B-Spline Surfaces in IFC: Implementing an Open-Source Viewer

Christoph Kaiser[1], Štefan Jaud[2] and Jonas Schlenger[1]
[1]Chair for Computational Modelling and Simulation, Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany
[2]The Hard Code GmbH, Moorenweis, Germany
E-mail(s): chr.kaiser@tum.de, stefan.jaud@outlook.com, jonas.schlenger@tum.de

Abstract: On the one hand, modern architecture, engineering, and construction (AEC) design often use free-form (curved) surfaces. The best-fitting general-purpose formal geometry description is a B-spline or non-uniform rational B-splines (NURBS) surface. However, in order to inspect such implicitly defined surfaces, a viewer constitutes an essential tool. On the other hand, the exchange format Industry Foundation Classes (IFC) is getting increasingly used in the building information modeling (BIM) processes. The IFC standard supports advanced boundary representations (BReps), which can contain B-spline or NURBS surfaces. There is a lack of IFC viewers which correctly visualize such geometries, especially among open-source solutions. We present an extension to the open-source software TUM OpenInfraPlatform (OIP) to fill this gap. We test the quality of our implementation with all officially available example files. Moreover, we devised additional examples which indicate an explicit improvement in comparison to other open-source viewers.

Keywords: B-spline surface, NURBS surface, Industry Foundation Classes (IFC), open-source viewer

1 Introduction

Modern architecture, engineering, and construction (AEC) design often use free-form (curved) surfaces. B-spline and non-uniform rational B-splines (NURBS) surfaces present the best-fitting general-purpose formal geometry description of such geometric representations in storage as well as in expressiveness. The implicit representation allows an infinite smooth surface evaluation for analysis and visualization. To inspect this implicitly defined geometry, a viewer constitutes an essential tool. However, many popular viewers support neither B-spline nor NURBS surfaces. Because of this deficiency, these advanced surfaces often have to be triangulated during the export process from the design software. During this step, the advantages of the implicit geometry description get lost, e.g. the ability for an easy modification and the small size of the exchange file [1].

The file format Industry Foundation Classes (IFC) is getting increasingly popular for AEC data exchange. It supports such complex surfaces with specialized entities [1]. However, not many IFC viewers support B-spline surfaces and even less the NURBS surfaces, especially in the field of open-source software. Thus, IFC exchanges often refrain from using these advanced surface definitions, e.g. both Coordination View and Reference View exclude these specialized entities. We fill this lack of readily available viewers with our solution. This paper serves as a guide to other (open-source) implementors to ensure unambiguous interpretation of advanced surface geometries across the industry.

The paper is structured as follows. The state-of-the-art is described in Section 2. We summarize the concepts of B-spline and NURBS surfaces and explain the corresponding parts of the IFC data model in Section 3. After that, Section 4 presents TUM OpenInfraPlatform (OIP) and describes how B-spline and NURBS surfaces are implemented. Additionally, it addresses the topological properties the computed geometry must fulfill. The validation of our implementation with multiple examples is shown in Section 5. Finally, Section 6 formulates a conclusion and lists remaining future work.

2 State of the Art

We selected 14 open-source and proprietary software solutions available, both online and desktop programs. For each software, we tested the visualization of B-spline and NURBS surfaces. The former is tested with the official example files *basin-advanced-brep* (basin) and *cube-advanced-brep* (cube) provided by buildingSMART International Ltd. [2]. The latter is tested with our own example file containing a NURBS surface, as described in 5 [3]. Table 1 shows the results of the selected IFC viewers. We denote the test as passed if the geometry is correctly shown. An error message, program crash, or a mere false visualization constitute a fail.

Not many IFC viewers support B-spline surfaces and even less the NURBS surfaces. Except for the proprietary viewer of BIMcollab, all viewers fail to visualize the NURBS correctly. Furthermore, most viewers treat the NURBS surface like an ordinary B-spline surface, resulting in incorrect geometry, as depicted in Figure 1. For users who do not know how the surface should look, this mishandling remains unnoticed and constitutes a false impression of correctness.

