buildings

MDPI

Article

Implementation of Open Data Exchange between Architectural Design and Structural Analysis Models

Goran Sibenik [1,*], Iva Kovacic [1], Valentinas Petrinas [1] and Wendelin Sprenger [2]

[1] Institute of Interdisciplinary Construction Process Management, Technische Universität Wien, 1040 Vienna, Austria; iva.kovacic@tuwien.ac.at (I.K.); valentinas.petrinas@tuwien.ac.at (V.P.)
[2] BIM 5D, Ed. Züblin AG, STRABAG Innovation & Digitalisation, 70567 Stuttgart, Germany; wendelin.sprenger@zueblin.de
* Correspondence: goran.sibenik@tuwien.ac.at

Abstract: Building information modelling promises model-based collaboration between stakeholders in the project design stage. However, data exchange between physical and analytical building models used for architectural design and structural analysis respectively rarely takes place due to numerous differences in building element representation, especially the representation of geometry. This paper presents the realization of a novel data exchange framework between architectural design and structural analysis building models, based on open interpretations on central storage. The exchange is achieved with a new system architecture, where the program *redDim* was developed to perform the interpretations, including the most challenging transformations of geometry. We deliver a proof of concept for the novel framework with a prototype building model and verify it on two further building models. Results show that structural-analysis models can be correctly automatically created by reducing dimensionality and reconnecting building elements. The proposed data exchange provides a base for missing standardization of interpretations, which facilitates the non-proprietary automated conversion between physical and analytical models. This research fills the gap in the existing model-based communication that could lead to a seamless data exchange.

Keywords: BIM; data exchange; structural analysis; physical model; analytical model; geometry interpretation

Citation: Sibenik, G.; Kovacic, I.; Petrinas, V.; Sprenger, W. Implementation of Open Data Exchange between Architectural Design and Structural Analysis Models. *Buildings* **2021**, *11*, 605. https://doi.org/10.3390/buildings11120605

Academic Editor: Junbok Lee

Received: 15 October 2021
Accepted: 24 November 2021
Published: 2 December 2021

1. Introduction

The construction industry, as one of the least digitized, aims to increase productivity by implementing building information modeling (BIM) and achieving digital collaboration [1]. An obstacle still present in achieving a seamless data exchange for digital collaboration is the lack of software interoperability [2]. The interoperability concept needs to be expanded beyond information systems towards business processes, employees and culture, and the management of external relationships [2]. The computer integrated construction needs to be directed towards the process-oriented systems to achieve a natural level with human elements that are found in the day-to-day businesses [3].

Attempts to improve data exchange and software interoperability have yielded domain-specific success, whereby for architectural design and structural analysis it has been limited. This exchange is still characterized by sharing of drawings, documents with bad interoperability and redundant work [4]. The existing solutions are not able to support the human element and to reliably support the data exchange process [5]. Therefore, this research aims to improve the current data exchange frameworks between architectural design and structural analysis models by overcoming the existing obstacles.

In the previous research, the need to focus on domain-specific data models instead of integrated building models, and the interpretations between them for improving software interoperability and eventually automating data exchange, was identified [6]. The need for multiple models, especially multiple geometries, has been recognized by other

researchers [7,8]. As interpretations or transformations of building models, we understand the methods that edit geometrical and non-geometrical data available in one domain-specific representation according to the representation requirements of another domain. Based on our novel data exchange framework, supporting multiple open domain-specific data models and interpretations between them [9], we follow with the implementation of this data exchange approach in the presented research. The emphasis of our novel approach is on non-proprietary or open interpretations, realized as transparent to end users, which can be edited or extended for heterogeneous practices typically found in the AEC industries.

Structural engineers cannot rely on completely automated processes unless they are transparent or comprehensible ("black-box scenario") [10]; as they carry the responsibility for the calculation and require a completely reliable model as a starting point. Therefore, a framework needs to support open interpretation rules, which could be tailored for additional workflows and practices. Hence, the research question we aim to answer is: How to define and realize open interpretations for the transfer of architectural design to structural analysis building models?

For this purpose we propose, implement and verify a novel data exchange concept supporting domain-specific interpretations on an open central data storage. Due to the lack of documented cooperative knowledge of structural engineers and architects, in this paper we are mapping several common rules for generating an analytical model from a physical model. The novel concept shows potentials in overcoming current misinterpretations in data exchange practices, in particular the geometrical interpretations that pose the main challenge, by reducing numerous software-tool-specific processes to a single central interpretation process. In this way, communication with the proprietary tools is simplified to the information mapping.

This paper is structured as follows: literature review is provided in Section 2; Research design is briefly described in Section 3; Section 4 will present the data exchange framework, followed by a proof of concept for a prototypic model in Section 5; Verification of the proposed approach through implementation on additional two models generated by a different modeler will be presented in Section 6; In Section 7, the results of the research will be discussed, and the work will be concluded with Section 8.

2. Literature Review

Data exchange between architectural design and structural analysis models has been researched from different aspects, mainly as part of research in BIM domain. Since this paper deals with the implementation of the novel framework, the following two subsections review first concepts and challenges of a model-based exchange between the domains in question, followed by the review of technologies that were considered for the novel system architecture.

2.1. Data Exchange Concepts and Challenges

Data exchange implies restructuring and translating data in a source schema in one application to a target schema in another application [11]. The level of achievement of a seamless data exchange varies between professional domains. Interoperability is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [2]. Numerous software tools belonging to different professional domains have also differing internal structures, which hinders the full interoperability across software tools [12]. However, the stakeholders belonging to various domains still aim to exchange information of interest. One of the greatest challenges is the transfer of a building model between architectural design and structural analysis [13]. The experts exchanging information of interest need to have mutual understanding of each other's domains in order to facilitate the communication [14]. The collaboration during the developed design stage (i.e., *Einreichsplan* in Austria) is particularly interesting, since both the architectural design plans and structural analysis are required for the building

permit [15]. Software tools in both domains allow for the use of building models; however, the model-based data exchange is still burdened with problems [6].

Levels of interoperability are described as technical, information and organizational, dealing with signals, data and processes, respectively [16]. Technical interoperability is secure data transfer, information interoperability deals with data processing and organizational interoperability with alignment of processes and workflows. While information interoperability has been achieved, with some issues remaining, organizational interoperability still lacks clear conceptualization. The distinction between geometrical and non-geometrical information is required for data interpretation [16]. Complex manipulation of geometrical information are rarely introduced in frameworks proposed to improve interoperability in the AEC industry, whereby the research focus remains on restructuring geometries by splitting, unifying, excluding or including them [7,17]. These methods do not suffice for achieving interoperability between architectural design and structural analysis models. Interpretations between these models are not pure information interoperability, they depend on the processes and workflows, and bend towards the organizational interoperability.

As in [18], models used for architectural design are referred to as physical (or architectural [19], or BIM models [20]), while the structural analysis models are analytical (or structural [19]). Representations of building elements have significant differences, especially regarding the geometry representation. While physical models aim to render the elements similarly to their real-world 3d shape, analytical models tend to simplify the geometries and reduce dimensionality of building elements to points, lines and surfaces. We identified the need to further explore these geometry interpretations in order to improve the interoperability itself. The interpretations (or transformations [20]) have been performed mostly intuitively based on the experience of structural engineers [13], and are therefore based on implicit knowledge, which is difficult to codify. The interpretations represent the process of defining analytical models based on information available in physical models.

The overview of considered literature investigating model interpretations is provided in Table 1, where different approaches towards requirements, interpretations and exchange formats are visible. The creation of structural analysis model is described as information extraction [19] or fixing (e.g., node adjustment in [20]), without considering the intuitive decision making in the workflows. The previous findings have shown that missing interpretation procedures are the main obstacle in achieving a seamless exchange [6]. Thorough documentation of the interpretations can be found in [20], giving a list of required interpretations for a transformation of an industry foundation classes (IFC) physical model to an IFC analytical model. They emphasize the importance of interpretations, however it is limited to swept-solid geometry defined with the IFC standard (naming physical and analytical representations as swept-solid and topological respectively). In order to further edit the interpretation methods for IFC geometry definitions and contribute to the framework, expert IFC knowledge is required. Authors may focus on developing new data exchange frameworks between architectural design and structural analysis [19,21,22]. Although geometry interpretations are an unavoidable part of each framework, they are not described in detail, serve the examined case studies, but cannot be validated for the general use or other business processes. The existing intuitive interpretation practices are not documented, nor related to the proposed interpretation methods. The interpretations serve for an "end-to-end" model transformation automation without allowing a detailed insight to end users. The interpretation rules are highly dependent on the geometry of building elements, but that aspect is either vaguely described or missing in the literature. As the reviewed research does not focus on the interpretations, corresponding data exchange solutions do not consider editing or creating new interpretations. Due to the heterogeneity of software tools and interpretation practices, such proposals have limited use on the market, similarly as proprietary data exchange solutions. Therefore, we implement open interpretations to data exchange between architectural design and structural analysis models.

Table 1. Literature describing the interpretations.

	Structural Requirements and Interpretations	Exchange Format
[23]	Geometry: beam and column; section properties; geometry interpretation; joints redefining based on connectivity	IFC can be exchanged on the server (XML-based) with Java3d graphical representation
[24]	Geometry: beam, column, slab, wall; section; material; filtering, no interpretation of geometry	IFC export converted to unified finite element model (XML-based), placed on server and imported to structural tools
[10]	Geometry and the associated attributes: no further details about requirements and interpretations	Basic CDF application (VB.Net) bidirectionally communicates between parametric design and structural analysis tool through proprietary file-format
[21]	Geometry: wall, slab, beam, column, brace; material; thickness; section; node modification; filtering based on geometry and material	IFC and other file formats can be exchanged and edited through unified information model on server (JSON-based) which uses OpenGL and WebGL for graphical representation
[25]	Interpretations of non-geometrical information and filtering	Multiple domain-specific SketchUp models communicating through central storage (XML-based with X3D and XPath)
[22]	Geometry: beam, column, wall, slab, opening; section property; material; node modification	Revit directly connected with YJK software tool, which further communicates with structural analysis tools
[19]	Geometry: beam, column, wall, slab; section profile; material; data extraction, no interpretation of geometry	IFC export converted to unified finite element model (XML-based) on IFC-structure model server and imported to structural tools
[20,26]	Geometry: beam, column, member, slab, wall; material; reducing dimensionality; material properties editing; connectivity adjustment	IFC physical to IFC analytical model with the help interpretation platform in C++
[27]	Geometry: wall, columns, openings, floors, roofs, stairs; material; material interpretation; geometry simplification mentioned, not explained in detail	IFC export is extended with additional material information for structural analysis
[28]	Geometry: plates, beams, columns, slabs; geometry discretization without reducing dimensionality	IFC with ACIS geometry kernel; no import to FEM tools
[29]	Geometry: beam, column, wall, slab; material; section properties; joints; reducing dimensionality	Web-based platform which bidirectionally interacts with several proprietary formats through unified information model (IFC-based)

Data exchange between architectural design and structural analysis is reported as the one with the lowest value/ratio and highest frequency [20,30]. This can be assigned to three problems: (i) the structural engineers work with established standards and recommendations which are not suitable for automatic data exchange as such; (ii) academic proposals of data exchange solutions are project- or software-tool-specific and vaguely integrate existing data exchange rules, making them unreliable for engineers as they are held responsible for their calculations; (iii) software industry does not manage to answer the heterogeneity of workflows and provides software-tool-specific solutions whose usefulness and usability depend on specific combinations of software applications. Regarding problem (i), in the previous work [9] the existing standards are investigated and included in the new data exchange framework. Rules with digitalization potential found in standards are mainly suitable for the validation step, before the interpretations take place, whereby the rules suitable for interpretation of geometry are rarely found. In the new framework, the issue (ii) is addressed by a detailed description of interpretation methods and their implemen-

tation as open methods. With such approach structural engineers have insight into the background processes and can edit them if necessary for their needs. The problem (iii) is in the novel data exchange framework resolved with non-proprietary interpretations that can be accessed and edited. Temporary organizations responsible for delivering a building project are generally not assembled based on the software tools utilized in domain-specific enterprises. Proprietary data exchange solutions do not deliver a satisfying performance in numerous possible combinations in the industry characterized by SMEs. However, enterprises encompassing multiple domains can adopt software tools based on their interoperability, but the problem of non-transparent model interpretations remains, and is often overcome by mastering the behavior of the software tools and adapting to adequate workarounds.

