



Project Acronym: **BIMERR**
 Project Full Title: **BIM-based holistic tools for Energy-driven Renovation of existing Residences**
 Grant Agreement: **820621**
 Project Duration: **42 months**

DELIVERABLE D5.1 Prototype of Enhanced BIM Platform 1

Deliverable Status: **Final**
 File Name: **D5.1_V1.docx**
 Due Date: **30/11/2020 (M23)**
 Submission Date: **30/11/2020 (M23)**
 Task Leader: **UCL (T5.1)**

| Dissemination level | |
|--|---|
| Public | X |
| Confidential, only for members of the Consortium (including the Commission Services) | |



This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621

The BIMERR project consortium is composed of:

| | | |
|-----------|--|----------------|
| FIT | Fraunhofer Gesellschaft Zur Foerderung Der Angewandten Forschung E.V. | Germany |
| CERTH | Ethniko Kentro Erevnas Kai Technologikis Anaptyxis | Greece |
| UPM | Universidad Politecnica De Madrid | Spain |
| UBITECH | Ubitech Limited | Cyprus |
| SUITE5 | Suite5 Data Intelligence Solutions Limited | Cyprus |
| HYPERTECH | Hypertech (Chaipertek) Anonymos Viomichaniki Emporiki Etaireia Pliroforikis Kai Neon Technologion | Greece |
| MERIT | Merit Consulting House Sprl | Belgium |
| XYLEM | Xylem Science And Technology Management Gmbh | Austria |
| CONKAT | Anonymos Etaireia Kataskevon Technikon Ergon, Emporikon Viomichanikonkai Nautiliakon Epicheiriseon Kon'kat | Greece |
| BOC | Boc Asset Management Gmbh | Austria |
| BX | Budimex Sa | Poland |
| UOP | University Of Peloponnese | Greece |
| UEDIN | University of Edinburgh | United Kingdom |
| UCL | University College London | United Kingdom |
| NT | Novitech As | Slovakia |
| FER | Ferrovial Agroman S.A | Spain |

Disclaimer

BIMERR project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission (EC). EC is not liable for any use that may be made of the information contained therein.

AUTHORS LIST

| Leading Author (Editor) | | | | |
|----------------------------------|--------------|------------|-------------|--------------------------|
| | Surname | First Name | Beneficiary | Contact email |
| | Katsigarakis | Kyriakos | UCL | k.katsigarakis@ucl.ac.uk |
| Co-authors (in alphabetic order) | | | | |
| # | Surname | First Name | Beneficiary | Contact email |
| 1 | Gounaridou | Apostolia | CERTH | agounaridou@iti.gr |
| 2 | Lilis | Giorgos | UCL | g.lilis@ucl.ac.uk |
| 3 | Rovas | Dimitrios | UCL | d.rovas@ucl.ac.uk |

REVIEWERS LIST

| List of Reviewers (in alphabetic order) | | | | |
|---|------------|------------|-------------|--------------------|
| # | Surname | First Name | Beneficiary | Contact email |
| 1 | Lampathaki | Fenareti | SUITE5 | fenareti@suite5.eu |
| 2 | Bosché | Frédéric | UEDIN | f.bosche@ed.ac.uk |

REVISION CONTROL

| Version | Author | Date | Status |
|---------|---------------|------------|----------------------|
| 0.1 | UCL | 15.10.2020 | Table of Contents |
| 0.2 | UCL | 15.11.2020 | Section 1, Section 2 |
| 0.4 | UCL | 18.11.2020 | Section 3, Section 6 |
| 0.6 | CERTH, UCL | 20.11.2020 | Section 4, |
| 0.8 | CERTH, UCL | 25.11.2020 | Section 5 |
| 0.9 | UEDIN, SUITE5 | 28.11.2020 | Final draft reviewed |
| 1.0 | UCL | 30.11.2020 | Submission to the EC |

TABLE OF CONTENTS

| | |
|--|-----------|
| <i>List of Figures.....</i> | 7 |
| <i>List of Tables.....</i> | 10 |
| EXECUTIVE SUMMARY | 12 |
| 1. Introduction..... | 13 |
| 1.1 Scope and Objectives of the Deliverable..... | 13 |
| 1.2 Relation to Other Tasks and Deliverables | 15 |
| 1.3 Structure of the Document | 16 |
| 2. Platform Architecture..... | 17 |
| 2.1 High-Level Architecture | 17 |
| 2.2 Modules Orchestration | 19 |
| 2.3 Core Module | 20 |
| 2.3.1 Web-based User Interface | 20 |
| 2.3.2 Data Model..... | 25 |
| 2.4 Supporting Software Components..... | 26 |
| 2.4.1 EXPRESS Schema Compiler..... | 26 |
| 2.4.2 IFC Java Library..... | 26 |
| 2.4.3 IFC Geometry Exporter..... | 27 |
| 2.4.4 IFC Optimizer..... | 27 |
| 2.4.5 IFC Unit Converter | 28 |
| 2.5 Revision Control | 28 |
| 2.5.1 Object-based Tracking Changes | 29 |
| 2.6 API Documentation | 29 |
| 2.7 Assumptions & Restrictions | 31 |

| | | |
|-----------|--|-----------|
| 2.8 | Installation Instructions | 31 |
| 2.9 | Licensing | 31 |
| 3. | Functional Components Specification | 32 |
| 3.1 | Data Quality Checking..... | 32 |
| 3.1.1 | IFC Schema Compliance | 32 |
| 3.1.2 | MVD Checking | 32 |
| 3.1.3 | Geometric Error Detection | 35 |
| 3.2 | Semantic Enrichment..... | 39 |
| 3.2.1 | Automatic Space Generation (ASG) | 40 |
| 3.2.2 | Common Boundary Intersection Projection (CBIP) | 42 |
| 3.3 | Geometry Engine | 43 |
| 3.3.1 | B-rep Generation | 43 |
| 3.3.2 | Model Viewer..... | 45 |
| 4. | Integration of BIM-MP in BIMERR workflows | 47 |
| 4.1 | IFC Creation..... | 48 |
| 4.1.1 | Scan-to-BIM..... | 48 |
| 4.1.2 | REVIT | 50 |
| 4.2 | IFC Validation | 54 |
| 4.3 | IFC Queries | 55 |
| 4.4 | IFC Transactions | 55 |
| 5. | Application Examples..... | 56 |
| 5.1 | Detailed Model..... | 57 |
| 5.2 | Simplified Model | 58 |
| 6. | Conclusions and Planning..... | 61 |

7. Bibliography 62

LIST OF FIGURES

| | |
|---|----|
| Figure 1 BIM Management Platform overall architecture..... | 17 |
| Figure 2 BIM-MP modules orchestration..... | 19 |
| Figure 3 Core Module's GUI and REST API back-end architecture..... | 20 |
| Figure 4 Core Module's components stack diagram..... | 21 |
| Figure 5 BIM-MP's active projects dashboard..... | 23 |
| Figure 6 BIM-MP's functional modules control panel..... | 24 |
| Figure 7 BIM-MP's internal file repository..... | 24 |
| Figure 8 IFC Classes generation using the EXPRESS Schema Compiler..... | 26 |
| Figure 9 IFC Java Library Components..... | 26 |
| Figure 10 IFC geometry exportation process..... | 27 |
| Figure 11 IFC file size optimization process..... | 28 |
| Figure 12 IFC unit conversion process..... | 28 |
| Figure 13 IFC object-based revision model representation..... | 29 |
| Figure 14 Definition of a Concept Template in IfcDoc..... | 34 |
| Figure 15 Definition of checking rules along with its parameters in IfcDoc..... | 34 |
| Figure 16 Validation of a custom Model View in IfcDoc..... | 35 |
| Figure 17 MVD Checker Module process..... | 35 |
| Figure 18 GED process diagram..... | 36 |
| Figure 19: Examples of Surface Errors in IFC geometric data..... | 37 |
| Figure 20: Examples of clash errors in IFC geometric data..... | 38 |
| Figure 21: Example of a space error..... | 38 |
| Figure 22 ASG enrichment process..... | 41 |

| | |
|--|----|
| Figure 23: Illustration of BIM-MP's ASG service..... | 41 |
| Figure 24 CBIP enrichment process..... | 42 |
| Figure 25: Illustration of BIM-MP' s CBIP service..... | 43 |
| Figure 26: B-rep generation process..... | 44 |
| Figure 27: Illustration of a correctly oriented boundary representation | 44 |
| Figure 28: IfcProducDefinitionShape and supported IFC subclasses..... | 45 |
| Figure 29: Process diagram of BIM-MP's model viewer | 45 |
| Figure 30 IFC rendering using BRG, glTF converter and xeogl engine | 46 |
| Figure 31 High-level IFC data flow in BIMERR..... | 47 |
| Figure 32 Geometry of structural components generated by Scan-to-BIM | 49 |
| Figure 33 Clash error detected by BIM-MP's GED module | 49 |
| Figure 34 Integration with Revit | 50 |
| Figure 35 Default Template and Project Units selection in Revit..... | 51 |
| Figure 36 Activation of "Areas and Volumes Computation" and Spaces visibility in Revit | 51 |
| Figure 37 Space placing and Space limits definition in Revit | 52 |
| Figure 38 Zone creation and Space assignment in Revit | 53 |
| Figure 39 Site definition and Project Point visibility activation in Revit | 53 |
| Figure 40 IFC entities export mapping in Revit..... | 54 |
| Figure 41 Iterative correction process of a BIM model | 57 |
| Figure 42 Detailed version of KRIPIS model which includes spaces and plenums..... | 57 |
| Figure 43 Detailed version of KRIPIS IFC model | 58 |
| Figure 44 Detailed version of KRIPIS IFC which includes 2 nd level space boundaries | 58 |
| Figure 45 Simplified version of KRIPIS model which complies with the scanning limitations.. | 59 |

| | |
|--|----|
| Figure 46 Simplified version of KRIPIS IFC model..... | 59 |
| Figure 47 Simplified version of KRIPIS IFC which includes the 2 nd level space boundaries | 60 |

LIST OF TABLES

ACRONYMS

| Acronym | Meaning |
|---------|--|
| AMQP | Advanced Message Queuing Protocol |
| API | Application Programming Interface |
| ARIBFA | Augmented Reality Enabled In-Situ Building Feature Annotation |
| ASG | Automatic Space Generation |
| ASHRAE | American Society of Heating, Refrigerating and Air-Conditioning Engineers |
| BEPS | Building Energy Performance Simulation |
| BIF | BIMERR Interoperability Framework |
| BIM-MP | BIM Management Platform |
| BIMERR | BIM-based holistic tools for Energy-driven Renovation of existing Residences |
| BRG | B-Rep Generation |
| CBIP | Common Boundary Intersection Projection |
| COP | Coefficient of performance |
| CRUD | Create, Read, Update, Delete |
| DAO | Data Access Object |
| DI | Dependency Injection |
| EMF | Eclipse Modeling Framework |
| ESB | Enterprise Service Bus |
| ETL | Extract, Transform, Load |
| GED | Geometric Error Detection |
| GUI | Graphical User Interface |
| GUID | Global Unique Identifier |
| IFC | Industry Foundation Classes |
| IoC | Inversion of Control |
| JMS | Java Messaging System |
| JNI | Java Native Interface |
| JPA | Java Persistence Layer |
| JSON | JavaScript Object Notation |
| LCA | Life Cycle Analysis |
| MEP | Mechanical, Electrical and Plumbing |
| MVC | Model-View-Control |
| MVD | Model View Definition |
| ORM | Object Relational Mapping |

| | |
|------|---------------------------------|
| REST | Representational State Transfer |
| SOA | Software-oriented Architecture |

EXECUTIVE SUMMARY

This document, Deliverable D5.1, describes the first version of a “Prototype of Enhanced BIM Platform 1” and is the output of the development activities linked to T5.1 “BIM Platform adaptation for Efficient Renovation support”.

The BIM Management Platform (BIM-MP) is part of the Digital Building Model Creation Tools of BIMERR, responsible for handling BIM data that conform to the IFC standard. BIM-MP provides an integrated data management solution for persistence, versioning, updating and checking of IFC data, which are created and modified by other BIMERR tools. The BIM-MP exposes an API through which, data exchange between BIM-MP and other tools through the BIMERR Interoperability Framework (BIF) is made possible. When the BIF forwards the IFC file into the BIM-MP repository, the file is post processed to ensure that the data meet end-user requirements; the platform also supports reporting and error handling functionalities. The BIM-MP does not only naïvely store the file, but rather works proactively to ensure data quality. These components are implemented in a modular and extensible sense: some of the modules create visual and textual reports regarding data quality issues in terms of data consistency, completeness and correctness, while others semantically enrich the IFC. All these modules are deployed as standalone containerized applications using the Service-Oriented Architecture (SOA) design principle.

This deliverable describes the first version of the BIM Management Platform, which introduces a cloud-BIM approach, implementing the SOA design principle. Analytic description of the functional modules, as well as examples of dynamic data workflows involving different software components and libraries of these modules, which are taking place when operational requests to the BIM-MP occur, are presented as a core part of this deliverable.

The deliverable investigates the differences, in semantic terms, between outputs of BIMERR tools (e.g. Scan-to-BIM) and commercial BIM authoring tools (Revit, ArchiCAD, etc.); this investigation concludes with the description of dynamic workflows. This deliverable presents the first version of the platform; a plan for further development of BIM-MP modules is presented.

1. INTRODUCTION

1.1 SCOPE AND OBJECTIVES OF THE DELIVERABLE

The main goal of this deliverable is to present the overall architecture of the first prototype of BIMERR's BIM Management Platform (BIM-MP), highlighting its functional modules, as well as to demonstrate the platforms' internal operational data flows which are executed within different use cases within BIMERR. Following the above rationale, it will become clear how BIM-MP responds to different requests originated from other BIMERR tools and supports their data processing needs in a well-defined and unified manner.

BIM-MP adopts openBIM standards to provide an integrated data management solution for storing, versioning, updating and checking BIM models in the form of IFC files. It offers native reference implementations in Java of the most used open-BIM standards such as the Industry Foundation Classes (IFC), Model-View Definitions (MVD), and the BIM Collaboration Format (BCF). A core area of concern in this deliverable is the bridging of the interoperability and data exchange gap between professional BIM authoring tools, which often resort to proprietary formats and siloed workflows within application ecosystems. In this deliverable, we focus on the possibility of facilitating high quality data exchanges between some of these tools, and BIM-MP which is based on openBIM standards.

While BIM Server (BIMserver, 2019) is an open-source cloud-BIM solution with a large community and support, it is mainly used for IFC data handling, versioning, and merging. The meta-model of BIM Server is based on the Eclipse Modelling Framework (EMF) which requires a wide range of dependencies. Additionally, the BIM Server uses external libraries to transform geometry representations for viewing purposes only. On the other hand, BIM-MP needs to provide custom-build libraries for handling IFC data and performing advanced geometric operations for detecting clash errors, generating B-rep representations of structural building elements and more, which requires a different solution than what is offered by the BIM Server.

BIM-MP contains a set of supporting software components to facilitate complex geometric operations such as the B-rep generation and the semantic enrichment of IFC files. For this reason, BIM-MP's internal API is based on the JNI programming interface to enable the connectivity with low-level geometric algorithms written in C/C++. Some functional algorithms

such as CBIP, ASG, BRG and GED are compiled as shared object libraries and packaged as functional modules of BIM-MP. This deliverable presents a harmonized state-of-the-art cloud-BIM platform architecture that aims to implement the generic abstractions and interfaces between the geometric algorithms and the open-BIM standards.

The deliverable D5.1 "Prototype of Enhanced BIM Platform 1" aims at reporting on the work that has been reached up to M23 on the BIM Management Platform, being developed in the context of T5.1 "BIM Platform adaptation for Efficient Renovation support", providing a general overview and documentation of the BIM Management Platform's first implementation. An updated version of the deliverable containing the documentation of the second version of BIM-MP will be released as D5.2 in M30.

