Streamer
European research on energy-efficient healthcare districts

D5.2

Semantic Web based Product Modelling Ontology (PMO) -
BIM Information Management and Quality Control

Deliverable Report: 5.2 Final Version

Streamer - Optimised design methodologies for energy-efficient buildings integrated in the neighbourhood energy systems.
The Streamer project is co-financed by the European Commission under the seventh research framework programme FP7.EeB.NMP.2013-5; GA No. 608739)

D5.2

Semantic Web based Product Modelling Ontology (PMO) -

BIM Information Management and Quality Control

Issue Date	1 November 2016
Produced by	AEC3, CSTB, DJG, DWA, KIT, NCC, TNO,
Main author	Matthias Weise, Thomas Liebich, Nick Nisbet (AEC3)
Co-authors	Bruno Hilaire, Audrey Vial (CSTB), Danny Werensteijin (DJG), Jan-Peter Pols (DWA), Steffen Hempel, Karl-Heinz Häfele (KIT), Stefan Dehlin (NCC), Thorsten Lang (IAA), Jan-Peter-Pols (DWA), Pim van den Helm (TNO),
Version:	Final
Reviewed by	Hassan Sleimann (CEA), Thorsten Lang (IAA)
Approved by	Freek Bomhof (TNO), Marc Bourdeau (CSTB)
Dissemination	O, PU

Colophon

The Streamer project is co-financed by the European Commission under the seventh research framework programme with contract No.: 608739 - FP7-2013-NMP-ENV-EeB. The information in this publication does not necessarily represent the view of the European Commission. The European Commission shall not in any way be liable or responsible for the use of any such knowledge, information or data, or of the consequences thereof.

Publishable executive summary

This document, deliverable 5.2 "*Semantic Web based Product Modelling Ontology (PMO)*", is dealing with BIM information management and quality control. Knowledge about hospital design (through typologies, labels, PoR template, and design rules) has been developed in WP1, WP2 and WP3. Based on this outcome, and following the state-of-the art analysis and the conclusion of previous deliverable 5.1 that recommended to follow a hybrid approach, we focus in this deliverable on formalizing information exchanges between tools at different steps of the design process and their application in the BIM/IFC world via the mvdXML format. This replaces the originally planned intent to develop a Semantic Web-based PMO that integrates all relevant design and checking knowledge in a single specification, but would have introduced another layer of complexity and the need for developing a related tool to be used in STREAMER, without any foreseen tangible advantages. Even if there is no PMO, at the end there is a tool (BIM-Q) that captures user requirements and turns them into a quality control process applicable to design alternatives expressed in IFC. Besides, the presented approach is reusing existing standards and specifications and therefore also enables to integrate available tools.

More precisely we introduce a layered approach that differentiates between different kinds of model checking. Each layer requires own checking knowledge and is typically managed by different authors or standardization bodies. It represents a generic solution and is based on principles adopted from the IDM/MVD methodology defined by buildingSMART. It is expected that it is a more flexible and practical approach that more easily can be adopted by the industry, also because it is in line with ongoing standardization efforts. .

The focus of developed specifications and prototypes is on collaboration support, more specifically on checking data exchange requirements that represents one layer in the proposed approach and supports information sharing between stakeholders. A main aspect is completeness of information derived from specific needs of design processes. Based on the presented approach, the sender and receiver of information will be able to check if all agreements are fulfilled or if some data is missing. They can use the same specification and neutral checking tools to validate results, ideally before submitting requested data to the receiver. This will help to reduce "requests for information" due to incomplete datasets.

Technically, main research questions have been: (1) how to do neutral model checking based on open BIM; and (2) how to capture and manage exchange requirements. mvdXML was selected as a basis for further developments, which so far had been used for documentation purposes only. STREAMER started to use mvdXML for model checking and was contributing to a new release published in spring 2016. By following this standardization path it is expected that the chosen approach is leading to better acceptance by the industry. Besides working on technical specifications another challenge was to adequately capture domain knowledge that is normally discussed using the language of domain experts. STREAMER decided to capture and manage both types of specifications in order to support later maintenance where it might be necessary to reuse, revise and update requirements.

The developed approach was validated for three data exchange scenarios: (1) client requirements for developing an initial design (PoR data), (2) basic room layout data to be used for basic energy simulation and (3) energy simulation results to be used for design evaluation and further design detailing. Practical tests have been carried out by the four STREAMER example projects using developed prototypes but also commercial tools. They show existing variety of model checking and necessary efforts to harmonize data flows.

Results of task 5.1 show how definition and checking of exchange requirements can be done. The developed approach is in line with buildingSMART developments and provides a sound basis for further specification work. Presented mvdXML specifications are still under revision and will be extended to reflect changes and extensions of the data flow. Ideally, requirements for the different energy simulation tools as well as expected result set are fully

harmonized and thus independent from used tools. Such neutral specifications could then be input to further standardization activities.

List of acronyms and abbreviations

- ADE: Application Domain Extension
- API: Application Programming Interface
- BCF: BIM Collaboration Format
- BIM: Building Information Modelling
- BPMN: Business Process Modelling Notation
- bSDD: buildingSMART Data Dictionary
- CAAD: Computer Aided Architectural Design
- CB-NL: Concept Library the Netherlands
- CityGML: City Geography Markup Language
- CMIS: Content Management Information System
- CMO: Concept Modelling Ontology
- CSV: Comma-separated Values (file format)
- DST: Decision Support Tool
- gbXML: green building XML
- GIS: Geographic Information System
- GML: Geography Markup Language
- HTTP: Hypertext Transfer Protocol
- HVAC: Heating, Ventilation, Air Conditioning
- EDC: Early Design Configurator
- EPSG: European Petroleum Survey Group
- IDM: Information Delivery Manual
- IFC: Industry Foundation Classes
- IFD (1): International Framework for Dictionaries
- IFD (2): Industrial, Flexible and Demountable Building (Dutch standards)
- JSON: JavaScript Object Notation
- LOD (1): Linked Open Data (well-known in the semantic web community);
- LOD (2): Level of Detail/Development (used in the AEC industry)
- MVD: Model View Definition
- NCM: National Calculation Method
- OGC: Open Geospatial Consortium
- OWL: Web Ontology Language
- REAP: Rotterdam Energy Approach and Planning
- RIF: Rule Interchange Format
- PMO: Product Modelling Ontology
- PoR: Programme of Requirements
- RDF: Resource Description Framework
- SACS©: System for the Analysis of Hospital Equipment
- SKOS: Simple Knowledge Organization System
- SPARQL: SPARQL Protocol And RDF Query Language
- STEP: Standard for the Exchange of Product Model Data
- SW: Semantic Web

- SWOP: Semantic Web-based Open Engineering Platform
- SWRL: Semantic Web Rule Language
- Turtle: Terse RDF Triple Language
- URI/URL: Uniform Resource Identify/Locator
- UTM: Universal Transverse Mercator
- W3C: World Wide Web Consortium
- XML: eXtensible Markup Language
- XSD: XML Schema Definition

Definitions

Building Information Modelling (BIM):

A digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle (further information see chapter 1.3).