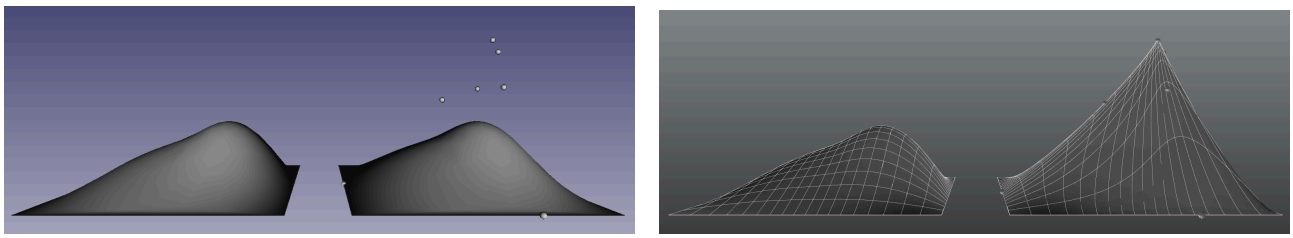

Figure 1: An example data set with a B-spline surface on the left and a NURBS surface on the right, both visualized in FreeCAD (left) and OIP (right).

Table 1: Test results of different IFC viewers for B-spline surface visualization. The viewers are grouped in open-source and proprietary software, as well as online and offline viewers. We have used the most recent software version available to the authors at the time of writing. The test results include B-spline surfaces and NURBS surfaces.

viewer	OIP[1]	FreeCAD[2]	IfcOpenShell[3]	IFC.js[4]	IFC Query[5]	xBIM[6]	BIMcollab[7]	BIMvision[8]	FZK Viewer[9]	Open IFC Viewer[10]	wikiifc[11]	Aspose Viewer[12]	Autodesk Viewer[13]	GOAT viewer[14]
open source	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
online viewer	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
B-spline surface	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
NURBS surface	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗

[1] Jaud, Hecht, Schlenger, *et al.* [4] [2] by opensourceBIM. URL: www.freecadweb.org
[3] *IfcOpenShell* [5] [4] URL: https://ifcjs.github.io/web-ifc-viewer/example/index
[5] by Bauhaus University Weimar. URL: www.ifcquery.com [6] Lockley, Benghi, and Černý [6]
[7] BIMcollab ZOOM. URL: https://www.bimcollab.com
[8] by Datacomp IT Sp. z o.o. URL: https://bimvision.eu/
[9] by Karlsruhe Institute of Technology (KIT). URL: https://www.iai.kit.edu/1302.php
[10] by OpenDesign alliance. URL: https://openifcviewer.com/ [11] URL: https://wikiifc.com/
[12] by Aspose Pty Ltd. URL: https://products.aspose.app/cad/viewer [13] by Autodesk, Inc. URL: https://viewer.autodesk.com/ [14] by VISOPLAN GmbH. URL: https://www.ifcviewer.de/

3 Fundamentals

3.1 B-Spline Surfaces and Non-uniform Rational B-splines Surfaces

In order to visualize the implicit geometry, the actual surface geometry in three-dimensional (3D) space needs to be evaluated. Rogers [7] defines the geometry of B-spline surfaces as:

$$Q(u,v) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} P_{i,j} \cdot N_{i,k}(u) \cdot M_{j,l}(v) \,, \tag{1}$$

based on a rectangular grid of control points $P_{i,j} = [x_{i,j}\ y_{i,j}\ z_{i,j}]^T$ in the 3D space. k and l are the order of the normalized basis functions, with u and v representing the current parameter position within the knot vectors X and Y. The basis functions (also called shape functions) $N_{i,k}$ and $M_{j,l}$ are defined with piecewise polynomials according to the Cox-De Boor Algorithm as:

$$N_{i,k}(u) = \frac{(u - x_i) \cdot N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u) \cdot N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}} \quad \text{with convention } \frac{0}{0} = 0, \text{ and} \tag{2a}$$

$$N_{i,1}(u) = \begin{cases} 1 \text{ , if } x_i \leq u < x_{i+1} \\ 0 \text{ , otherwise.} \end{cases} \tag{2b}$$

The equations 2a and 2b can be applied correspondingly for $M_{j,l}$ with the variables v and y instead of u and x. The polynomial degree of the basis functions N and M are derived as $k-1$ and $l-1$, respectively. NURBS surfaces are defined similarly as:

$$Q(u,v) = \left(\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} h_{i,j} \cdot P_{i,j} \cdot N_{i,k}(u) \cdot M_{j,l}(v) \right) \left(\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} h_{i,j} \cdot N_{i,k}(u) \cdot M_{j,l}(v) \right)^{-1}, \tag{3}$$

with each control point $P_{i,j}$ having a corresponding weight $h_{i,j}$. This determines the influence the corresponding control point has on the surface's geometry, i.e. a high weight value results in a surface closer to the corresponding control point.