1.2 RELATION TO OTHER TASKS AND DELIVERABLES

T5.1 “BIM Platform Adaptation for Efficient Renovation Support” and therefore D5.1 “Prototype of Enhanced BIM Management Platform 1” are related to BIMERR deliverables organized in the following table:

| Del. Number | Deliverable Title | Relations and Contributions |
|--------------------|--|--|
| D3.6 | BIMERR System Architecture 2nd Version | The second version of the BIMERR architecture provided an overview on the BIMERR components, how they communicate to each other and which components need access to the BIMERR BIM-MP. |
| D4.4 | BIMERR Building Semantic Modelling tool 1 | Semantic modelling of BIM data depends on IFC data provided by BIM-MP through BIF. |
| D4.6 | BIMERR Information Collection & Enrichment Tool 1 | BIM-MP exchanges IFC data with other BIMERR tools through BIF |
| D4.8 | Integrated BIMERR Interoperability Framework 1 | BIM-MP responds to IFC queries coming from BIF. |
| D5.3 | Innovative Scan-to-BIM tools for Automated BIM v1 | BIM-MP receives and checks the initial IFC file originated from BIMERR’s Scan-to-BIM tool. |
| D5.5 | Building Information Collection Application for building residents 1 | BIM-MP provides building topology data to BICA tool. |
| D5.7 | Building resident energy-related behaviour profiling framework 1 | BIM-MP organizes building spatial zoning data required for the occupancy profile data obtained from PRUBS. |
| D5.9 | Refinement of Augmented-Reality Smart Glasses for the ARIBFA app | BIM-MP checks using specific MVD the data added to the IFC files by ARIBFA app. |
| D7.1 | Populated Material/Component Database 1 | BIM-MP’s IFC data quality MVD checking services is required for all building components. |
| D7.3 | Building Energy Modelling Module 1 | BEPE module needs BIM-MP’s BEPS data quality checking and enrichment services for the BIM of the baseline and the renovation scenarios. |

1.3 STRUCTURE OF THE DOCUMENT

Section 1 describes the scope and objectives of the deliverable and its relations to other tasks and deliverables.

Section 2 "Platform Architecture" illustrates the high-level architecture of BIM-MP and presents an overview of its core and functional modules. The prototype implementation of the Core Module of BIM-MP is presented, as well as the description of the supporting software components used for handling and storing of IFC data. The section concludes with the documentation of the API it exposes, the assumptions considered during the first release and the licensing.

Section 3: "Functional Components Specification" describes the operation of each functional Module of BIM-MP. Depending on their role, the functional modules are grouped into three categories and described in three respective subsections: Data Quality Checking, Semantic Enrichment and Geometry Engine.

In Section 4: "Dynamic Workflows", the BIM-MP's internal component executions and data exchanges are analyzed depending on the different calls from other BIMERR tools and their various IFC data processing requests. It describes the BIM data flows within BIMERR from IFC creation to final version exportation. The final IFC includes the simulated data of the selected renovation scenario, used by PWMA to optimize the renovation processes.

Section 5 presents application examples of BIM-MP's operations for the KRIPIS pre-validation building. The designer following the provided BIM guidelines and using the online version of the platform, produced an error-free model in terms of geometry. During the design phase, the designer created two design alternatives of the BIM model in Revit: a) an accurate model, and b) a simplified model due to physical limitations of the scanning process. Both models have been checked, validated and enriched using BIM-MP's functional modules.

Section 6 provides the conclusions along with the plan for the 2nd iteration of the BIMERR BIM Management Platform.

2. PLATFORM ARCHITECTURE

BIM-MP is responsible for providing functionalities in relation to the storing, checking, updating, and querying of BIM models. These models conform to the openBIM International Foundation Classes (IFC) standard (ISO 16739:2018) which contains a rich set of classes designed to provide a robust interoperability solution for data exchange between different built environments software applications. This section describes the back-end system architecture of a cloud-BIM management solution that enables the interoperability between functional modules and external clients.

2.1 HIGH-LEVEL ARCHITECTURE

BIM-MP includes a core and a set of reusable modules to support synchronous and asynchronous requests in a unified manner. Some modules can respond to requests immediately while others require more time depending on the complexity of the BIM models. All of them include a set of low- and high-level libraries to perform their business logic operations. Figure 1 shows a generalized view of the proposed architecture of the BIM-MP with its core and reusable modules.

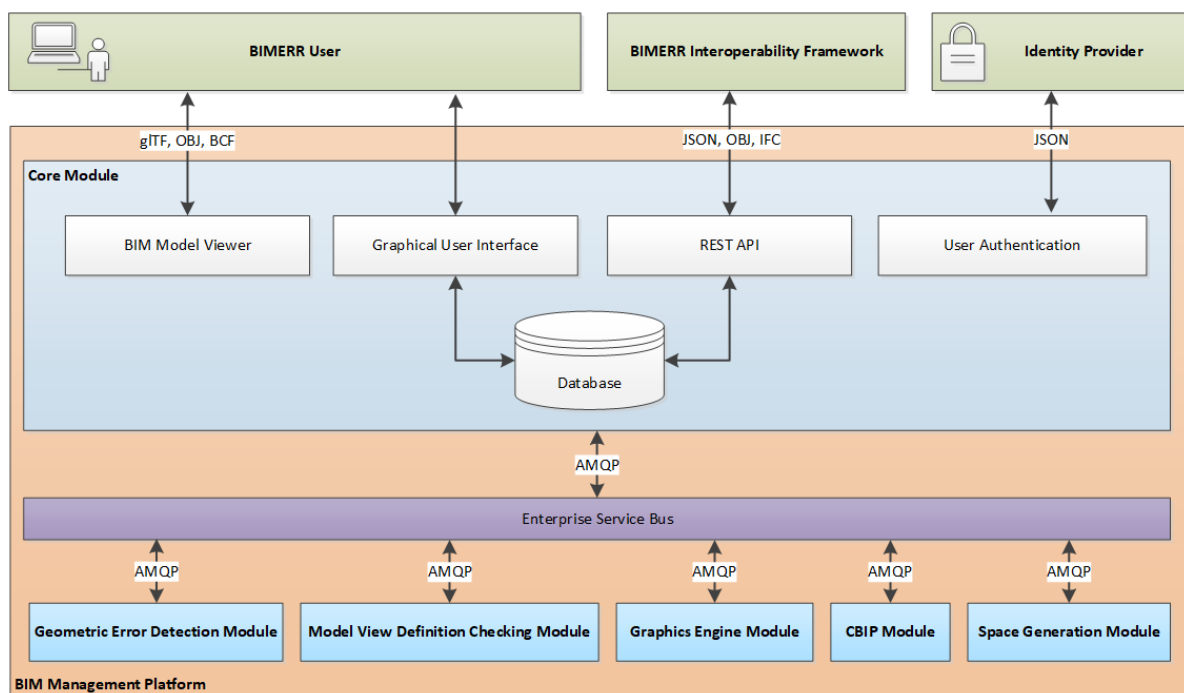


Figure 1 BIM Management Platform overall architecture

The high-level libraries are responsible for handling and querying BIM models and for extracting the required information for the execution of the low-level operations. The low-level libraries support multithread processing and exchange information with the high-level libraries through the JNI programming framework. The modules are deployed in the cloud infrastructure as a standalone Spring Boot Application following the Software-Oriented Architecture (SOA) design principle.

For the communication of the different modules, BIM-MP uses the Red Hat Fuse Enterprise Service Bus (ESB) which is in charge of providing the asynchronous communication using queues as well as for handling the routing of the messages. The ESB acts as a message broker providing the required queues and facilitates interoperability. The ESB includes Extract-Transform-Load (ETL) logic and custom mappers to transform the payload of the messages in a proper form according to the requirements of the receivers. In the proposed architecture, BIM-MP modules are standalone applications based on the Spring Boot Framework. Each of them contains high-level and low-level libraries to perform its business logic operations. We defined the following modules as a basis for the prototype implementation of the platform:

- 1) **Core Module** contains the GUI implementation and REST API of BIM-MP. It uses high-level libraries to parse BIM models and extract the necessary information used by the functional modules.
- 2) **Geometric Error Detection (GED) Module** integrates geometric model-checking functionalities into the platform helps BIM designers to create an error-free model in terms of geometry in the scope of building energy performance simulation analysis. The GED module reports detected errors to the designer in a visual form using OBJ and BCF data.
- 3) **Model View Definition Checking Module** helps BIM designers to validate BIM models in terms of data completeness based on predefined rules following the mvdXML specification.
- 4) **Graphics Engine Module** uses the geometric information of BIM models to generate B-Rep solids of the structural and non-structural building elements. It stores the B-Rep solids in BIF using OBJ files. Additionally, the embedded WebGL viewer of the BIM-MP uses the B-Rep solids for a visual representation of the model through the GUI.
- 5) **CBIP Module** uses geometric operations to enrich the BIM model with 2nd-level space boundaries and shading surfaces.

6) **Automatic Space Generation Module** uses the geometric information to generate the geometry of the spaces and enriches the BIM model.

Each of these modules exchange information using data structures which conform to open standards. The Core Module itself controls the binding between the BIM-MP front-end system and the functional modules.

2.2 MODULES ORCHESTRATION

As mentioned earlier, BIM-MP implementation follows the SOA design approach. The distributed nature of the platform requires orchestration techniques and a messaging framework that provides a loose coupling between components, to achieve performance and reliability. The use of asynchronous messaging offers many benefits, but also brings challenges such as the delivery sequence of messages and the concurrency between the different BIM-MP modules.

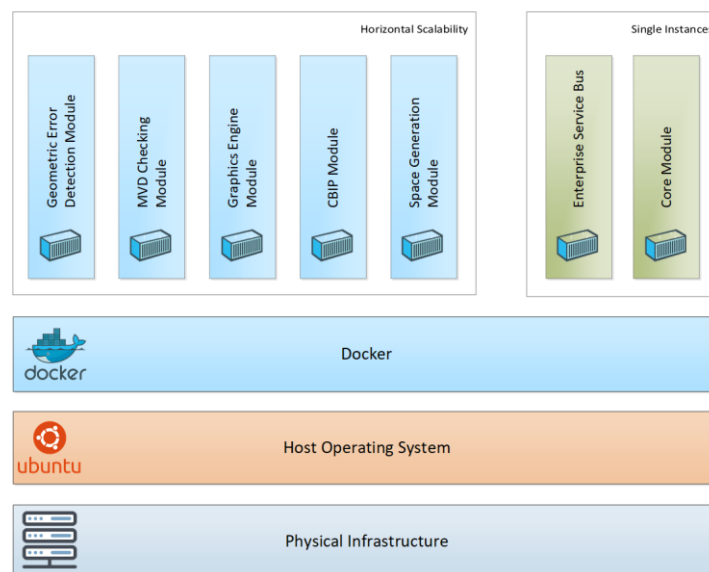


Figure 2 BIM-MP modules orchestration

A secure and private containerized environment hosted on dedicated server infrastructure is used to provide the BIM-MP functionality. This setup utilizes a lightweight virtualization system that does not require hypervisors on hardware. The container images facilitate the portability and distribution of workloads in a standardized manner, and allow developers to package all software components and dependencies into reusable units.

2.3 CORE MODULE

The Core Module is deployed as a cloud application and provides the web-based interface and the API of BIM-MP. It uses a set of software components and libraries to handle the BIM models and to extract the information required by the functional modules.

2.3.1 Web-based User Interface

We developed the Core Module in Java EE, adopting the microservice design pattern. The Core Service includes the first version of the GUI and provides authorized access to BIMERR users and tools through the BIMERR Identity Provider. The Core Module is based on the Spring Boot Framework and uses its embedded webserver to support the Model-View-Control (MVC) design pattern.

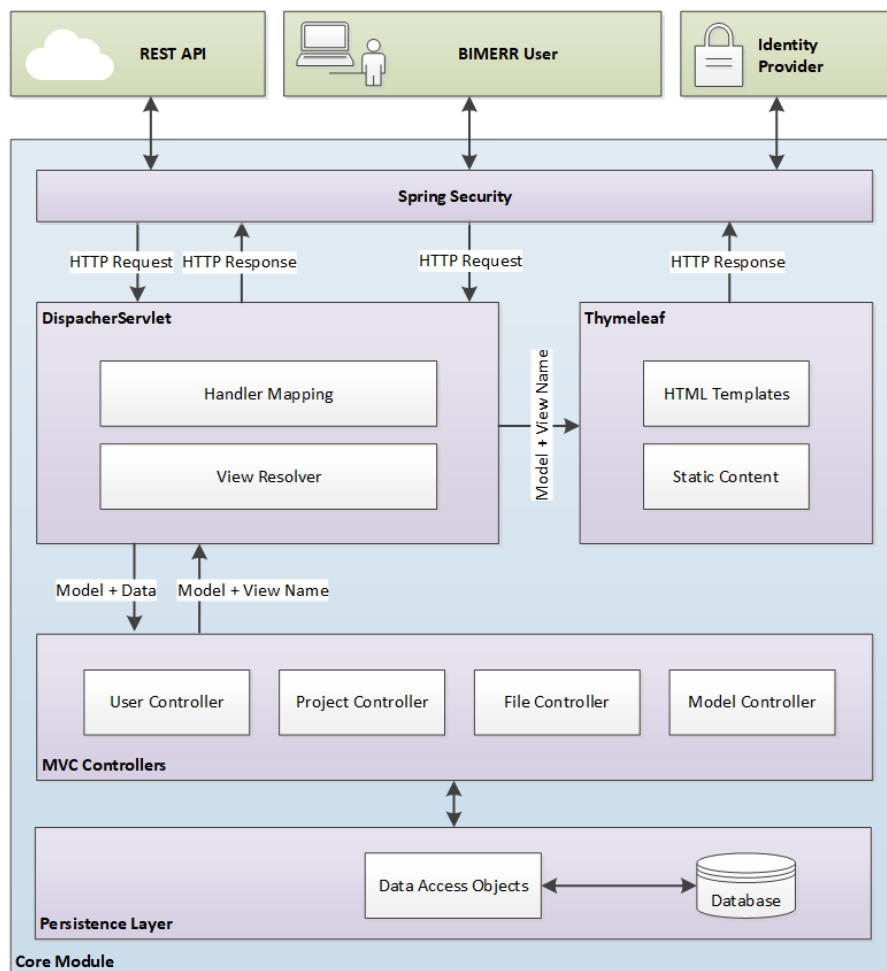


Figure 3 Core Module's GUI and REST API back-end architecture

Figure 3 shows the main components of the GUI and REST API back-end architecture. The Spring MVC Framework comprises three parts:

- 1) The Dispatcher Servlet which forwards incoming client requests to specific controllers based on the URL pattern configuration of the Mapping Handler.
- 2) The Controller which collects through an API or DI the data from the BIM-MP services and stores them in the Model object; and
- 3) The View Resolver which uses the Model object to render the corresponding template and forwards the response to the Dispatcher Servlet and then back to the client.

Moreover, the Core Module uses the Spring Security Framework and the Spring Keycloak Adapter to handle access policies for the BIM-MP. The Identity Provider to grant access to BIM-MP for accessing information such as user data, user roles and groups. Figure 3 illustrates the interfaces between the MVC components of the Core Module and the BIMERR framework. Apart from the requirement to support modern web technologies, the Core Module includes a set of components to configure a database for storing its data. Moreover, it includes AMQP adapters to initialize asynchronous communication interfaces with the functional modules through the Enterprise Service Bus. Figure 4 illustrates the Core Module components in a stack diagram.

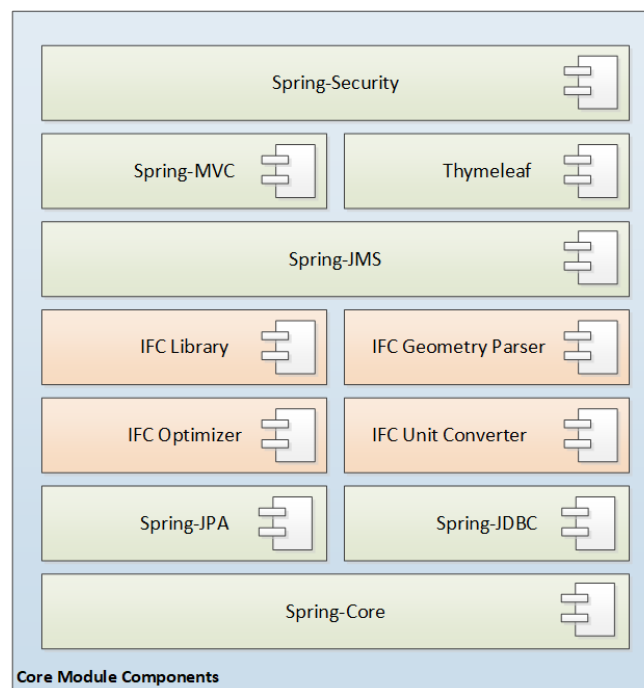


Figure 4 Core Module's components stack diagram

Spring Core is the foundation component of the Spring Framework. SC provides support for developing applications using technologies such as Dependency Injection (DI) and Inversion of Control (IoC).

Spring JDBC is the component of Spring Framework responsible for connecting to a database and SQL commands executions. Spring JDBC provides low-level functionalities such as opening the database connection, preparing SQL statements for querying, handling exceptions and finally closing the database connection.

Spring JPA (Java Persistence API) is a component of Spring Framework that automates the creation and manipulation of the JPA-based repositories. It provides the Create, Read, Update and Delete (CRUD) functionalities on the defined entities of the Core Module by simplifying the manual creation of Data Access Objects (DAO) for Object-Relational Mapping (ORM).

