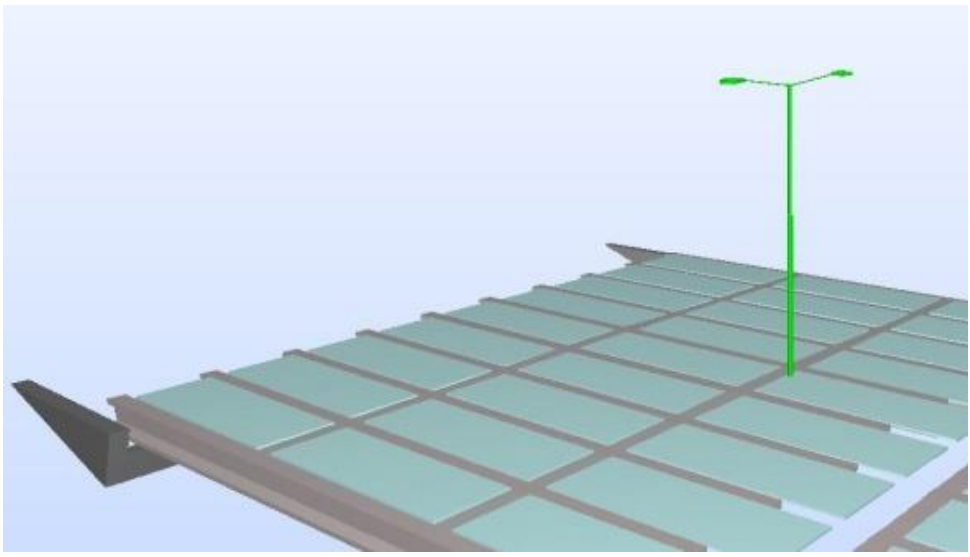
| | |
|--|---|
| 2.1.8 Parallel Check | |
| 2.1.9 Parallel Check (negative) | |
| Number of Objects | 2 |
| Description | <p>The operator evaluates if the two given objects are parallel to each other. This is done by determining if the axes vectors of the two are parallel. In the case when the entire bridge is selected as an object, one of the following can be chosen as an axis vector: longitudinal, lateral, elevation.</p> <p>Parallel Check (Negative) evaluates the inverse of Parallel Check and returns true if the objects being evaluated are not found to be parallel.</p> |
| Structure | <p>if <object1> <Axis/Vector> is parallel to <object2> <Axis/vector></p> <p><Axis/Vector> list: Extrusion axis, longest axis and shortest axis</p> |
| Interface |  |
| Example |  <p>In this example, the operator, Parallel Check, returns that the circled objects are parallel.</p> |

| | |
|--------------------------|--|
| 2.1.10 | Material Check |
| Number of Objects | 1 |
| Description | The operator evaluates the value of the material property of an object to find out if it equals the input value. |
| Structure | <p><Material Check> if <Object1> is made of "[Material Name]"</p> <p>Where: <i>Material Name</i>: Material that needs to be checked</p> |
| Interface |  |
| Example |   <p>In this example, the rule checks if the material property of the bridge girder (top image) is "Concrete – Precast Concrete – 35 MPa".</p> |

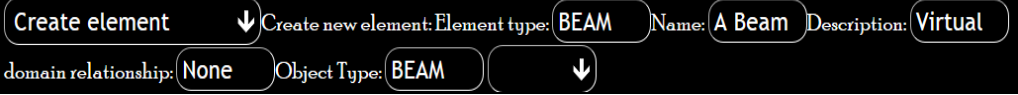
2.2 THEN Clause operators

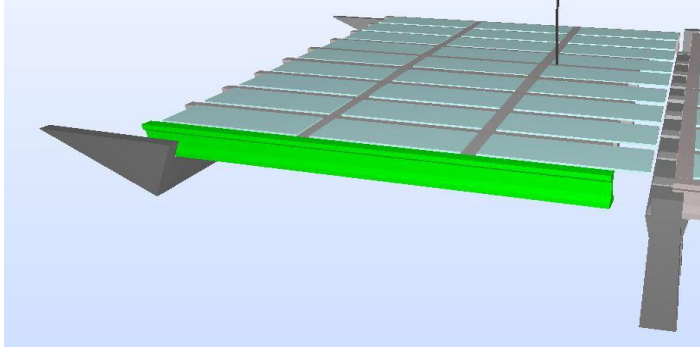
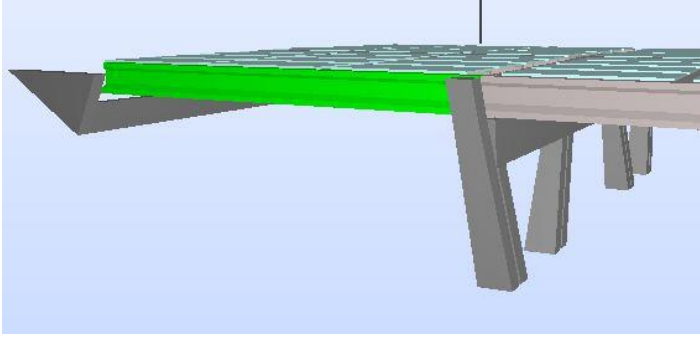
The THEN clause is executed only if the presumed condition or multiple conditions given in the IF clause are fulfilled in the first part of the rule (IF clause). The primary functions of most of the THEN clauses in SeeBIM 2.0 include creation of axes, classification of objects into their respective fields, fixation of any occluded parts or areas, aggregation and disaggregation, and retrieving related objects. Like IF clauses, THEN clauses in a rule may contain more than one statement and multiple tasks may therefore be executed. Following is a detailed documentation of all the THEN clauses operating in SeeBIM 2.0.

| 2.2.1 Create Axis | |
|--------------------------|--|
| Number of Objects | 1 |
| Description | The operator identifies different geometries, such as girders, capping beams, transverse beams, etc. and creates either a lateral or a longitudinal axis by projecting the shape on the XY plane and creating an axis along the centre line of the object. As an input, any part of the bridge, or the entire bridge, can be selected. In case the user selects 'Bridge' as the object, the operator creates longitudinal and lateral axes for all relevant components. |
| Structure | <create axis>from<object1> as "" AND/OR/END |
| Interface |  |
| Example |  <p>In this example, the operator creates an axis underneath the abutment after fulfilling the Field Check condition.</p> |

| | |
|--------------------------|--|
| 2.2.2 Is a | |
| Number of Objects | 1 |
| Description | <p>The operator classifies objects in their respective groups, such as element type, object type and domain type. It can also be used to name and tag, and to add description, orientation and proportion to elements.</p> <p>As an input, any part of the bridge or the entire bridge can be selected.</p> <p>The user has the following sub-options available for classification within the operator: Element Type, Object Type. Tag, Domain Type, Name, Description, P2lline, Orientation, Proportion and User Field.</p> |
| Structure | <p><Is a> add <Object1> field [<i>property name</i>] to [<i>value</i>]</p> <p>Where:</p> <p>Property name: Element Type, Object Type. Tag, Domain Type, Name, Description, P2lline, Orientation, Proportion or a User named field</p> <p>Value: A text or numerical value</p> |
| Interface |  |
| Example | <p>In this example, the operator assigns a tag of 'LightFixure' to the lamp post.</p>  |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|-----------------|---------------------------|-------------|--|-----------------|------------|-------|------------------|--------|--|----------|-------------------------|-------------|---------------------------|------|-------------------------|-------|--------|-------------|--|-----------------|------------|-------|------------------|--------|--|----------|-------------------------|-------------|---------------------------|------|-------------------------|-------|--------------|--|
| <p>Before:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Type</td><td>ARC-L-R1-H150-AL1-xx-</td></tr> <tr><td>Description</td><td></td></tr> <tr><td>Functional Type</td><td>NOTDEFINED</td></tr> <tr><td>Layer</td><td>E-LITE-EQPM-OTLN</td></tr> <tr><td>System</td><td></td></tr> <tr><td>Geometry</td><td>Boundary Representation</td></tr> <tr><td>Application</td><td>Autodesk Revit 2016 (ENU)</td></tr> <tr><td>GUID</td><td>2cr2CYr\$5ANg_hFMM0m9fe</td></tr> <tr><td>BATID</td><td>206872</td></tr> </table> <p>After:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Description</td><td></td></tr> <tr><td>Functional Type</td><td>NOTDEFINED</td></tr> <tr><td>Layer</td><td>E-LITE-EQPM-OTLN</td></tr> <tr><td>System</td><td></td></tr> <tr><td>Geometry</td><td>Boundary Representation</td></tr> <tr><td>Application</td><td>Autodesk Revit 2016 (ENU)</td></tr> <tr><td>GUID</td><td>2cr2CYr\$5ANg_hFMM0m9fe</td></tr> <tr><td>BATID</td><td>LightFixture</td></tr> </table> <p><i>Note: BATID represents the 'tag' field in the Solibri viewer.</i></p> | Type | ARC-L-R1-H150-AL1-xx- | Description | | Functional Type | NOTDEFINED | Layer | E-LITE-EQPM-OTLN | System | | Geometry | Boundary Representation | Application | Autodesk Revit 2016 (ENU) | GUID | 2cr2CYr\$5ANg_hFMM0m9fe | BATID | 206872 | Description | | Functional Type | NOTDEFINED | Layer | E-LITE-EQPM-OTLN | System | | Geometry | Boundary Representation | Application | Autodesk Revit 2016 (ENU) | GUID | 2cr2CYr\$5ANg_hFMM0m9fe | BATID | LightFixture | |
| | Type | ARC-L-R1-H150-AL1-xx- | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Functional Type | NOTDEFINED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Layer | E-LITE-EQPM-OTLN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | System | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Geometry | Boundary Representation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Application | Autodesk Revit 2016 (ENU) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | GUID | 2cr2CYr\$5ANg_hFMM0m9fe | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | BATID | 206872 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Functional Type | NOTDEFINED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Layer | E-LITE-EQPM-OTLN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | System | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Geometry | Boundary Representation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Application | Autodesk Revit 2016 (ENU) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | GUID | 2cr2CYr\$5ANg_hFMM0m9fe | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | BATID | LightFixture | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| 2.2.3 Create element | |
|-----------------------------|--|
| Number of Objects | 1 |
| Description | <p>The operator creates a new element, and gives the user the option of choosing the element type, domain relationships and object type. The user can further add the name and description of the element. Note: this operator does not add geometry, it only adds the object itself, so that it can be seen in the object hierarchy.</p> |
| Structure | <p><Create element> Create New Element: Element type [<i>element type</i>] Name: [<i>name</i>] Description: [<i>description</i>] domain relationship: [<i>relationship</i>] Object Type: [<i>object type</i>]</p> <p>Where element type: a legal IFC entity type name: any name text description: a text description of the new element relationship: Object type: a user-defined object type, usually defined in the domain MVD (model view definition) document.</p> |
| Interface |  |
| Example | <p>The operator will add a new line to the ifc file. The following is an example of the object that will be created with the above interface line:</p> <pre>#831= IFCBEAM('1mbQDPWNbCoBmizWeqYVhH',#41,'A Beam',\$,'Virtual',\$,\$,'BEAM');</pre> |