2.2. Data Exchange Technologies

Data exchange processes between architectural design and structural analysis have been document- (or file-) based until now. This document-based approach is digitalized with the data-exchange platforms where the advantages of a data-centric approach have still not been fully exploited [31]. The implementation of databases and a data-centric exchange of information provide a possibility to perform synchronous analysis, knowledge representation and large-scale data management. A set of international standards ISO 19650 deals with the organization and digitization of information about buildings and civil engineering works and recommends a common data environment (CDE) for data transfer throughout the whole life cycle of buildings. Part two [32], which describes delivery phase of the assets, is of particular importance for this research. For a successful collaboration via a data-centric CDE, an information delivery planning and responsibility matrix is required along with a higher level of standardized processes. However, the standard [32] does not describe the processes on the scale of specific geometries and building elements, as needed for the exchange between the architectural design and structural analysis. The presented research aims to further build on the recommended framework for information management systems with the processes required for this particular data exchange. The data-centric approach is also seen as a way to distance the data from proprietary solutions. Because of the high data fragmentation in the AEC industry, flexible data integration and sharing is required [4]. The NoSQL (not only SQL) databases have been recognized as a possible solution for the heterogeneous data originating from various proprietary solution [4,33]. The NoSQL database provides scalability, flexibility and performance compared to a relational database. In our work, we use a MongoDB semi-structured database to provide a data-centric open exchange.

Semi-structured schemas are not strict and evolve with time, which is suitable for the AEC industry. Popular serialization formats for a semi-structured schema are JSON (JavaScript Object Notation) and XML (Extensible Markup Language) [34]. Advantages of using the JSON format for being lightweight and human-readable are recognized and it is utilized for data management frameworks in the oil and gas engineering sector [35]. This framework is realized with MongoDB, and certain advantages compared to other JSON databases such as multiple indexing and SQL-similar query language are recognized. IFC building models may be serialized as JSON, and the resulting ifcJSON format arguably shows advantages towards the existing ifcXML serialization [36]. In our work, the JSON format will be used to support a web-based data-centric framework.

A popular approach when it comes to dealing with semantical differences between different applications is the use of ontologies [34]. The ontologies are omitted in this work due to the lack of necessary access control, especially needed in a decentralized collaboration environment [37]. This deficit is currently the subject of improvement efforts and could increase the implementation potential of ontologies in the future. The focus of the works investigating ontologies in the AEC industry has been primarily set on non-geometrical information, while the most complex parts in the data exchange in question are geometrical interpretations. The geometry description of building elements with ontologies

is a recent work [38]. Therefore, we focus on the solutions which offer access control and settled structures for describing geometry.

Discrepancy between life spans of long-lasting products such are buildings and software tools used for their creation should not be reflected on digital product representations which are needed during the whole product life cycle. X3DOM technology can be used to represent geometry in web browsers and help with the plant layout design process [39]. They use a dynamic representation to store geometry information by recording the modelling history. Geometry kernels such as 3d ACIS Modeler (ACIS) have already been used to overcome the differences between architectural design and structural analysis [28,40] and to manage complex geometric interpretation. An additional geometry kernel can be used to generate meshes for finite element modelling, with the focus on boundary-representation (BRep) models [41]. A geometry kernel is useful to store and manipulate the geometrical information for complex geometry methods, therefore the serialization used in our work will not correspond entirely to the existing IFC standard. We chose Open Cascade Library (OC) as the free and open source geometry kernel for the storage and editing of geometric information.

3. Research Design

The presented paper describes the implementation of the novel framework for data exchange supporting open interpretations, including the proof of concept and verification. The main aim of the paper is to facilitate interpretations or transformations of building models independent of proprietary tools used for architectural design or structural analysis; which are not realized as editable, open or extendable in practice, nor any of the existing structural software packages. Table 2 gives a brief overview of the research design. The conceptual framework is thoroughly described in [9]. The current work describes the details of the framework implementation, and tests its applicability in practice.

Table 2. Research design.

Framework concept	Data exchange concept based on domain-specific classification, non-proprietary interpretations between domains and automation of central storage and proprietary solutions [9]
Implementation: Proof of concept	Implementation of the framework with detailed explanation of system architecture (Section 4) Development of algorithms describing individual open interpretations for building elements and belonging geometries found in a single prototype model (Section 5)
Verification	Testing of interpretations developed with the protype model with two additional models and identifying required modifications needed to support new models (Section 6)

Each research step is built around open interpretations, with the aim to identify, implement and verify interpretation methods. The system architecture which is developed for the novel framework is explained in Section 4. Interpretations are developed with the proof of concept, and their usefulness and usability are verified with two further verification models. Open interpretation methods and their realization is a research gap which we bridge with this paper.

4. Novel Data Exchange Framework

A novel theoretical framework and the corresponding system architecture, which incorporate interpretation rules in an open data exchange, were developed in the previous work [9]. Hence, a brief description of the framework as a base for further development will be presented, followed by a detailed description of the corresponding system architecture.

4.1. Concept

The proposed theoretical framework for interdisciplinary data exchange is based on three parts: (i) classification, (ii) interpretation and (iii) automation. The classification part defines the terminology and domain-specific requirements. Semantical and geometrical properties of building elements are clearly distinguished within the classification system [42] as they undergo different interpretations. The interpretation part defines the relations between domain-specific classification systems, including the semantical and geometrical interpretations. The interpretations support the transition from one to another domain-specific classification system and corresponding data. This process can be fully or partially automated. The classification and interpretation parts take place in central storage and are not dependent on proprietary software tools. Finally, the automation part implies automatic communication between the domain-specific models on the central database with the proprietary models and software tools. The concept of the new framework can be compared to the IFC-based data exchange with, however, several key differences: classification is developed as domain-specific and interpretations take place between multiple centrally-stored domain-specific models, while the popular IFC subschemas called model view definitions (MVD) and IFC involve multiple domains, and the interdisciplinary interpretations are not formalized. Proprietary domain-specific tools in our concept communicate with the domain-specific models. The idea is similar to the MVDs which have not been developed as domain-specific and the interpretations between them have not been considered [6]. An additional difference is that the data exchange is not file-based. Figure 1 depicts different parts within the framework.

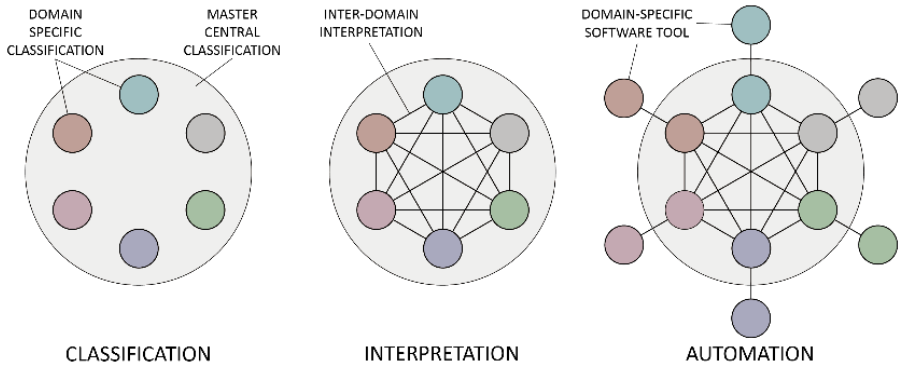


Figure 1. Framework schema (from [9]).

4.2. System Architecture

A digital system comprising software tools, schemas and geometry kernels is developed to support the above presented framework. The system architecture incorporates all three parts: classification, interpretation and automation, and in that order it will be described.

Models can be connected in asynchronous or static and synchronous or dynamic way. Synchronous connection means that a storage (containing a model) or a software tool can be subscribed to another storage or tool and the changes made in the native tool or storage are recognized and updated in the receiving one. On the other hand, asynchronous connection does not allow automatic information updates, and the exchange process must be repeated in order to display the changes. In the described system, synchronous connection represents a connection via an API (application programming interface), while file-based exchange is considered asynchronous. A non-proprietary IFC building model is a starting point in the data exchange framework. A corresponding flexible system architecture can be used with all software tools supporting export to IFC (Figure 2a). However, it does not provide a synchronous connection between the native model in Revit or architectural model in IFC and the central storage, and some data losses and inconsistencies take place during the export from native to IFC models.

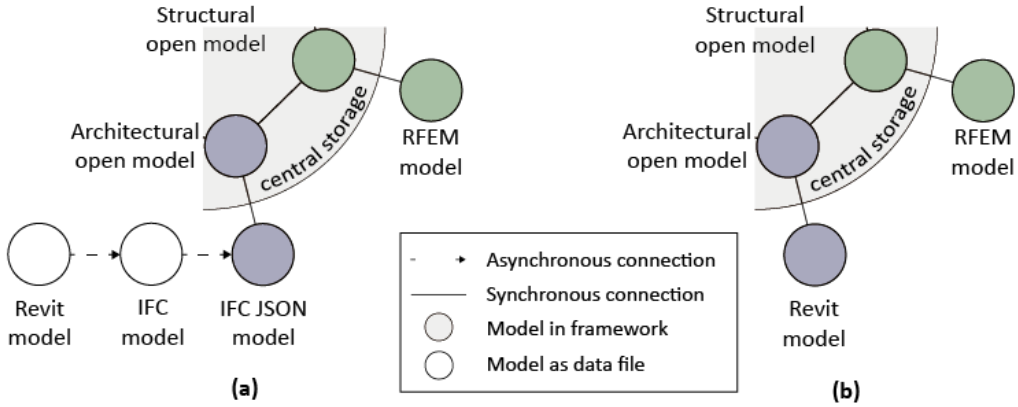


Figure 2. Building models within the (**a**) implemented and (**b**) optimal system architectures.