Spring Security is a component of Spring Framework that provides authentication, authorization, and protection against common attacks. The Keycloak Spring Adapter uses the Spring Security in the background, to validate the access token of the user against the Identity Provider. It also checks if the user has the necessary permissions to perform a request.

Spring MVC provides a framework for creating web applications using the MVC design pattern. It includes objects to support the main concepts of modern web applications and is extendable to support external frameworks. The server-side page rendering of the Core Module is performed using the Spring Thymeleaf Template Engine.

Thymeleaf is a template engine for modern web applications, used for implementing the end-user interfaces, which constitutes the View part of the MVC design pattern.

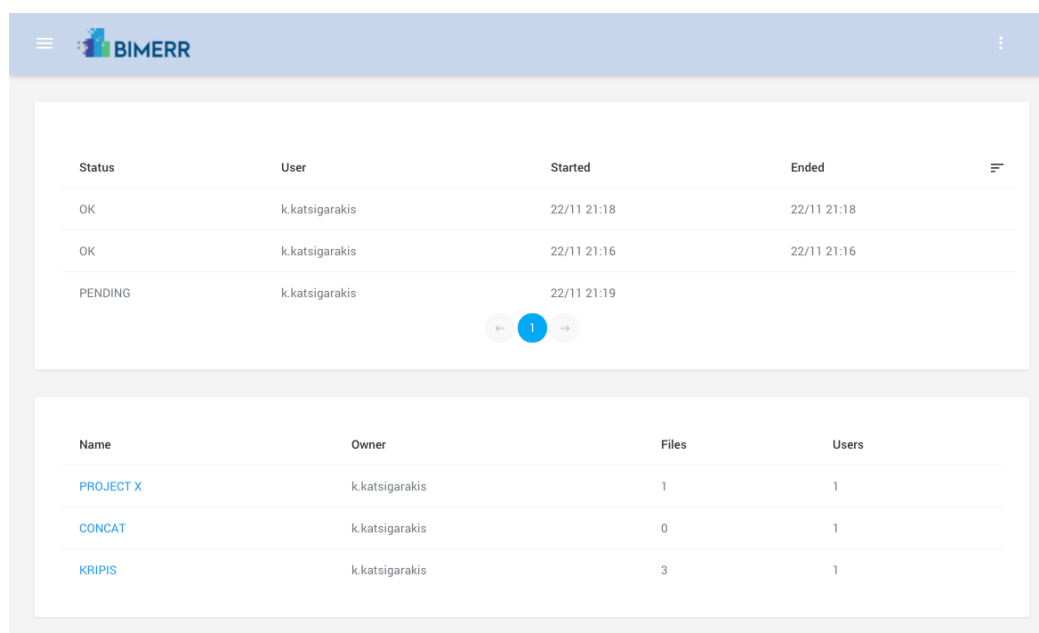
Spring JMS (Java Message Service) provides a framework for interconnecting the Core Module with messaging systems and brokers to exchange messages asynchronously using the Advanced Message Queueing Protocol.

Table 1 indicates the technologies, frameworks and software packages used for the development of the BIM-MP's Core Module.

Table 1 Technologies and libraries used in the BIM-MP, along with their licenses

| Technology / Software | Version | License |
|-----------------------|---------|--------------------|
| Spring Framework | 5.3.1 | Apache License 2.0 |
| Spring Boot | 2.2.1 | Apache Licence 2.0 |
| Spring Security | 5.5.0 | Apache License 2.0 |
| Thymeleaf | 3.0.11 | Apache License 2.0 |
| Red Hat Fuse | 7.5.0 | Apache License 2.0 |
| Docker | 19.03 | Apache License 2.0 |
| MySQL | 8.0.24 | GPLV2 |
| Hibernate | 5.4.8 | LGPL 2.1 |
| xeogl | 0.9 | MIT License |
| Clipper | 6.1.3 | BSL V1.0 |
| RapidXML | 1.13 | BSL V1.0 |

Within BIMERR, the Keycloak Identity Provider is responsible for handling the authentication of registered users and providing access to their renovation projects. BIM-MP uses the Keycloak REST API to retrieve the assigned roles and to query the active renovation projects by the user UUID. The BIM-MP User Interface reacts on synchronous events, e.g. progress bars and push notifications. When a user authenticates successfully, BIM-MP's dashboard is listed. The dashboard contains information about active processes and renovation projects assigned to the specific user (see Figure 5).



The screenshot shows the BIMERR dashboard interface. At the top, there is a header with the BIMERR logo and a menu icon. Below the header, there are two main sections. The first section is a table with columns: Status, User, Started, Ended, and an icon column. It contains three rows of data, all for user 'k.katsigarakis'. The first two rows are 'OK' with timestamps '22/11 21:18' and '22/11 21:16'. The third row is 'PENDING' with timestamp '22/11 21:19'. Below this table is a pagination control showing '1' in a blue circle. The second section is another table with columns: Name, Owner, Files, and Users. It contains three rows of data for projects: 'PROJECT X', 'CONCAT', and 'KRIPIS', all owned by 'k.katsigarakis'.

| Status | User | Started | Ended | |
|---------|----------------|-------------|-------------|--|
| OK | k.katsigarakis | 22/11 21:18 | 22/11 21:18 | |
| OK | k.katsigarakis | 22/11 21:16 | 22/11 21:16 | |
| PENDING | k.katsigarakis | 22/11 21:19 | | |

| Name | Owner | Files | Users |
|-----------|----------------|-------|-------|
| PROJECT X | k.katsigarakis | 1 | 1 |
| CONCAT | k.katsigarakis | 0 | 1 |
| KRIPIS | k.katsigarakis | 3 | 1 |

Figure 5 BIM-MP's active projects dashboard

The main page of each renovation project provides access for the execution of BIM-MP's functional modules. Each Module reports the progress of its execution in real-time using a synchronized progress bar, as shown in Figure 6.

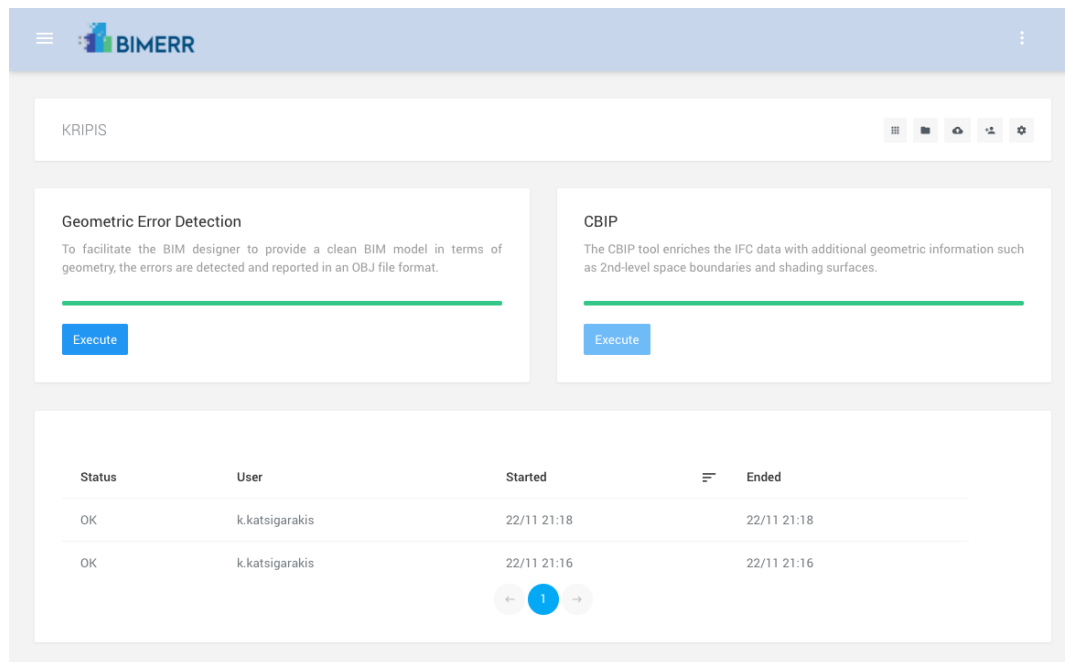


Figure 6 BIM-MP's functional modules control panel

The BIM-MP provides a separate file repository for each renovation project. The BIM-MP's functional modules use this repository to store the outputs and reports of their executions. The user can either download or delete the generated files.

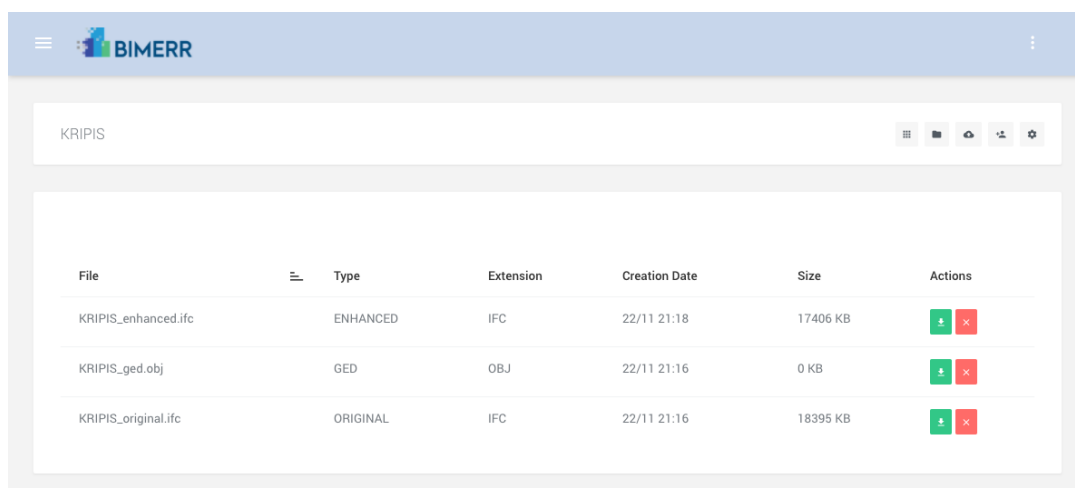


Figure 7 BIM-MP's internal file repository

2.3.2 Data Model

The Core Module of BIM-MP includes entities, attributes, and relationships between entities, to represent the logical data model which stores the data needed to perform the BIM-MP operations. Using the Spring Framework¹ and Hibernate², the automatic deployment of the database is driven based on Java Persistence API (JPA) annotations. In this context, Java classes represent the tables, while some fields inside the classes represent the relations between different tables. The framework supports all types of relations such as one-to-one, many-to-one, one-to-many and many-to-many. When using this approach, the relational database can be transparently managed from Java, increasing the abstraction level of the persistence layer.

The Core Module requires a connection to a MySQL Server. MySQL Server is a Relational Database Management System (RDBMS) that support multi-tenancy. The Hibernate framework utilizes the MySQL dialect to access the BIM-MP's database to perform transactional operations and queries.

¹ A comprehensive programming and configuration model for enterprise applications <https://spring.io>

² Domain model persistence for relational databases <https://hibernate.org>

2.4 SUPPORTING SOFTWARE COMPONENTS

BIM-MP uses software components and libraries to handle the complexity of the functional modules and to increase the automation and reliability of the business logic operations. It contains five primary software components described below:

2.4.1 EXPRESS Schema Compiler

The EXPRESS Schema Compiler is a standalone application that uses the Java EE Code Model framework to generate the IFC Java classes directly from the EXPRESS data. It can parse successfully the most frequently used IFC releases, from IFC2X3 to IFC4X1. First, the application transforms the EXPRESS data into in-memory objects using an internal data model representation. Then applies a set of predefined transformation rules on the objects to instantiate the representation of the Code Model. In the end, the Code Model generates the IFC classes and organize them into Java packages based on the release of the IFC schema.

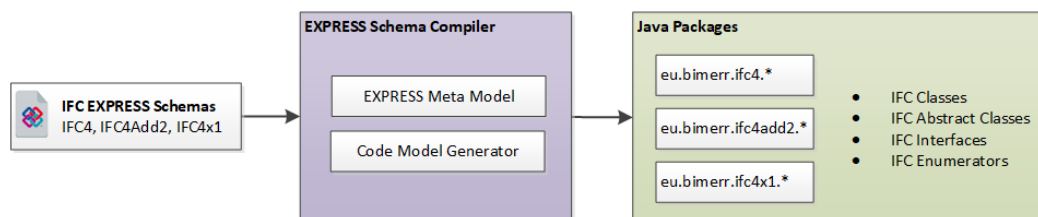


Figure 8 IFC Classes generation using the EXPRESS Schema Compiler

2.4.2 IFC Java Library

The IFC Java Library uses the generated IFC Java classes to parse efficiently the STEP data and to instantiate the representation of the BIM model. The current version of the IFC library can handle the most frequently used IFC releases, from IFC2X3 to IFC4X1.

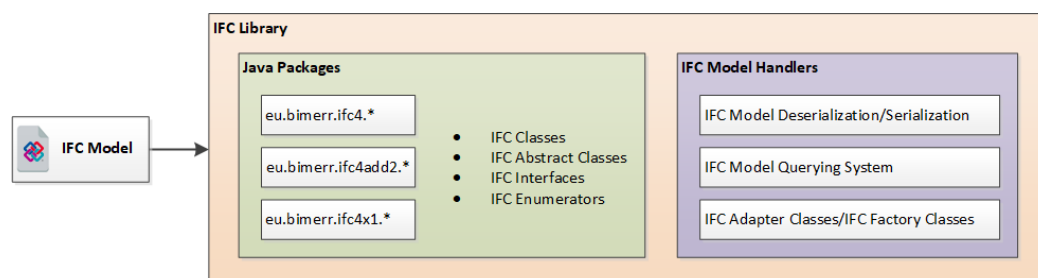


Figure 9 IFC Java Library Components

It provides an API that includes a set of useful functionalities for manipulating the loaded objects. Moreover, it supports the initialization of the reverse relations by adding the instance of an object to the corresponding collection of the inverse connected instance of another object.

2.4.3 IFC Geometry Exporter

The IFC Geometry Exporter extracts the geometric content of a BIM model and its semantics in XML format using the IFC Java Library, to provide the necessary geometric data to the low-level algorithms through the JNI programming interface.

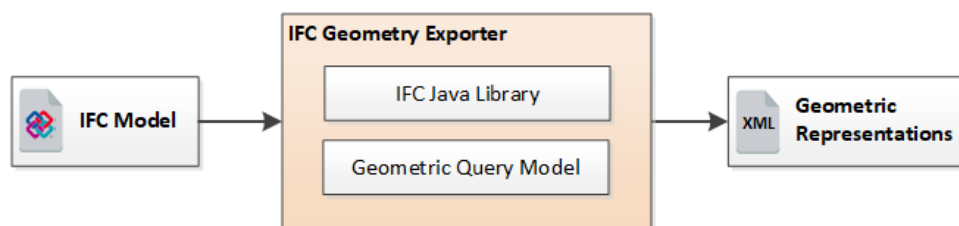


Figure 10 IFC geometry exportation process

The efficient processing of IFC data consists of separate operations performed by individual libraries written in different programming languages. The serialization of IFC is easier using high-level languages such as Java or C# while the complex geometric operations often require low-level programming languages such as C/C++. For this reason, the IFC Geometry Exporter produces an XML to facilitate the data handling on the low-level geometric operations.

2.4.4 IFC Optimizer

The IFC Optimizer performs lossless compression of large IFC file sizes, to speed up the loading process and the execution of some ETL tools such as the IFC Geometry Exporter. The IFC Exporters of BIM authoring tools often generate multiple instances of the same object. The IFC Optimizer uses the IFC Java Library to compare the hash values of the IFC objects and merges them when they are equal. Furthermore, it updates the express identifiers of the deleted objects to maintain the original connections.

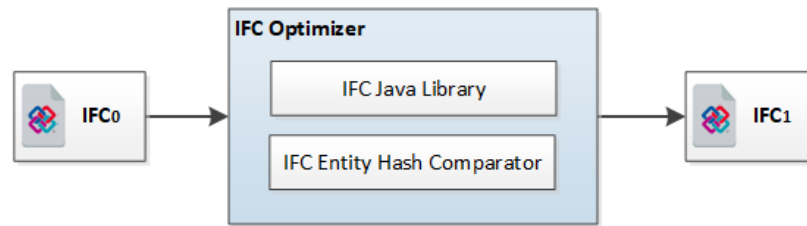


Figure 11 IFC file size optimization process

2.4.5 IFC Unit Converter

In general, the tools and algorithms require the input BIM model to conform to the defined MVD that ensures the semantics and completeness of the used IFC data. The units of values in an IFC file depend on the IFC Exporters of the BIM authoring tools. The IFC Unit Converter uses the methods of the IFC Java Library to convert the values of basic and derived units into a new unit system.