Ontology:

Ontology's are the structural frameworks for organizing information and are used […] as a form of knowledge representation about the world or some part of it. (Wikipedia[1])

Syntactical compliance:

Compliance with the file format, lowest level of model checks relevant for file-based data exchange (further information see chapter 2.2).

Structural compliance:

Compliance with a given data format or ontology. The type of checks depends on the used modelling language and may enables to do encode sophisticated model checks (further information see chapter 2.2).

Data exchange requirements:

Checks completeness of data required by a specific process, thus typically linked to workflow and task descriptions (further information see chapter 2.2).

Model consistency:

Fundamental checks of proper model semantics (further information see chapter 2.2).

Design rules:

Recommendations for a good design that may be prioritized or overruled (further information see chapter 2.2).

Regulations:

Rules with a legal background that must be satisfied to be built (further information see chapter 2.2).

Project specific client requirements or PoR:

Requirements for a specific building that are defined by the client (further information see chapter 2.2).

[1] http://en.wikipedia.org/wiki/Ontology_%28information_science%29 (accessed: 2014-01-20)

Contents

1. Introduction

1.1 Goals

The goals of this task 5.1 has been defined (1) To build a knowledge base on ontology work in the domain, and present a robust framework for the practical implementation by the design team, stakeholders, building occupants; (2) To use this knowledge base to develop an ontology-based energy information system and associated tools for design energy-efficient buildings and districts, which will lead to ontology enabled interoperability; and thus, (3) To proof the eligibility of ontologies in the preliminary design stage of both new and retrofitted buildings.

1.2 Vision

As Streamer looks into research in accordance with practical needs, the interest in ontologies is driven by the domain ontology, i.e. the need for conceptualizing, organizing, and stating knowledge that is essential to energy-efficient hospital design using modern BIM and GIS technologies. Abstract philosophical and upper ontologies are therefore out of scope.

1.3 Problem Statement

BIM, is now mainly understood as a methodology to design, construct and maintain facilities using shared information assets with latest software tools and services in a more collaborative environment.

A commonly accepted definition is:

- *Building Information Modelling (BIM) is a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition. A basic premise of BIM is collaboration by different stakeholders at different phases of the life cycle of a facility to insert, extract, update or modify information in the BIM to support and reflect the roles of that stakeholder.* (NBIMS[2]).

The “shared knowledge resource for information about a facility” and in particular the “shared knowledge resource about energy-efficient hospitals” will therefore set the maximum boundaries for developing and using ontologies in Streamer. It is, however, still a very comprehensive and large boundary. Thus, the overall focus still needs to be limited.

One particular aspect of applying BIM is to focus on the “I” in B”I”M - the **information**. BIM is essentially **information management** for construction and operation projects. Information has to be managed through the whole life-cycle. In that life-cycle the same piece of information (e.g. the thermal performance of the facade or the total u-value of the roof plate, etc.) is often created several times independently by several participants using their own software tools. Confusion arises not only by re-entering of the values (where also the errors occur here), but also by different naming conventions or by different degrees of certainty of the provided values.

The management of the information thereby need to include:

[2] National BIM Standard – United States. National Building Information Model Standard Project Committee, http://www.nationalbimstandard.org/faq.php#faq1 (accessed: 2014-01-20)

1. *Stating the information requirement specific to either a life-cycle phase, or to a particular task* (what has to be delivered, by whom, when and in which level of detail or certainty),
2. *Creating the information* (when creating the information as property for a model element in a BIM authoring tool, which template is used, how is it named, does it comply with the information requirement),
3. *Comparing the information* (e.g. compare the “as required” room areas within the space program with the “as designed” area values of the design alternatives created by the designer),
4. *Validating the information deliveries* (e.g. make an automatic completeness check, that the BIM data within the virtual building model deliver by the architect or engineer has all required properties with values within an acceptable value range),
5. *Quality checks as extended validation services* (e.g. checking against building code or other design and engineering rules),
6. *Exchanging the rich information models between project participants in open standards* (such as IFC and CityGML) to prevent re-entering of information.

These problem statements lead to the following challenges that can be expressed by research questions.

1.4 Challenges

Potential user related questions need to be answered:

- What are the benefits to hospital clients and other stakeholders, if BIM / GIS with proper information management is used in new design and retrofitting of hospitals?
- Is there a common understanding among hospital clients, designers and contractors about the purpose and need for BIM information requirement management?
- Is it ‘Level of Detail’, or ‘Level of Development’, or ‘Level of Definition’, or how are BIM information requirements stated and communicated between hospital clients and designers or contractors?
- Why are current techniques to develop BIM guidelines insufficient – or is this deemed to be sufficient if stated using conventional spreadsheets (as in Excel)?

Potential technology related questions need to be answered:

- Can semantic web technology and languages, such as OWL, be efficiently used to support BIM based information management, are those technologies robust enough, can it be proven to be realistic in time and budget?
- Can existing solutions and techniques, such as the IFC and cityGML, be integrated with results from ontology works (in particular semantic web technologies) without reinventing the wheel?
- Can emerging technologies to formalize BIM data delivery and validation, such as mvdXML, be used and enhanced, and how could it benefit from semantic web technologies?
- Can BIM validation and rule checking be enhanced? How does it perform compared with existing solutions, such as Solibri Model Checker, BIMserver.org, etc.? Can those tools be enhanced and integrated into such a solution?

1.5 Followed Approach

The development plan for T5.1 include (see Figure 1):

- Analysis of the state-of-the-art in ontological research and neighbourhood developments (information modelling and standardization, rule-based assistance and validation),

- Setting real-life end-user scenarios for applying such ontologies to energy-efficient hospital design and retrofitting,
- Defining the specification for applying / enhancing existing open standardization frameworks IFC and CityGML to satisfy the end-user scenarios by utilizing results from the ontology work
- Implementing a prototype to validate the approach and to be used within the Streamer demonstrators

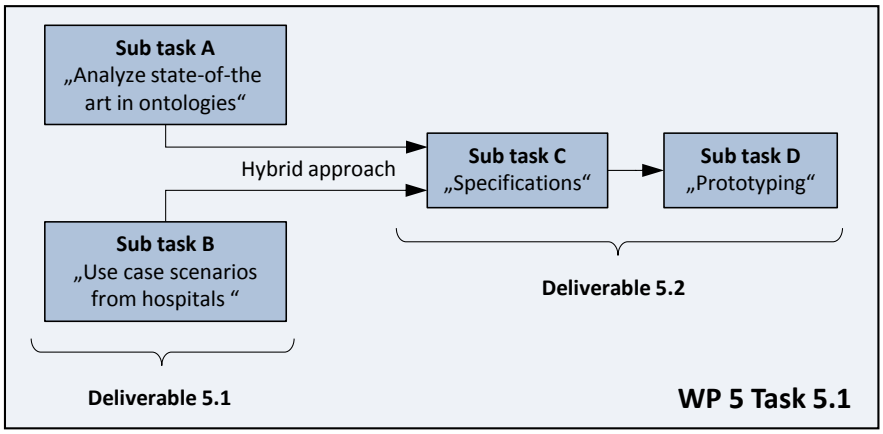


Figure 1: Structure of T5.1 (cross-links to other tasks and work packages are not shown)