| 2.2.4 Lengthen Occluded Girder | |
|---------------------------------------|--|
| Number of Objects | 1 |
| Description | <p>Where the 3D geometry of bridge girders has been rebuilt from point cloud data, their ends are often missing due to occlusions of the girder's ends from the laser scanner. For these cases, this operator corrects the lengths of any occluded ends of bridge girders by determining the original design intent for the minimum spacing between the ends of contiguous girders or the minimal support length, and extending the girders to the correct length. The original elements are copied by the operator, and edited to add the new reconstructed shape.</p> <p>The IF part of the rule should test if object1 is a girder.</p> |
| Structure | <Lengthen Occluded Girder> <Object1> |
| Interface | <div style="background-color: #333; color: white; padding: 5px; border: 1px solid #ccc;"> Field Check ↓ if Object1 ↓ Object Type ↓ equals Girder THEN ↓ Lengthen Occluded Gir ↓ of Object1 ↓ ↓ </div> |
| Example | <p>Before:</p>  <p>After:</p>  <p>In this example, the operator lengthens the occluded girder to reach the capping beam.</p> |

3 Development of SeeBIM 2.0

ASCE Journal of Computing in Civil Engineering

ENHANCEMENT OF A SEMANTIC ENRICHMENT TOOL FOR BUILDING INFORMATION MODELING

Rafael Sacks¹, Ling Ma², Raz Yosef³, Andre Borrmann⁴, Simon Daum⁵, Uri Kattel⁶

¹Assoc. Prof. of Faculty of Civil and Environmental Engineering, Technion, cvsacks@technion.ac.il

²Postdoctoral Researcher of National Building Research Institute, Technion, lingma@technion.ac.il

³Lab Assistant of National Building Research Institute, Technion, razyos@gmail.com

⁴Professor of Chair of Computational Modeling and Simulation, Technische Universitat Munchen, andre.borrmann@tum.de

⁵Research Assistant of Chair of Computational Modeling and Simulation, Technische Universitat Munchen, simon.daum@tum.de

⁶Graduate student of Faculty of Civil and Environmental Engineering, Technion, uri.kattel@gmail.com

Abstract: Semantic enrichment of building models adds meaningful domain- or application-specific information to a digital building model. It is applicable to solving interoperability problems and to compilation of models from point cloud data. The SeeBIM (semantic enrichment engine for BIM) prototype software encapsulates domain expert knowledge in computer readable rules for inference of object types, identity and aggregation of systems. However, it is limited to axis-aligned bounding box geometry and the adequacy of its rule-sets cannot be guaranteed. This research has solved these drawbacks by a) devising a new procedure for compiling inference rule sets that are known *a priori* to be adequate for complete and thorough classification of model objects, and b) enhancing the operators to compute complex geometry and enable precise topological rule processing. The procedure for compiling adequate rule sets is illustrated using a synthetic small-scale concrete highway bridge model. A full-scale highway bridge model, with 333 components of 13 different types and compiled from a laser scanned point cloud, was used to validate the approach and test the enhanced SeeBIM system. All of the elements were classified correctly, demonstrating the efficacy of the approach to semantic enrichment.

Keywords: Building Information Modeling, semantic enrichment, solid geometry, topology.

Introduction

Semantic enrichment of building models refers to the automatic or semi-automatic addition of meaningful information to a digital model of a building or other structure by software that can deduce new information by processing rules (Belsky et al. 2016). The inputs are an existing building model, information about the building from other sources (such as a database), and a set of rules that encapsulate expert knowledge of the domain. The rules use the existing information and evaluate the topological, spatial, geometric and other relationships between the model's objects. The output is a digital building model that incorporates the new information – new objects, property values, and/or relationships.

Development of semantic enrichment for models is motivated by the information interoperability problem (Eastman et al. 2011), which hampers the use of Building Information Modeling (BIM), and by the difficulties faced by vendors of commercial BIM software in implementing the standard solution – exchanges based on the Industry Foundation Classes (IFC) (BuildingSmart 2013). Semantic enrichment draws on the foundations laid by research of semantic query languages for BIM (Mazairac and Beetz 2013), semantic rule-checking systems for BIM (Eastman et al. 2009; Pauwels et al. 2011), and BIM model query using spatial and topological relationships (Borrmann and Rank 2009; Daum and Borrmann 2014).

Whereas semantic enrichment is generally considered to be applied to add missing information to building model instance files, it has also been applied to extend the schema of

building information models. Zhang and El-Gohary (2016), for example, identified missing concepts in the IFC schema that were needed to express building code requirements. Semantic enrichment is also useful for compilation of ‘as-is’ or ‘as-built’ BIM models from spatial point cloud data (PCD) collected on site through state-of-the-art surveying technologies, such as laser scanning and photo/videogrammetry (Brilakis et al. 2010; Zeibak-Shini et al. 2016). These large data sets must be converted into 3D primitives and then identified as context-specific objects. Current practice requires intensive operations by experienced BIM modelers, and the problem has attracted many research efforts to automate the procedure (Bosche and Haas 2008; Kashani et al. 2014). Yet the outputs of these systems are not semantically rich BIM models. Information regarding the objects’ identification, relationships and other alphanumeric data are typically missing.

Previous Work

SeeBIM 1.0 (Belsky et al. 2016) (Semantic Enrichment Engine for BIM) is an early software prototype whose primary aim was to establish the feasibility of the approach. As depicted in Figure 1, the tool parses an IFC file to extract objects’ shapes, relationships and other attributes. It then applies forward chaining to infer additional facts about the model, using sets of rules compiled in advance by experts in the domain of interest. It records the results in an enriched IFC file.

Experiments conducted using SeeBIM for two domains – precast concrete modeling (Belsky et al. 2016) and automated detailed design (Aram 2015) – showed how the approach could be used to add information to a model in an IFC file. The input in these efforts consisted of IFC files exported according to the Coordination View (CV) 2.0, which defines the exchange of 3D geometry data and is the only Model View Definition (MVD) commonly supported by BIM authoring tools (BuildingSmart 2010). The output in each case was an enriched IFC file that conforms to the MVD defined for precast concrete. More recently, Ramaji and Memari (Ramaji and Memari 2016) illustrated a similar idea: identification of structural features, such as a beam-column joints, in a building model exported from an architectural BIM tool, and enrichment of the model for import into a structural analysis tool. The common thread in these applications is that the exporting tool does not need to conform to the MVD of the importing tool, which means that export functions can remain generic. This is a major advantage for BIM software vendors because they find it difficult commercially to justify tailoring of export functions to narrow domains or specific importing software requirements.

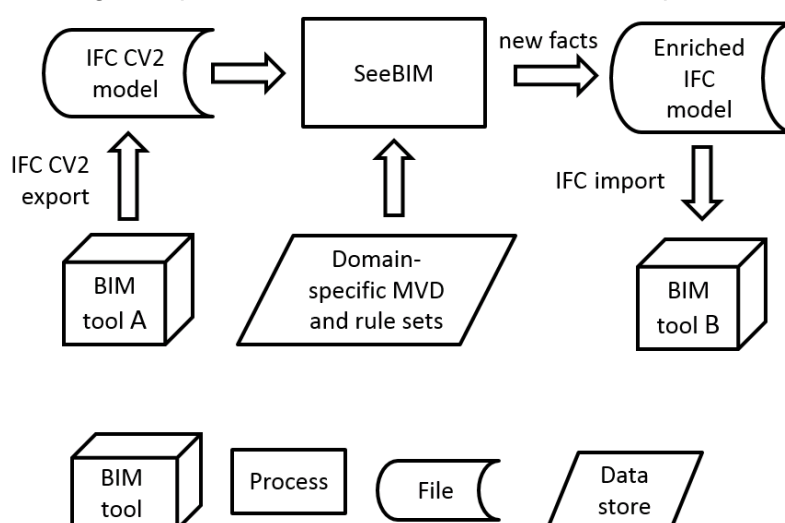


Figure 1. The SeeBIM Process. IFC CV2 files conform to the Coordination View 2.0 model view definition.