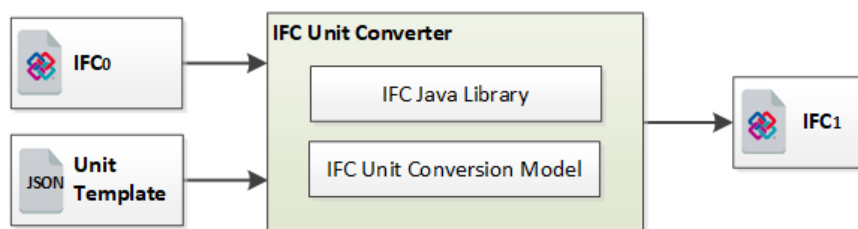


Figure 12 IFC unit conversion process

2.5 REVISION CONTROL

BIM-MP can handle the geometry and the semantic information of a BIM model that is provided in IFC. In the IFC schema specification, the IFC objects may reflect a final state, but they also may reflect a transient state. For instance, during a BIMERR renovation project, external BIM tools may continuously revise IFC objects. In the case where multiple tools are making updates to the same information, there is a concept in IFC schema to support local copies of the modified IFC objects. This revision scheme identifies changes declared on a per-object basis instead of identifying changes in text. An IFC object is considered modified when: a) any of its direct attributes change; b) any of its referenced resources change; and c) items are added or removed from any collection. Within this revision scheme, each IFC object is marked with a change action indicating if the IFC object was ADDED, MODIFIED, DELETED or not changed.

2.5.1 Object-based Tracking Changes

The BIMERR tools can use this revision scheme to track the type of changes that might have occurred to the IFC objects during the last active session. In this case, the BIM-MP will know how an IFC object might have been affected by other BIMERR tools.

The process is the following: First, the Scan-to-BIM tool creates an IFC that is open for modifications. Scan-to-BIM should set the ChangeAction attribute of the IfcOwnerHistory (see Figure 13) to NOCHANGE to establish a baseline so that BIM-MP can easily identify the upcoming modifications. Next, when a BIMERR tool updates the IFC, should set the ChangeAction of the new IFC objects to ADDED and similarly modified objects to MODIFIED and deleted objects to DELETED and the OwningApplication to the application characteristics. Moreover, the BIMERR tool is responsible for updating the LastModifiedDate attribute to the time of the modification. Thus, when the BIM-MP receives the modified IFC, it can determine which objects have been added and modified and either merge or reject these changes, as necessary.

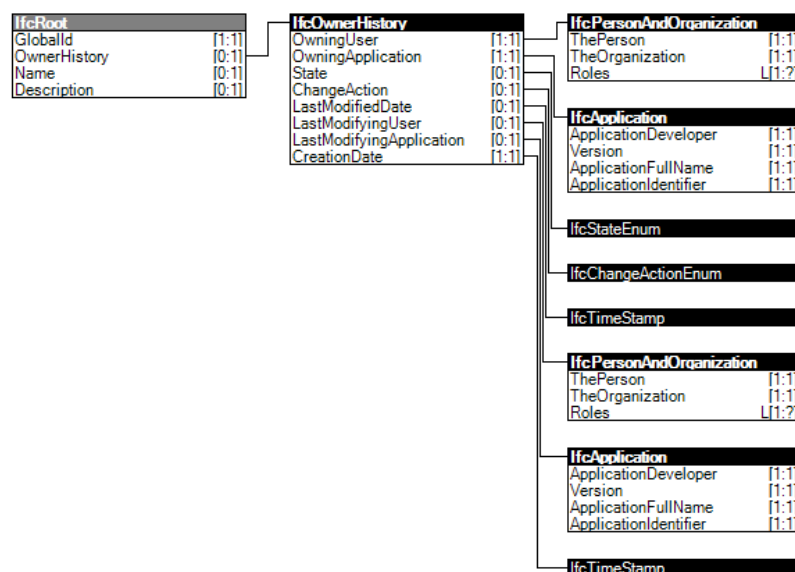


Figure 13 IFC object-based revision model representation

2.6 API DOCUMENTATION

The first version of BIM-MP exposes one generic REST API allowing the BIF to query the IFC data for populating the corresponding Building Data Model. The request body of the POST

query must be a valid JSON message. The JSON should include at least one parameter to determine the targeted version of the model.

Base Query

This query will return all instances of the IFC model that has been validated by the GED and MVD modules of BIM-MP.

```
{
  "version": "VALIDATED"
}
```

Query by Type

This query will return the instances of the requested entity of the IFC model that has been previously validated by the quality checking modules of BIM-MP.

```
{
  "version": "VALIDATED",
  "type": "IfcSlab"
}
```

The above query allows the developer to define multiple entities in a single request.

```
{
  "version": "VALIDATED",
  "type": ["IfcSlab", "IfcWall"]
}
```

Additionally, the BIM-MP supports returning the instances of the subtypes of the requested entity. For example, the instances of the entity IfcWallStandardCase are not returned from the above query. To include subtypes, the external application should add the corresponding parameter.

```
{
  "version": "VALIDATED",
  "type": ["IfcSlab", "IfcWall"],
  "subtypes": true
}
```

Alternative, if the GUID is known, the external application can perform queries by providing a list containing a single or multiple GUIDs.

```
{  
  "version": "VALIDATED",  
  "guid": ["272moRIQL9YgZOIcCFKXJZ", "2jbZxm12o6KSSyddJ6f7ky"]  
}
```

2.7 ASSUMPTIONS & RESTRICTIONS

The first version of BIM-MP has a number of assumptions and restrictions which are presented in the following:

- The current version of the BIM-MP's Core Module contains a 3D viewer with limited functionalities i.e. colour and texture support. The model navigation panel, as well as textures and colours of the building elements will be included before submitting the final version of the BIM-MP (M30).
- The current version of the BIM-MP's REST API supports queries of IFC objects by type or GUID. Additional REST API endpoints such as updating, editing of IFC objects will be included in the final version of the BIM-MP.
- The BIM-MP is fully compatible with IFC4, however some geometric transformations are not supported.
- BIM-MP supports only, non-curved IFC geometric representations. Some IFC curved geometric representations are approximated using segmentation processes.
- A first version of the mvdXML of BIMERR, has been initiated, which is not final. As BIMERR tools evolve, updates of this initial version will be performed, conforming to the tool's new requirements.

2.8 INSTALLATION INSTRUCTIONS

The BIM-MP is deployed in the cloud and is available as a web-based application, thus no installation is required.

2.9 LICENSING

The BIM-MP is a closed source component.

3. FUNCTIONAL COMPONENTS SPECIFICATION

In this section, the function of individual BIM-MP components is analyzed. BIM-MP components can be grouped into four categories: (1) Data quality checking, (2) Semantic Enrichment, (3) Geometry engine and (4) Revision control components. These categories and their respective BIM-MP functional components are described in the following subsections.

3.1 DATA QUALITY CHECKING

Since IFC data files are common data pools of multiple software components which alter their content either by adding or removing information, their data quality should be checked regularly. Such data quality checking services designed according to BIMERR needs are supported by BIM-MP. These data quality checking operations can be classified into three categories: schema compliance where the consistency of the IFC data is checked based on the IFC data schema, model view definition (MVD) checking where the completeness of the IFC data files are checked, and Geometric Error Detection operations where the geometric correctness checks are performed to IFC data files. These operations are analyzed in the following subsections.

3.1.1 IFC Schema Compliance

Within BIMERR, the exchange of BIM data among the different tools is based on the openBIM standards such as the ISO IFC4 ADD2 TC1 data exchange format. Within this standard, BIM data are coded in a STEP file, the structure of which is defined according to an EXPRESS schema conforming to the EXPRESS data modelling language. Taking this into account, the IFC Schema Compliance Checker of BIM-MP validates the STEP data of the input IFC files against the corresponding EXPRESS schema defined in the standard. This functionality is integrated into the IFC Library of BIM-MP and includes validation of datatypes, class names, the range of numerical variables and the sizes of the data collections.

3.1.2 MVD Checking

The IFC specification provides a multi-domain information model for capturing building data such as geometry, materials, components, properties and more. To support specific data exchange requirements between different tools and processes only a subset of the IFC

specification is required in terms of used entities and properties. The Model View Definition (MVD) specification (buildingSMART, 2020) allows to define reusable Concept Templates and Rules for describing precisely the data exchange requirements. Along with the maintenance of the IFC specification, buildingSMART has published two general-purpose Model View Definitions described below:

- **IFC Reference View** is mainly used by tools and processes that do not require modifications of the geometry. The geometric representation is optimized for analysis and display purposes but excludes parametric geometry definitions.
- **IFC Design Transfer View** includes instances with support for editing the geometric representations of building elements and spaces. It is the preferred MVD in BIMEER, because it enables the enrichment of the BIM model with new geometric information and property sets.

BIM-MP provides an MVD Checking Module to help BIMERR users to automatically validate the completeness of a BIM model against predefined rules using the mvdXML specification. Three steps are needed to achieve automatic MVD validation of a BIM model: a) The creation of the mvdXML file using the IfcDoc tool (IfcDoc, 2020); b) The application of the rules on the IFC instances using the algorithms of the IFC Library; and c) The generation of the validation report.

IfcDoc has been developed by buildingSMART International, to improve the computer-interpretable implementation of the IFC specification. The user can create custom MVDs and assign new concepts to them. Each concept contains a) an applicable IFC entity connected to a Concept Template; and b) the checking rules along with its parameters and logical operators.

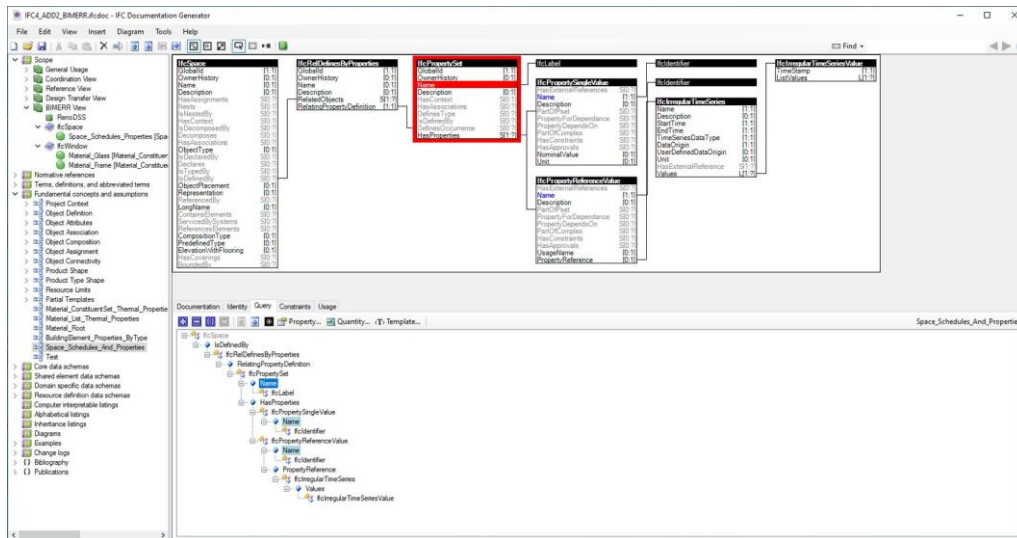


Figure 14 Definition of a Concept Template in IfcDoc

Each Concept Template consists a graph of entities and attributes, with constraints and parameters set for particular attributes and instance types. Figure 14 shows the interface of the IfcDoc tool for the definition of a new Concept Template.

The applied checking rules include parameters and logical-operators, e.g. instances that are checked by type or properties that are checked by value. Figure 15 shows the interface of the IfcDoc tool for the definition of the checking rules.

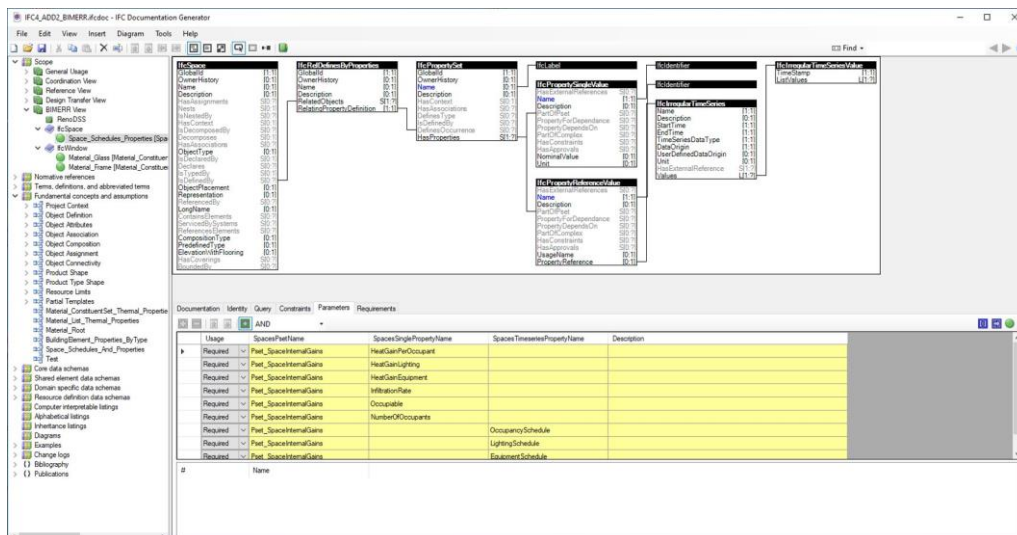


Figure 15 Definition of checking rules along with its parameters in IfcDoc

Before the exportation of the mvdXML, the user can validate the defined MVD using the embedded IFC validator in IfcDoc. The IFC validator requires an IFC file, the name of the Model Deliverable D5.1 ■ 11/2020 ■ UCL

View and the data exchange requirements. Figure 16 shows the IFC validator dialog box of the IfcDoc.

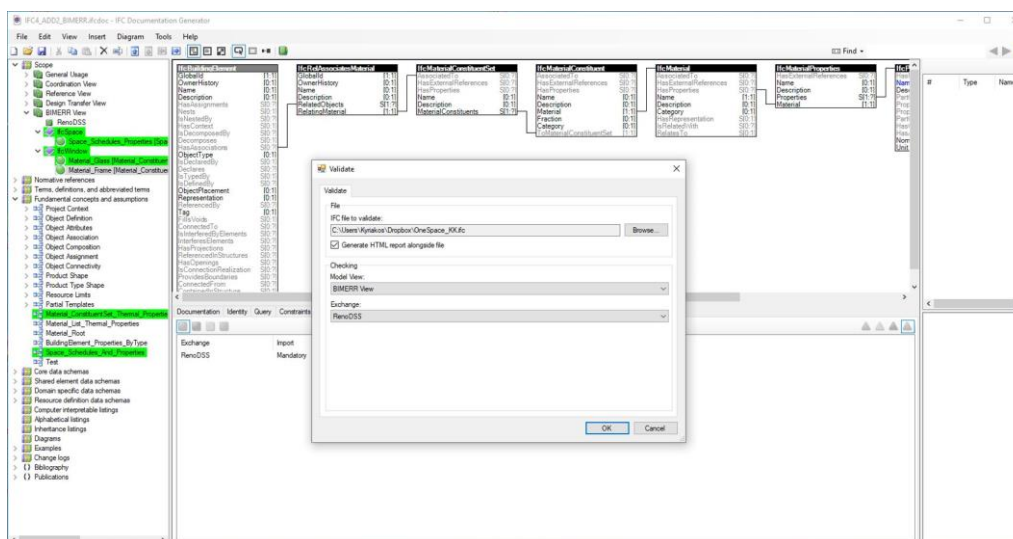


Figure 16 Validation of a custom Model View in IfcDoc

When the validation ends, the green markup ensures that the defined concepts are listed in the IFC. Following the same approach, the MVD Checking Module of BIM-MP takes as inputs the IFC file and the mvdXML file. Then the IFC Library serializes the STEP data and initializes the inverse relations of the IFC objects to perform the checking. The output of the Module is a detailed report of the validation process for each defined concept.

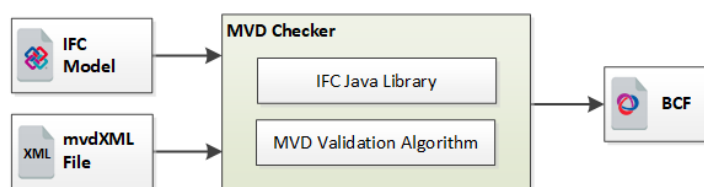


Figure 17 MVD Checker Module process

The MVD Checking Module examines whether the data exchange requirements of a BIMERR tool are satisfied. These data requirements include, among others, material thermal properties, MEP equipment property sets and operating schedules.