The previous deliverable D5.1 covers the state of the art analysis and the identification of relevant end user scenarios and recommends specifications and existing early prototypes. This deliverable D5.2 on “Semantic Web based PMO (Product Modelling Ontology)” will focus on the chosen specifications and the actual prototyping of solutions. It has been concluded in the state-of-the art analysis that a hybrid approach shall be followed that reuses and extends current BIM/GIS developments. A main challenge that has been identified is information management that is checking if provided information fulfils expected requirements and constraints. It is a very knowledge intensive task, still mostly done manually. We also decided to follow the IDM/MVD methodology suggested by buildingSMART to be in line with ongoing standardization efforts. The two sub tasks that are in scope of this deliverable include:

Subtask C: Specifications

The specific challenge in developing an IDM/MVD for model checking is to capture requirements in a semi-formal, but still flexible and user-friendly way that can be linked to a data structure like IFC. Ideally, there are templates that can be configured to represent data requirements as outlined in the D5.1. The specific challenges for the specification subtask are:

- identify typical data requirements – to be identified and extracted from other deliverables
- capture the data requirements in a semi-formal (re-usable) structure (requirements ontology)
- provide mapping definitions to IFC based on mvdXML templates
- configure those definitions to exchange requirements being relevant for specific tasks

Following this solution approach we also expect to identify shortcomings for the applicability of mvdXML. This may relate to the expressiveness and clarity of the used rule grammar, the configuration approach and the level of reusability. This may lead to extension approaches for mvdXML to be discussed within buildingSMART.

Subtask D: Prototyping

Prototypical solutions shall be implemented to support the definition, management and use of developed checking specifications. Main developments activities are related to the following features:

- support requirements capturing that is to be exported to mvdXML to enable model checking
- improve and extend the mvdXML checking possibilities
- provide reporting functionality about identified issues using the BCF format

1.6 Organisation of the Deliverable

This chapter gives a short introduction to task 5.1. Content has been taken mainly from the Description of Work and our last deliverable D5.1. It was updated and shortened to focus on work presented in the D5.2.

Chapter 2 introduces the STREAMER framework for information management and quality control. Based on the state of the art review and decision taken in earlier it provides a high-level overview about the proposed approach. It takes into account results of other work packages and shows how this work fits into the overall research of the STREAMER project. It also introduces the main work- and data flow as well as important definitions related to model checking in order to clarify the scope of developments. Chapter 3 provides technical details about checking of exchange requirements as proposed and developed in this task. It is focusing on mvdXML as a neutral and standardized solution for BIM information management. Chapter 4 describes the captured knowledge that is relevant for checking exchange requirements for the processes described in chapter 2. It also shows the differences between the examples discussed in WP7, which are mainly caused by the use of different tools and, in case of retrofitting scenario, reuse of existing building data. Chapter 5 describes the prototype implementations. They not only support model checking but also the definition of exchange requirements as wells as the management of identified issues. Chapter 6 gives a conclusion of the presented work. Technical details of the developed checking specification is attached in the appendix section.

1.7 Relationship to other work in Streamer

There are several relationships to other tasks and work packages. Most important relationships are shortly mentioned here. More details are given in the next chapters.

WP1 and WP2 provide input to our exchange requirements from an engineering point of view, whereas WP4 defines the boundaries for collaboration and data management, in particular the relevant work- and data flow (see chapter 2). They gave input and feedback from domain experts who shall use our methods and tools. WP5 and WP6 are dealing with technical specifications and prototype developments. T5.2 develops solutions for shared data management and communication, and T5.3 is encoding design knowledge. They are an essential part of the overall framework for BIM information management and quality control. WP6, namely the Early Design Configurator is also implementing checking solutions for exchange requirements, which partially overlaps with the generic and neutral mvdXML-based solution developed in T5.1 (see chapter 5). WP7 shows project-specific boundaries, mainly coming from used tools and supported exchange formats and, in case of retrofitting available building data (see chapter 4).

1.8 Changes to the DoW

Based on our state-of-the art analysis carried out in the beginning of task 5.1 it was decided to adjust the way how to achieve described goals. A major change has been the decision to reject the idea to develop a Product Modelling Ontology using Semantic Web technologies. The reasons are explained in chapter 3.1 and the previous deliverable 5.1. This change is reflected in the extension to the original title of the deliverable.

Also, it became obvious that there is a strong need to clarify the connections to developments from other work packages. This was leading to the decision to work on a high-level overview about the data management framework as presented in chapter 2. Accordingly, the reader should be aware that the content presented in this deliverable went through a revision and has changed compared to the original DoW.

2. STREAMER Framework for Information Management and Quality Control

BIM information management and quality control have been identified as main challenges for task 5.1 activities. By nature, this task is influenced not only by technical but also by organisational, regulatory, contractual and other aspects. Dealing with those aspects is therefore not only targeting technical questions such as how to encode relevant knowledge as being in focus of this task but must also consider how to identify and maintain that knowledge in order to be transparent and maintainable. This chapter presents the STREAMER framework for BIM information management and quality control that has been developed taking into account existing tools and standards as well as the insights and knowledge gathered in other STREAMER work packages. It also explains why we have chosen to follow a hybrid or layered approach instead of developing a monolithic, highly specialized new STREAMER Prodcut Modelling Ontology.

2.1 The big picture and overall workflow

The STREAMER project is based on the concept of Building Information Modelling that is intended to enable better collaboration throughout the whole lifecycle of a building. It means to use a virtual representation of the building that contains all relevant information in an agreed data format. But it not only enables to share and reuse information but also to check if contributions coming from different domain experts fit together and meet the client and other requirements. However, the BIM does not have to be complete or consistent all the time as there will be intermediate design stages to test design alternatives for instance focusing on a specific aspect like energy consumption, user comfort, life-cycle costs etc. In order to find the best solution for the building the design activities have to be prioritized, coordinated and controlled. The design and decisions making processes for new and refurbished hospitals are discussed in WP1 (see D1.4 for retrofitting scenarios), WP2 (D2.3 & 2.6 for decisions matrices regarding MEP, envelope and configuration solutions) and WP4 (D4.1 for collaboration and information management) of the STREAMER project.

While WP3 is analysing relevant KPIs and the interaction between strategic, tactical and operational level activities WP4 is focusing on workflow aspects including a discussion of involved stakeholders and their responsibilities throughout the programming, design, construction and operational stages. BIM is supposed to change traditional processes and therefore have to be thoroughly redesigned, in particular when it comes to information exchange that in an optimal case enables a fully integrated design process that for instance is a prerequisite to check hundreds of design alternatives to find the best solution as needed in the Rotherham retrofitting example (see WP7).