An optimized, software-specific system architecture with synchronous connection between Revit and a central model can provide fully synchronous exchange (Figure 2b). The system architecture where Revit is directly coupled with the central storage uses Revit API. This work considers one-directional exchange. With the use of APIs and fully synchronous connection, it is possible to extend the exchange to both directions, which is a future work, and requires the investigation of interpretations in the other direction. In order to leave the option of using other architectural tools than Revit open, a suboptimal flexible exchange with IFC is implemented at this stage of the research.

4.2.1. Classification

Classification defines an underlying structure of building models, in this case architectural and structural building models. We opted for MongoDB and JSON, deeming it appropriate for the heterogeneous building models. MongoDB uses documents and groups them in various collections. A data-exchange workflow would benefit from the clear distinction between geometrical and semantic data [42]; therefore, these two types of data are treated separately within the JSON document. The main shared characteristics are that they describe the same building element and both are defined with the JSON format. An example of a building element is presented in Figure 3.

```
{
    "_id": "1000",                                          mandatory attribute building element ID
    "type": "COLUMN",                                       mandatory attribute entity type
    "load_bearing": true,                                   optional* attribute load-bearing property
    "material": "concrete",                                 optional* attribute material
    "cross_section": {                                      optional* attribute cross section
        "type": "RECTANGLEPROFILEDEF",                      *required for structural analysis
        "width": 20,
        "height": 20
    }
    "geometry": {                                           mandatory attribute geometry
        "BRepBuilderAPI_Transform": [...],    OC class name: array of parameters for the class initialization
        "methods": ["Shape"],                               array of methods for the OC class
    }
}
```

Figure 3. Example of a building element with JSON serialization.

1. Semantic information

Information necessary for a data exchange between physical and analytical models, besides geometric information, is information about the element ID, type, load-bearing property, cross section for linear elements and material. This amount of information is sufficient to properly define a model in a structural analysis software tool. Mandatory properties for each data exchange are element type, ID and geometry, and specific for

the exchange to structural analysis are load-bearing properties and material, as well as cross section details for linear elements. The majority of terms used to define the semantic information originate from the IFC 2x3 schema.

Terms defining building element types are: IfcColumn, IfcWall, IfcSlab, IfcBeam and IfcOpeningElement. Element types in a structural model are IfcStructuralCurveMember or IfcStructuralSurfaceMember and StructuralSurfaceMemberOpening.

The element ID originates from IFC JSON, except that the prefix '#' is removed since it is not supported in MongoDB. The ID remains the same in the structural collection after the interpretation.

Load-bearing properties of building elements are MongoDB boolean values. Only the elements which are not "load-bearing: false" are relevant for structural models.

Cross sections for linear elements are defined based on their profile description in the IFC geometrical definition and the corresponding parameter. The cross-section property is a JSON object consisting of a type description and optionally a parameters array.

Materials are defined with a name string, which does not follow any particular standard. SAF project standardizes the format for exchange between various structural analysis software [43], and defines materials: "concrete", "steel", "timber", "aluminium", "masonry" and "other". The architectural model defines the materials mainly for visualization purposes, the structural models and software tools provide more detailed material description with the structural performance properties. Structural materials are assigned by referencing a commonly used material type, for instance "C30/37" for each material named "concrete" in the architectural model.

- Geometric information

OC was used for geometry definitions to enable significant editing and complex interpretation processes. Each building element has its geometry defined in the form {class: parameters array and methods: parameters array}. In such a way, geometries can be automatically retrieved with the help of recursive methods. A similar approach with OC and linked data is developed within SCOPE project [44]. If the geometry definitions are valid, they result in OC topological shapes. The geometries exported after the interpretation are performed are also defined with the OC kernel, in the same form as the geometries of elements in the central architectural representation.

4.2.2. Interpretation

Prior to implementation of interpretation, implicit structural engineering knowledge has to be documented according to specific project requirements and building element typologies, in order to be codified. The documented knowledge results in algorithms which relate the two domain-specific models and are suitable for coding and automation.

To implement the interpretations, we created the *redDim* (*red*uce *Dim*ensionality) program, using the C# programing language. *redDim* provides multiple functionalities to read the architectural model, edit it and write a new structural model. *redDim* is able to read the existing geometrical and non-geometrical information, filter the load-bearing building elements, interpret semantic information, perform multiple methods for interpretation of geometric information, assign additional material properties for the structural model and finally export the structural model to the MongoDB structural collection.

The greatest advantage of *redDim* is the interpretation of geometrical information on the central storage, since the interpretations are the greatest challenge in conversion from architectural to structural analysis models. The interpretations take place with the OC geometry definitions with the help of OC methods. The resulting model is an open structural analysis model, which can be imported by mapping the information to the structural analysis software tool. In that way, the geometry interpretations are not anymore part of the problematic software import.

4.2.3. Automation

Automation part allows coupling of proprietary software with the central storage. As mentioned, IFC building models are used in the system architecture. Programs *IFCtoJSON*, *strucToRISE*, *redDim* and *RFEMimporter* are developed to support the framework (Figure 4). An IFC model is reformatted to a JSON format with a software tool *IFCtoJSON*, and placed on MongoDB as "ifc_json" collection. *strucToRISE* extracts and restructures "ifc_json" information, and creates a central architectural open model. IFC geometry is mapped to OC geometry with *strucToRISE* by using similar classes and preserving parametric element representation (e.g., extrusions, see Figure 13). IFC models contain less information than the native building models—in the case that the IFC export does not suffice, a direct connection with the native model is needed. The architectural model is used for the interpretations and creation of a structural central model. The structural central model can be mapped to any software tool that provides an API. For test purposes and to achieve a full exchange, the resulting structural model was transferred to Dlubal RFEM structural analysis program using a program *RFEMimporter*, with the help of an RFEM API.

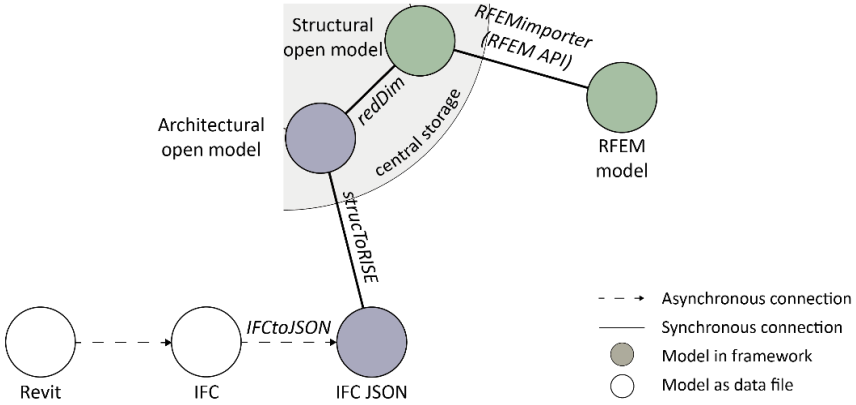


Figure 4. System architecture.

IFCtoJSON, *strucToRISE* and *RFEMimporter* will be described in the follow-up.

- *IFCtoJSON*

In order to use the IFC file on the MongoDB, it is reformatted to the JSON syntax with a self-made converter [45]. All IFC entities and properties are reformatted using the IFC 2x3 schema. In such a way, "key: value" combinations are defined with JSON syntax. A result of the conversion is a JSON file that can be placed on the MongDB. This Mongo collection, named "ifc_json", is a starting point of the novel data exchange.

- *strucToRISE*

From the "ifc_json" collection a central open architectural model is created. Geometry defined in IFC is converted to OC classes and non-geometrical building element properties are restructured (ID, building element type, load-bearing property, cross section and material). The building elements are filtered from "ifc_json" based on the element type.

Load-bearing properties are defined with IfcRelDefinesByProperties; load bearing property is a boolean value defined with an IfcPropertySingleValue. Material is related to the building element with IfcRelAssociatesMaterial; the material is defined as IfcMaterial, IfcMaterialLayerSetUsage or IfcMaterialLayerSet, depending on whether it is a single or multi-layered building element, and if it occurs multiple times in the model. Material names originate from the native software tool Revit and do not follow any standardized terminology. Material naming on central storage follow the naming of SAF (2020) recommendation. However, the conventions for the material terminology are needed so they would not cause interoperability problems.

All the geometries were separately treated and converted to OC classes. Proprietary software that converts STEP to OC exists; however, in this work it was not closely investigated.

- *RFEMimporter*

The RFEM Dlubal API is called RF-COM. IFC classes StructuralCurveMember, StrucutralSurfaceMember and StrucutralSurfaceMemberOpening are defined in RFEM as member, surface and opening elements respectively. Geometric information is transferred to RFEM without any complex interpretation, primarily by reassigning the existing information between OC and internal RFEM kernels. RFEM internally describes cross sections as a single string, e.g., "RECTANGLE 100/200", which was created based on the existing cross-section information. Materials are also defined as a single string. For instance, all concrete elements are defined as "C30/37" or in RFEM terminology "NameID | C30/37@TypeID | CONCRETE @NormID | EN 1992-1-1". If a material does not exist in the standard RFEM library it needs to be defined manually. The element ID is kept in RFEM as a comment which can be assigned with the RFEM API.

Figure 5 depicts an exemplary building element through different steps in the data exchange process. Table 3 lists the created software tools with their specific performing tasks used to realize the proposed framework:

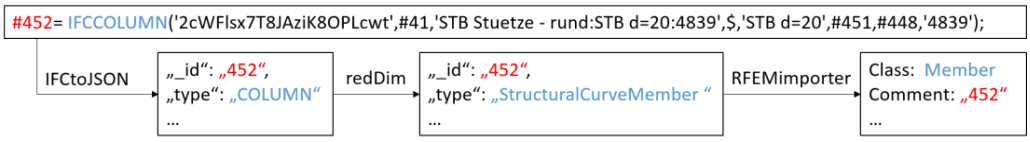


Figure 5. Object ID and type during the data exchange.

Table 3. Summary of developed applications.

Name	Task
IFCtoJSON	Refactors IFC files and IFC schema to JSON format
strucToRISE	Filters and restructures building elements and uses only necessary attributes mainly with IFC terminology, converts IFC geometry definitions to OC
redDim	Core program that performs non-proprietary interpretations between architectural design and structural analysis models
RFEMimporter	Maps the centrally stored structural analysis model to Dlubal RFEM, converts OC geometry and non-geometrical properties to software-specific objects

5. Proof of Concept: Data Exchange Implementation

The following section presents implementation of the novel data exchange framework. The proof of concept builds on the system architecture as presented in Figure 4 to entirely automate the interpretation of the prototype building model (PM). *IFCtoJSON* reformats all IFC files, independent of the building model. However, the other three programs: *strucToRISE*, *redDim* and *RFEMimporter* are developed with the PM. The complete workflow implemented with the PM, from its IFC representation, original and interpreted model in *redDim* and finally resulting import in RFEM Dlubal, is presented in Figure 6.