3.1.3 Geometric Error Detection

Geometric errors contained in IFC files affect significantly Building Energy Performance Simulations (BEPS) since they alter the geometric content of BEPS models, i.e. the second level

space boundary topology (Bazjanac, 2010), which is derived from the geometric content of BIM (IFC) files using well documented and tested algorithms (Lilis, Giannakis, & Rovas, 2017). Since BIM IFC models within BIMERR will be used for BEPS purposes, the geometric content of these models should be also checked for geometric errors which affect BEPS model generation process. BIM-MP contains all necessary software components to perform the appropriate geometric error detection process (GED process) (Lilis, Katsigarakis, & Rovas, 2018). According to the process diagram of BIM-MP's GED process displayed in Figure 18, the GED process receives the geometric content and semantics of the IFC model, extracted by the IFC Geometry Exporter and outputs an error report in OBJ format (Wavefront, 1992). This error report is forwarded to the WebGL viewer, for user display.

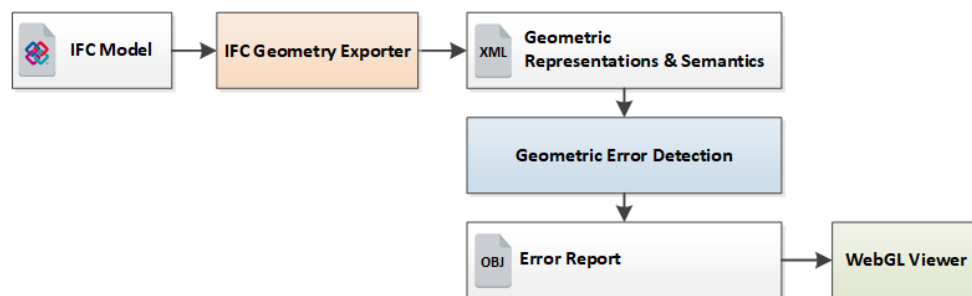


Figure 18 GED process diagram

The geometric errors appearing in IFC data and the respective checks performed by BIM-MP can be classified into three categories: (a) Surface errors, (b) Clash errors and (c) Space errors (Lilis, Giannakis, & Rovas, 2015). These geometric error types affect the generation of BEPS models, as they alter their geometric content (the buildings' second-level space boundary topology) and consequently affect the energy model generation of BEPE module of BIMERR. We describe each of the above geometric error categories in the following sections.

A. SURFACE ERRORS

Sometimes, the boundary geometric representations of architectural elements in IFC files have missing surfaces, or some of their surfaces are inverted: their normal vectors point inwards when the normal vectors of the other surfaces of the boundary representation point outwards, and vice versa. In these cases, the boundary representation has a surface error which affects further geometrical checking and should be detected and reported. Examples of surface errors are illustrated in Figure 19. For example in part II of Figure 19, three surfaces of a building slab are inverted. This inversion causes a misinterpretation of what part of the 3D space is inside or

outside the building slab. Another example of a surface error is illustrated in part III of Figure 19, where a building slab have a missing boundary surface, defined by a set of isolated line segments (line segments that are boundaries of only one boundary surface).

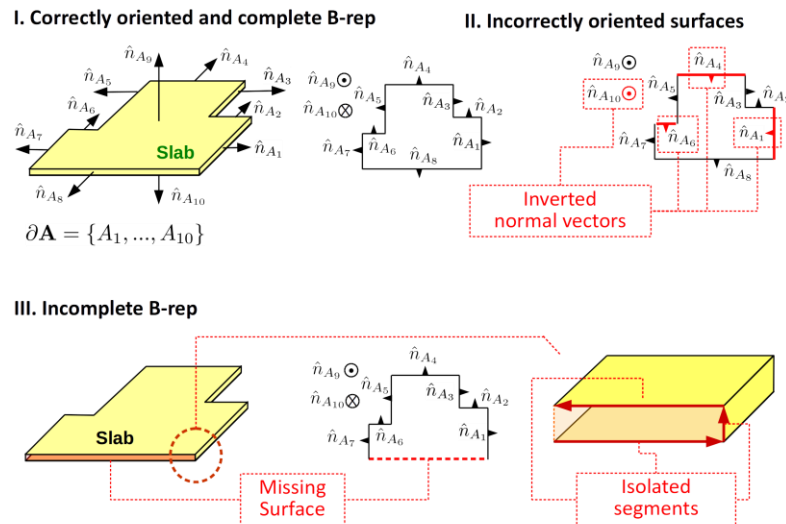


Figure 19: Examples of Surface Errors in IFC geometric data

B. CLASH ERRORS

Clash errors occur when the geometrical representations of two or more architectural elements intersect. Containment is a specific case of clash error, detected by BIM-MP, where an architectural component is contained inside another (see example (II) in Figure 20). Generally, a clash between two entities may yield intersecting space boundary surfaces affecting under some conditions the accuracy of the BEPS model generation of BIMERR's BEPE module. If the intersection surfaces of the clash are not attached to building spaces, then the clash error is not affecting BEPS (see example (I) in Figure 20). In case the intersection surfaces of the clash are attached to neighbouring building spaces then, as in example (II) in Figure 20, the clash error affects BEPS.

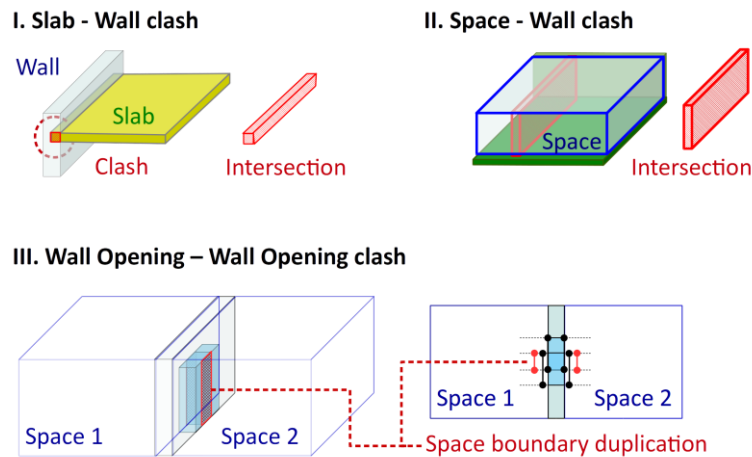


Figure 20: Examples of clash errors in IFC geometric data

C. SPACE ERRORS

Space errors occur when the internal building space volumes, if existing, are not completely bounded by surrounding building architectural elements (walls, slabs,). In these cases, a space error is detected, and the unbounded surfaces of the space volumes are reported. A space error and its respective correct space definition are illustrated in Figure 21.

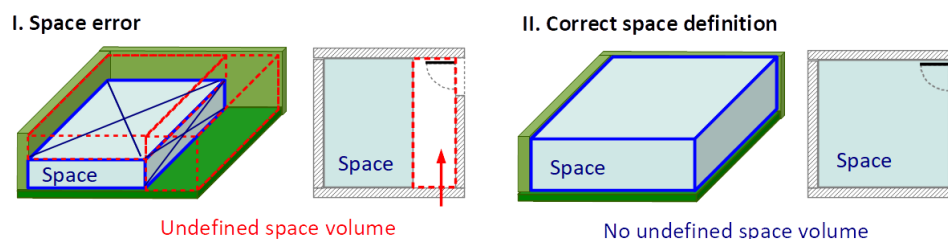


Figure 21: Example of a space error

GEOMETRIC ERROR DETECTION PROCESS

If at least the internal building space volume description is present in the IFC file, then a *BEPS-dedicated geometric detection process* is executed by BIM-MP, which includes the following three stages:

- Surface error detection
- BEPS clash error detection

- Space error detection

In the first stage (*surface error* detection), the surface integrity of the boundary representations of all architectural elements, including the building spaces, is checked. Missing surfaces or surfaces with inverted surface normal vectors (inverted surfaces) are reported. The geometric solids passing these checks pass to the next stage.

In the second stage, a *BEPS-specific clash error detection* is performed. Only clashes with intersecting surfaces attached to neighbour spaces are detected (neighbour space condition). This kind of clash errors affects the BEPS model generation. If the clash involves a building space entity, then it is reported in this stage without taking into account the neighbour space attachment condition. An example of a clash which satisfies this space boundary conditions and is reported in this stage is presented in the example II of Figure 20. On the contrary, the clash example I of Figure 20 violates the neighbor space attachment condition and therefore is not reported.

Finally, the BEPS geometric error detection process concludes in the third stage, where the space error detection process, described in the previous section, is applied to all building space volume geometric descriptions contained in the IFC file.

We understand that data quality is of paramount importance for the BIMERR automated processes to properly function. Since error-free data seems like a utopic objective, we try to address the quality improvement in BIMERR through a top-bottom perspective (guidelines for preparing the BIM models) with a bottom-up perspective (explicit checking at the BIM-MP level). It is possible that model-checkers (e.g. Solibri) can also be used to ensure modelling errors (geometric or otherwise) are detected. While the developments here focus primarily on using the data for BEPS, they also prove the concept that model checking can mostly be performed on the cloud. Checking for additional conditions will be included if the needs arise in BIMERR – the software architecture is permissive of this. The development and integration of a generic rule-engine is considered out of scope.

3.2 SEMANTIC ENRICHMENT

BIM-MP offers IFC data enrichment services, where certain data classes of input IFC files are populated with data which are obtained from existing IFC data classes. These enrichment

services are offered within the BIMERR framework to ensure that all IFC BIMs have the necessary information for the generation of BEPS models required for the assessment of building retrofitting projects within BIMERR. Two data enrichment services are offered by BIM-MP: Common Boundary Intersection Projection (CBIP) and Automatic Space Generation (ASG) services, both analyzed in the following sections.

3.2.1 Automatic Space Generation (ASG)

Frequently, the geometric representations of the internal building space volumes are missing or are defined incorrectly in the respective IFC BIM data classes. This happens because the design on an inner building space with BIM authoring tools is a tedious task involving filling all the space cavities (gaps), between the internal building space volume and its surrounding building architectural elements (walls, slabs, ...). These cavities can be too complicated, rendering the design of the inner building space impossible, even with the best BIM design software suites.

To overcome such difficulties, BIM-MP offers the Automatic Space Generation (ASG) service, which enriches an input IFC file with the geometric data of all building inner shells defined by the inner building space volumes.

According to Figure 22, ASG process receives as input the geometric content and its semantics exported in XML format by the IFC Geometry Exporter, of an initial IFC model IFC_0 . Then, the building's space geometric representations in XML format produced by ASG algorithm, together with the initial model IFC_0 are used as input to the BIM-MP enrichment service to produce the final enriched IFC model IFC_1 . ASG implements this IFC data enrichment by populating `IfcSpace` data classes which contain the boundary representations of the inner shells of the building spaces translated to the local coordinate systems of the respective spaces' level. ASG leaves no space gaps between the generated building space volumes and the surrounding building architectural elements.

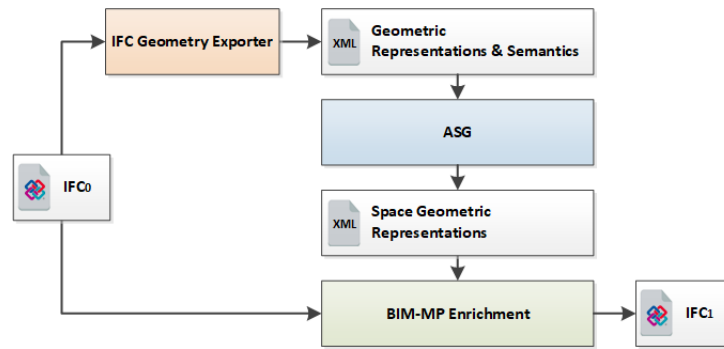


Figure 22 ASG enrichment process

Initially, the ASG algorithm extracts the joint boundary surfaces among all the possible pairs of boundary representations of architectural elements of the building of interest, illustrated with blue color in parts I and III of Figure 23. Then, these joint boundary surfaces are subtracted from the respective boundary representations of the elements they belong to, yielding the remaining set of surfaces illustrated with green colors in parts II and IV of Figure 23.

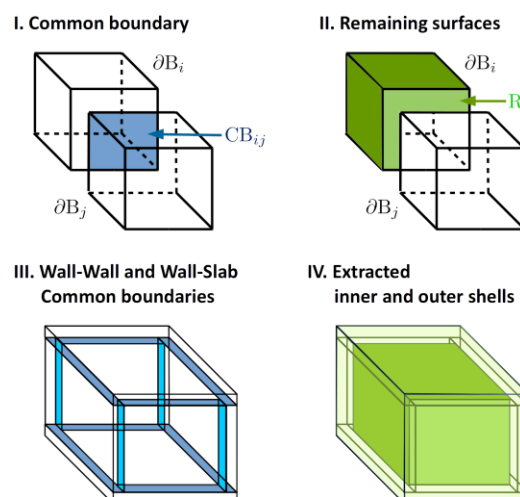


Figure 23: Illustration of BIM-MP's ASG service

The remaining surfaces of all architectural elements of a building form inner and outer surface shells. The obtained inner shells define the desired inner building space volumes. The separation between inner and outer shells is performed using the normal vectors of the surfaces of the shell, provided that all building entities have no surface errors (passed Geometric Error Detection stage 1). If all normal vectors of the surfaces of the shell point inward, the shell is characterized as inner and represents a space volume, otherwise the shell is outer and is a part of the building façade.

3.2.2 Common Boundary Intersection Projection (CBIP)

When the BIM files are generated using BIM authoring tools (Revit, ArchiCAD), necessary geometric content for the BEPS model generation, – the building’s second-level space boundary topology (Bazjanac, 2010) and the external shading surfaces – may be either missing or is incorrectly exported due to flaws in the IFC exporter of the BIM authoring tool. In this case, the generation of the second-level space boundary surface topology of the building, from the architectural geometric content contained in the IFC file and the population of the respective IFC data classes (IFCRelSpace-Boundary2nd-Level), are performed by BIM-MP’s CBIP tool (Lilis, Giannakis, & Rovas, 2017).

According to Figure 24, the CBIP process receives as input the geometric content and its semantics exported in XML format by the IFC Geometry Exporter, of an initial IFC model IFC₀. Then, the output of CBIP process: the building’s boundary surface topology (second-level space boundaries and external shading surfaces) in XML format, together with the initial model IFC₀ are used as input to BIM-MP enrichment service to produce the final enriched IFC model IFC₁.

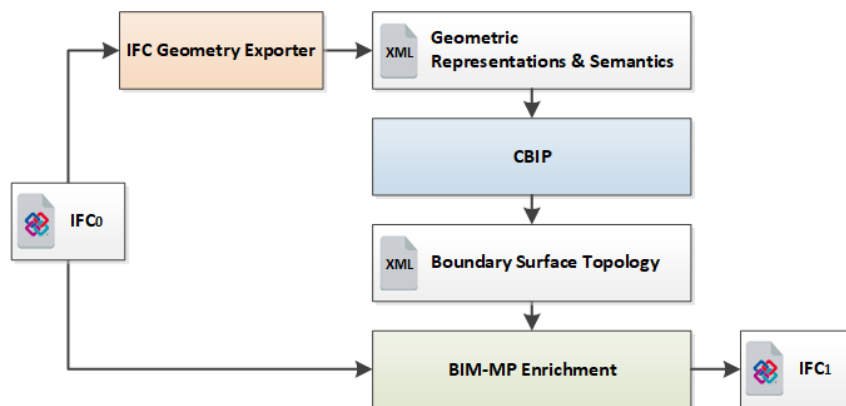


Figure 24 CBIP enrichment process

The CBIP operation is illustrated in Figure 25: from a set of architectural elements (part A) CBIP performs the necessary geometric operations and extracts the second-level space boundary surface topology (part B). The second-level space boundary topology is a part of the boundary surface topology (illustrated in part B of Figure 25), which includes the internal and external space boundaries and the virtual space partitions. The 3D points of the extracted surfaces of this topology are used to populate the appropriate IFC data classes which together with all the

necessary semantic links to the elements of the building and their respective material layer set bedding, are contained in the enriched IFC file outputted by the CBIP service.