However, SeeBIM 1.0 has some important limitations that have become apparent in the first large-scale application of the tool, within the framework of an EU FP7 'Infraction' research project - "SeeBridge" (Technion 2015). The project aims to develop the ability to generate semantically rich bridge models from point cloud data. Computer vision technology is used to generate 3D shapes; SeeBIM is used to enrich the model by identifying bridge elements and their functional relationships. The limitations encountered in bridge model enrichment include:

- Firstly, the compilation of rule sets is at present essentially a social exercise that entails interviewing domain experts to elicit their knowledge and compiling it in the form of if-then rules. The process depends on intuition and subjective judgment, and neither the completeness nor the precision of rule sets can be guaranteed. Since the success or failure of the approach is dependent on the robustness of the tools, a rigorous method is needed for compiling rule sets, one that allows testing for adequacy.
- Secondly, the input is restricted to the IFC model file. In the worst case, this contains only the geometry, location and orientation of the 3D shapes. However, alphanumeric information, such as the year of construction or a building's location, can be vital in supporting semantic enrichment, providing essential clues to support inference rule processing. Such information is often available in some other data source, such as a highway agency's Bridge Management System (BMS), and should be imported with the model.
- Finally, the prototype uses axis-aligned bounding boxes (AABB) to approximate a model's geometry. This results in errors in many cases where objects have a non-convex shape or they are not axis-aligned. A shape's boundary and dimensions are inappropriately enlarged when it is non-axis aligned, with the result that many spatial topology operators return incorrect results. For example, a false positive result that two objects are in contact may be obtained if the first object is partially overlapped by the second object's AABB. SeeBIM depends heavily on the ability to process geometric and topological information, since geometry and placement are the only guaranteed information presented in all input models. This handicap therefore

severely limits the tool's application for domains such as highway bridges which commonly include many non-convex shapes (e.g. concrete girders).

The goal of the research presented in this paper was to address these and other issues that must be resolved before semantic enrichment can become practical. The solutions that have been devised are presented and discussed in the following sections. Some of the solutions have been implemented in software and they are used throughout the paper to illustrate the concepts.

Methodology

Research and development of the SeeBIM system follows a standard Design Science approach as defined specifically for the context of information science (Peppers et al. 2007). The methodology has the following six basic steps: identify problem and motivate; define objectives of a solution; design and develop a prototype software; demonstrate; evaluate; and communicate.. This paper focuses on the iteration of the design and development, demonstration and evaluation steps. The designed artefact is the SeeBIM 1.0 prototype, which is outlined above and reported in detail in Belsky et al. (2016). In the current work, the prototype has been enhanced based on the requirements for a specific application domain, that of inspection of reinforced concrete highway bridges.

The need to use data from an alphanumeric database (the Bridge Management System) as well as the 3D geometry model (compiled on the basis of the PCD) was identified through compilation of a formal Information Delivery Manual (IDM) for the domain of interest in the SeeBridge project (Sacks et al. 2016). The need to use explicit BREP geometry for correctly processing topological queries also arises from the IDM in that it identifies bridge elements that have concave geometry features and are not aligned with the major bridge axes. The third requirement – the need for a rigorous method to compile inference rules – is the result of consideration of the complexity that results in a real-world case, with large numbers of bridge component types, which renders informal rule compilation error-prone and inadequate.

In full-scale implementation for the use-case of compilation of BIM models from point cloud data (Scan-to-BIM), the semantic enrichment step begins once a 3D solid geometry model has been prepared. For this research, the 3D geometry models were compiled using BIM authoring tools. The models were exported to IFC without any of the semantic information, so that they could serve as input for the semantic enrichment process. At the same time, the BIM models provided the 'ground truth' for validation of results.

Rule-based inference

The success of model enrichment depends on the completeness and effectiveness of the inference rules used. Rule sets for expert system applications are commonly derived from knowledge acquisition interviews with domain experts (Hayes-Roth 1985). The procedural knowledge acquired is expressed in the form of IF-THEN rule clauses that form logical chains of inference. The complexity of the rules increases with the number of object types and features, and developers have limited ability to evaluate the process logic inherent in systems with large numbers of inference rules. In the case of rule sets for semantic enrichment of BIM models, the approach does not guarantee the completeness or adequacy of a rule set nor the reliability of the results.

The approach defined below is a procedure for deriving rule sets for identification of BIM object types (classification). Classification rules use two types of IF clauses: clauses that test for features of a single object, and clauses that test for topological relationships between pairs of objects. Rules used to identify object types therefore often depend on the prior

identification of other relevant, related objects. If the rule set is inadequate, some objects cannot be identified and enrichment will be partial, and in some cases interdependency within the rules can result in infinite loops. A rigorous and robust approach to compiling rules sets is preferable. Ideally, developers should be able to guarantee that if enough evidence is available in the data, then the set of rules will be adequate to identify all objects in the domain and the rule set will not be redundant. This is the goal of the procedure developed and described below.

In this approach, rules for identifying BIM object types are compiled in seven steps:

- 1) A set of pairwise topological relationships that are most apparently relevant for object identification is defined in consultation with domain experts.
- 2) The experts are asked to express their knowledge in the form of matrices, one for each of the relationships. Each matrix represents a particular pairwise relationship that can be applied to all the object pairs. The values in the cells are the logical results of the relationship for each pair.
- 3) The values for each particular cell in the resulting set of matrices are strung together to generate a string in each corresponding cell of a composite pairwise spatial/topological relationship matrix. This is an NxN matrix (where N is the number of possible object types).
- 4) Each string is then compared with all the other strings. Any string that is unique implies that if the set of relationship result values it represents is found to hold for any pair of object instances in a BIM model that is being enriched, then the identity of both of the objects can be determined.
- 5) If any object type does not have at least one unique string, then additional pairwise relationships must be added, repeating the process from step 2. This is done repeatedly if necessary, until all object types have at least one unique string.
- 6) Next, a subset of unique rule strings is selected from the whole set of unique strings, such that each object type is represented in at least one rule. .
- 7) Finally, a SeeBIM rule is compiled directly from each unique string in the subset.

To illustrate this procedure, we present their application to a small-scale synthetic bridge model (Figure 2). This model consists of eight typical types of bridge elements (A-H), as shown in Table 1, so that any kind of pairwise relationship can be represented as an 8x8 matrix. For example, Table 1 shows a matrix for the contact relationships involved in this

model. The relations are expressed as “IF object 1 is of type A and object 2 is of type B”, then there are three possible values:

'y' := object 1 is always in contact with object 2;

'n' := object 1 is never in contact with object 2;

'x' := object 1 may or may not be in contact with object 2;

As shown in Table 1, a column will always be in direct contact with a capping beam ('y'); a primary girder will never be in direct contact with a column ('n'); a primary girder may or may not be in contact with another primary girder ('x').

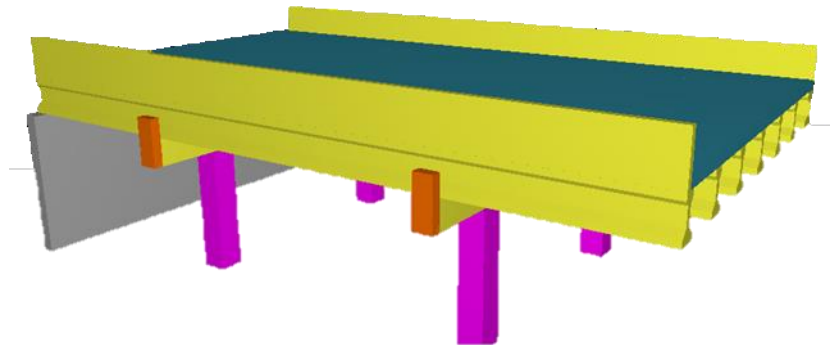


Figure 2 A synthetic bridge model

Table 2 lists ten different spatial features and pairwise relationships used to test the approach for the synthetic bridge model, and additional possible result values used across all the features and relationships are shown in Table 3. The relationship results are compiled in an 8x8 matrix with 10-digit strings in each cell, i.e. one digit for each relationship, as shown in

Table 4. Note that the first two relationships (the first two digits in each cell of

Table 4) reflect the objects' relative orientation to the bridge as a whole, so that the first digit in each row is the same and the second digit in every column is the same.