The process definition in WP4 also considers available tools that in many cases are related to the scope or the type of the project. For example room programming of a hospital being characterized by many special healthcare requirements and expensive medical equipment is asking for different tools than a normal office building. STREAMER is focusing on hospitals and energy efficient design. The examples presented in WP7 show that there can be many different project setups. There are new and refurbished buildings, different regulations due to being located in different countries or another tool setup in particular for the room programming and energy simulation. In addition to the project-type based configuration, there are project specific settings that not necessarily are regarded as BIM data. Namely client requirements such as KPIs and the room programming as well as organisational aspects such as workflow definitions including schedules and responsibilities are to be mentioned here. Those settings are typically first action items in a project. Figure 2 and Figure 3 provide a high-level view on the settings being relevant for the project management as discussed in more detail in deliverables of WP3 and WP4 (D3.6, D4.1). Those settings then control the dataflow between participants relying on different kinds of ideally checkable requirement specifications that are discussed in the next section. The smallest piece in the data management activity is the execution of

a particular task by a particular stakeholder done with help of some software. A task typically imports some (BIM) data, is then processing the data using additional task specific knowledge including the experiences and decisions by the client and finally is exporting the processed data back to the BIM repository. This description of task fits to many activities in a project, even the initial project setup can be treated as tasks with input and output data.

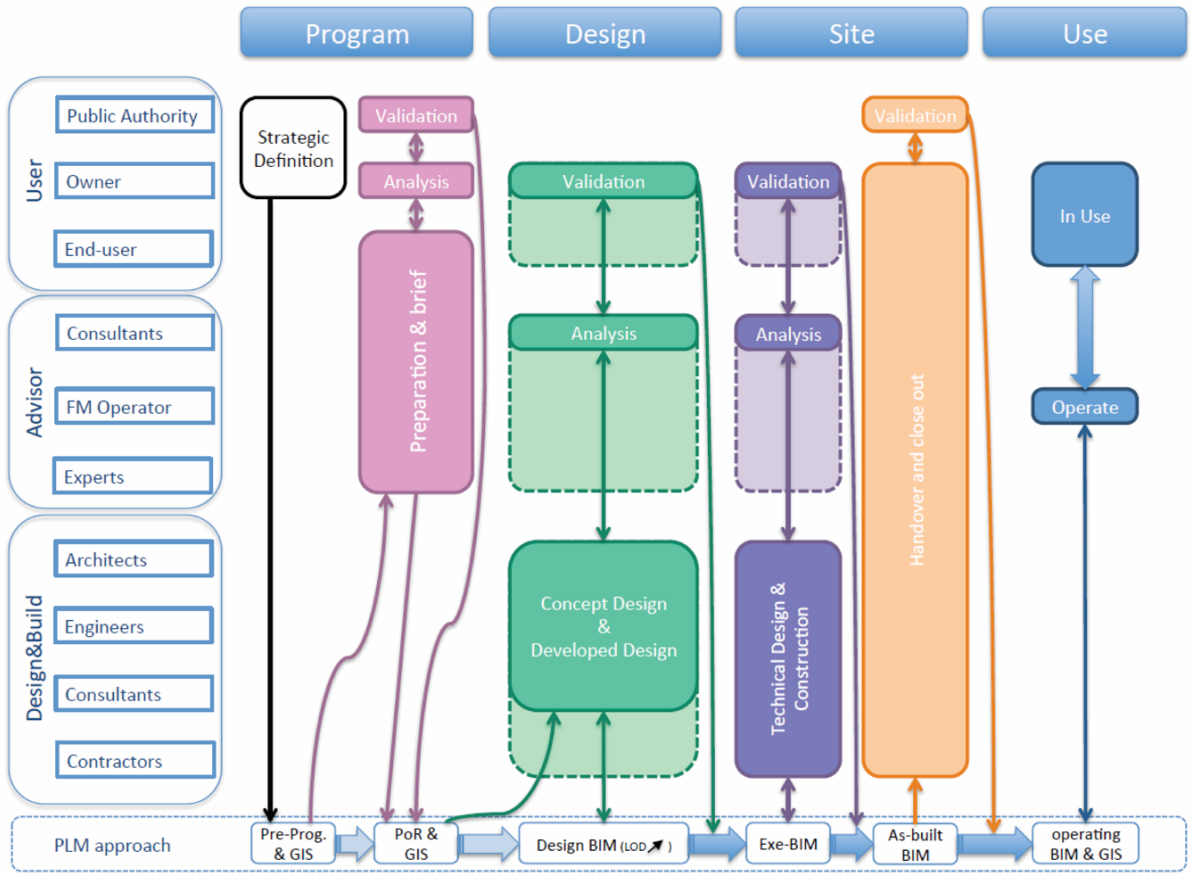


Figure 2: Schematic representation of the Streamer project stages and involved stakeholders in the decision-making process (see D4.1)

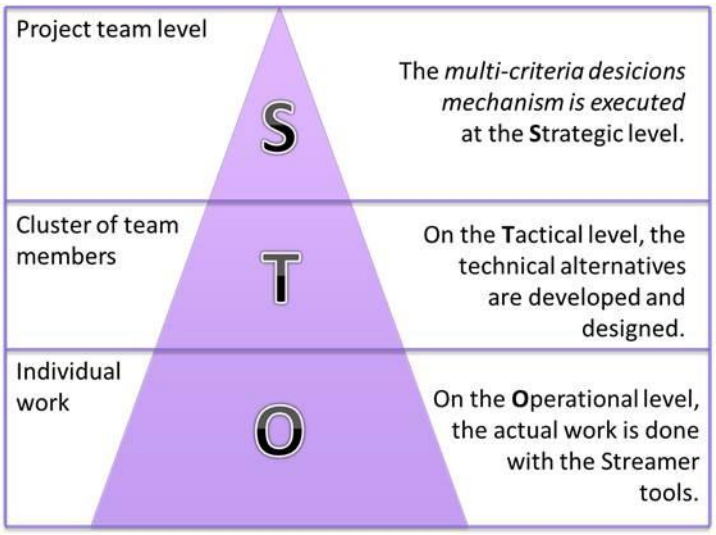


Figure 3: The hierarchy level of the STREAMER decision-making organization (see D3.6)

When focusing on tasks and its input and output data, the principle task sequence as shown in Figure 4 can be identified. It also shows the knowledge added by STREAMER as well as the data that is exchanged between those tasks. Information management means to control and check the produced data in order to make sure that all expected quality criteria are fulfilled. For instance if energy simulation results do not include the annual energy consumption, then a key figure is missing for the design evaluation task. Some data might be optional if it can be extracted from product catalogues or is given by default values. Such default data is for instance available in the CEN simulation tool, which applies window default properties for the U-value, solar energy transmittance and heat capacity that for instance are not exported by the Early Design Configurator. The different aspects for checking data are discussed in chapter 2.2.