5.1. Methodology

The PM is a model of the iconic Villa Savoye by Le Corbusier (Table 4). It represents the modernist work of architecture, where concrete columns define the main support system with concrete slabs. Additionally, basement walls and staircase walls on the roof terrace are part of the structural system. This building is chosen primarily as its structural system is commonly used in practice, and involves both load-bearing columns and walls; it also consists of a rich set of standard geometrical forms, mainly polygons extruded in different

directions with straight and curved segments, defining multiple building element types. Some geometries such as columns with square cross sections and rectangular walls are already addressed in the literature (Table 1), but a suitable geometry-specific algorithm for performing interpretations has not been found.

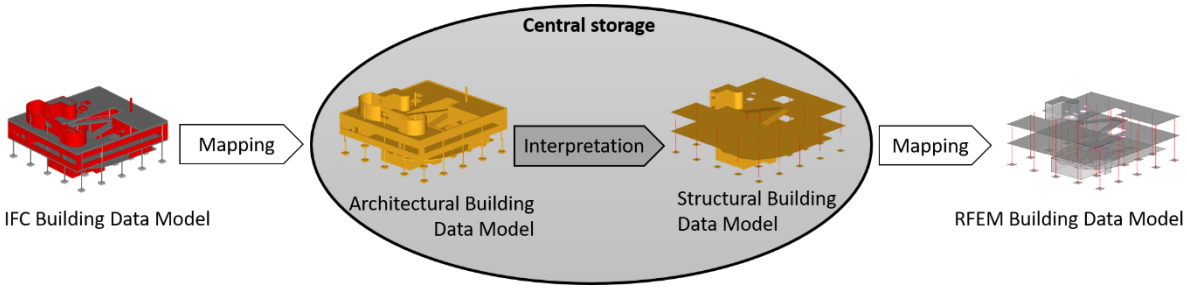
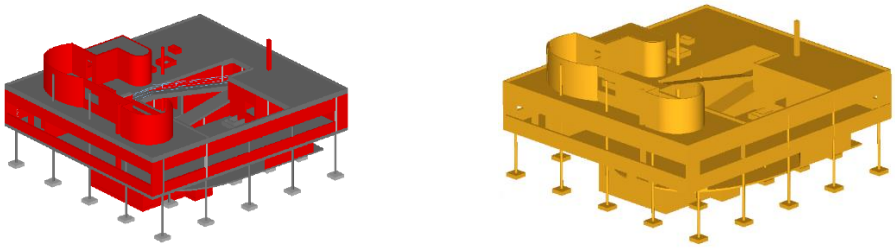


Figure 6. Overview of the PM workflow.

Table 4. Screenshots of PM architectural models.



(a) PM in Solibri (b) PM in *redDim*

5.2. Classification

strucToRISE creates the central open architectural model from the "ifc_json" collection, which follows the previously described classification. The PM architectural collection consists of 421 building elements: walls, openings, columns and slabs, all having properties: element type, ID, material and geometry, and only the linear elements cross-section. The load-bearing property is not defined in 26% of building elements, while 27% of the rest have value "true". Materials of load-bearing elements are "concrete" and "masonry", whereas building materials of non-load bearing elements are not focus of this work. All geometries are defined with OC definitions, and the resulting shapes are automatically created. These properties within the architectural model suffice for data exchange purposes.

The structural model consists of 125 building elements: StructuralCurveMembers, StructuralSurfaceMembers and StructuralSurfaceMemberOpenings, with the corresponding properties. This data is mapped and rendered as structural model in RFEM Dlubal.

5.3. Interpretations

Before the implementation of interpretations, necessary implicit knowledge needs to be documented in such a way that it can be codified, computationally readable and automated. The interpretation algorithms originating from implicit knowledge depend on project properties and building elements. Hereby proposed algorithms will be verified with additional models in order to evaluate their standardization potential.

5.3.1. Documentation of Implicit Knowledge

The geometry interpretation methods for the data exchange from the physical into an analytical model, developed and applied within the framework, include linearization, planarization, reconnecting elements and perimeter adjustment. The methods are based on the previous literature review (e.g., [20,22]), and were chosen in order to support

specific geometries present in the PM, such as columns with round and rectangular cross section, straight and curved one-layered walls with uniform thickness, horizontal and tilted uniform-thickness slabs and single foundations.

- Linearization

Linear elements in the PM are columns with round and square cross sections. They are represented as OC topological shapes cylinders and boxes. A single algorithm converts the shapes to edges, consisting of several steps (Figure 7): (i) top and bottom faces are identified by analyzing and comparing all the faces constituting the shape; in the case of non-vertical linear elements (such as beams) two parallel faces with the smallest area can be identified; (ii) vertices defining the centers of mass of the faces is found; and (iii) an edge is created between the vertices representing the linear element.

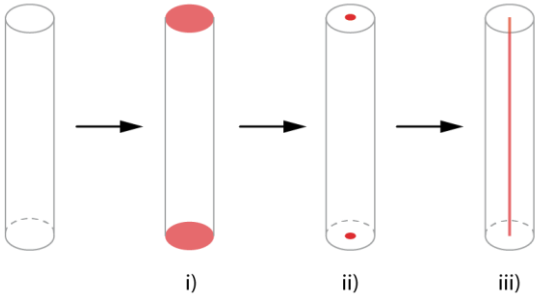


Figure 7. Linearization of building elements.

- Planarization

Planar elements in PM are walls and slabs (including flat roof), both represented as topological shapes of OC, some containing openings. The planarization method involved (Figure 8): (i) largest face of the shape is located; (ii) the element thickness is found by analyzing distances from the central face vertex to its projection on the neighboring surfaces; and (iii) the largest face is offset for half of the thickness to the inner part of the topological element. The result is a planar element axis as the OC face. Analytical representation of openings is an intersection between the physical opening geometry and planarized host element, so it would be coplanar with the host element surface.

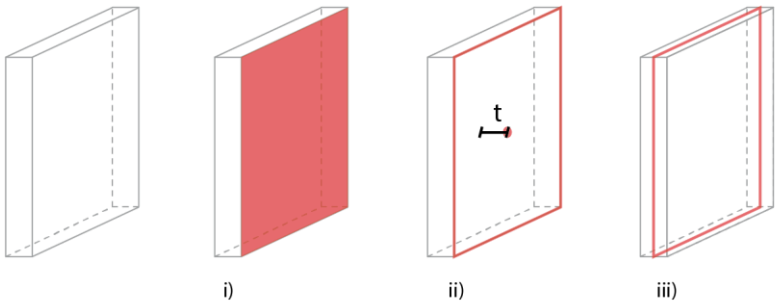


Figure 8. Planarization of building elements.

- Reconnecting Elements

Physical building elements are investigated for their neighboring elements. Based on the results, the analytical elements are reconnected. Reconnect elements consists of (Figure 9): (i) finding neighboring elements in the physical model by checking the minimum distance between elements, and considering the element type (e.g., columns cannot be reconnected to other columns or walls, but can to slabs) and inclination (tilted slabs); (ii) addressing each element and their neighbors in analytical representation; depending on whether an element is linear or planar: (iii)(a) linear elements or edges in OC are extended

until the intersection point with the neighboring planar element; (iii)(b) the planar elements are extended or trimmed: the closest edges are projected, adjacent edges extended or trimmed and the remaining edges unchanged. Vertical planar elements are connected both to horizontal and vertical neighboring planar elements. Openings are reconnected to the neighboring elements, and the complex geometry of hosts with openings is updated.

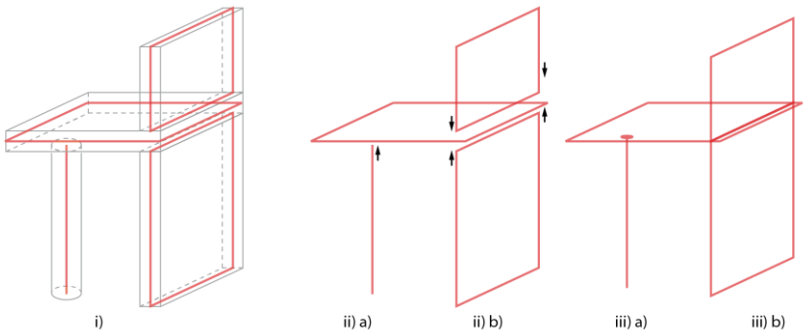


Figure 9. Reconnecting elements in structural representation.

- Perimeter Adjustment

The information about the neighboring building elements is used in order to adjust the perimeter of horizontal planar elements. The edges of horizontal planar elements are adjusted to the building elements supporting them. The elements near the perimeter edges are identified, and linear vertical elements connected to the two closest vertical load-bearing elements on the same side of the slab and in the same neighbor list to create *load-bearing fronts*. These *fronts* determine if the slab edge should be adjusted to the underlying columns. For perimeter adjustment the following steps are applied (Figure 10): (i) vertical elements close to the perimeter in the analytical model are found; (ii) "load-bearing fronts" are created from the vertical linear elements; (iii) planar vertical elements and "*fronts*" are projected on the horizontal planar element; (iv) depending on the distance between intersection and the edge of the slab, and the tolerance, some neighbors are eliminated; parallelism of the edges and the projections is checked; and (iv) the face is extended or trimmed to the edges which fulfill the conditions from (iii).

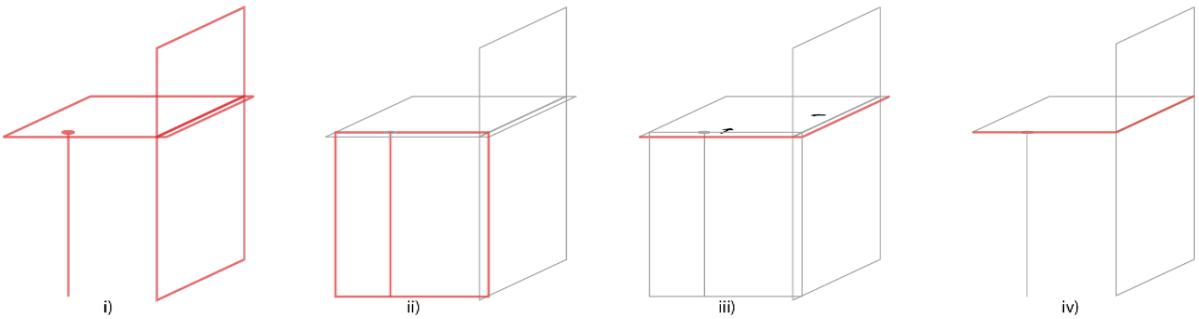


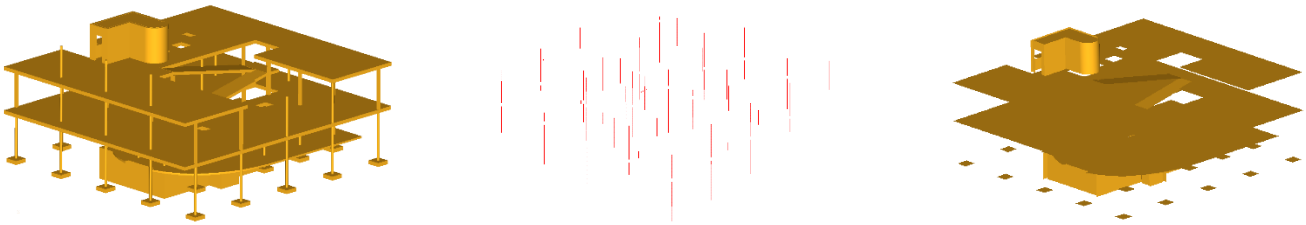
Figure 10. Perimeter adjustment of horizontal structural elements.