Table 1. Matrix for conditions of the contact relationship between bridge objects.

| | | | Type of object 1 | | | | | | | |
|------------------|---|----------------|------------------|---|---|---|---|---|---|---|
| | | | A | B | C | D | E | F | G | H |
| Type of object 2 | A | Primary Girder | x | y | y | n | n | n | x | x |
| | B | Capping Beam | | n | n | x | x | y | n | n |
| | C | Deck Slab | | | x | n | n | n | n | x |
| | D | Shear Key | | | | n | n | n | n | n |
| | E | Abutment | | | | | n | n | y | n |
| | F | Column | | | | | | n | n | n |
| | G | Bearing | | | | | | | n | n |
| | H | Safety barrier | | | | | | | | n |

Table 2. Conditional pairwise relationships between concrete girder bridges object types.

| No. | Conditional Relation |
|-----|--|
| 1 | Which axis of the bridge is object 2 parallel to? |
| 2 | Which axis of the bridge is object 1 parallel to? |
| 3 | Which object is larger in volume? |
| 4 | Which object's extrusion axis is longer? |
| 5 | Which object is closer to the lateral axis of the bridge? |
| 6 | Which object is closer to the longitudinal axis of the bridge? |
| 7 | Which object's bounding box is absolutely higher? |
| 8 | Which object's centroid is higher? |
| 9 | Do both objects have the same extrusion direction? |
| 10 | Are the objects in contact? |

Table 3. Additional possible result values used in the relationship matrices.

| Value | e | 1 | 2 | i | j | k |
|---------|-------|----------------|----------------|--------------------------|---------------------|----------------------|
| Meaning | Equal | Element type 1 | Element type 2 | bridge longitudinal axis | bridge lateral axis | bridge vertical axis |

All of the strings are then compared with one another to identify unique string values. When comparing strings, any relationships that have an 'x' value for either or both of the object types are ignored, as their values are obviously different. If a relationship's string is unique, then it will only be evaluated as true for those instances in which the pair of objects being tested for are of the types to which the cell belongs. This means that if the relationship evaluates as true, then the pair of objects being compared can be classified with full confidence. It is this property which allows users to compile a set of rules that can be considered *a priori* to be adequate.

The theoretical minimum number of unique pairwise relationships needed for adequacy of a rule set – i.e. the ability to classify all the objects in a model correctly and confidently – is half of the number of object types. More may be needed if some of the object types occur in more than one unique relationship. Furthermore, in cases where the model itself has inaccuracies or is incomplete, then any particular rule derived from a unique relationship string may not evaluate as true for all of the object pairs, and so the objects concerned may not be classified. In such cases, having additional unique rules, beyond the theoretical minimum, is useful. Some redundancy can improve the rate of success of classification.

For the case of the synthetic bridge, the four cells highlighted in

Table 4 with grey shading, are unique and they form an adequate set of unique relationship strings for classifying all the objects in a model of a bridge of this type, because they cover all the bridge element types(i.e., A-B, D-E, F-G and C-H) in pairs.

Table 4. Conditional relationship matrix for the eight types of bridge elements.

| | A | B | C | D | E | F | G | H |
|-----------------|-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Primary Girders | A iiXxxxney | ij11xx22n | ii1xxx11y | ik22x2n2n | ij112x22n | ikx2xx22n | ik222x22n | ii11x211y |
| Transverse | B | jjeeeeney | jix2xx11n | jk22x211n | jj1e2xn2y | jk22x222n | jk222xn1n | ji211211n |
| Deck Slab | C | | iixxxxnex | ik22x222n | ij112x22n | ik22xx22n | ik222x22n | ii1x2n1y |
| Shear Keys | D | | | kkeeeene | kj112122n | kk11x122y | kk222122y | ki111111n |
| Abutments | E | | | | jjxxxenyn | jk2212n2n | jk22ex11n | ji211211n |
| Column | F | | | | | kkeeeeney | kk222x11y | ki111211n |
| Bearings | G | | | | | | kkeexxney | ki111211n |
| Safety barriers | H | | | | | | | iiieeeney |

Finally, in step 7, these four relationship strings are translated into inference rules. For example, rules for identifying columns and bearings are translated from the string kk222x11yn in cell F-G as follows:

IF object 1 is parallel to bridge vertical axis (z)
 & object 2 is parallel to bridge vertical axis (z)
 & object 2 has larger volume than object 1
 & object 2 has longer extrusion axis than object 1
 & object 2 is closer to lateral axis of the bridge (y)
 & object 1 bounding box is absolutely higher than object 2
 & object 1 centroid is absolutely higher than object 2
 & object 1 extrusion axis is parallel to object 2 extrusion axis
 & object 1 is not in contact with object 2

THEN Object 1 is a bearing and Object 2 is a column

The process described above results in rule sets that contain sufficient tests to identify all of the possible object types in the domain. This ability to ensure adequacy is an important enhancement of the SeeBIM approach to semantic enrichment.

Merging BIM model data with information from external sources

The minimal starting point for semantic enrichment of building models is an IFC file containing building entities with solid geometry. However, most of the organizations that manage constructed facilities use databases of one form or another to describe their assets. These systems generally contain useful data that can and should be used to support semantic enrichment of BIM models. For example, state departments of transport (DOTs) use Bridge Management Systems (BMS) to manage their bridge networks. The BMS data shown in

Table 5 identifies a bridge with characteristic data in tabulated formats that can be helpful for semantic enrichment of a model of the bridge.

This data can provide prior information that is valuable for inference of bridge object types and relationships. For example, the pre-stressed concrete superstructure type suggests the presence and possible types of girders, and the bridge span length provides a candidate measure for identifying the bridge girders. The year of construction and the location further constrain the type of bridge elements (e.g., AASHTO girders were not available in this location until the 1960s).

Table 5. Examples of bridge data in a BMS database.

| Attribute | Data Example | IFC Property Types |
|----------------------|--|----------------------------|
| Bridge span | 17.6m | IfcPositiveLengthMeasure |
| Superstructure type | Prestressed concrete | IfcPropertyEnumeratedValue |
| Year of construction | 1993 | IfcDate |
| Location | Afek road bridge above route 79 in Kiryat Bialik | IfcText |
| Ownership | National roads company | IfcPropertyEnumeratedValue |

SeeBIM imports standard IFC files which must have building entities with BREP or extruded solid geometry. As a minimum, the entities will be IfcBuildingElementProxy entities. It uses a late-binding method (RDF 2015) (STEP Tools 2016)) to parse IFC files on the fly through the ISO Standard Data Access Interface (SDAI) (STEP Tools 2016), which means that it can import models from any IFC version provided that the EXPRESS schema definition files are available.

SeeBIM incorporates the data from external databases by appending it to the appropriate IFC entities. First, the properties, their data types and their values are imported into the runtime internal database of the application. This makes them available for testing within the IF clauses of the rules. During rule-processing, rules may add additional alphanumeric data to any of the model's entities. Finally, once rule-processing is complete, the data is exported in the form of IFC property sets. IFC property value entities are collected in IfcPropertySet entities, which are associated with building entities using IfcRelDefinesByProperties entities. According to the IFC schema, entities and property sets have a many-to-many relationship. Each entity can have more than one property set and each property set can be assigned to more than one entity. For example, many prefabricated components of a concrete bridge will share the same property sets and property values. However, many BIM authoring tools duplicate the same property set for each entity, creating unnecessarily large files. SeeBIM 2.0 identifies, resolves and removes these duplications, so that the IFC file size is reduced.

Enhanced Geometric and Topological Operators

An object's classification is related to its geometry, functions and other properties. In the worst case, only the geometry is guaranteed to be provided in a BIM model. Hence, the deduction of other information depends on unique model features, and the success of semantic enrichment depends on the ability to identify these features, including objects' shape features and pairwise topological and spatial relationships.

Enhancement of the semantic enrichment engine required removing the restrictions imposed by the prototype's axis-aligned bounding box representation of the geometry by using a minimal volume bounding box (MVBB) representation, in the first instance, and implementation of more sophisticated spatial and topological operators to account for explicit and potentially concave geometry representation, in the second instance.

Shape representation

Objects' shape features include the shape extents and orientation, which can be derived from the minimal volume bounding box (MVBB) of the object. Toussaint (1983) first proposed

the rotating caliper algorithm that can be used to construct the smallest-area enclosing rectangle in 2D. O'Rourke (1985) extended the algorithm to 3D such that the MVBB has at least two adjacent faces flush with edges of the 3D shape. He presented an algorithm for generating the MVBB in a brute force way. Based on O'Rourke's findings, Jylanki (2015) developed a more efficient algorithm to generate the MVBB, and this algorithm is used in SeeBIM 2.0.