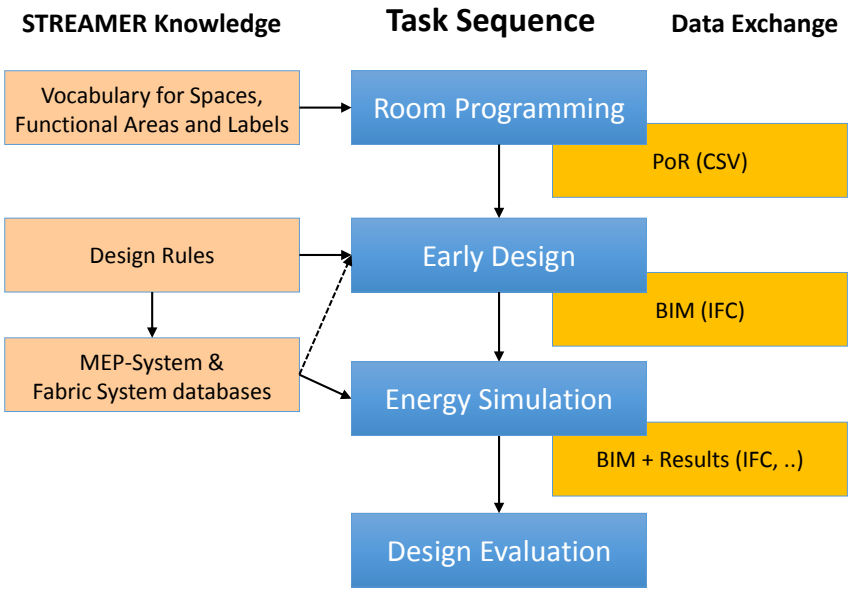


Figure 4: Principle task sequence for finding energy efficient design solutions. Further types of simulation like LCC simulation or comfort checks might be done in parallel to energy simulation to be able to evaluate all KPIs.

The task sequence for retrofit is similar but slightly different. It begins by federating all the available information sources (see Figure 5). Because of their diversity, a global dictionary is needed to map the attribute names into line, and a local (project) dictionary is needed to bring specific zone and system names into agreement.
Options for system and fabric upgrades are selected by expert users. The STREAMER systems catalogue is used to associate appropriate performance attributes.
The STREAMER, IFC and UK SBEM NCM vocabularies are used to prepare each option model for simulation and costing. The results are merged into the model for design evaluation.

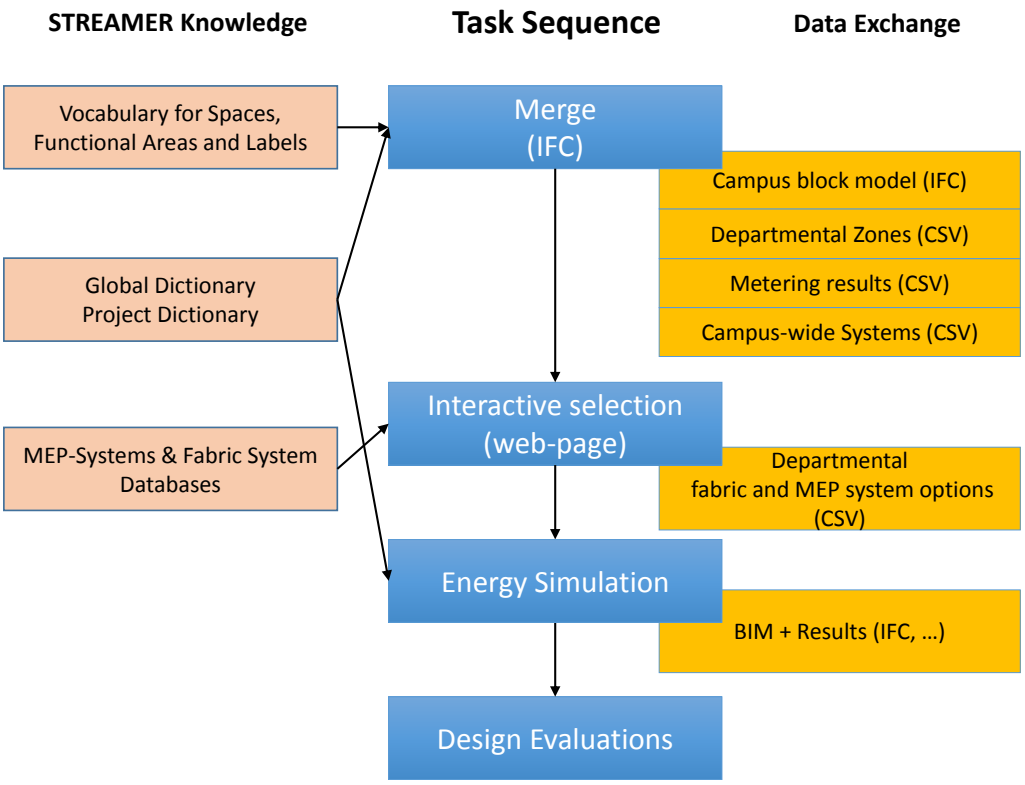


Figure 5: Task sequence for the Rotherham retrofit use case.

The principle task sequence is integrated in the decision-making process as shown in Figure 6. At the Strategic level there are decisions regarding the goals of the project, definition of the KPI's and its weight factors as well as the organization of the communication. Accordingly, "Room Programming" is seen as a strategic task that needs to be done in the beginning but may change when getting new insights by the "Design Evaluation" and may starts a new design iteration. On the sub-ordinate level, the Tactical level, the technical alternatives are developed and designed. It is based on design rules that for instance are defined for the layout of rooms in a hospital. The "Early Design" but also the selection of design rules, if project specific, are seen as tactical tasks. At the third and last level, the Operational level, the engineering analyses such as the energy calculation and the MEP solutions are designed. The results of those tasks provide the basis for decisions in the tactical and strategic level, which will collect, compare and weight the different results of a design alternative.

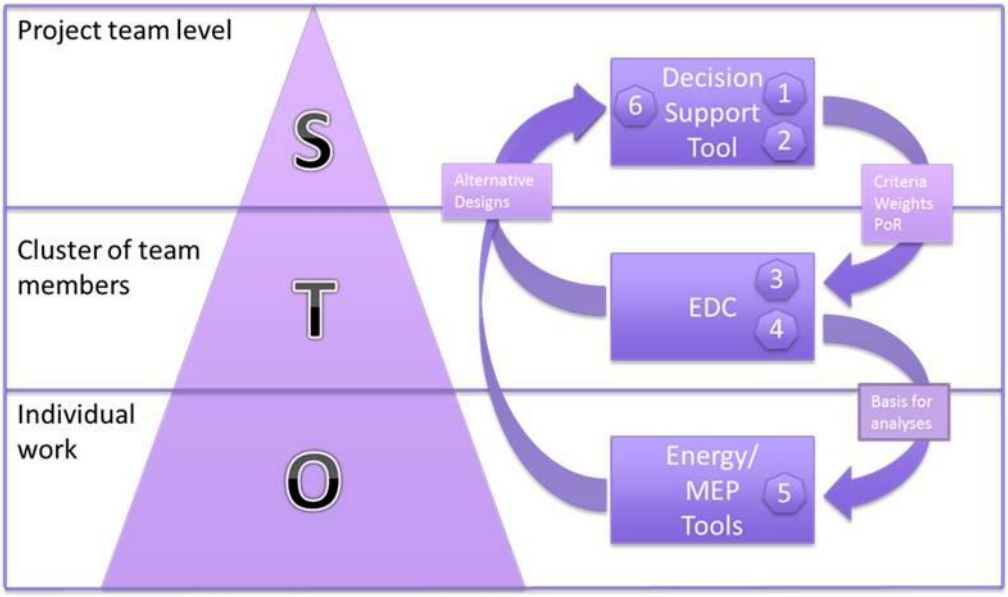


Figure 6: The iterative decision-making process and the inter-level connections (see D3.6)