5.3.2. redDim

redDim program is developed for the PM of Villa Savoye with all necessary interpretation methods. The architectural model available on MongoDB is read with MongoDB API and NET framework. OC libraries are required to create and edit OC classes. *redDim* provides the method "original" showing the original geometry found in the architectural model. This geometry includes all elements, irrelevant of their load-bearing properties. Since *strucToRISE* and *redDim* were created for PM, validation was not required. *redDim* is able to read the complete model geometry (Table 4b). The method "filtered" renders all

elements which do not have the load-bearing property "false" (Table 5a). This model is the working model for the geometry interpretations. Semantic interpretations take place on the go with geometric interpretations. The interpretations follow the implicit knowledge: linearization, planarization, reconnecting elements and perimeter adjustment.

Table 5. Results of *redDim* methods.



(a) Filtering result (b) Linearization result (c) Planarization result

Linearization affects the columns (Table 5b). Two smallest faces of the topological shape are found with ShapeExplorer and tested for parallelism. Centers of mass for both faces are determined with BRepGProp surface properties method, resulting with two points. The edge is created with the two points, describing a linear building element. The element type is redefined as a structural curved member.

Planarization reduces dimensionality of slabs and walls (Table 5c). The openings, both in slabs and walls, are planarized separately. The largest face of each slab and wall is found with ShapeExplorer and BRepGProp. The parallel face is found by comparing normal directions of underlying surfaces. The central point of largest face is projected to the parallel face and the distance between the points is found, defining the thickness of the building element. The largest face is offset for half of element thickness, creating the structural representation of the building element. Planar elements, such as walls and slabs, change their building element type to structural surface member in the structural representation. Since openings are dependent on their host elements, they are planarized by finding an intersection of the planarized host element representation and physical geometry of an opening.

Linearization and planarization reduce dimensionality of all load-bearing elements. Elements are reconnected in their structural representation with reconnect elements and perimeter adjustment. *redDim* tests all elements for the distance in the physical model, and each element is assigned a neighbor list. Each building element is connected to their neighboring elements: columns are extended to the slabs by redefining edges in their structural representation with an intersection point and a more distant edge point; Walls are extended to each other and to the slabs by identifying the edge closest to the neighbor and its adjacent edges, finding the intersection line between two planar surfaces, defined by the wall and the neighbor, identifying the intersection points of curves of adjacent edges and neighboring surface, using these points to define an edge on the intersection line, and extending or trimming adjacent edges (similar to linear elements). This method is used to extend walls, openings belonging to the walls and ramps. Horizontal slabs are interpreted with the adjust perimeter method (Figure 11).

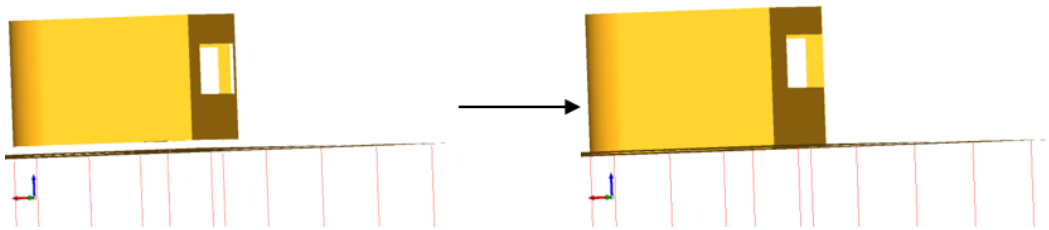


Figure 11. Before and after reconnecting elements.

Slabs and openings belonging to slabs are treated with perimeter adjustment method, using already defined lists with neighbors (Figure 12). Only the neighbors close to the perimeter, which is a wire of a face defining the structural representation, are filtered. Each column in the list of neighbors is connected to the two closest elements in the same list, which are on the same side of the slab, creating "load-bearing fronts". These columns are hereafter considered as planar "load-bearing fronts" and not as linear elements. Intersection lines between the slab and the planar neighbor are analyzed. Slab edges are adjusted to the ones satisfying the closeness and parallelism conditions. Elements with curved edge had to be specially treated. If an edge neighbors a curved edge, a curved edge is not extended. The curved edge is defined on the intersection line between the two surfaces; its end points are designated as an intersection of a line passing through the center point of the arc and the original end point intersection with the neighboring surface. The adjacent edges are adjusted to these intersection points.

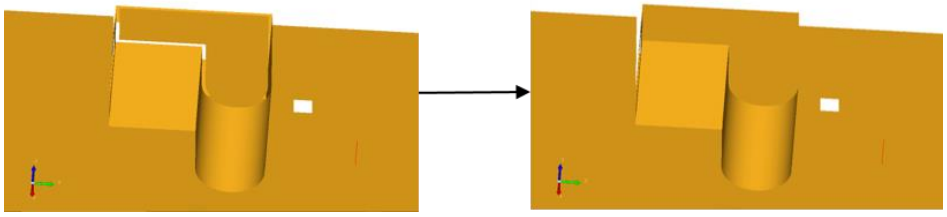


Figure 12. Before and after perimeter adjustment.

Last method in *redDim* exports the structural model. Exported are OC classes and methods defining the geometry, and additional parameters such as ID of building element, entity type, thickness calculated within *redDim* and profile cross section for linear elements from the architectural model. All load-bearing elements in their structural representation are available in a new collection on MongoDB.

5.4. Automation: StructToRISE and RFEM Importer

The central database contains architectural design and structural analysis models, which are related to the external domain-specific models. IFC in JSON format is used as the external architectural model, and RFEM Dlubal contains the external structural analysis model. *structToRISE* accesses the IFC model and defines the "ifc_json" collection to central architectural collection, creating 421 building elements from 20046 entities in the "ifc_json" collection. For each building element its reference number or ID and its entity type is transferred to the MongoDB. Geometries existing in the original model such as IfcMappedItem, IfcExtrudedAreaSolid, IfcBooleanClippingResult, IfcFacetedBrep, IfcHalfSpaceSolid and IfcPolygonalBoundedHalfSpace are identified by "body" representations of building elements. These geometries are redefined as OC classes. *RFEMimporter* maps the central structural building model to RFEM internal model with all the necessary properties, merely by mapping the geometries and properties to the RFEM internal classes. Converting OC to RFEM classes is a simple procedure compared to geometry interpretations. As an example, Figure 13 shows extruded area solid in its IFC and OC representations, where the focus is on geometry information.

```
#6=     IFCCARTESIANPOINT((0.,0.,0.));
#19=    IFCDIRECTION((0.,0.,1.));
#23=    IFCDIRECTION((1.,0.));
#31=    IFCAXIS2PLACEMENT3D(#6,$,$);
#32=    IFCLOCALPLACEMENT(#33320,#31);
#35=    IFCPERSON($,'','***@hotmail.com',$,$,$,$,$);
#37=    IFCORGANIZATION($,'','',$,$);
#38=    IFCPERSONANDORGANIZATION(#35,#37,$);
#41=    IFCOWNERHISTORY(#38,#5,$,.NOCHANGE.,$,$,$,1591271314);
#94=    IFCAXIS2PLACEMENT3D(#6,$,$);
#95=    IFCDIRECTION((6.123031769111189E-17,1.));
#97=    IFCGEOMETRICREPRESENTATIONCONTEXT($,'Model',3,0.001,#94,#95);
#102=   IFCGEOMETRICREPRESENTATIONSUBCONTEXT
        ('Body','Model',*,*,*,*,#97,$,.MODEL_VIEW.,$);
#140=   IFCAXIS2PLACEMENT3D(#6,$,$);
#141=   IFCLOCALPLACEMENT(#32,#140);
#239=   IFCCARTESIANTRANSFORMATIONOPERATOR3D($,$,#6,1.,$);
#975=   IFCCARTESIANPOINT((0.,0.));
#977=   IFCAXIS2PLACEMENT2D(#975,#23);
#978=   IFCCIRCLEPROFILEDEF(.AREA.,'STB d=20',#977,10.);
#979=   IFCCARTESIANPOINT((-136.,610.,0.));
#981=   IFCAXIS2PLACEMENT3D(#979,$,$);
#982=   IFCEXTRUDEDAREASOLID(#978,#981,#19,280.);
#986=   IFCSHAPEREPRESENTATION(#102,'Body','SweptSolid',(#982));
#988=   IFCAXIS2PLACEMENT3D(#6,$,$);
#989=   IFCREPRESENTATIONMAP(#988,#986);
#992=   IFCMAPPEDITEM(#989,#239);
#994=   IFCSHAPEREPRESENTATION(#102,'Body','MappedRepresentation',(#992));
#996=   IFCPRODUCTDEFINITIONSHAPE($,$,(#994));
#998=   IFCAXIS2PLACEMENT3D(#6,$,$);
#999=   IFCLOCALPLACEMENT(#141,#998);
#1000=  IFCCOLUMN('2cWFlsx7T8JAziK8OPLcZu',#41,'STB St\X2\00FC\X0\tze –
        rund:STB d=20:5288',$,'STB d=20',#999,#996,'5288');
#33319= IFCAXIS2PLACEMENT3D(#6,$,$);
#33320= IFCLOCALPLACEMENT($,#33319);
```

```
"geometry": {
    "BRepBuilderAPI_Transform": [{
        "BRepBuilderAPI_Transform": [{
            "BRepPrimAPI_MakeCylinder": [10, 280],
            "methods": ["Shape"] }, {
            "gp_Trsf": [],
            "methods": [{
                "SetTransformation": [{
                    "gp_Ax3": [{
                            "gp_Ax2": [{
                                "gp_Pnt": [] }, {
                                "gp_Dir": [0, 0, 1] }, {
                                "gp_Dir": [1, 0, 0] }] }],
                        "methods": [{
                    "SetLocation": [{
                            "gp_Pnt": [-136, 610, 0] }] }] }, {
                "gp_Ax3": [{
                    "gp_Ax2": [{
                            "gp_Pnt": [] }, {
                            "gp_Dir": [0, 0, 1] }, {
                              "gp_Dir": [1, 0, 0] }] }] }] },
        false],
        "methods": ["Shape"] }, {
        "gp_Trsf": [],
        "methods": [{
            "SetTransformation": [{
                "gp_Ax3": [{
                        "gp_Ax2": [{
                            "gp_Pnt": [] }, {
                            "gp_Dir": [0, 0, 1] }, {
                            "gp_Dir": [1, 0, 0] }] }],
                    "methods": [{
                            "SetLocation": [{
                                "gp_Pnt": [0, 0, 0] }] }] }, {
                  "gp_Ax3": [] }] }] },
    false],
    "methods": ["Shape"] }
```

Figure 13. IFC (**left**) and OC (**right**) geometry definition.