3.2 Industry Foundation Classes

IFC is a vendor-neutral data schema that enables the exchange of digital building models throughout the entire lifecycle of a building. The number of software products which support IFC and the range of possible use cases rise rapidly [1]. The International Organization for Standardization (ISO) standardizes the current release of IFC 4.0 in ISO 16739-1:2018 [8].

IFC distinguishes between semantic and geometric descriptions [1]. While the semantic descriptions are not of interest for this study, we focus on a few concepts from the geometric descriptions as presented in Figure 2. IFCADVANCEDBREP allows representing volumes with curved surfaces. The IFCCLOSEDSHELL points to one or several instances of the type IFCFACE, e.g. each side of a cube could be one surface part described by an individual IFCFACE instance. IFCADVANCEDFACE is a subtype of IFCFACESURFACE and inherits its attributes accordingly. Both are subtypes of IFCFACE. IFCADVANCEDFACE has no additional attributes but adds limitations to the allowed attribute types. The actual surface geometry needs to be described by an IFCSURFACE, e.g. by an IFCBSPLINESURFACE-WITHKNOTS or IFCRATIONALBSPLINESURFACEWITHKNOTS that describe a B-spline surface and a NURBS surface, respectively. An IFCFACEOUTERBOUND describes the topology of a boundary loop, which can limit the extent of the surface. [1], [9]

4 Process and Implementation

4.1 TUM OpenInfraPlatform

OIP is an open-source application for viewing and analyzing building information modeling (BIM) models. To achieve this, OIP allows to read, visualize, navigate, and handle IFC and point cloud data