A more technical view on the task sequence is shown Figure 7. It not only shows the used tools but also two options that can be used for central data management offering additional collaboration features. The process is the following:

- Step 1: The Briefbuilder tool enables to specify the program of requirements and is used in the Rjinstate example. It is a commercial tool that can export the PoR data to a CSV format (see D1.6). Alternatively, a normal spreadsheet tool like Excel can be used based on a STREAMER spreadsheet template (more details in chapter 4 of this deliverable).
- Step 2: The Early Design Configuration (EDC) developed in WP6 produces various building design proposals trying to find an optimal solution for the defined requirements. It generates an IFC file that is uploaded to the central data storage.
- Steps 3 and 4: Energy simulation is done with different tools, depending on the example hospital. The tools export the results either back to the IFC file or provide separate result sets using a proprietary data format.
- Step 5: At the end of the workflow the Decision Support Tool enables to compare results and to evaluate against the defined KPIs.

At any time the BIM Collaboration Format (BCF) can be used to trigger a discussion about any part in the BIM. It is an additional information on top of the IFC model that is based on issues, similar to issue tracking systems in software development. Each issue contains relevant details like a view point, screenshots, additional notes including tracked replies and also BIM snippets being able to share parts of the BIM. The BCF is an open XML file format developed by buildingSMART to support workflow communication in BIM processes. Any IFC viewer with BCF support like for instance eveBIM from the partner CSTB, the IFC Explorer from KIT and many more can read and write BCF and thus can be integrated into our scenario. Similar to BIM files and other documents the BCF issues can be management by the CMIS or by the CSTB document server. For this, an API needs to be instantiated from each client tool in order to communicate with the server. Additionally, the CSTB document server offers a specific function for BCF support that allows to store the link between IFC files and their corresponding BCF annotations.

While BCF and the shown infrastructure and tools enable to discuss and resolve issues the crucial questions is how to specify and control the quality of exchanged information between participants and the different levels. To answer this question is the main topic of this deliverable.

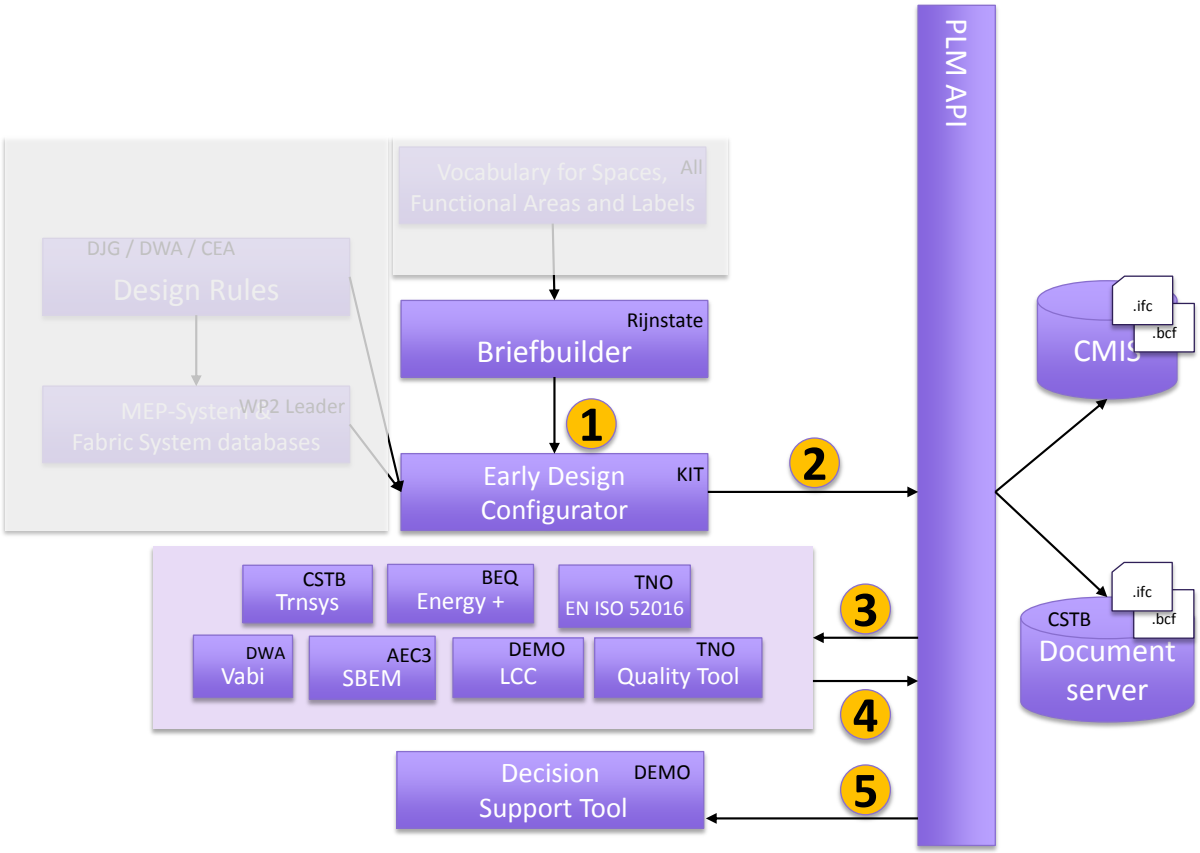


Figure 7: Developed prototypes and used commercial tools in the principle task sequence and data flow showing the use of the central data management environment through the PLM-API

2.2 Types of model checking and the proposed layered approach

Model checking is a key functionality for BIM information management. It enables to identify missing or wrong data. Ideally, model checking is done systematically whenever data is imported and exported to ensure the expected level of quality. Additionally, it has to be an automatic and reliable process in order to be able to deal with the amounts of data shared with BIM, in particular if dealing with complex buildings like hospitals. The principle of model checking is simple: a BIM data set is checked against a set of rules that must be satisfied. If a rule is violated, then an error message is created that for instance could be reported using BCF issues as presented in the previous chapter. While the principle of model checking is clear misunderstanding exists about what is checked in that process.

In order to reduce misunderstanding about model checking a layered approach is suggested that differentiates between the following types. Please note that the differentiation criteria is not always clear from a semantic point of view so that in some cases the decision is simply based on the fact where or how checking rules are encoded and who is responsible for maintaining the rule set.

- **Syntactical compliance to the file format**:

  This is the lowest level of model checks and is based on the file serialization format. Examples are for instance the STEP physical file format (SPFF by ISO 10303-21) that is mainly used by IFC, or the eXtensible

Markup Language (XML by W3C) used CityGML, gbXML and many other recently developed standards. Such errors are typically reported by the import software may failing to parse the file due to a wrong file structure, wrong references, incorrect escape sequence etc.