6. Verification

An additional aim of this work is to test the applicability of the demonstrated approach to external models (created by various modelers). Since the application is still conceptual and a work in progress, debugging was not considered as model-related change (e.g., error handling was done on the go), but rather new and changed interpretation methods which were required for the new models.

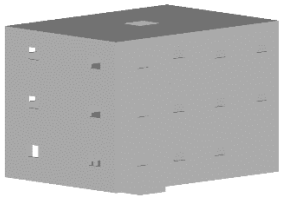
6.1. Methodology

The interpretations defined with the PM are tested on additional two building models. Two new exports of IFC models from Revit are created and used in the novel framework—Verification Building Model 1 (VM1) and Verification Building Model 2 (VM2) (curtesy of STRABAG AG). Native Revit models originate from a different planner or planners, where a company specific workflow and standards were implemented. The models, as real use cases, are provided by a structural engineering company and display a similar structural system as in the PM. "ifc_json" models are imported to MongoDB. *strucToRISE* supported all geometries for creating the architectural model. *redDim* is adapted to the new building elements and geometries (Section 6.2). *RFEMimporter* created for the PM was sufficient for the proper import of the new models. The methods defined for the PM were tested on complete automation of the building model interpretations. The methodology primarily involved software testing and visual checks of the resulting model. The imported structural models in RFEM Dlubal are, the same as PM, verified by authors and the external structural engineer.

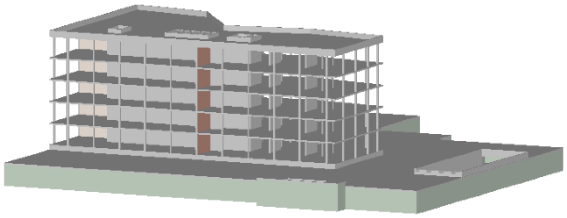
VM1 is a three-story building with concrete load-bearing walls. VM2 is a five-story building with an underground garage and structural system as a combination of load-bearing walls and columns. IFC models and the corresponding OC representations of verification models are shown in Table 6.

The verification models extend implicit knowledge documented for the PM, and the previously described methods are updated.

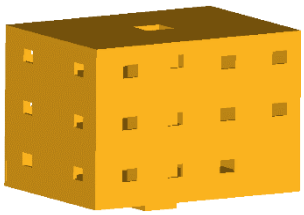
Table 6. Screenshots of VM1 and VM2 architectural models.



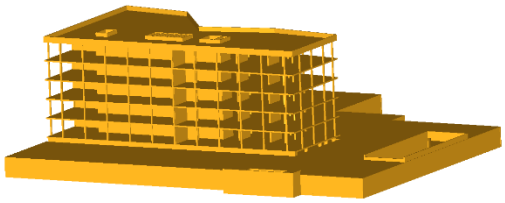
(a) VM1 in Solibri



(b) VM2 in Solibri



(c) VM1 in *redDim*



(d) VM2 in *redDim*

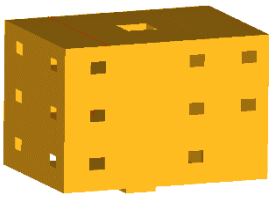
6.2. Results

The interpretations defined with the prototype building covered more than 80% of building elements of the VM1. The VM2 had only a few additional adjustments, after the changes for the VM1 were implemented. VM1 has 84 architectural and 83 structural, while VM2 has 613 architectural and 366 structural building elements. Results show a great potential in standardizing the majority of interpretations for the most common geometries and building elements.

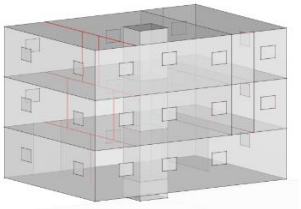
6.2.1. Verification Model 1 (VM1)

The geometry of the VM1 was supported entirely by *strucToRISE* and *RFEMimporter*. *redDim* behaved unexpectedly and it was adjusted to the new interpretations. With the adjustments it was able to automatically create a structural model (Table 7a), which could be imported to RFEM Dlubal with *RFEMimporter* (Table 7b).

Table 7. Screenshots of VM1 structural model.



(a) VM1 in *redDim*



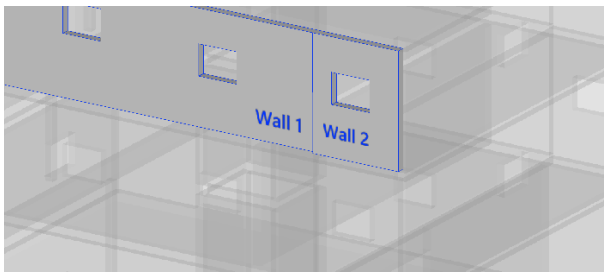
(b) VM1 in RFEM

Differences in VM1 compared to the PM were identified as follows:

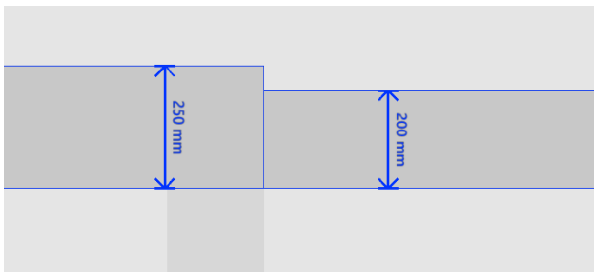
- Continuing walls

The intersections of parallel walls with the same thickness do not exist (Table 8a). However, parallel walls were present in the lists with neighbors and since it is a standard modelling approach *redDim* required an extension—the neighbors are tested for their parallelism and divided into two groups. Linear elements are parallel if their main directions are same, and planar if their normals to the face are parallel. Linear and planar neighbors are considered parallel if the direction of linear element is orthogonal to the normal of the planar face.

Table 8. Building elements which required additional methods in VM1.



(a) Continuing walls



(b) Continuing slabs

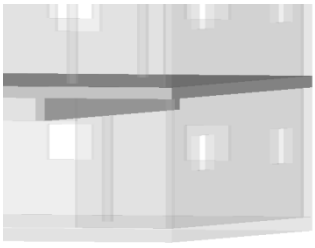
- Aligning continuing slabs with varying thickness

Neighboring building elements with varying thicknesses were not part of PM. The staircase slab is a neighboring standard floor slab, and their thicknesses are 20 and 25 cm respectively (Table 8b). The slabs have their bottom faces aligned, however, after planarization; the resulting faces display a 2.5 cm distance between each other. Parallel neighbors which are not in the same plane after the planarization need to be aligned. Thus, as an addition to the reconnecting elements method, in fact, as its first step, an alignment of neighboring structural faces is performed. The smaller face, area-wise, is moved to be coplanar with the larger one. An additional connectivity problem was found due to the varying slab thicknesses: the underlying wall is modelled as a single element, and connected only to one of the two slabs. If the alignment is performed as the first step, this problem is avoided. The eccentricity of the slab was added as additional information based on the distance between the original and final position.

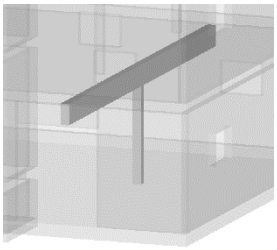
- Aligning beam to slab

In the PM there were no beams; however, in the VM1 beams supporting slabs were present (Table 9a). When the beam is linearized, it loses its connection to the slab in its structural representation. The beam is aligned to the slab from the neighbor list. The distance between the beam and the slab is added as the eccentricity of the beam, when the beam is translated to the face which represents the structural representation of slab.

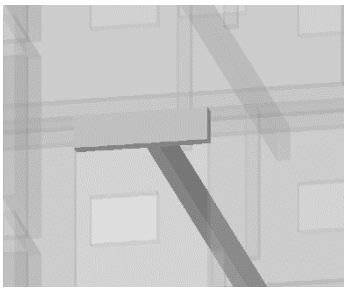
Table 9. Building elements which required additional methods in VM1.



(a) Beam supporting slab



(b) Column supporting beam



(c) Beam neighboring beam

- Extending column to beam

In the VM1, a column-supporting a beam exists (Table 9b). The column is reconnected to the beam after the beam is aligned with the slab. For checking the extension distance, beam eccentricity is considered, since the beams have previously been aligned with the slab.

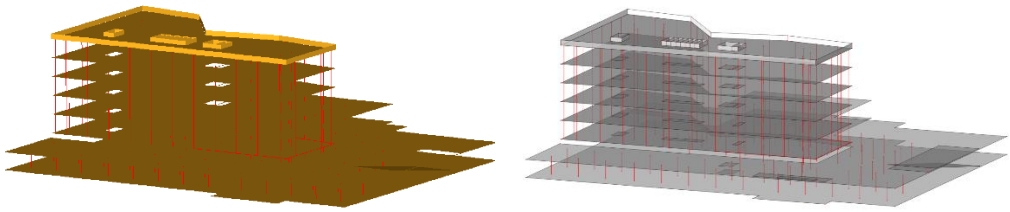
- Extending beam to beam

Since there were no beams in the PM, there were also no connections between two beams. In VM1, two beams supporting the same slab are neighboring each other (Table 9c). Both beams are aligned to the face of the corresponding slab in its structural representation, which made them coplanar. Because of that there was no problem in finding the beam intersection.

6.2.2. Verification Model 2 (VM2)

Changes in *redDim* to completely automate the interpretation of the physical into analytical model involved addressing the issues of linearization of beams with non-parallel smallest faces and walls with multiple connectivity options. After the changes in *redDim*, a structural model could be successfully created (Table 10a) and imported to RFEM (Table 10b).

Table 10. Screenshots of VM2 structural model.



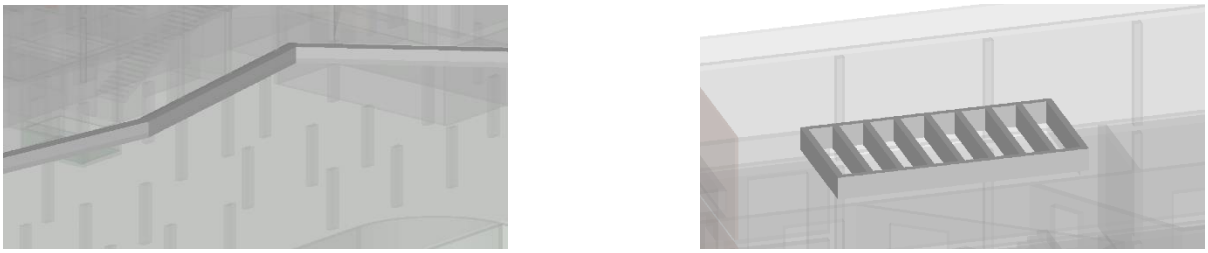
(a) VM2 in *redDim* (b) VM2 in RFEM

VM2 building was addressing additionally the issues of:

- Linearization of beams with non-parallel smallest faces

Linearization was finding two smallest faces of a shape to create the linear element representation. The method additionally involved checking if the faces were parallel to each other. This approach did not work for beams found in VM2 since there were beams that were not orthogonally connected to each other and therefore their smallest faces were not parallel (Table 11a). Parallelism was excluded as a condition for linearization and the method worked properly for the PM and VM1 without it.