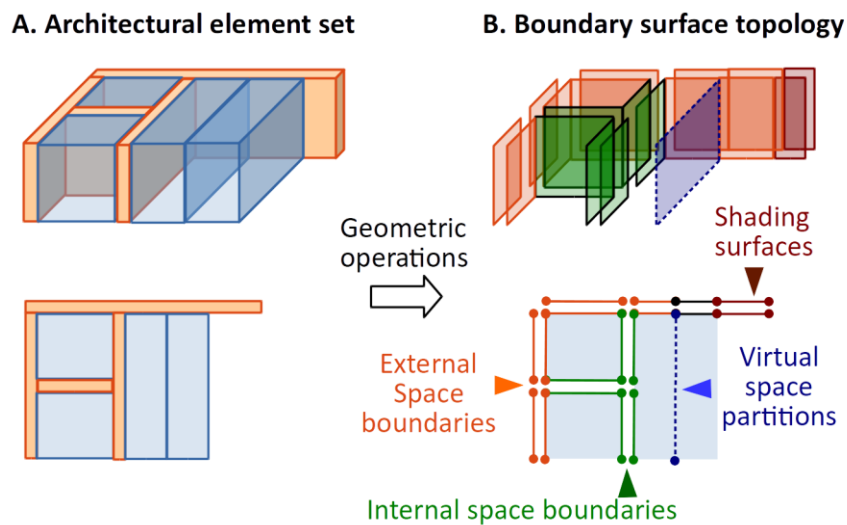


Figure 25: Illustration of BIM-MP's CBIP service

3.3 GEOMETRY ENGINE

The geometric data content of IFC files is not in a graphics compatible representation format to avoid verbosity and be as short as possible without losing critical information. To convert the IFC geometric data into a graphics compatible format, BIM-MP has a dedicated geometry engine which transforms all geometric representations of the architectural elements into boundary representations first using a B-rep generation process and finally into a graphics compatible format (OBJ, glTF) using a model viewer. These low-level geometric operations are performed using a dedicated C++ geometric library which based on clipper - one of the fastest and robust open-source freeware libraries for clipping and offsetting lines and polygons in two dimensions. The Clipper library developed by Angus Jonhson and based on Vattis' algorithm (Vatti, 1992). The B-rep generation and the model viewer components are described in the following sections.

3.3.1 B-rep Generation

The B-rep generation (BRG) module of BIM-MP is responsible for transforming the various geometric representations of building elements existing in IFC data into correctly oriented boundary representations. This process is illustrated in Figure 26: BRG receives as input the

geometric content and its semantics of an IFC model, exported in XML file by the IFC Geometry Exporter, and returns the B-reps of this content, in OBJ format.

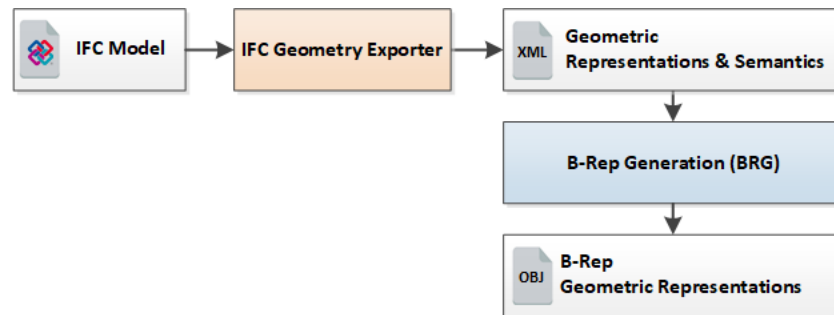


Figure 26: B-rep generation process

The BRG module outputs correctly oriented B-reps, which conform to the righthand rule, as displayed in Figure 27.

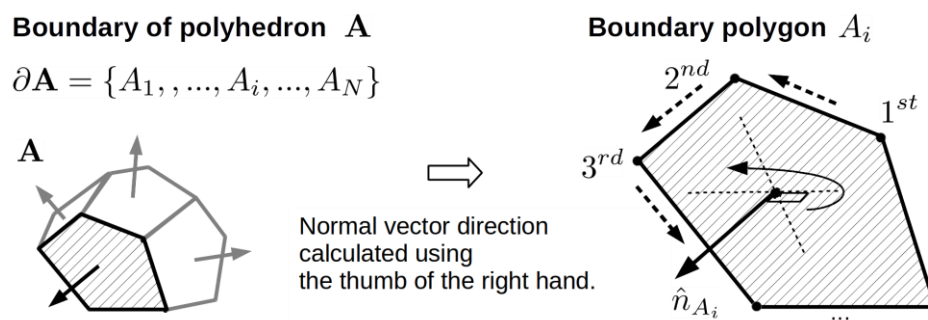


Figure 27: Illustration of a correctly oriented boundary representation

BRG transforms to B-reps, multiple solid geometric representation types defined by the `IfcProductDefinitionShape` and its subclasses, as displayed in Figure 28: `IfcProductDefinitionShape` and supported IFC subclasses.

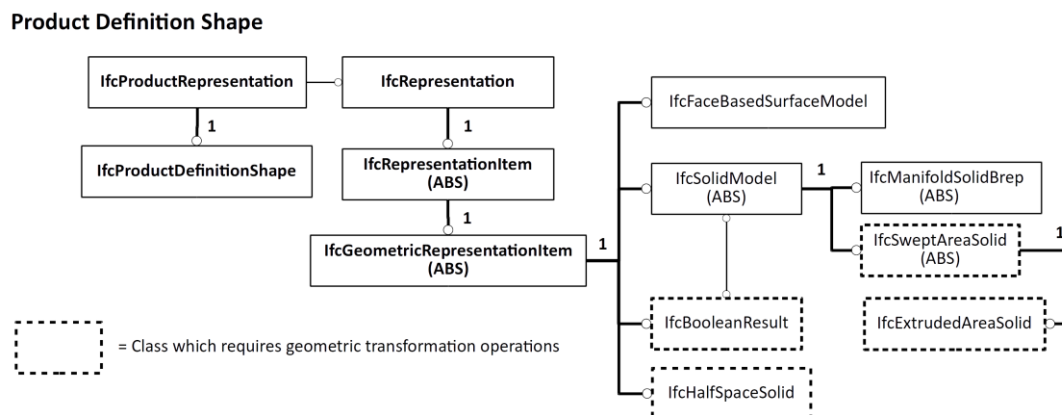


Figure 28: IfcProductDefinitionShape and supported IFC subclasses.

Essentially, the BRG module performs the necessary complex geometric transformation operations, depending on the input solid representation, to produce the three-dimensional points of the surfaces of the boundary representations contained in the output OBJ file.

3.3.2 Model Viewer

The Model Viewer is a BIM model visualization component based on the open-source project xeogl (xeogl, 2018). It uses WebGL for rendering 3D graphics natively within any compatible web browser. This component loads the geometry of a BIM model from glTF data (Robinet, 2014), data, allowing the BIM-MP user to navigate with the camera and to explore the structural and non-structural elements of a building and their properties. The Model Viewer is written in JavaScript and deployed as part of the Core Module. It has access to the project repository to load and visualize the 3D geometric representation of the BIM model. The process is the following: The BIM-MP triggers the execution of the B-Rep Module automatically when the IFC data are available in the project repository. Next, the B-Rep Module generates and stores the OBJ data into the project repository. Finally, BIM-MP uses an open-source glTF converter (CesiumGS) to generate the glTF data. The operation of BIM-MP's model viewer is summarized in Figure 29.



Figure 29: Process diagram of BIM-MP's model viewer

The 3D graphics rendering has been tested successfully in the Core Module as shown in Figure 30.

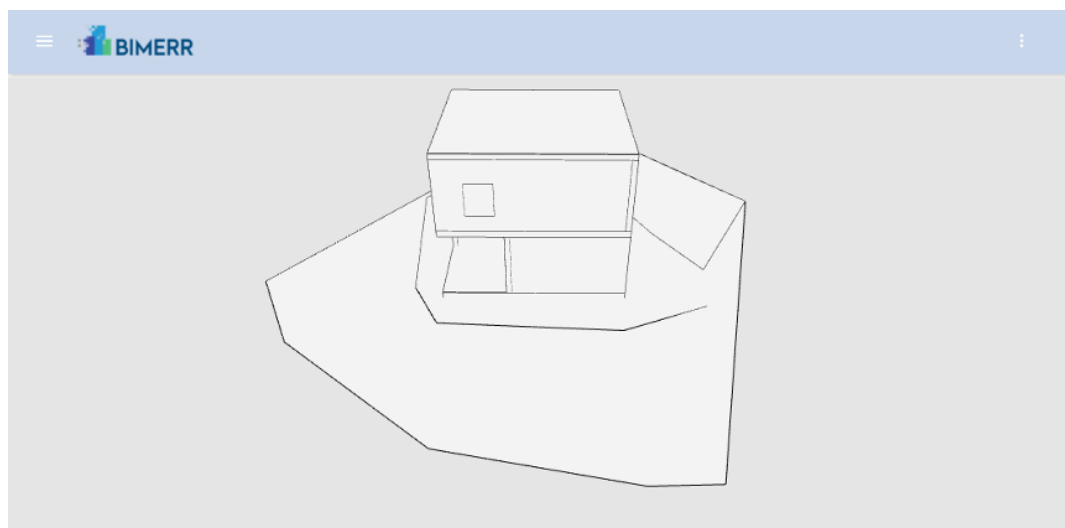


Figure 30 IFC rendering using BRG, glTF converter and xeogl engine

4. INTEGRATION OF BIM-MP IN BIMERR WORKFLOWS

This section gives a high-level overview of the IFC data flow in BIMERR. The dynamic workflow can be grouped into three categories: a) IFC data creation; b) IFC data validation, and c) IFC data manipulation and transaction management. As shown in Figure 31, BIF is used as the central data repository and is responsible for storing files and for controlling access rights.

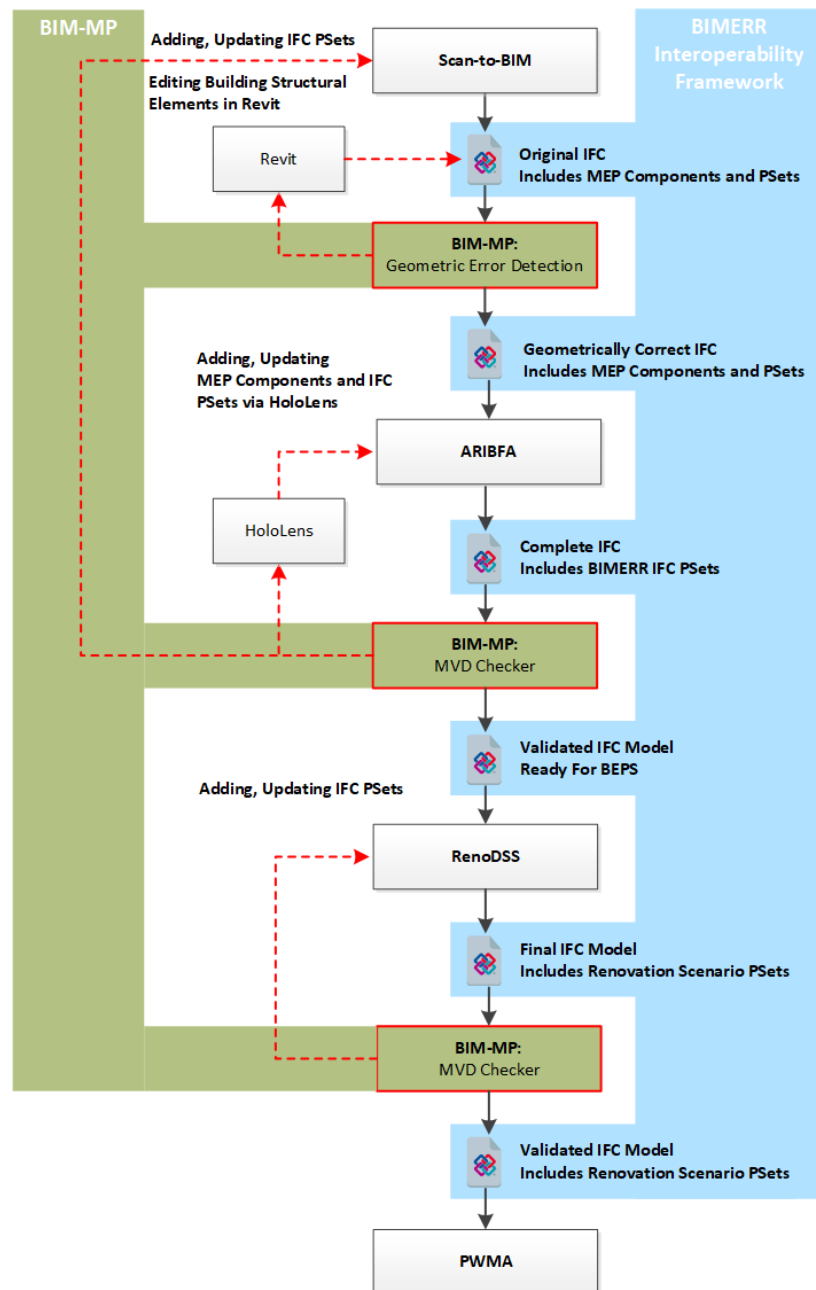


Figure 31 High-level IFC data flow in BIMERR

Within BIMERR, Scan-to-BIM tool is responsible for populating the MEP components in the initial IFC file. However, sometimes during the on-site inspection, new components are recognized. At this point, ARIBFA acts as a complementary tool providing a state-of-the-art solution to help BIMERR users to add missing information in the model using augmented-reality glasses. After the IFC file creation by Scan-to-BIM, the platform performs a series of data quality checks, including schema compliance and geometry error detection checks. Afterwards, ARIBFA adds to the model the missing information as well as the new components and their properties. Next, BIM-MP performs the completeness checking on the new version to ensure that the IFC model is ready for energy evaluation by RenoDSS. After the evaluation and scenario selection, the RenoDSS creates a new version of the model, and the BIM-MP performs a new MVD completeness check to ensure that all IFC PSets of the selected renovation scenario are included. This modified BIM model is requested from the PWMA tool, to calculate the time and the cost of the selected renovation procedures. The services offered or supported by BIM-MP presented as IFC workflow processes in the following sections.

4.1 IFC CREATION

The workflow starts with the creation of the IFC file. Within BIMERR, the Scan-to-BIM tool or a BIM authoring tool like Revit may be used to create the initial IFC files. This applies to both the pre-validation and demonstration buildings. The IFC file generation processes of both tools are error-prone due to various limitations. The following sections analyze the procedures used by BIM-MP for validating and correcting the IFC data.

4.1.1 Scan-to-BIM

The Scan-to-BIM tool is one of the IFC creation tools used in BIMERR. The tool is described analytically in D5.3 is an open-source software framework for generating IFC files from point cloud data. Scan-to-BIM consists of two components: Scan-to-BIM (Structural) which populates respective IFC data classes with the geometric representations of structural building elements and Scan-To-BIM (MEP) which populates respective IFC data with the details of the building's MEP components.

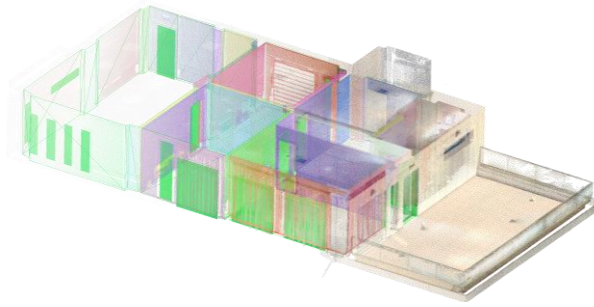


Figure 32 Geometry of structural components generated by Scan-to-BIM

The structural and MEP IFC data from the Scan-to-BIM tool are generated in a semi-automated manner involving human intervention, manual adjustments and corrections. Hence, these IFC data should be checked by the BIM-MP for detecting geometric errors that may affect the building energy simulation model. The BIM-MP's functional modules generate visual reports to help the user resolve the detected geometric issues in Revit as illustrated in Figure 33.