This level of model checks is typically not recognized by the user as the tools normally export files with a correct syntax.

- **Structural compliance to the data format**:

  The data file not only has to follow the used syntax but must also follow the structure defined by the underlying data schema. Each data schema like for instance IFC or CityGML define a set of entities, attributes and references that define a structure representing agreed elements that can be used for data exchange. For instance a wall is represented in IFC by an IfcWall entity and can have a given set of information like name, description, placement, geometry etc. A data schema can contain a lot of checks, not only against the use of correct entity and property names, but also the cardinality of references (e.g. setting if something is mandatory) or simple consistency checks. Structural errors are typically avoided by the file extension like *.ifc, but may pop-up if incompatible versions are used (e.g. IFC2x3 instead of IFC4). Such kinds of checking rules are defined and maintained by the standardization body and will normally change with a new release.

- **Data exchange requirements for processes[3]:**

  A BIM data structure like IFC is per se neutral to processes because it should be applicable for different data sharing scenarios. Accordingly, mandatory data is restricted to the absolute minimum by the data schema. It means that if an IFC data file is compliant with the data schema it not necessarily contains the data that is required for a particular process. This needs to be defined as a so called Exchange Requirement that is part of an Information Delivery Manual (IDM) and thus linked to processes. It defines what data has to be delivered for a particular processes. For thermal analysis it could be defined that each space must have space boundaries, occupancy profile information, heat gains, etc. If that data is missing, the thermal behaviour of the building cannot be analysed, or values may be defaulted or must be collected first. An Exchange Requirement may also restrict allowed ranges of values or can include implementer agreements that clarify the use the data schema, but it does not contain consistency checks that require further analysis of the data. Checking that all space boundaries of a space are closed and have the right orientation is not part of an Exchange Requirement. An Exchange Requirement for IFC-based data exchange can be encoded in the mvdXML format, which is an open standard defined by buildingSMART.

  Consistency checks:

  There are many examples of fundamental inconsistencies that can happen in a BIM model. In some cases like clash detection they might been seen as an own task for instance to coordination different domains, in particular if user interaction is necessary. There are other cases where data is not properly updated after design changes due to having redundancies in the data structure, or maybe there are wrong settings for instance in the structural analysis system that would lead to an undetermined behaviour of the load transfer. Thus, consistency checks require deep engineering knowledge and often require more complex analysis than checking data requirements. They are typically implemented in the used design tools and is something that is done as part of the task.

[3] or: completeness of model

- **Design rules**:

  Design rules are examples of recommendations that may be prioritized or overruled at some point. Other recommendations may come from specialist experts, insurers or best practice documents. Other examples include environmental assessment schemes where points are offered and then awarded for selected aspects, and those points contribute to a weighted (balanced) scorecard. The final score is then graded semantically, for example from excellent down to poor. Examples of design rules are discussed in task 5.3 and WP6. They are implemented in the Knowledge Editor and the Early Design Configurator. They are based on accepted rules for good design for instance based on functional requirements and are dealing with all aspects of a building. They can include personal preferences and specific expert knowledge. Design rules are typically defined for a type of buildings defining the applicability of the rules. They should influence design decisions and have an impact to the KPIs. Accordingly, they are typically applied when working on the design. When submitting the design proposal (export the result of a task) the domain expert is responsible to make sure that all design rules of his domain are fulfilled.

- **Compliance with regulations:**

  Regulations are similar to design rules, but have a legal background and may depend on the location of the building, jurisdiction and date. There are for instance country specific regulations for fire escape routes. Compliance with regulations is checked by the responsible authority in order to get the necessary approval. It is of course important to make sure that all regulations are fulfilled before submitting the proposal to the checking authority. Several efforts have been undertaken to encode regulations as checkable rules for BIM data. However, there are still a number of research questions how to deal and manage regulations in an efficient and transparent way.

  Regulations, requirements and Recommendations are the subject of a buildingSMART International paper being developed for future standardisation. These types of checking must handle complex applicability, selection and exceptions alongside to qualify the requirements. So far three layers have been identified, mark-up which annotates the original documents with referencing and grammatical roles, interpreted layer, where formal logical rules are documented, and operational layer where the rules are presented in a form that universal rule engines can process. mvdXML (see also chapter 3.3) is effectively an interpreted layer which is operable by specialised applications.

- **Check against the Program of Requirements**:

  The Program of Requirements is an example of project specific requirements that are defined in the beginning of the project by the client. It means that the generic functionality for comparing client requirements with the actual design can be offered by tools, but the requirements representing the “checking rules” need to be defined and managed within the project. Defining the PoR can thus be seen as any other task in the project producing some “BIM data” representing client needs. As for any other data it needs to be checked if all information is according to the checking rules for PoR data. Ideally, a PoR tool such as dRofus or Briefbuilder enables to compare client needs with the actual design solution.

The focus of the task 5.1 is on managing and checking data exchange requirements based on the mvdXML format. It is not yet done in practice and can significantly improve data exchange between participants.

2.3 How to capture checking knowledge

The principle of model checking is clear and simple. However, there are a couple of challenges for defining and maintaining the checking rules. This chapter is discussing criteria for capturing checking knowledge.

- **Openness and transparency:**
  Model checking might be offered as a black box without the possibility to verify the used rule set. If not properly documented, there is a risk that a rule set may not be applicable to a give design and thus is producing wrong results. Ideally, the checking knowledge is open so that it can be reviewed and may even be adjusted by the users. It also enables to use different checking engines may offering different features for checking or reviewing the model. An argument against openness is to protect specific expert knowledge because it is part of their business model. Accordingly, such knowledge may not be available free of charge.
  A key element of openness is the need for acceptability, which depends mainly on a clear connection between the source requirements, usually a written document, and the results. The intervention of derivative interpretations and re-keying reduces credibility and robustness.

- **Usability**:
  Encoding of checking rules can easily lead to complex expressions or programming code. Such code is rather difficult to read for domain experts who have the knowledge about checking rules. They are typically not familiar with used expressions and the details of the underlying data structure. Ideally, the code can be reviewed on a more abstract level or for instance is based on a Domain Specific Language (DSL) that fits to the terms and scope of the domain. The design rules developed in WP6 are a good example of using typical adjacent relationships of spaces to encode the rule or checking knowledge. Each of those relationships is then implemented in some programming code but still easily configurable.
  A checking application must be able to express its results in terms of reasons for failure or as means to resolve that failure. In many cases there may be several alternative resolutions of varying difficulty (improve the insulation, repurpose the space, reclassify the building, and relocate the project).

- **Expressiveness**:
  The used rule language may not allow to specify a particular rule due to missing expressions. If for instance mathematical functions are not supported then it is not possible to do rule checking that require the calculation of values. Thus, the main question when choosing the rule language is whether it can deal with all checking rules or not.