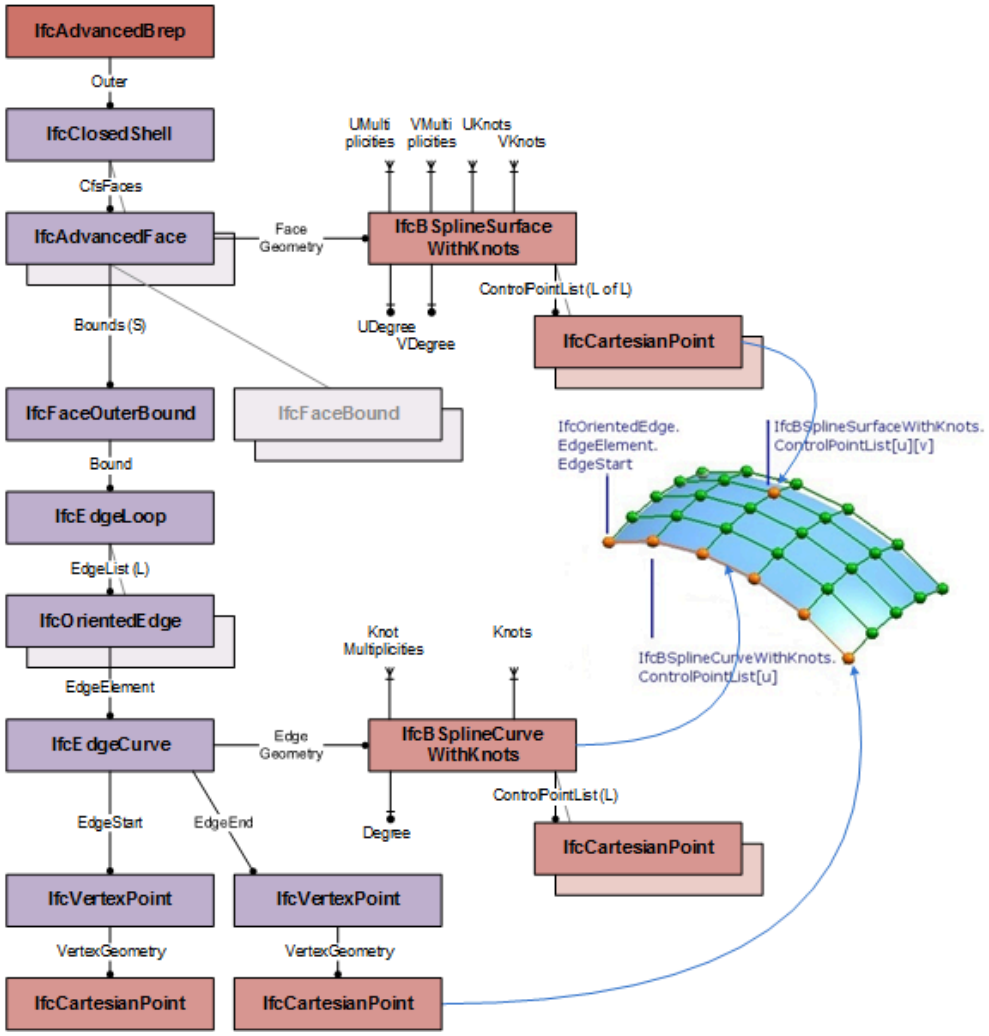

Figure 2: Topological (violet) and geometrical (red) entities orchestrate an IFCADVANCEDBREP with B-spline surfaces as parts of its geometrical description [9].

(PCD). Furthermore, both can be compared by simultaneous loading and overlapping. OIP provides a prototypical playground for developers, in which features and functionalities can be changed, added, or removed as required. One mentionable feature is the support of IFC candidate versions beyond the current release version IFC 4.0. [4]

The software is separated into the parts early-binding generator, graphical user interface (GUI), core components, and additional plugin modules. The core component is written in the programming language C++. Its main task is converting geometric descriptions from IFC to a triangulated geometry, represented in the geometry kernel carve. [10]

4.2 Implementation

In OIP, the core component mirrors the hierarchical IFC structure, as mentioned in Subsection 3.2. There are converter functions for each abstract supertype, which hand over the input parameters depending on their subtype. For example, the IFCADVANCEDBREP provides the attribute outer closed

shell, which is handed over to the corresponding converter function. After that, each part of the object's surface is converted individually. Algorithm 1 describes the processing of a vector of IFCFACEs in more detail, which takes its inheritance structure into account. The entity IFCFACE can be of the subtype IFCFACESURFACE, which in turn can be of the subtype IFCADVANCEDFACE. As mentioned in Subsection 3.2, if the limitations of an IFCADVANCEDFACE are fulfilled, the instance can be treated the same as an IFCFACESURFACE.

Listing 1: Decision tree of IfcFace.

```
Input: ifcFaces
Output: itemData
function convertIfcFaceList
  initialize triangulatedGeometry
  foreach ifcFace in ifcFaces
    if ifcFace is IfcFaceSurface
      ifcFaceSurface ← ifcFace as IfcFaceSurface
      if ifcFaceSurface is IfcAdvancedFace
        ifcAdvancedFace ← ifcFaceSurface as IfcAdvancedFace
        if ifcAdvancedFace has valid attributes
          call computeIfcFaceSurface(ifcAdvancedFace, triangulatedGeometry)
        end if
      else
        call computeIfcFaceSurface(ifcFaceSurface, triangulatedGeometry)
      end if
    else
      call computeIfcFace(ifcFace, triangulatedGeometry)
    end if
  end foreach
  add triangulatedGeometry to itemData
end function
```

The function `computeIfcFaceSurface` calculates the surface geometry depending on its type. Since the geometry can be larger than the boundary loop, it must be trimmed accordingly. One specialized function is `convertIfcBSplineSurface`, which converts instances of the abstract supertype IFCBSPLINESURFACE according to its inheritance structure similar to the algorithm above. The IFCBSPLINESURFACE can be of the subtype IFCBSPLINESURFACEWITHKNOTS. The entity IFCBSPLINESURFACEWITHKNOTS can be of the subtype IFCRATIONALBSPLINESURFACEWITHKNOTS. According to the type, the function loads the attributes from the IFC entity and hands them over to the corresponding compute function, which is independent of the IFC instances. The equations 1 and 3 are used to calculate the actual geometry of the surface in 3D space, according to the algorithms by Rogers [7]. As mentioned in Subsections 3.2, each surface part of the entire object surface is represented in one IFC entity and converted individually. As a final step, these single surfaces are merged into one topological connected polyhedron and coincident vertices are removed to achieve the requirements of an IFCADVANCEDBREP.