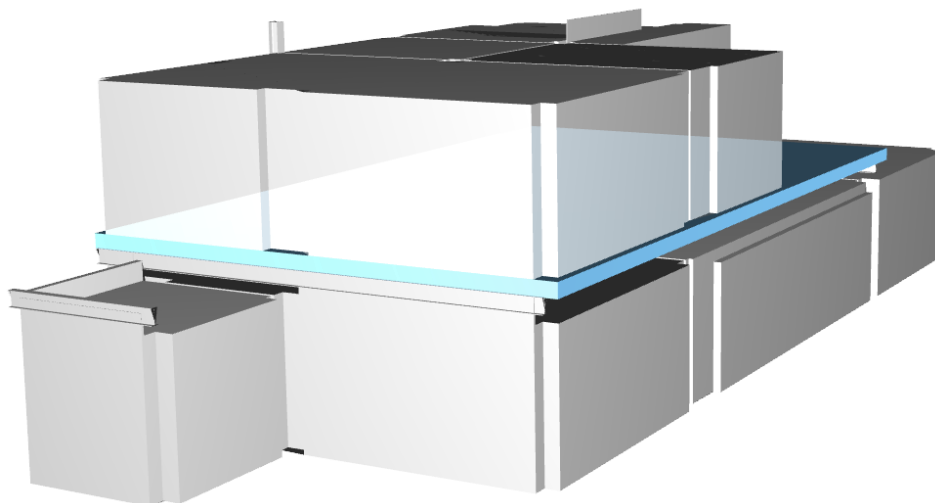


Figure 33 Clash error detected by BIM-MP's GED module

The visual inspection of the detected errors helps the user to correct the BIM model using Autodesk's Revit BIM authoring tool. BIM-MP provides a modified version of the open-source project "IFC for Revit" ensuring the correct importation of the geometry for all structural building elements. This is possible because the Scan-to-BIM IFC exportation process complies to a specific geometric MVD that is fully compatible with the modified version of "IFC for Revit" open-source project. The user can edit the structural building elements to make the necessary adjustments and to export the modified IFC file. Moreover, the IFC Importer has been modified to map all custom IFC PSets to Revit's internal data model and the IFC Exporter has been

modified to export these IFC PSets to the output IFC file. Apart from this, because Revit's IFC exporter discards the 2nd level space boundaries contained in the input IFC file, BIM-MP's semantic enrichment process is required to enrich the modified BIM model with such information as shown in Figure 34.

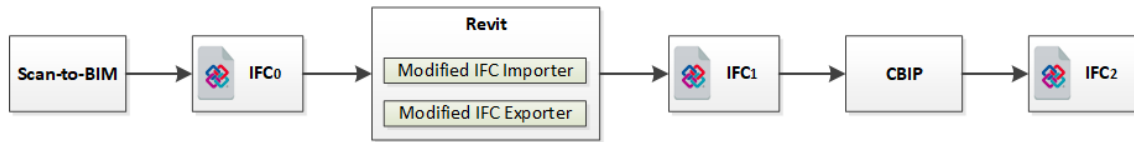


Figure 34 Integration with Revit

4.1.2 REVIT

Autodesk's Revit is used for the creation of the IFC files of BIMERR's pre-validation and demonstration sites. In this section, a step-by-step model design and exportation process, applied on KRIPIS pre-validation building, is presented. This workflow uses the modified version of "IFC-for-Revit" open-source project for exporting information from Revit's internal libraries such as material thermal properties, schedules, infiltration rates and more.

Project Settings

The first step in Revit is to define the template and the project units. In this case a default metric template was selected. Additionally, the Project Units (Manage → Project Units) must be defined according to the International System of Units metric system (SI) – at least for the Common and Energy disciplines (see Figure 35).

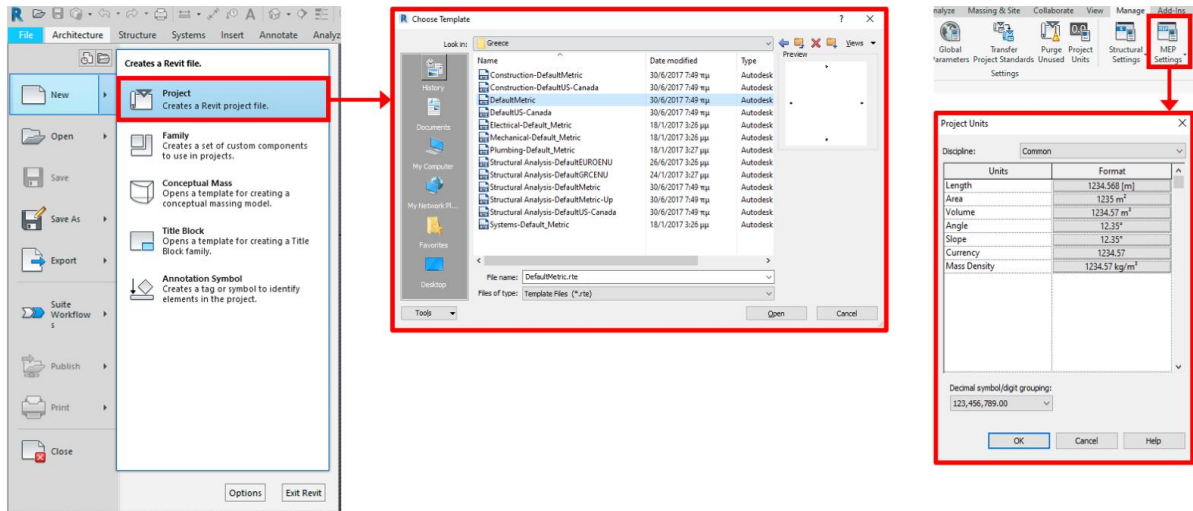


Figure 35 Default Template and Project Units selection in Revit

Space Definition

The space representation is used to describe the geometric representation of the air volume occupied the building spaces. Spaces should be defined for each area of the building, including unoccupied areas such as plenums. Regarding the space definition, firstly it is important to activate the visibility and graphics settings of a view in order to define the elements and categories visible in this view and their graphical appearance. On the Model Categories tab of the Visibility Graphics dialog, the user by scrolling down to Spaces and by checking the box becomes able to visualize the spaces of the building in this specific view (see Figure 36). In addition, the "Areas and Volumes Computations" option must be activated (Analyze tab → Spaces & Zones → Area and Volume Computations)

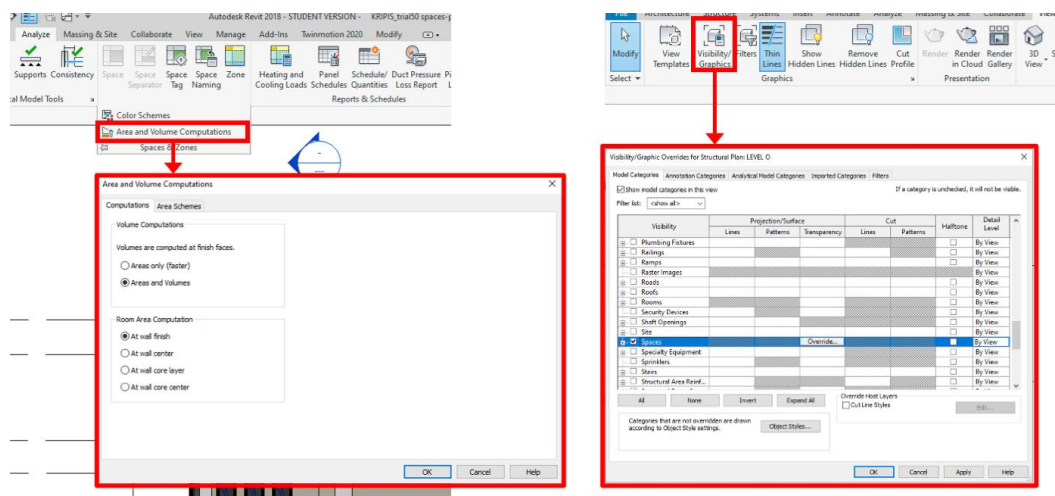


Figure 36 Activation of "Areas and Volumes Computation" and Spaces visibility in Revit

In order to place a space, with the floor plan view active, the designer should follow the path (Analyze → Spaces and Zones → Space). It is recommended to create spaces manually and to avoid the automatically placing option. To be able to see the placed Spaces green as at the figure below, the Hidden Line Visual Style must be selected. It is also important to verify or redefine from the Properties palette the upper limits and boundaries of each Space, where necessary (see Figure 37).

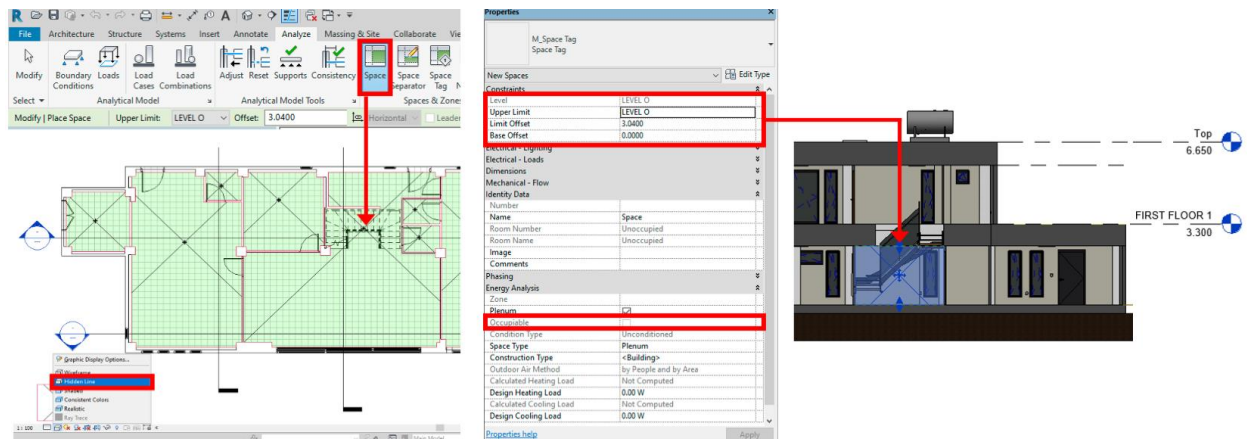


Figure 37 Space placing and Space limits definition in Revit

Apartment Definition

Overall, it is recommended to create Zones (groups of spaces) and to assign each space to a created zone. By using the zone tool, the user can define spaces that belong to a specific zone which is controlled by environmental control systems. Zones can be edited after their creation, by adding or removing spaces from them (Analyze → Spaces and Zones → Zone, displayed in Figure 38). To modify the zones and their included spaces, the user can activate the System Browser palette from (View → User Interface). In this way the type of each space can be defined also easier (see Figure 38).

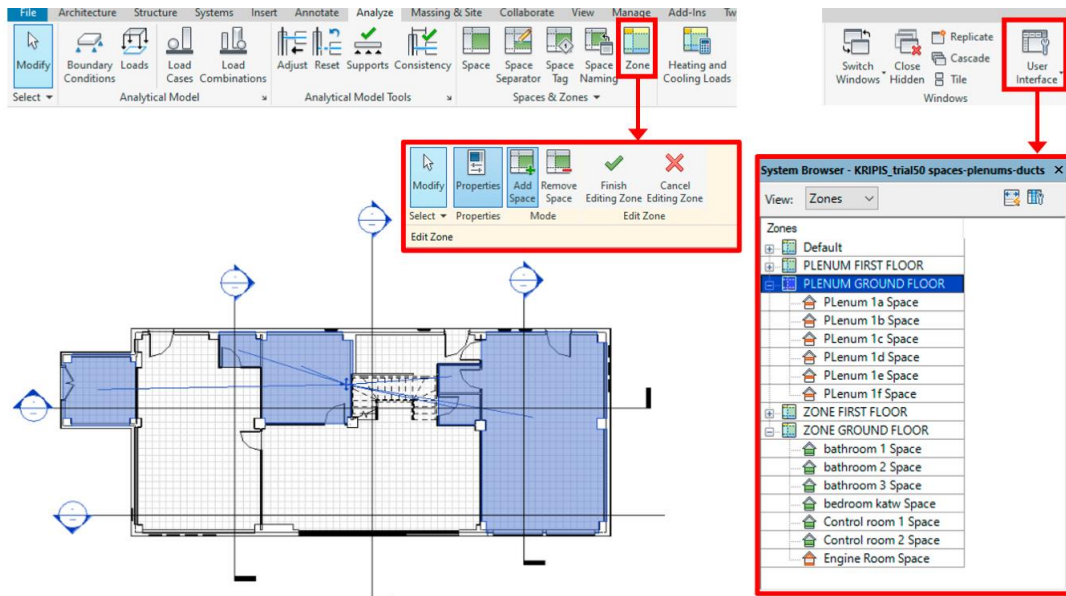


Figure 38 Zone creation and Space assignment in Revit

Site Definition

In case the building site is attached to the building, the attaching site surfaces should be correctly defined. A building site can be defined in Revit by selecting (Massing & Site → Toposurface). After the building site is defined a building pad should also be defined in Revit by selecting (Massing & Site → Building Pad). For rendering purposes of the IFC file after the exportation the Survey Point and the Project Base Point of the model must be placed into or close to the building's footprint, as illustrated in Figure 39. To visualize the Survey Point and the Project Base Point of the model, the designer can open the Visibility Graphics menu (Model Categories → Visibility Graphics) and scroll down to Site in order to check the relevant boxes.

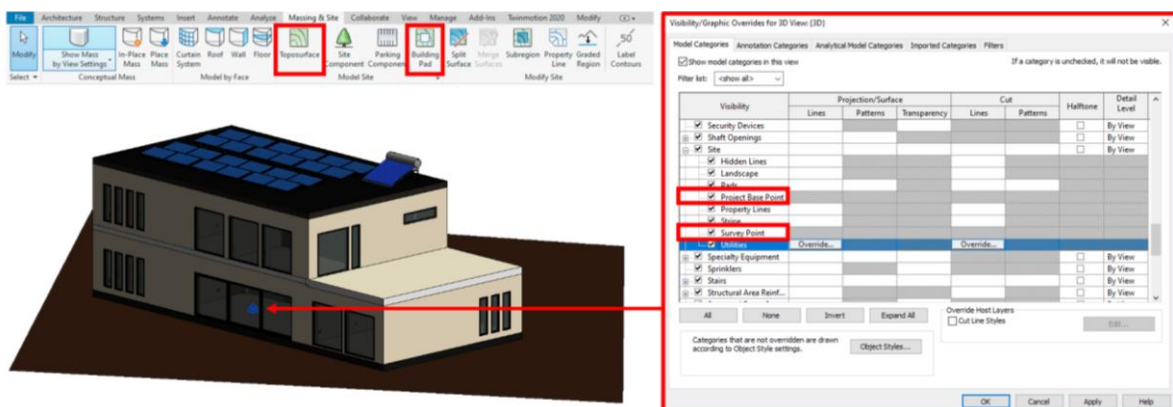


Figure 39 Site definition and Project Point visibility activation in Revit

IFC Exportation

Autodesk's Revit provides IFC4 exportation supporting both DTV and RV model views. The designer is able to decide which Revit Categories should be exported and which IFC entity they will be transformed to, as displayed in Figure 40.

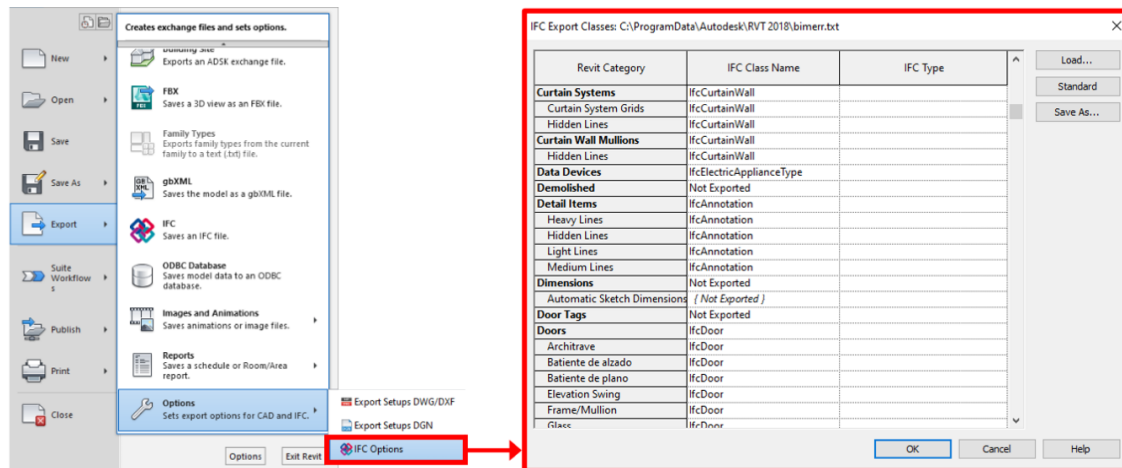


Figure 40 IFC entities export mapping in Revit

4.2 IFC VALIDATION

When the IFC file is exported either from Scan-to-BIM tool or from Revit, it is forwarded to the quality checking services of BIM-MP. These services constitute the IFC validation workflow in which a sequence of checks is performed to ensure the overall quality of the IFC file.

The first process in the sequence is the IFC schema compatibility check, where the structure of the input IFC is examined against the IFC4 schema. The content of the data objects as well their semantic annotations are verified according to the EXPRESS definitions of the IFC4 standard.

If the compatibility check is passed, BIM-MP forwards the IFC file to the Geometric Error Detection module. In this stage, the existence of the three geometric error types defined in the previous section (surface errors, clash errors and space errors) is examined within the geometric content of the IFC input data. When the Geometric Error Detection module finishes its checks, the BIM-MP's notification system informs the user to download the error report. If the IFC file is error-free the user can start the on-site survey for detecting additional MEP components of the building using the augmented-reality glasses.