The generated MVBB can be represented by three components: orientation ($axis[0]$, $axis[1]$ and $axis[2]$, each of which is a three dimensional vector), coordinates of the centroid point ($pos[0]$, $pos[1]$ and $pos[2]$), and the extent of the box in the local axes ($r.x$, $r.y$ and $r.z$), as shown in

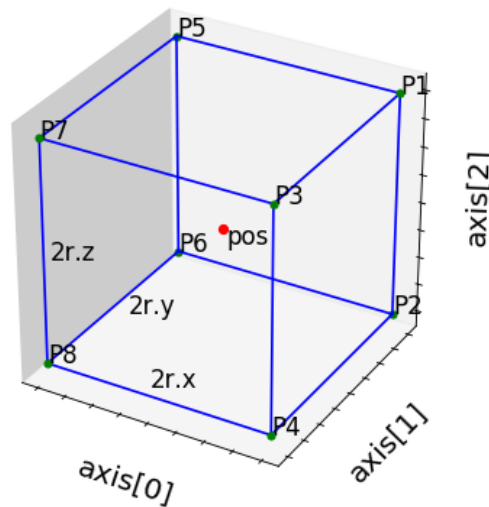


Figure 3. The properties of a MVBB

The local coordinates of the eight vertices of the MVBB can be derived as:

$$\mathbf{P} = \begin{bmatrix} r.x & r.x & r.x & r.x & -r.x & -r.x & -r.x & -r.x \\ r.y & r.y & -r.y & -r.y & r.y & r.y & -r.y & -r.y \\ r.z & -r.z & r.z & -r.z & r.z & -r.z & r.z & -r.z \end{bmatrix}$$

The transformation from the local coordinate system to the global coordinate system can be represented as a 4×4 matrix in the homogeneous space:

$$\mathbf{T} = \begin{bmatrix} axis[0][0] & axis[1][0] & axis[2][0] & pos[0] \\ axis[0][1] & axis[1][1] & axis[2][1] & pos[1] \\ axis[0][2] & axis[1][2] & axis[2][2] & pos[2] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To simplify the mathematical operation for computing the global coordinates of all the vertices, the vectors in \mathbf{P} were augmented to the homogeneous space by increasing their dimensionality. For example: $\mathbf{P}'_1 = [r.x \ r.y \ r.z \ 1]$, so that \mathbf{P}' is a 4×8 matrix. Then, the global coordinates of the vertices in the homogeneous space can be derived as:

$$\mathbf{V}' = \mathbf{T} \times \mathbf{P}' = \begin{bmatrix} x_1 & x_2 & \dots & x_8 \\ y_1 & y_2 & \dots & y_8 \\ z_1 & z_2 & \dots & z_8 \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

The actual coordinates of each point can be derived by reducing the dimensionality of each vector, for example, $\mathbf{V}_1 = [x_1 \ y_1 \ z_1]$. In addition, the six faces can be derived as follows:

- $(\mathbf{V}_1 \ \mathbf{V}_2 \ \mathbf{V}_4 \ \mathbf{V}_3)$ and $(\mathbf{V}_5 \ \mathbf{V}_6 \ \mathbf{V}_8 \ \mathbf{V}_7)$ are faces whose normal direction is $axis[0]$
- $(\mathbf{V}_1 \ \mathbf{V}_2 \ \mathbf{V}_6 \ \mathbf{V}_5)$ and $(\mathbf{V}_3 \ \mathbf{V}_4 \ \mathbf{V}_8 \ \mathbf{V}_7)$ are faces whose normal direction is $axis[1]$
- $(\mathbf{V}_1 \ \mathbf{V}_5 \ \mathbf{V}_7 \ \mathbf{V}_3)$ and $(\mathbf{V}_2 \ \mathbf{V}_6 \ \mathbf{V}_8 \ \mathbf{V}_4)$ are faces whose normal direction is $axis[2]$

Spatial and Topological Relationships and Operators

Extensive data sets can be precisely analyzed, explored and processed by a formal query language. To handle spatial data, languages such as Spatial SQL and GeoSPARQL are used in geographical information systems (GIS) (Egenhofer 1994; Perry and Herring 2012). There have also been attempts to facilitate query languages in the AEC domain (Borrmann 2010; Mazairac and Beetz 2013). However, none of these methods could process the 3D representations used in civil engineering in an adequate way, especially with respect to qualitative spatial predicates. This was a major deficiency, as spatial relations between building elements play a significant role in most of the design and engineering tasks of the AEC domain. To close this gap, a BIM query language called Query Language for 4D Building Information Models (QL4BIM) was developed (Daum and Borrmann 2013).

Among other features, QL4BIM makes it possible to select specific building elements by applying qualitative spatial predicates as part of filter expressions. These relationships provide a high level of abstraction between the technological view on building geometry using numerical coordinates, and the way humans reason about spatial entities and the relations between them. Typical examples of queries concerned with spatial semantics are:

- Which columns *touch* slab 34?
- Get all walls which are *contained* in the first storey.
- Does the space representation of room 107 *intersect* with any heating equipment?
- Get all objects *within a distance* of 1.5 meters from wall 232.

The ability to identify and compute spatial relationships between building objects is also essential for a semantical enrichment. For that reason, QL4BIM operators are facilitated in the enrichment process of SeeBIM.

As a query language, QL4BIM was designed to be employed by domain experts and offers a carefully selected vocabulary to formulate queries at a high level of abstraction (Daum and Borrmann 2015). In SeeBIM 2.0, there is no need to incorporate this kind of end user interface. Instead, the QL4BIM operators are introduced as library functions which can be invoked directly from the SeeBIM rule composition interface. The functionality offered includes metric, directional and topological operators which are illustrated in Figure 4. **Error! Reference source not found..**

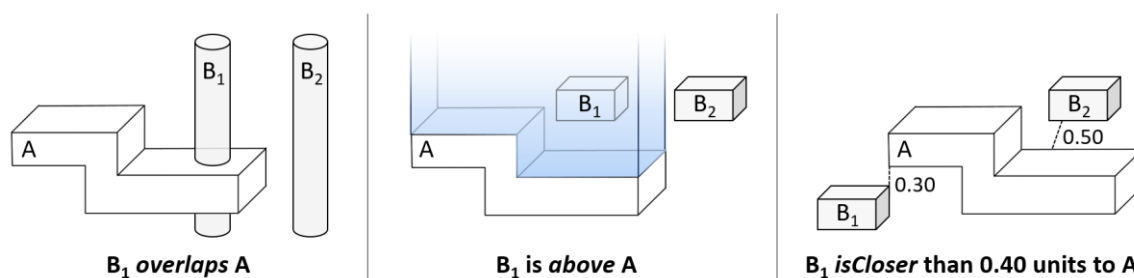


Figure 4. Examples of the three classes of spatial operators provided by QL4BIM. In all cases object B₁ passes the predicate whereas object B₂ is rejected.

In contrast to several applications in the GIS/AEC domain, the QL4BIM operators are not restricted to 2D geometry or bounding box abstractions (ISO/OGC ; Nepal et al. 2012). Instead, 3D geometry is processed as triangulated boundary representation, and operators evaluate correctly with convex and non-convex shapes.

The topological and directional operators are based on the mathematical definitions stated by Egenhofer (1989) and Borrmann (2006). In the first case, the approach is called the 9-Intersection Model (9-IM) and facilitates theories of algebraic topology and set topology (Gaal 1964). The 9-IM applies the notion of the neighborhood of a point to describe

topological concepts such as the interior A° , the boundary ∂A and the exterior A^- of a point set A . Topological predicates are defined by the set oriented intersections of the interior, the boundary and the exterior of two operands. Here, an intersection can yield an empty (\emptyset) or a nonempty set ($\neg\emptyset$). Figure 5 shows the 9-IM matrix for a 2D scenario with two regions A and B .

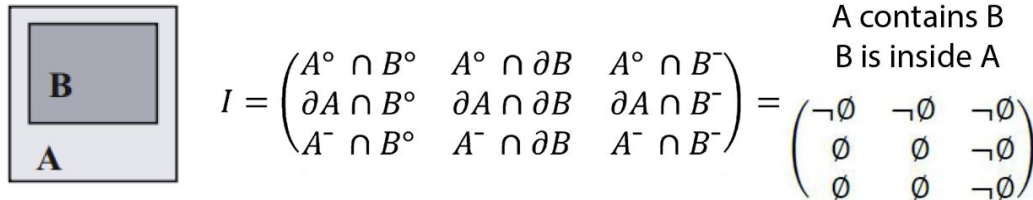


Figure 5. Deducing the topological relationship between two regions by the 9-Intersection Model (containment/inside case).

A 9-IM matrix represents the topological invariants of the topological relations, reflecting that the set oriented intersection results remain constant under transformations. Theoretically, there are $2^9 = 512$ possible configurations, but only eight are encountered when closed regions are examined in 2D. The same number of configurations arises for closed solids in 3D. **Error! Reference source not found.** shows the eight possible topological onstellations of two solids and the corresponding matrices.

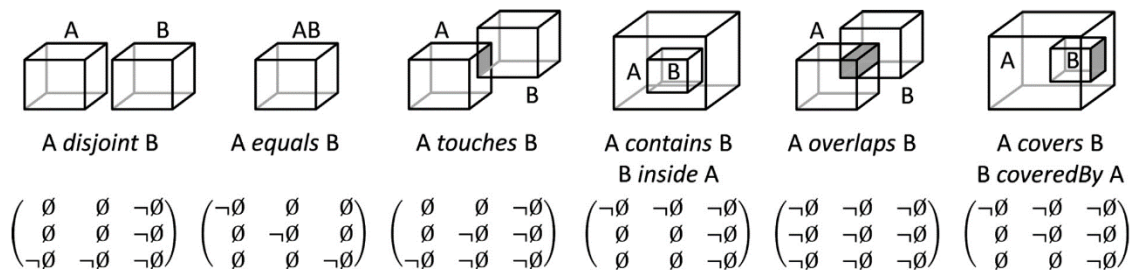


Figure 6. The eight topological templates for solids introduced by the 9-IM.