5 Results

5.1 Examples

The presented implementation in OIP visualizes advanced boundary representations (BReps) with B-Spline surfaces and NURBS surfaces described in an IFC file. buildingSMART International Ltd. [2] provides the official examples basin and cube, which use B-Spline surfaces. Figures 3 and 5 show the officially provided screenshots of the examples. In comparison, figures 4 and 6 show the corresponding visualizations in OIP.

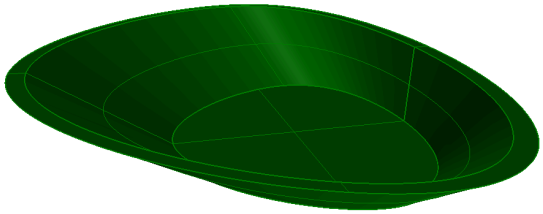

Figure 3: Example *basin advanced brep* [2].

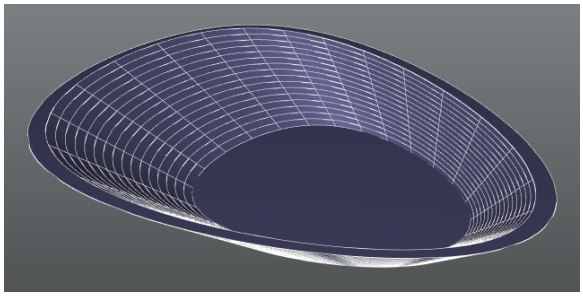

Figure 4: Example *basin advanced brep* visualized in OIP.

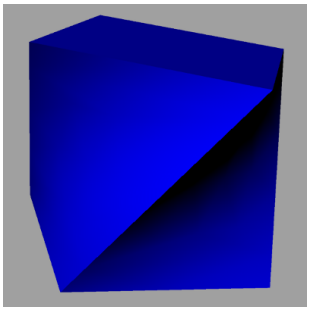

Figure 5: Example *cube advanced Brep* [2].

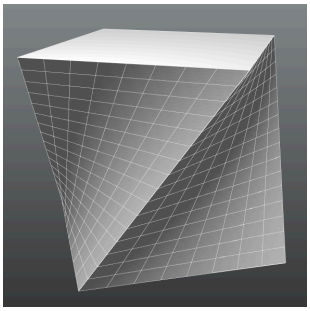

Figure 6: Example *cube advanced Brep* visualized in OIP.

buildingSMART International (bSI) does not provide an official example that includes a NURBS surface. For this reason, we developed the example *rational-BSpline-surface-with-knots* by ourselves to test our solution. The IFC file is available on the OIP GitHub repository [3]. Figure 1 of OIP visualizes the result, which shows a B-spline surface on the left and a NURBS surface on the right. Both geometries are generated with an identical control point grid. By definition, the B-spline surface uses equal weights of the value one at all control points. In contrast, this NURBS surface uses a weight value of 50 at the control point with the increased z coordinate.

5.2 Validation

The IFC documentation provides only a few screenshots and no analytically calculated points for verification. Therefore, checking by eye is the first approach for validation. Comparing Figures 3 and 4

of the example basin, as well as Figures 5 and 6 of the example cube, the geometries of the official screenshots and the visualizations in OIP seem equal. This is the first indication of the correctness of our geometry computation.