After the on-site inspection and detection of new MEP components, the enriched IFC is forwarded to the MVD Completeness Checker, where the attributes of the IFC objects as well as their semantics are examined to access whether they comply to the conditions defined in an mvdXML file, which is generated according to RenoDSS data requirements.

The IFC files which have passed the three data quality control checking stages of the IFC validation workflow, can be forwarded to BIF for populating the Building Data Model.

4.3 IFC QUERIES

When the BIM model is loaded and the checking process is completed, the Building Data Model population workflow is initiated. During this workflow, the BIF is able to perform IFC data queries to the BIM-MP API and collect their responses, to populate the Building Data Model, as long as the BIM-MP has defined specific data collection jobs in the BIF UI.

To facilitate IFC data annotation and minimize the 3D rendering time of other BIMERR tools, BIM-MP, using the embedded geometric engine, produces for each building floor the 3D geometry of the floor's structural building elements. This 3D geometry, for each building floor, is then exported as OBJ data files and stored into the BIF. Then, BIF creates the semantic links between the URIs of the respective floor OBJ files and the corresponding instances of the Building Data Model. ARIBFA uses these OBJ files to optimize the rendering time of displaying structural building objects to the augmented-reality glasses.

4.4 IFC TRANSACTIONS

BIM-MP provides an API for creating and updating IFC PSets for existing IFC objects. When ARIBFA creates annotations of the detected MEP components using the augmented-reality glasses and updates the corresponding data models in BIF, BIM-MP initiates an IFC update process to populate the enriched IFC with the additional information in the form of IFC PSets.

5. APPLICATION EXAMPLES

The BIMERR tool validation and demonstration will take place in two steps: a) the pre-validation phase and b) the validation phase on real renovation sites.

Regarding the pre-validation activities, one used demonstration site is the KRIPIS Smart Home, located in Greece, owned and operated by CERTH. It is a representative of a single-family, detached residential building and is already equipped with many IoT, Smart Home solutions that provide a lot of information about its operational characteristics. KRIPIS is not going to be renovated. However, it contributes to experimentation, evaluation and usability testing of the BIMEER tools by providing feedback to the development partners for improvements. In the case of KRIPIS, the suspended ceiling of the building was a notable issue. More precisely, even though there is an empty space known as “plenum” between the false ceiling and the slab, only the false ceiling is visible by the resident. The Scan-to-BIM tool was not provided with point cloud data of this upper plenum. However, the existence of this unoccupied and unconditioned space is considered to be necessary for an accurate energy evaluation.

Hence, two different BIM models have been created in Revit following the BIM guidelines mentioned above: a) a detailed BIM model and b) a simplified BIM model according to the scanning limitations, to analyse the performance of BIM-MP’s geometric checking and the semantic enrichment processes.

The BIM designer follows an iterative process for correcting the BIM models, as shown in Figure 41. After the first IFC exportation from Revit, the designer uploads the generated IFC to BIM-MP’s internal repository, and the GED module is executed. This Module produces an OBJ file that contains 3D information of the detected errors. The designer can view the error report either in BIM-MP’s internal WebGL viewer or in any other OBJ viewer. When the generated error report is empty the BIM model is characterized as geometric error-free and the designer can proceed with the execution of the CBIP module. At this point, a visual inspection of the model is recommended for ensuring the overall quality of the IFC file.

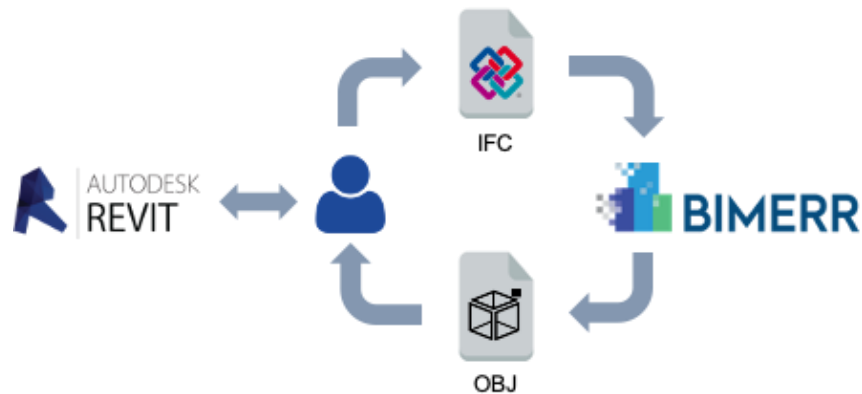


Figure 41 Iterative correction process of a BIM model

5.1 DETAILED MODEL

Taking into account the design characteristics of the KRIPIS building, a detailed BIM model has been designed in Revit following the instructions of Autodesk's official documentation regarding the Space and Plenum definition. Occupied spaces were placed from the ground floor level up to the suspended ceiling (plenum ground floor level). A similar procedure was followed for the first floor. Moreover, plenums were placed between the false ceiling and the upper slab to define the unoccupied spaces as illustrated in Figure 42.

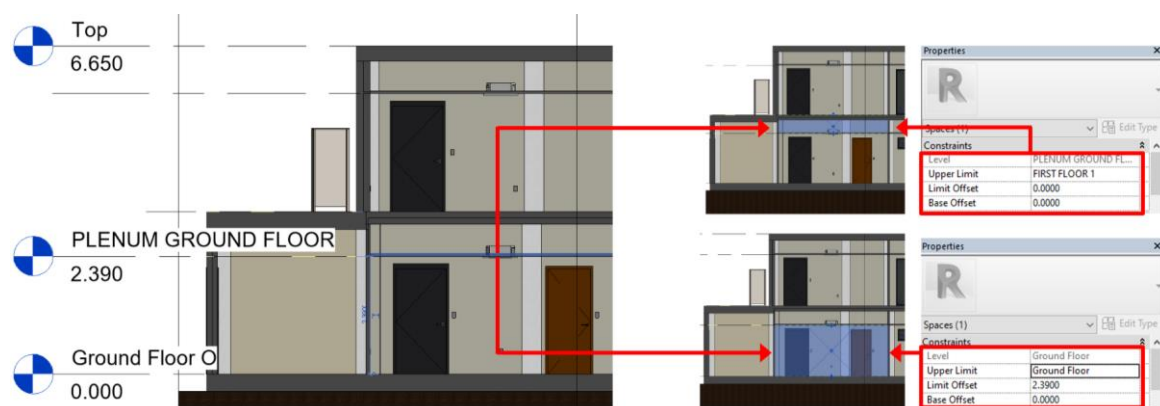


Figure 42 Detailed version of KRIPIS model which includes spaces and plenums

The detailed version of KRIPIS model contains the actual geometry of all structural building elements. In this case the thickness of the slab is accurately provided as shown in Figure 43.

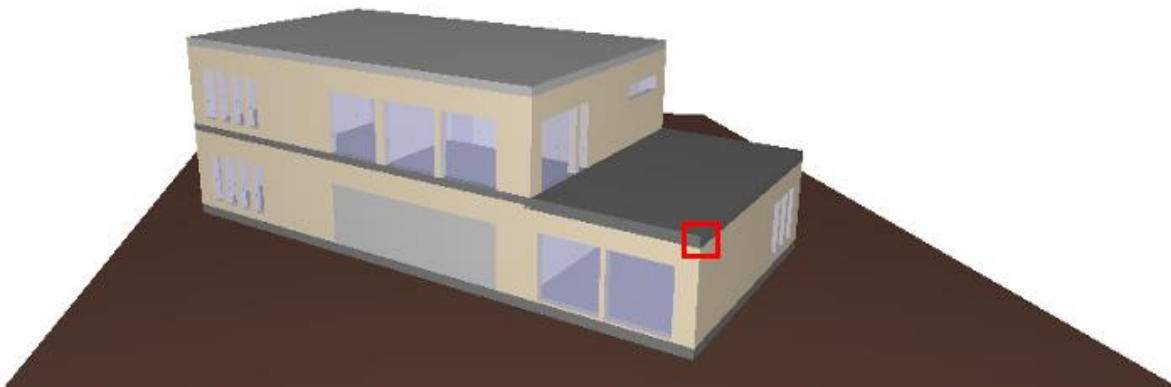


Figure 43 Detailed version of KRIPIS IFC model

When an error-free version of the model is created, the designer executes the CBIP module to enrich the IFC file with the 2nd level space boundaries. Figure 44 illustrates the 2nd level space boundaries of a space that contains a plenum.

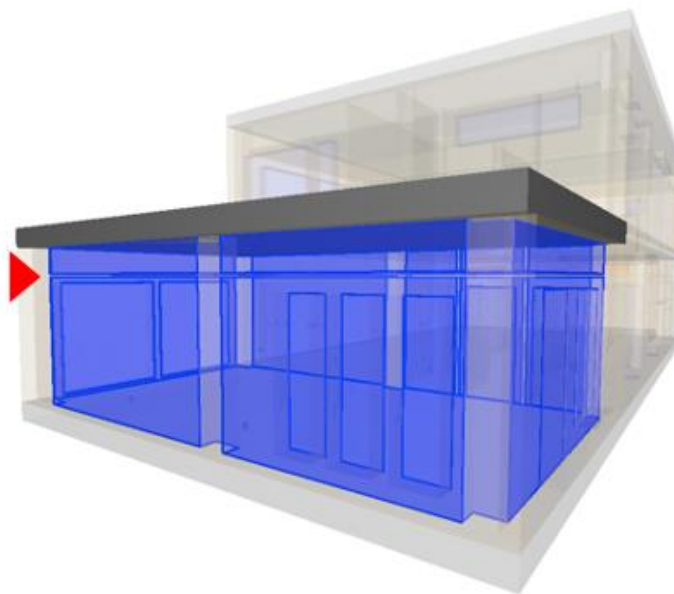


Figure 44 Detailed version of KRIPIS IFC which includes 2nd level space boundaries

5.2 SIMPLIFIED MODEL

On the other hand, by following the Scan-to-BIM approach, another approximate Revit model was alternatively created since these plenum areas will never be scanned. The main concept is to have a solid slab consisting of the original slab, an air layer and the suspended ceiling in an integrated way (see Figure 45).



Figure 45 Simplified version of KRIPIS model which complies with the scanning limitations

In this case, the IFC contains only the occupied and conditioned spaces of the building while plenums do not exist. The simplified version of the KRIPIS model contains a solid slab as shown in Figure 46.

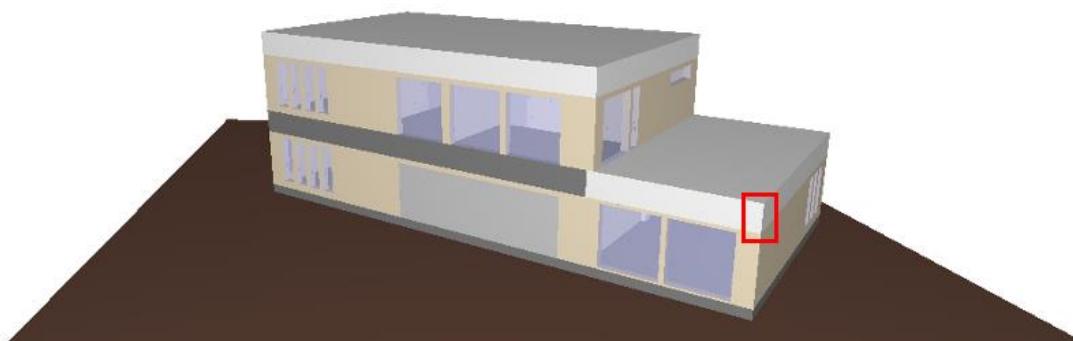


Figure 46 Simplified version of KRIPIS IFC model

When an error-free version of the model is created, the user executes the CBIP module to enrich the IFC file. Figure 47 illustrates the 2nd level space boundaries of a space that does not contain plenum.

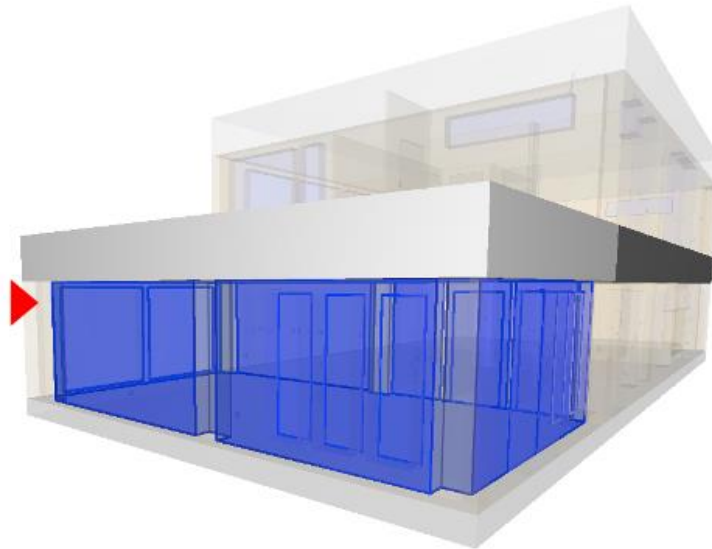


Figure 47 Simplified version of KRIPIS IFC which includes the 2nd level space boundaries

In conclusion, in future work, it would probably be interesting to compare the results from the two BIM model versions and their impact on the energy estimates of the project.

6. CONCLUSIONS AND PLANNING

This deliverable introduced the main functional components of BIMERR's BIM management platform, and their use is illustrated in dynamic data workflows. Application examples on BIMERR's demonstration buildings, were also presented. BIM-MP components are integrated into a cloud-based framework following a service-oriented architecture (SOA), which offers IFC data quality checking, query, storing and enrichment services, to BIMERR users and tools.

The data quality checking mechanisms include geometric error detection, MVD-based completeness and data consistency checking operations applied on IFC input data files, to ensure that IFC data are suitable for BIMERR's BIM needs. On the other hand, BIM-MP's data enrichment services include the Automatic Space Generation (ASG) service, where the building's internal building volumes are reconstructed from their surrounding constructions and the Common Boundary Intersection Projection (CBIP) service, where the building's second-level space boundary topology is obtained. The data enrichment services are designed to aid the BEPS model generation especially when the IFC file is originated from a BIM authoring tools such as Autodesk's Revit.

BIM-MP offers two types of responses to external requests: a textual type, where the output is formatted in JSON, and a visual type useful for displaying geometrical output which is formatted either: as an OBJ file to be compatible with most of other external graphics displaying tools, or as a GLTF file supported by BIM-MP's native graphical environment, or as a BCF compatible to external BIM authoring tools.

The work presented here introduces a first version of BIM-MP. In the future and as BIMERR tools evolve, more connectivity options will be added to BIM-MP, to support faster and on demand visual as well as text responses to BIMERR's tool requests.

The final version of the BIM-MP will be provided at M30 and will include all missing functionalities of the BIM-MP's Core Module. The next version will be tested on all pre-validation buildings. The exporters and importers of BIM authoring tools such as Revit and ArchiCAD, will be configured to ensure flawless IFC data transfer from and to BIMERR framework.

7. BIBLIOGRAPHY

- Bazjanac, V. (2010). Space boundary requirements for modeling of building geometry for energy and other performance simulation. *CIB W78 27th International Conference*.
- BIMserver*. (2019). Retrieved from BIMserver: <https://github.com/opensourceBIM/BIMserver>
- buildingSMART*. (2020). Retrieved from Model View Definition: <https://technical.buildingsmart.org/standards/ifc/mvd/>
- CesiumGS. (n.d.). *Obj2glTF*. <https://godoc.org/github.com/sturfeeinc/glTF/obj2glTF>.
- IfcDoc*. (2020). Retrieved from <https://www.buildingsmart.org/standards/groups/ifcdoc/>
- Lilis, G. N., Giannakis, G., & Rovas, D. (2015). Detection and semi-automatic correction of geometric inaccuracies in IFC files. *International conference of IBPSA - Building Simulation*, (pp. 2182-2189).
- Lilis, G. N., Giannakis, G., & Rovas, D. (2017). Automatic generation of second-level space boundary topology from IFC geometry inputs. *Automation in Construction*, 108-124.
- Lilis, G. N., Katsigarakis, K., & Rovas, D. (2018). A tool for IFC building energy performance simulation suitability checking . *12th European Conference on Product and Process modelling*, (pp. 57-64).
- Robinet, F. (2014). gltf: Designing an open-standard runtime asset format. *GPU Pro 5*, 375-392.
- Vatti, R. B. (1992). A generic solution to polygon clipping. *Communications of the ACM*, 56-63.
- Wavefront, T. (1992). *Wavefront OBJ File Format Summary*. <https://www.fileformat.info/format/wavefrontobj/egff.htm>.
- xeogl*. (2018). Retrieved from Data Driven WebGL: <https://xeogl.org>