The definition of the directional operators uses a projection-based model. There is a strict and a relaxed version of each directional predicate. In both cases, reference object A and target object B are spatial objects and $a \in A, b \in B$. Then, the formal definitions of the Above operators read as shown in Figure 6 **Error! Reference source not found.**, whereas the indices of a and b denote the respective dimension. Figure 7 **Error! Reference source not found.** includes an example with five object pairs $A-B_1$ to $A-B_5$. Table 6 shows the results of the relaxed and the strict version of the Above operator, if it is employed on these pairs.

$$\text{aboverelaxed}(A,B) \Leftrightarrow \exists a,b : a_x = b_x \wedge a_z < b_z$$

$$\text{abovestrict}(A,B) \Leftrightarrow \forall b : (\exists b : a_x = b_x \wedge a_z < b_z) \wedge (\nexists b : a_x = b_x \wedge a_z \geq b_z)$$

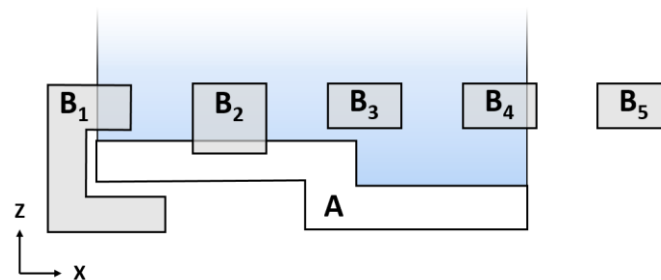


Figure 7. Mathematical definition for of the projection-based directional predicates and an example for the Above case (2D case for clearness).

Table 6. Results of the two Above operators for the spatial constellation in Figure 7 Error! eference source not found..

| | B ₁ | B ₂ | B ₃ | B ₄ | B ₅ |
|---------------------------------|----------------|----------------|----------------|----------------|----------------|
| aboverelaxed(A,B _i) | true | true | true | true | false |
| abovestrict(A,B _i) | false | false | true | false | false |

From the mathematical definitions, algorithms are deduced that process triangulated meshes and determine the spatial predicate between two spatial objects. In the case of the topological operators' triangle intersection- and inside/outside-tests are applied. For example, the Touch predicate is verified if at least two triangles meet, no triangles intersect and B is located outside of A.

The directional functionality is realized by triangle extrusions, prism/triangle- and ray-tests. For being Above in the relaxed version, at least one triangle of B must intersect with a prism of A. These prisms are created by extruding the triangles of A in the correlated direction of the predicate. In the case of the Above predicate, this is the positive Z-direction. To deal with the emerging computational complexity that arises if extensive datasets and detailed geometry representations are handed to the operators, the spatial indexing structure R*-Tree is incorporated (Beckmann et al. 1990).

Support for Tolerances in Spatial Operators

For imperfect datasets, the support of user-defined tolerances in the processing of directional and topological predicates is needed. This is especially the case if geometry is reconstructed from a laser-scanned point cloud. Besides numerical discrepancies, parts of the objects' surfaces may be obscured in these data sets.

In addition to imperfect geometry reconstruction, the need for tolerances also derives from the practical design and construction considerations. Here, minimal gaps and intersections of a specific extent must be approved, whereas the applied tolerances depend on the modeling domain and the actual use case.

To support semantic tolerances and to yield robust results despite numerical imprecisions, a particular mesh handling is developed for QL4BIM within the SeeBridge project. The approach is based on the use of an inner and an outer mesh. These meshes are created via shifting original triangles by a user defined amount. Here, the inner boundary is given the index *i*, the outer one the index *o*. The developed tolerance supporting topological operators are denoted as TST-operators, the tolerance supporting directional operators as TSD-operators.

The algorithms for the TST-operators begin with the original geometry representations and check for several topological predicates. If none of the permissible predicates is valid, the investigation is aborted and the predicate is rejected. If the first condition is met, a modified boundary is created by triangle shifting. Whether to move inside and / or outside depends on the actual predicate and the operand. In the last step of the TST-processing, one of several predicates has to be confirmed with the changed geometry. Figure 8 and Figure 9 show the predicates allowed at the beginning (1), the geometry modification per operand (2) and the predicates that have to be finally checked (3).

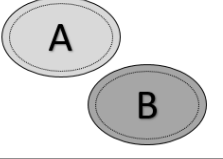
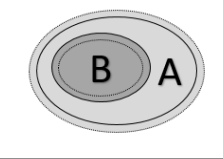
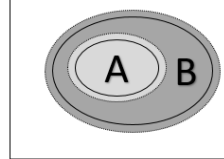
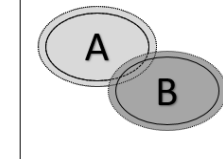
| | | | |
|--|--|--|--|
|  |  |  |  |
| <ol style="list-style-type: none"> 1. A disjoint B 2. Create $A_o B_o$ 3. A_o disjoint B_o | <ol style="list-style-type: none"> 1. A contains B 2. Create $A_i B_o$ 3. A_i contains B_o | <ol style="list-style-type: none"> 1. A inside B 2. Create $A_o B_i$ 3. A_o inside B_i | <ol style="list-style-type: none"> 1. A overlaps B 2. Create $A_i B_i$ 3. A_i overlaps B_i |
| strong disjoint | strong contains | strong inside | strong overlaps |

Figure 8. Three-stage definitions of the strong TST-operators in QL4BIM

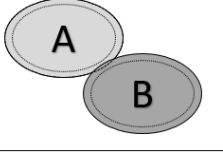
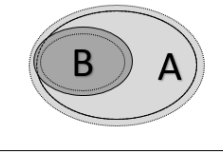
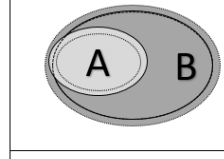
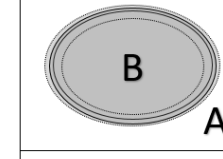
| | | | |
|--|--|--|---|
|  |  |  |  |
| <ol style="list-style-type: none"> 1. A disjoint B A touches B 2. Create $A_o B_o$ 3. A_o overlaps B_o A_o touches B_o | <ol style="list-style-type: none"> 1. A contains B A covers B 2. Create $A_i B_o$ 3. A_o overlaps B_i A_o covers B_i | <ol style="list-style-type: none"> 1. A inside B A coveredBy B 2. Create $A_o B_i$ 3. A_o overlaps B_i A_o coveredBy B_i | <ol style="list-style-type: none"> 1. ! A disjoint B ! A touches B 2. Create $A_o A_i B_o B_i$ 3. A_o contains B_i & A_i inside B_o |
| weak touch | weak covers | weak coveredBy | weak equal |

Figure 9. Three-stage definitions of the weak TST-operators in QL4BIM

As indicated by the figures, the operators can be divided into two groups, the strong and the weak ones. In case of the strong operators, the topological predicate returns true independent of the application tolerances. In the case of the weak ones, the predicate is confirmed only thanks to the tolerances applied. Thus, the operators of the first group are denoted as strong, and the operators of the second group as weak variants of their originals. The strong group includes the predicates disjoint, contains, inside and overlaps, the weak group touches, covers, coveredBy and equals.

The definition of the TSD-operators is equal for all directions and includes only a geometry modification and a subsequent directional analysis. In the modification step, A_i is produced and B is not altered. Then, the original directional predicate is executed. TSD-operators are weak variants of their originals.

The necessary offset meshes for this approach can be generated by several approaches (Egenhofer et al. 1989; Rossignac and Requicha 1986). In QL4BIM, the Multiple Normal Vectors of a Vertex Method (MNVN) method is applied to balance between the geometric accuracy of the created boundaries and the computational costs (Kim et al. 2004).

Full-scale bridge test

The enhanced semantic enrichment tool was tested using a model of a concrete girder highway bridge on Route 79 in Haifa, Israel. The bridge was scanned using a terrestrial laser scanner. To obtain a panoramic view of the entire bridge, several scans were taken from different positions and a complete point cloud was derived by registration of the collected PCD sets. A 3D model of the bridge geometry was compiled manually in a BIM authoring tool from the PCD. The model, shown in Figure 10, contains 333 bridge elements of ten different types (listed in Table 7).

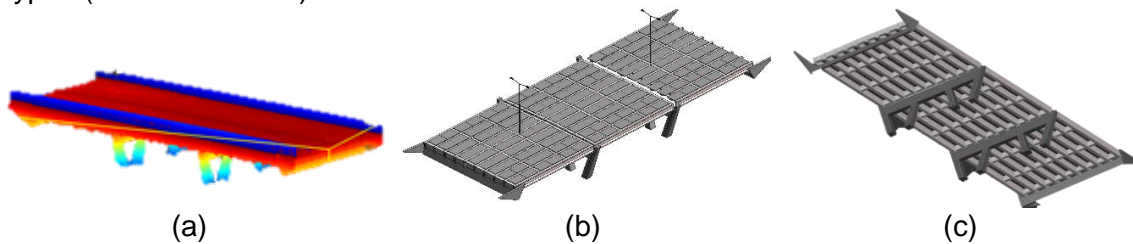


Figure 10. The manually prepared bridge model geometry: a) point cloud, b) Revit model top view, c) IFC model bottom view.