Figure 1 of OIP compares a B-spline surface with a NURBS surface. The single increased weight value of the NURBS surface leads to the noticeable peak of the geometry. For further verification of our implementation, we added small markers (spheres with radius = 0.05) in the IFC file (see Figure 1), which highlight a few points on the NURBS surface. The coordinates of these markers are calculated separately from OIP with MATLAB and visually verified with the NURBS surface generator of the 3D creation suite Blender. The center of all spheres lies directly on the visualized surface, which further confirms the correctness of the computed NURBS surface. The shown markers have the following coordinates:

$Q(0.0, 1.0) = (5.0, 2.0, 0.0)$	$Q(0.5, 1.0) = (6.974, 2.0, 1.690)$	$Q(1.5, 1.0) = (7.988, 2.0, 2.887)$
$Q(1.9, 1.0) = (8.098, 2.0, 2.652)$	$Q(2.0, 1.0) = (9.0, 2.0, 0.0)$	$Q(1.5, 0.0) = (7.8123, 0.0, 0.0)$
$Q(1.5, 0.2) = (7.925, 1.421, 1.925)$	$Q(1.5, 1.8) = (7.925, 2.579, 1.925)$	$Q(1.5, 2.0) = (7.813, 4.0, 0.0)$

All three examples demonstrate the ability of OIP to visualize B-spline surfaces and NURBS surfaces. Moreover, they provide validation within the scope of available reference and test data.

6 Conclusions

As discussed, the calculation and visualization of advanced BReps with B-spline surfaces and NURBS surfaces in OIP is more refined in comparison to other free viewers. However, as OIP is a development playground, the IFC standard is not supported fully. For example, not all subtypes of IFCSURFACE have been implemented yet, thus BReps using these subtypes in the attribute face surface could not be visualized. Additionally, the boundary loop trimming that was mentioned above is still to be implemented. The current implementation requires the face surface to fit into the boundary loop, which is the case for all officially provided examples. Applying a suitable trimming algorithm is the next step to improve the variety of possible advanced BReps visualized in OIP. Furthermore, this step eases the support of the pending subtypes mentioned in this section. To further verify the correctness of the visualization, a more automated and quantitative verification process will be developed in the future.

In conclusion, it has been shown that the OIP can overcome common shortcomings of existing free IFC viewers in regards to the visualization of advanced BReps with B-spline surfaces and NURBS surfaces. The OIP open-source project offers excellent transparency to guide other software providers in implementing B-spline and NURBS surfaces. Furthermore, an additional IFC example file has been created that helps software users to assess an IFC viewer's correctness visually.

References

[1] A. Borrmann, M. König, C. Koch, and J. Beetz, *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*, 2nd ed. Springer, 2021.

[2] buildingSMART International Ltd., *IFC 4 - Examples*, Accessed: 2022-03-12, 2020. [Online]. Available: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/link/annex-e.htm.

[3] TUM OpenInfraPlatform, *Rational-bspline-surface-with-knots marks.ifc*, Accessed: 2022-07-10, 2022. [Online]. Available: https://github.com/tumcms/Open-Infra-Platform/blob/development/testdata/IFC4ExampleFiles/rational-bspline-surface-with-knots%20marker.ifc.

[4] Š. Jaud, H. Hecht, J. Schlenger, and J. Amann, "TUM Open Infra Platform: an open source package for simultaneous viewing and analysis of digital models in the civil engineering domain", *Journal of Open Source Software*, vol. 7, no. 72, p. 3061, 2022. DOI: 10.21105/joss.03061. [Online]. Available: https://doi.org/10.21105/joss.03061.

[5] *Ifcopenshell*, Accessed: 2022-05-16, 2021. [Online]. Available: https://ifcopenshell.github.io/docs/.

[6] S. Lockley, C. Benghi, and M. Černý, "Xbim.essentials: A library for interoperable building information applications", *Journal of Open Source Software*, vol. 2, no. 20, p. 473, 2017. DOI: 10.21105/joss.00473. [Online]. Available: https://doi.org/10.21105/joss.00473.

[7] D. F. Rogers, *An introduction to NURBS: with historical perspective*. Morgan Kaufmann Publishers Inc, 2001.

[8] ISO, "Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries – Part 1: Data schema", International Organization for Standardization, Geneva, CH, Standard, Nov. 2018.

[9] buildingSMART International Ltd., *IFC 4 - IfcAdvancedBrep*, Accessed: 2022-03-12, 2020. [Online]. Available: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/link/ifcadvancedbrep.htm.

[10] H. Hecht and Š. Jaud, "Tum openinfraplatform: The open-source bim visualisation software", in *Proceedings of the 31st Forum Bauinformatik*, Berlin, Germany, 2019. [Online]. Available: https://publications.cms.bgu.tum.de/2019_Hecht_Jaud_FBI.pdf.