Table 11. Building elements which required additional methods in VM2.



(a) Beams with non-parallel smallest faces (b) Walls with multiple connectivity options

- Reconnecting walls with multiple connectivity options

In the VM2, some walls had multiple neighbors that were satisfying the connectivity relations from the previous two models (Table 11b). Due to this, a wall was not trimmed to the correct neighbor, whereby it lost connection to the neighbor closest to the edge. Therefore, an additional condition was added, which checks if the cut-off or extended part is within the tolerance range. The tolerance is defined as the area calculated with the intersection and thickness of the neighbor, with the consideration of the angle between the elements. This approach eliminated the wrong connections and worked with the previous models as well.

7. Discussion

The research question of "*how to define and realize open interpretations for the transfer of architectural design to structural analysis building models*?" is answered by implementing a novel data exchange framework, demonstrating it with a prototype building and verifying it with two additional building models. Thereby, a complete development of a new data exchange approach—from the novel framework implementation, to proof of concept and verification—was presented. The new approach shows potential in overcoming misinterpretations that are the state of the art in data exchange practices. The focus of the paper is set on the open geometrical interpretations which can be understood, edited and expanded to satisfy heterogeneous workflows, and currently pose a main challenge for a seamless data exchange.

Four interpretation methods were initially proposed based on the geometrical elements of the PM. The four methods cover some common building elements and relatively simple but commonly present geometries such as columns with constant cross-sections and walls and slabs with constant thickness. The interpretations take place during the conversion of a physical model to an analytical model. They have not been standardized nor documented for the existing workflows, which causes inconsistencies in the data exchange practices between the architectural design and structural analysis domains. The interpretation methods are implemented with a novel framework, which is described in the previous work [9].

After the verification of the implemented exchange with two additional external building models, VM1 and VM2, a great similarity is found between the interpretations. More than 80% of both verification models were successfully interpreted with the methods defined for the PM. After the automated exchange is achieved for all three building models, the geometrical interpretations can be summarized in two method groups: reducing dimensionality and reconnecting elements. In the PM, reconnecting elements and perimeter adjustment were recognized as separate methods, while in the verification models, alignment of building elements was also required. All three methods, reconnecting elements, perimeter adjustment and alignment, are actually different approaches for reconnecting elements. Which method is to be applied depends on the neighbors and their position compared to the element to be extended or trimmed. Further analysis can be performed and additional verification models tested to obtain more detailed results. However, the approach until now has shown significant potential for the interpretation automation.

The overview of the interpretations and final summary is presented in Figure 14. The required interpretations can be categorized into two groups: dimensionality reduction and reconnection of elements. Dimensionality reduction is defined with linearization and planarization. We also previously recognized the need for punctualization (marked with *), where 3D elements are represented as points; however, this method was not present in the tested models. On the other hand connections between the elements need to be reestablished. In the analyzed models, three reconnection methods were identified: aligning parallel building elements, simple reconnection of elements with low number of neighboring elements and complex reconnection (or in PM perimeter adjustment) where each edge of the building element is tested individually. This overview gives the final proposal for the geometrical interpretations between physical and analytical models. As the main motivation is to realize the non-proprietary interpretations, the methods allow editing and extending if other workflows or geometries require changes. However, based on the verification results, the proposed methods cover most building elements and geometries found in building models.

The main advantage of the novel framework is that it moves the procedural steps for interpretation from single software tools (either architectural design or structural analysis) to the central database. Interpretation methods are facilitated with the open source geometry kernel on the central database. Therefore, the end users (or coordination/BIM managers) have insight into the methods taking place, understand the model behavior and can adjust it to their workflows if needed. This is currently in practice mostly impossible

since the import or export procedures work as "black-boxes" defined in the proprietary software tools.

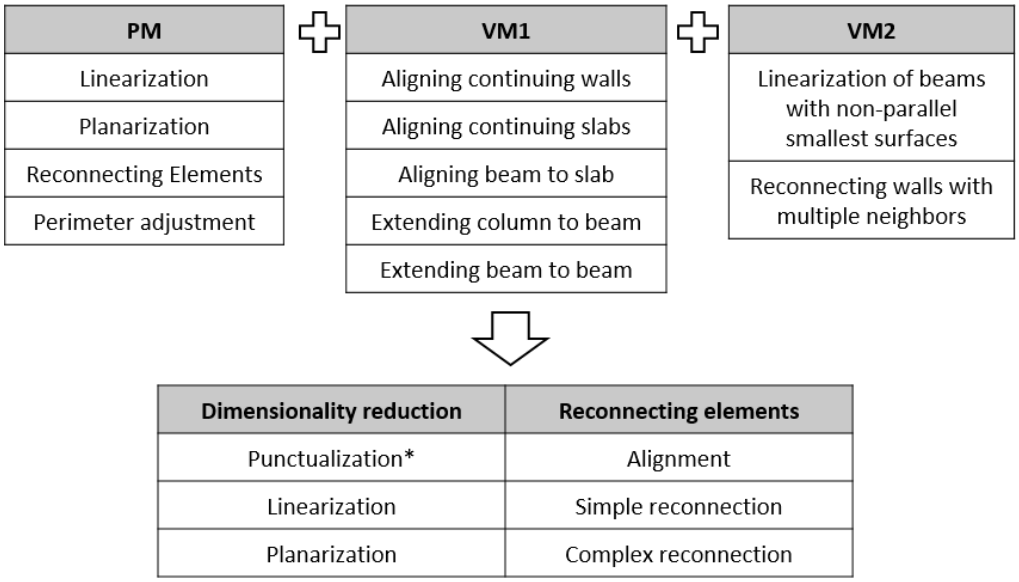


Figure 14. Summary and overview of results.

The innovative aspects of this research lie primarily in the documentation and codifying of intuitive interpretation rules based on implicit expert knowledge, making it explicit, thus enabling editing, expanding and automation via other software tools or geometry kernels. Additionally, the implementation of the interpretations in the novel open exchange framework making them comprehensible to end users represents further innovative contribution.

All building elements in the use-case building have been successfully interpreted, through application of methods for dimensionality reduction and reconnecting elements. Some interpretations were more elaborate to automate than the others, partly because of the complexity of the traditional intuitive interpretation process and partly because of the technical reasons such as geometry kernel suitability and predefined methods in the kernel. The level of difficulty depended on several factors: (a) difficulty of translating IFC to OC geometry; (b) understanding the traditional implicit interpretation process; and (c) difficulty in bridging the gap between the traditional implicit knowledge and explicit method.

8. Conclusions

This paper aims to bridge a knowledge gap through documenting and codifying the implicit interpretations of architectural design models for structural analysis models. This knowledge is found to be a requirement in overcoming the data exchange difficulties between architectural design and structural analysis building models.

Methods required to perform geometrical interpretation are dimensionality reduction and reconnecting elements. Although the described geometrical interpretations represent only a small part of numerous possibilities and undocumented knowledge taking place during the information exchange between physical and analytical models, the geometries present in the PM, namely vertical columns with a constant cross section and walls and slabs with constant thickness, can often be found in building models. After the approach was verified with two additional building models, where new elements such as beams and different spatial relations between the building elements were introduced, interpretations methods defined with PM were not able to cover only several building elements. Hence, it is possible to predefine the majority of geometrical interoperations in such a way. The openness of the proposed methods distinguishes the framework from proprietary solutions

and allows for changes which might be required for various practices. Automating the interpretations might significantly simplify the data exchange process by avoiding the most common and repetitive tasks, leaving a small portion of building elements, geometries and interpretations to structural engineers to manually deal with.

One of the limitations of the current implementation is the lack of interventions on single building elements. The model interpretations are completely automated, which could pose a problem for other practices and building models. Structural engineers need more control over the interpretation process. The methods defined in this research are limited to the use case models. For wider practical implementation it is necessary to extend them with additional ones and to cover more geometries. The interpretations are based on the IFC building models that depend on the export performance of native software tools and their IFC interface, which are different from the building models in their native software. Therefore, this step could be replaced with the direct connection of the software tool to the central database.

Next steps will include the optimization of the system architecture together with the application of interpretation steps in order to get the direct open exchange model from Revit. The framework, after being enhanced with additional methods, will be tested on the models from design offices. In that way the scalability of the current approach will be tested. The usefulness and usability of the novel software can be tested only after a user-friendly interface is provided, which is also one of the future steps, where the engineers can tackle a single element and gain better control over the interpretations. Structural analysis feedback is not part of the investigated data exchange, since additional structural analysis information is required. Reaching automated structural feedback after the data exchange is investigated in [46] and will be extended with the future investigations. Additionally, a bidirectional exchange will be tested, including data transfer from the analytical to physical model.

Author Contributions: Conceptualization, G.S.; methodology, G.S.; software, G.S. and V.P.; validation, G.S., I.K. and W.S.; formal analysis, G.S.; investigation, G.S.; resources, W.S.; data curation, G.S. and V.P.; writing—original draft preparation, G.S.; writing—review and editing, G.S., I.K. and W.S.; visualization, G.S. and V.P.; supervision, I.K. and W.S.; project administration, I.K.; funding acquisition, I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was conducted as part of the Doctoral College "Computational Design" on TU Wien (TUW) and as part of the project DATAFILTER which was funded by Strabag SE.

Data Availability Statement: The data presented (developed software tools) in this study are available on request from the corresponding author. The data are not publicly available due to privacy issues. Building models were obtained from Züblin AG and are available from the authors with the permission of Züblin AG.