Table 7. Bridge elements in the bridge model (note that the bearings and abutment plinths were not visible in the PCD due to occlusion).

| Concrete girder bridge object types | | Number of visible objects |
|-------------------------------------|-----------------------------------|---------------------------|
| Deck / super-structure | Primary girder | 30 |
| | Capping beam | 2 |
| | Transverse beam | 99 |
| | Partial depth precast deck panel | 162 |
| Sub-structure | Bearing | None |
| | Plinth on capping beam | 20 |
| | Plinth on abutment | None |
| | Shear key on capping beam | 4 |
| | Shear key on abutment (wing wall) | 4 |
| | Abutment | 2 |
| Non-structural | Column | 8 |
| | Lamp posts | 2 |
| | Safety barriers | None |

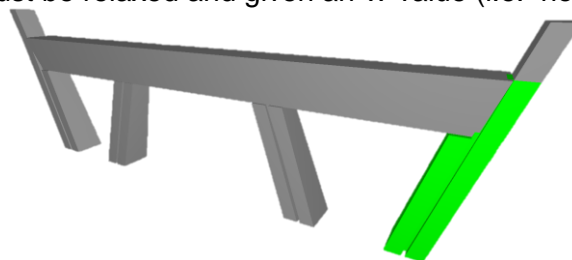
The full-scale bridge has 13 object types, whereas the synthetic bridge used to explain the process above had only eight. Therefore, a new set of rules is needed, and the set must have at least seven rules, whereas the synthetic bridge required only four. To obtain a sufficient set of unique rules in this case required 19 conditional relations, and these are listed in

Table 8.

Table 8. Conditional pairwise relationships between concrete girder bridge object types.

| No. | Conditional Relation |
|-----|--|
| 1 | Are the two objects in contact? |
| 2 | Is object 1 in contact with the object 2's side face? |
| 3 | Is object 1 in contact with the object 2's front / back face? |
| 4 | Is object 1 in contact with the object 2's bottom face? |
| 5 | Is object 1 in contact with the object 2's top face? |
| 6 | Are the two objects in parallel along their extrusion direction? |
| 7 | Are the two objects in parallel along their long edges? |
| 8 | Is object 1's centroid higher than object 2's? |
| 9 | Is object 1's extrusion longer than object 2's? |
| 10 | Is object 1's volume greater than object 2's? |
| 11 | Is object 1 vertically extruded? |
| 12 | Is object 1's extrusion direction parallel to the road axis? |
| 13 | Is object 1's extrusion direction parallel to the skew angle of the bridge supports? |
| 14 | Is object 1 horizontal? |
| 15 | Is object 1 the bridge? |
| 16 | Is object 2 the bridge? |
| 17 | Is object 1 wider than object 2? |
| 18 | Is object 1 taller than object 2? |
| 19 | Is object 2 is a capping beam? |

Any ambiguity in understanding and modeling the bridge objects will affect the classification result. Figure 11 shows an example of this: in the test, the outer concrete columns were modeled such that the shear key is on top of the column, while the capping beam does not rest on the topmost face of the column. However, in the initial compilation of conditional relation #5 for the case of the capping beam and the column, the logical assumption was "the capping beam (object 1) is always in contact with the top face of the column (object 2)". Similarly, the inverse condition (#4) was assumed to be always true ('y' value). This would be true for all cases only if the capping beam were modeled as an extrusion extending to both ends, such that the shear key would be above the capping beam and the capping beam would be in contact with the top face of all of the columns (as it is with the two middle columns), but this bridge illustrates that the condition is not in fact always true. Therefore, in order to cope with the more general case, the conditional value for this pair of objects for conditions #4 and #5 must be relaxed and given an 'x' value (i.e. not always).

**Figure 11. Part of the substructure in the bridge model**

The information delivery manual prepared for the SeeBridge project (Sacks et al. 2016) lists thirteen relevant object types for a concrete girder bridge such as this one. Of those, none of the bearings or the plinths on the abutments were visible in the PCD for the test bridge because they were occluded by other objects. Nevertheless, they are included in the rule

sets so that the rules will be valid for the general case of concrete girder bridges. The entire bridge is also considered as an object since its boundary and orientation are of great importance for inference of other objects. Therefore, the examination uses a 14x14 matrix with 19-digit strings in each cell.

The result values in all the matrices were filled in consultation with three experts: a senior partner in a bridge structural design practice with over 15 years of bridge design experience, a professor of structural engineering and construction management, and a structural engineer with 5 years of experience in reinforced concrete design.

A part of the matrix is shown in Figure 12 with unique strings identified in the shaded cells. Each string represents the 19 conditions of a rule for object classification of the two elements in the row and column to which the cell belongs. The string in the Primary Girder – Transverse Beam cell, shown in bold text, is an example of a unique string. It can be translated as:

- IF** object 1 is in contact with object 2
- & object 1 is in contact with object 2's side face
- & object 1 is not in contact with object 2's front or back face
- & object 1 is not in contact with object 2's top face
- & object 1 is not in contact with object 2's bottom face
- & the two objects are not parallel along their extrusion direction
- & the two objects are not parallel along their long edges
- & object 1's extrusion axis length is longer than object 2's extrusion axis length
- & object 1's volume is greater than object 2's volume
- & object 1 is not vertical
- & object 1's extrusion direction is parallel to the road axis
- & object 1's extrusion direction is not parallel to the skew angle of the bridge supports
- & object 1 is not the bridge
- & object 2 is not the bridge,
- THEN** object 1 is a primary girder and object 2 is a transverse beam

| | | | obj1 | | | | | |
|---------------|---------------------|--------------------------|---------------------|--------------------|---------------------------|--------------------|--------------------|--------------------|
| | | | Element Group | Bridge | Deck Superstructure | | | |
| | | | Element description | Bridge | Primary Girder | Capping beam | Transverse beam | Deck slab panel |
| Element Group | Element description | # | | | 111 | | 201 | 301 |
| obj2 | Bridge | Bridge | 00000000000000yy00 | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | Deck Superstructure | Primary Girder | 111 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Capping beam | | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Transverse beam | 201 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Deck slab panel | 301 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | Substructure | Bearing | | xxxxxxxxxy0100ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Plinth (capping beam) | | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Plinth (abutment) | | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Shear Key (capping beam) | | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Shear Key (abutment) | 401 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | Non-structural | Abutment | 403 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Column | 405 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | | Lamp post | | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny |
| | Safety barrier | 507 | xxxxxxxxxy0000ynyy | xxxxxxxxxyxxxxxxxx | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | xxxxxxxxxxxxxxxxny | |

Figure 12. The conditional relation strings for the thirteen bridge element types.

The matrix yielded 14 unique rule strings. Seven of these were selected and implemented in the SeeBIM interface. These seven are sufficient for classifying all of the 13 object types, because each pairwise rule identifies two object types. Finally, the BIM model was loaded and processed for matching and enrichment in the rule-processing engine, which iterates over the set of rules, using two nested loops to process rules for every possible pair of elements, and in both possible orders for each pair. It infers new information in each cycle, stopping only once no additional information can be inferred. Thus the sequence of the rules is unimportant, and each rule may be checked several times in the enrichment process. For the case of this girder bridge, with 333 elements of ten different types, it proved possible to compile a set of rules that could perform complete classification with 100% precision and recall.

Discussion

While the specific set of rules derived and implemented for the case of reinforced-concrete highway girder bridges is sufficient for correct classification of all thirteen object types, the value of this work is in the procedure, not in the rule set. The procedure enables users to compile SeeBIM rule sets for classifying the objects in a building information model that is not typed, and to do so with the knowledge that the rule set is comprehensive and effective. There are two aspects that could be improved: a) sufficiency of the rule set for any given domain, and b) redundancy of rule set for computation.

In theory, for n object types, $n/2$ rules should suffice to identify all object types if only pairwise rules are used. In practice, one object type 'A' may have unique pairwise relationships with two (or more) objects, e.g., 'B' and 'C', and if neither 'B' nor 'C' has any unique pairwise relationships with objects other than 'A', then, identification of 'B' and 'C' both depend on 'A'. In this case, some pairwise rules overlap (share an object type), and more than $n/2$ rules will be needed. Also, some dependencies may be nested (e.g., 'B' depends on 'A', and 'C' depends on the fact that 'A' and 'B' have been classified). In this case, 'C' will be classified in the second or later iteration of the system (for example, relationship No.19 in

-
- a) Table 8 can only be true after a first iteration in which 'capping beams' have already been identified). This too will result in the need for more than $n/2$ rules. Finally, given possible inaccuracies of the 3D object data, some rules may not work for particular instances of objects that are inaccurately modeled: redundancy in the number of rules will improve the probability of classifying the objects.
- b) The number of possible rule strings that could be generated is much larger than the number of rules needed. In theory, the number of rules, with redundancy, could be in the range from $n/2$ to n^2 , and for each object type there are $2n-1$ possible cells with rule strings. If k pairwise relationships are evaluated and each relationship has two conditions, a string could have 2^k different combinations. As a result, the $2n-1$ cells will almost always have more than one unique strings that come from the 2^k different combinations. In this work, a subset of unique rules was manually chosen from the full set such that the subset covers all the object types.