Acknowledgments: The authors wish to express their appreciation to Theodor Sansakrit Strohal for his support and contribution to the research on the side of Strabag SE, and Dario Bubalo and the department "Technische Dienste—Konstruktion und Technologie" from Züblin, subsidiary of Strabag SE, for providing the building data models. Open Access Funding by TU Wien.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Agarwal, R.; Chandrasekaran, S.; Sridhar, M. Imagining Construction's Digital Future (Blog, June). McKinsey&Company. Available online: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/imagining-constructions-digital-future (accessed on 14 October 2021).
2. Grilo, A.; Jardim-Goncalves, R. Value proposition on interoperability of BIM and collaborative working environments. *Automat. Constr.* **2010**, *19*, 522–530. [CrossRef]
3. Boddy, S.; Rezgui, Y.; Cooper, G.; Wetherill, M. Computer integrated construction: A review and proposals for future direction. *Adv. Eng. Softw.* **2007**, *38*, 677–687. [CrossRef]

4. Lin, J.-R.; Zhang, J.-P.; Zhang, X.-Y.; Hu, Z.-Z. Automating closed-loop structural safety management for bridge construction through multisource data integration. *Adv. Eng. Softw.* **2019**, *128*, 152–168. [CrossRef]
5. Ren, G.; Ding, R.; Li, H. Building an ontological knowledgebase for bridge maintenance. *Adv. Eng. Softw.* **2019**, *130*, 24–40. [CrossRef]
6. Sibenik, G.; Kovacic, I. Assessment of Model-Based Data Exchange between Architectural Design and Structural Analysis. *J. Build. Eng.* **2020**, *32*, 101589. [CrossRef]
7. Sanguinetti, P.; Abdelmohsen, S.; Lee, J.; Lee, J.-K.; Sheward, H.; Eastman, C. General system architecture for BIM: An integrated approach for design and analysis. *Adv. Eng. Inform.* **2012**, *26*, 317–333. [CrossRef]
8. Scherer, R.J.; Schapke, S.-E. A distributed multi-model-based Management Information System for simulation and decision-making on construction projects. *Adv. Eng. Inform.* **2011**, *25*, 582–599. [CrossRef]
9. Sibenik, G.; Kovacic, I. Interpreted Open Data Exchange between Architectural Design and Structural Analysis Models. Special Issue: 'CIB World Building Congress 2019: Constructing Smart Cities'. *ITcon* **2021**, *26*, 39–57. [CrossRef]
10. Holzer, D.; Tengono, Y.; Downing, S. Developing a Framework for Linking Design Intelligence from Multiple Professions in the AEC Industry. In *Computer-Aided Architectural Design Futures (CAADFutures) 2007*; Dong, A., Moere, A.V., Gero, J.S., Eds.; Springer: Dordrecht, The Netherlands, 2007; pp. 303–316. [CrossRef]
11. Fagin, R.; Kolaitis, P.G.; Popa, L. Data exchange: Getting to the core. *Acm. Trans. Database Syst.* **2005**, *30*, 174–210. [CrossRef]
12. Ren, R.; Zhang, J.; Dib, H.N. BIM interoperability for structural analysis. In Proceedings of the ASCE Construction Research Congress: Construction Information Technology, New Orleans, LA, USA, 2–4 April 2018; Wang, C., Harper, C., Lee, Y., Harris, R., Berryman, C., Eds.; ASCE: Reston, VA, USA, 2018; pp. 470–479. [CrossRef]
13. Kepplin, R.; Schnellenbach-Held, M.; Held, M. Building Information Modeling—Umsetzung in der Tragwerksplanung. *Bautechnik* **2017**, *94*, 220–226. [CrossRef]
14. Luyten, L. CAAD and Conceptual Design Collaboration between Architects and Structural Engineers. In Proceedings of the 33rd eCAADe Conference—Volume 2, Vienna, Austria, 16–18 September 2015; Martens, B., Wurzer, G., Grasl, T., Lorenz, W.E., Schaffranek, R., Eds.; pp. 215–224. Available online: http://papers.cumincad.org/cgi-bin/works/Show?ecaade2015_206 (accessed on 14 October 2021).
15. Kovacic, I.; Vasilescu, D.; Filzmoser, M.; Suppin, R.; Oberwinter, L. BIM in teaching-lessons learned from exploratory study. *Organ. Technol. Manag. Constr.* **2015**, *7*, 1358–1366. [CrossRef]
16. Golzarpoor, B.; Haas, C.T.; Rayside, D.; Kang, S.; Weston, M. Improving construction industry process interoperability with Industry Foundation Processes (IFP). *Adv. Eng. Inform.* **2018**, *38*, 555–568. [CrossRef]
17. Solihin, W.; Eastman, C.; Lee, Y.-C. A framework for fully integrated building information models in a federated environment. *Adv. Eng. Inform.* **2016**, *30*, 168–189. [CrossRef]
18. Eastman, C.; Teicholz, P.; Sacks, R.; Lee, G. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 3rd ed.; John Wiley and Sons Inc.: Hoboken, NJ, USA, 2018.
19. Qin, L.; Deng, X.-Y.; Liu, X.-L. Industry foundation classes based integration of architectural design and structural analysis. *J. Shanghai Jiaotong Univ. (Sci.)* **2011**, *16*, 83–90. [CrossRef]
20. Ramaji, I.J.; Memari, A.M. Interpretation of structural analytical models from the coordination view in building information models. *Automat. Constr.* **2018**, *90*, 117–133. [CrossRef]
21. Hu, Z.Z.; Zhang, X.Y.; Wang, H.W.; Kassem, M. Improving interoperability between architectural and structural design models: An industry foundation classes-based approach with web-based tools. *Automat. Constr.* **2016**, *66*, 29–42. [CrossRef]
22. Liu, Z.-Q.; Li, Y.-G.; Zhang, H.-I. IFC-based integration tool for supporting information exchange from architectural model to structural model. *J. Cent. South Univ. Technol.* **2010**, *17*, 1344–1350. [CrossRef]
23. Chen, P.-H.; Cui, L.; Wan, C.; Yang, Q.; Ting, S.K.; Tiong, R.L.K. Implementation of IFC-based web server for collaborative building design between architects and structural engineers. *Automat. Constr.* **2005**, *14*, 115–128. [CrossRef]
24. Deng, X.Y.; Chang, T.-Y.P. Creating Structural Model from IFC-based Architectural Model. In Proceedings of the Joint International Conference on Computing and Decision Making in Civil and Building Engineering (ICCCBE), Montreal, QC, Canada, 14–16 June 2006; Rivard, H., Miresco, E., Cheung, M.M.S., Eds.; pp. 3687–3695. Available online: http://itc.scix.net/data/works/att/w78-2006-tf577.pdf (accessed on 14 October 2021).
25. Lee, J.; Jeong, Y.; Oh, M.; Hong, S.W. A Filter-Mediated Communication Model for Design Collaboration in Building Construction. *Sci. World J.* **2014**, *2014*, 808613. [CrossRef]
26. Ramaji, I.J.; Memari, A.M. Interpreted information exchange: Implementation point of view. *J. Inf. Technol. Constr.* **2020**, *25*, 123–139. [CrossRef]
27. Ren, R.; Zhang, J. A New Framework to Address BIM Interoperability in the AEC Domain from Technical and Process Dimensions. *Adv. Civ. Eng.* **2021**, *2021*. [CrossRef]
28. Romberg, R.; Niggl, A.; van Treeck, C.; Rank, E. Structural Analysis based on the Product Model Standard IFC. In Proceedings of the 10th International Conference on Computing in Civil and Building Engineering (ICCCBE), Weimar, Germany, 2–4 June 2004; Beucke, K., Firmenich, B., Donath, D., Fruchter, R., Roddis, K., Eds. Available online: https://e-pub.uni-weimar.de/opus4/frontdoor/deliver/index/docId/243/file/icccbe-x_028_pdfa.pdf (accessed on 14 October 2021).

29. Zhang, X.-Y.; Hu, Z.-Z.; Wang, H.-W.; Kassem, M. An Industry Foundation Classes (IFC) Web-Based Approach and Platform for Bi-Directional Conversion of Structural Analysis Models. In Proceedings of the International Conference on Computing in Civil and Building Engineering, Orlando, FL, USA, 23–25 June 2014; Issa, R., Flood, I., Eds.; ASCE: Reston, VA, USA; pp. 390–397. [CrossRef]
30. McGraw Hill Construction. *The Business Value of BIM for Construction in Major Global Markets: How Contractors Around the World Are Driving Innovation with Building Information Modeling, Smart Market Report*; McGraw Hill Construction: Bedford, MA, USA, 2014. Available online: https://www.icn-solutions.nl/pdf/bim_construction.pdf (accessed on 14 October 2021).
31. Chassiakos, A.P.; Sakellaropoulos, S.P. A web-based system for managing construction information. *Adv. Eng. Softw.* **2008**, *39*, 865–876. [CrossRef]
32. ISO. *ISO 19650-2:2018 Organization and Digitization of Information about Buildings And Civil Engineering Works, Including Building Information Modelling (BIM)—Information Management Using Building Information Modelling—Part 2: Delivery Phase of the Assets*; ISO: Geneva, Switzerland, 2018.
33. Jeong, S.; Hou, R.; Lynch, J.P.; Sohn, H.; Law, K.H. An information modeling framework for bridge monitoring. *Adv. Eng. Softw.* **2017**, *114*, 11–31. [CrossRef]
34. Shelden, D.; Pauwels, P.; Pishdad-Bozorgi, P.; Tang, S. Data standards and data exchange for Construction 4.0. In *Construction 4.0: An Innovation Platform for the Built Environment*; Sawhney, A., Riley, M., Irizarry, J., Eds.; Routledge: London, UK, 2020; pp. 222–239.
35. Rasys, E.; Hodds, M.; Dawood, N.; Kassem, M. A Web3d Enabled Information Integration Framework for Facility Management. *AJCEB Conf. Ser.* **2014**, *2*, 1–12. [CrossRef]
36. Afsari, K.; Eastman, C.M.; Castro-Lacouture, D. JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. *Automat. Constr.* **2017**, *77*, 24–51. [CrossRef]
37. Werbrouck, J.; Taelman, R.; Verborgh, R.; Pauwels, P.; Beetz, J.; Mannens, E. Pattern-based access control in a decentralised collaboration environment. In Proceedings of the 8th Linked Data in Architecture and Construction (LDAC2020) Workshop, Dublin (virtual), Ireland, 17–19 June 2020. Available online: http://ceur-ws.org/Vol-2636/09paper.pdf (accessed on 14 October 2021).
38. Wagner, A.; Bonduel, M.; Pauwels, P.; Rüppel, U. Representing construction-related geometry in a semantic web context: A review of approaches. *Automat. Constr.* **2020**, *115*, 103130. [CrossRef]
39. Mouton, C.; Parfouru, S.; Jeulin, C.; Dutertre, C.; Goblet, J.-L.; Paviot, T.; Lamouri, S.; Limper, M.; Stein, C.; Behr, J.; et al. Enhancing the plant layout design process using X3DOM and a scalable web3D service architecture. In Proceedings of the 19th International ACM Conference on 3D Web Technologies (Web3D '14), Vancouver, BC, Canada, 8–10 August 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 125–132. [CrossRef]
40. Mora, R.; Rivard, H.; Bédard, C. Computer Representation to Support Conceptual Structural Design within a Building Architectural Context. *J. Comput. Civ. Eng.* **2006**, *20*, 76–87. [CrossRef]
41. Xu, Z.; Rao, Z.; Gan, V.J.L.; Ding, Y.; Wan, C.; Liu, X. Developing an Extended IFC Data Schema and Mesh Generation Framework for Finite Element Modeling. *Adv. Civ. Eng.* **2019**, *2019*, 1434093. [CrossRef]
42. Sibenik, G.; Kovacic, I. Proposal for a discipline-specific open exchange framework. In Proceedings of the 35th ISARC, Berlin, Germany, 20–25 July 2018; pp. 570–577. [CrossRef]
43. SAF. Available online: https://www.saf.guide/ (accessed on 14 October 2021).
44. SCOPE. SCOPE—Semantic Construction Project Engineering. Available online: https://www.projekt-scope.de/en/home-en/ (accessed on 4 October 2021).
45. Sibenik, G.; Petrinas, V. IFCtoJSON (GitHub Repository). Available online: https://github.com/ibauTUW/IFCtoJSON (accessed on 14 October 2021).
46. Sibenik, G.; Kovacic, I.; Petrinas, V. Automated model preprocessing for structural analysis. In Proceedings of the 38th ISARC, Dubai, United Arab Emirates, 2–4 November 2021; pp. 153–160. [CrossRef]