Note that at this stage, the approach does not consider the efficiency of computation, because the runtime for large models remains very short. In theory, an algorithm could be developed to select an optimal subset of unique rules. Such an algorithm should also be able to evaluate whether some sub-string exists for any rule string selected such that it is still unique within the set of strings (i.e. it may be possible to compile unique rules with fewer than k conditions). In this way, the efficiency of the computation will be improved.

Conclusions

Semantic enrichment is an important process that can relieve the problem of information interoperability and greatly improve the functionality of BIM models throughout a facility's life-cycle. This paper presented the enhancement of a semantic enrichment tool. The enhancements include a novel and rigorous method for compilation of inference rules, adoption of external data for enrichment, and additional operators for identification of shape features and spatial relationships that are common in geometrically complex facilities like bridges. The system was validated using a full-scale 3D model of a concrete girder highway bridge.

The process developed for rule definition results in rule sets that contain sufficient tests to identify all the possible object types in the domain. This is an important enhancement of the SeeBIM approach to semantic enrichment. Naturally, however, such a system is still subject to the quality of the input data. The objects can be completely and correctly classified only when the models have sufficiently small errors in the locations and geometry of the bridge components to allow the geometry and topological relationship operators to perform correctly with suitable tolerances. However, model deficiencies cannot be completely avoided. For example, two objects expected to be touching may be modeled as overlapped or disconnected objects; in this case the rule checking may give a false negative error. Setting large tolerance values could avoid such results, but setting the tolerance too large is likely to

result in false positive errors. Notwithstanding the robustness of the rule compilation process, success of the object classification process remains dependent on the quality of the geometric model.

Future work will address additional aspects of semantic enrichment. For the general Scan-to-BIM use case, in addition to object classification, rules are needed for object aggregation, for numbering/naming objects, for generating abstract objects, and to apply corrections where objects are occluded. In addition, researchers should consider attempting to apply machine-learning approaches to semantic enrichment for BIM in general and to each of these challenges in particular.

References

- Aram, S. (2015). "A Knowledge-based system framework for semantic enrichment and automated detailed design in the AEC projects." PhD., Georgia Institute of Technology.
- Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. "The R*-tree: an efficient and robust access method for points and rectangles." *Proc., ACM Sigmod Record, Acm*, 322-331.
- Belsky, M., Sacks, R., and Brilakis, I. (2016). "Semantic Enrichment for Building Information Modeling." *Computer-Aided Civil and Infrastructure Engineering*, 31(4), 261-274.
- Borrmann, A. (2006). "Extended formal specifications of 3D Spatial Data Types." Technische Universität München.
- Borrmann, A. (2010). "From GIS to BIM and back again-A Spatial Query Language for 3D building models and 3D city models." *5th International 3D GeoInfo Conference*. Berlin.
- Borrmann, A., and Rank, E. (2009). "Specification and implementation of directional operators in a 3D spatial query language for building information models." *Adv. Eng. Inform.*, 23(1), 32-44.
- Bosche, F., and Haas, C. T. (2008). "Automated retrieval of 3D CAD model objects in construction range images." *Automation in Construction Selected papers from the 26th ISARC 2009*, 17(4), 499-512.
- Brilakis, I., Manolis, L., Sacks, R., Savarese, S., Christodoulou, S., Teizer, J., and Makhmalbaf, A. (2010). "Toward automated generation of parametric BIMs based on hybrid video and laser scanning data." *Adv. Eng. Inform.*, 24(4), 456-465.
- BuildingSmart (2010). "Coordination View Version 2.0." <<http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0>>. (1 Feb. 2017, 2017).
- BuildingSmart (2013). "Industry Foundation Classes Release 4 (IFC4)." <<http://www.buildingsmart-tech.org/ifc/IFC4/final/html/>>. (1 Feb. 2017, 2017).
- Daum, S., and Borrmann, A. (2013). "Definition and Implementation of Temporal Operators for a 4D Query Language." *Proc. of the ASCE International Workshop on Computing in Civil Engineering*. Los Angeles, California.
- Daum, S., and Borrmann, A. (2014). "Processing of Topological BIM Queries using Boundary Representation Based Methods." *Adv. Eng. Inform.*, 28(4), 272-286.
- Daum, S., and Borrmann, A. (2015). "Simplifying the Analysis of Building Information Models Using tQL4BIM and vQL4BIM." *Proc. of the EG-ICE 2015 Eindhoven*, The Netherlands.

- Eastman, C., Lee, J. M., Jeong, Y. S., and Lee, J. K. (2009). "Automatic rule-based checking of building designs." *Automation in Construction Selected papers from the 26th ISARC 2009*, 18(8), 1011-1033.
- Eastman, C., Teicholz, P., Sacks, R., and Liston, K. (2011). *BIM Handbook*, John Wiley & Sons, Inc.
- Egenhofer, M. (1989). "A Formal Definition of Binary Topological Relationships." *Proc. of the 3rd Int. Conf. on Foundations of Data Organization and Algorithms (FODO)*.
- Egenhofer, M. (1994). "Spatial SQL: A Query and Presentation Language." *IEEE Trans. Knowl. Data Eng*, 6(1), 86–95.
- Egenhofer, M., Frank, A., and Jackson, J. P. (1989). "A Topological Data Model for Spatial Databases." *Proc. of the 1st Int. Symp. on the Design and Implementation of Large Spatial Databases* Santa Barbara, California.
- Gaal, S. A. (1964). *Point set topology*, Academic Press, New York.
- Hayes-Roth, F. (1985). "Rule - based systems." *Communications of the ACM*, 28(9), 921-932.
- ISO/OGC "Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture." 19125.
- Jylanki, J. (2015). "An Exact Algorithm for Finding Minimum Oriented Bounding Boxes." <<http://clb.demon.fi/projects/an-exact-algorithm-for-finding-minimum-oriented-bounding-boxes>>. (1 Feb. 2017, 2015).
- Kashani, A., Crawford, P., Biswas, S., Graettinger, A., and Grau, D. (2014). "Automated Tornado Damage Assessment and Wind Speed Estimation Based on Terrestrial Laser Scanning." *Journal of Computing in Civil Engineering*, 04014051.
- Kim, S.-J., Lee, D.-Y., and Yang, M.-Y. (2004). "Offset triangular mesh using the multiple normal vectors of a vertex." *Computer-Aided Design and Applications*, 1(1-4), 285–291.
- Mazairac, W., and Beetz, J. (2013). "BIMQL - An open query language for building information models." *Adv. Eng. Inform.*, 27(4), 444-456.
- Nepal, M. P., Staub-French, S., Pottinger, R., and Webster, A. (2012). "Querying a building information model for construction-specific spatial information." *Adv. Eng. Inform.*, 26(4), 904–923.
- O'Rourke, J. (1985). "Finding minimal enclosing boxes." *International journal of computer & information sciences*, 14(3), 183-199.
- Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., and Van Campenhout, J. (2011). "A semantic rule checking environment for building performance checking." *Automation in Construction Selected papers from the 26th ISARC 2009*, 20(5), 506-518.
- Peppers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). "A design science research methodology for Information Systems Research." *Journal of Management Information Systems*, 24(3), 45-77.
- Perry, M., and Herring, J. (2012). "OGC GeoSPARQL-A geographic query language for RDF data." *OGC Implementation Standard. Sept.*
- Ramaji, I. J., and Memari, A. M. (2016). "Interpreted Information Exchange: Systematic Approach for BIM to Engineering Analysis Information Transformations." *Journal of Computing in Civil Engineering*, 30(6), 04016028.
- RDF (2015). "IFC Engine DLL." <<http://rdf.bg/ifc-engine-dll.php>>. (1 Feb. 2017).

- Rossignac, J. R., and Requicha, A. A. G. (1986). "Offsetting operations in solid modelling." *Computer Aided Geometric Design*, 3(2), 129–148.
- Sacks, R., Kedar, A., Borrmann, A., Ma, L., Singer, D., and Kattel, U. (2016). "SeeBridge Information Delivery Manual (IDM) for Next Generation Bridge Inspection." *33rd International Symposium on Automation and Robotics in Construction*, A. Sattineni, ed., Auburn University, Auburn, AL.
- STEP Tools, I. (2016). "ISO 10303-22 - Standard data access interface (SDAI)." <<http://www.jsdai.net/support/about-step/sdai>>. (1 Feb. 2017, 2016).
- STEP Tools, I. (2016). "STEP and STEP-NC Software for e-manufacturing." <<http://www.steptools.com/>>. (1 Feb. 2017, 2016).
- Technion (2015). "SeeBridge – Semantic Enrichment Engine for Bridges." <<http://seebridge.net.technion.ac.il/>>. (1 Feb. 2017, 2016).
- Toussaint, G. T. "Solving geometric problems with the rotating calipers." *Proc., IEEE Melecon*, A10.
- Zeibak-Shini, R., Sacks, R., Ma, L., and Filin, S. (2016). "Towards generation of as-damaged BIM models using laser-scanning and as-built BIM: First estimate of as-damaged locations of reinforced concrete frame members in masonry infill structures." *Adv. Eng. Inform.*, 30(3), 312-326.
- Zhang, J. S., and El-Gohary, N. M. (2016). "Extending Building Information Models Semiautomatically Using Semantic Natural Language Processing Techniques." *Journal of Computing in Civil Engineering*, 30(5), C4016004.