- **Decidability**:
  Statements like for instance available in the OWL-Full language may not be fully decidable thus leading to issues when it comes to model checking. Somebody may just want to express some facts about the BIM without dealing with its implementation. However, if a result is expected a proper rule must be defined, or such features and rules must be ignored by the rule engine. A critical factor is the ability of applications to handle True/False or True/unknown/False facts. Without the ability to handle unknowns and track their significance, rule systems become information hungry and unwieldy.

- **Performance**:
  Depending on the kind of check, for instance if done when importing a BIM file, checking results might be

needed in short time. In such cases the execution time becomes important as it otherwise would slow down the work of the users. The most critical performance parameters besides the complexity of the rules are the size of the model and the number of rules.

- **Maintainability**:
  If rules or regulations change the implementation of the rule must be adjusted. If the code is well documented and structured it is easier to identify the relevant code parts and to make necessary changes. It is a typical software maintenance issue and thus should reuse principles and tools that are successfully applied in that area. For instance, a modular software design and the use of a code versioning system as well as issue tracking solutions are state-of-the art in the software development. An important decision when starting to encode checking rules is the selection of a proper rule language that for instance supports a modular design and the re-use of code parts. Another aspect is systematic testing, which is typically based on a well-defined set of test cases that is used to verify the code. This for instance enables to easily identify negative side-effects when changing code parts.

- **Scalability**:
  Scalability describes the behaviour of model checking when new rules are added or the model size increases. A solution may work properly for small BIM models, but fails for bigger models. Or a larger number of rules becomes unmanageable. Scalability also affects the preparation of norms: given the rapid evolution of norms such as regulations, requirements and recommendations, it must be possible to absorb new material in a timely and efficient process.

3. Specifications for capturing quality control knowledge

This chapter is discussing options to capture checking knowledge. As outlined in chapter 2.2 the focus in task 5.1 is on checking exchange requirements using mvdXML. The decision for using mvdXML is based on the agreement to use the open and neutral IFC format for BIM data exchange. IFC already offers an ecosystem of tools that support design, construction and maintenance processes. Thus, they can easily be integrated into the STREAMER design process without further data conversion processes. IFC data sets, if available, can also be reused in the retrofitting scenario. Other options like using the IFC modelling language EXPRESS or Semantic Web-based solutions like OWL or RuleML have been disregarded based on our analysis results. However, this chapter will provide further details of the selected solutions based on a layered model checking approach compared to a more monolithic Product Modelling Ontology (PMO).

The chapter starts with describing the limitations of developing a PMO and then introduces to the knowledge that is captured to support the STREAMER process, namely the Exchange Requirements defined by domain experts, its representation as mvdXML as well as the design rules used by the Early Design Configurator (EDC).

3.1 Knowledge encoded in an ontology

The idea of a Product Modelling Ontology (PMO) was to provide a means to represent industry product models with all their constraints about their assembly, their properties, their relations, etc., under the form of an extended ontology. This idea dates back to the SWOP project (Böhms et al. 2008) that was proposing an OWL-based solution to encode that knowledge. It was used and demonstrated as a semantic layer containing all the necessary knowledge to generate valid design solutions and alternatives. Optimization techniques were then used to identify the “best” options according to the selected objectives (see also WP6 and EDC developments). This is in line with the overall goals of the STREAMER project.

After the state-of-the art review and further testing with OWL-based solutions (see D5.1) it was decided to follow a hybrid approach, which is not only based on established standards and tools but also follows a more layered approach based on the types of model checking as described in chapter 2.2. This solution is expected to be more flexible and robust in particular for industrial needs, and is also in line with ongoing standardization efforts that are actively supported by the STREAMER project. The followed approach is still based on file-based data exchange, which means to use the STEP physical file format for IFC and the XML file format for other data structures like CityGML. This enables to easily integrate available tools into the STREAMER workflow because no data conversion or re-implementation is necessary. Both, the data standards and related file formats cover the knowledge of the first two basic checking layers. As file-based data exchange remains an important way of communication the quality control of agreed data flows was identified as a fundamental issue to improve collaboration between stakeholders. This hasn’t been a topic of PMO. The knowledge encoded in a PMO is mainly supporting individual design steps in order to end-up with a consistent and good design. In STREAMER this layer is mainly handled by the design rules developed in T5.3 and WP6. Additionally, capturing and maintaining relevant knowledge has been identified as a fundamental topic that is supported by the various STREAMER developments, namely the BIM-Q tool (see chapter 5.1) and the Design Rule Editor (D6.1). Also, expressiveness of standard OWL, SPARQL, SWRL and available tools was found to be too limited to semantically define design rules for instance requiring geometric calculations.

3.2 Exchange requirements

According to the Information Delivery Manual (IDM, ISO 29481-1) and Model View Definition (MVD) methodology, the specification work follows subsequent steps and involves different stakeholders starting with a high-level view on the business processes and goes down to software implementation details. The result of each step is an agreement or technical specification that forms the basis for further communication and detailing.

Task 5.1 is focusing on the definition of exchange requirements. Accordingly, relevant processes, involved actors and the data flow as presented in Figure 4 of chapter 2.1 are available as a reference. An exchange requirement is for instance the data that must be generated in the process called *Room Programming*[4]. It is defined by domain experts who have to describe what information must be defined in that process. They also have to agree on terms, their meaning and the expected structure of required data. Lean thinking places the onus on the receiver to specify their requirements, and the sender to support this requirement as far as they can. In case of PoR the task results in a set of room types that are classified by criteria such as comfort, safety, hygiene class, accessibility and others (Di Giulio 2015). For each of those classification criteria allowed ranges of values with their meaning have to be defined. For instance, a room classified as “A4” means that it should be accessible for staff only. In many cases existing classification systems can reused as a reference (for further details see chapter 4).

The structure of defined requirements may fit to other processes. Therefore, it is reasonable and recommended to harmonize and reuse such specifications. If room type information is not only needed for space layout but also for energy estimation it should be linked as a requirement to both processes. Traditionally, the main purpose of this step is to prepare implementation of software interfaces, which means to translate the terms of domain experts to a data structure like IFC. This step is done by modelling experts who are familiar with the data structure. For IFC-based MVD developments it means to switch to the ifcDoc tool that enables to work on an mvdXML specification, which will lose the link to the exchange requirements defined by the domain expert. The approach suggested by the STREAMER project is to capture exchange requirements and the mvdXML configuration in the same tool (see chapter 5.1).

3.3 mvdXML

With support of the STREAMER project the mvdXML 1.1 specification was published in spring 2016 (Chipman et al.). It enables to specify exchange requirements of a Model View Definition (MVD). Besides a couple of minor improvements and simplifications compared to the initial release 1.0 the most notable change is the enhanced capability for model checking. This feature of mvdXML is becoming more and more interesting. However, the focus is still on MVD documentation purposes for instance for creating the HTML documentation of the Design Transfer and Reference View of IFC4 (see Figure 8). It is therefore an important specification document for software implementation. mvdXML can also be used for generating an IFC subset schema for checking structural compliance (see chapter 2.2) and for data filtering and querying as required for the interaction with a PLM system (see task 5.2).

[4] Or Program of Requirements (PoR)

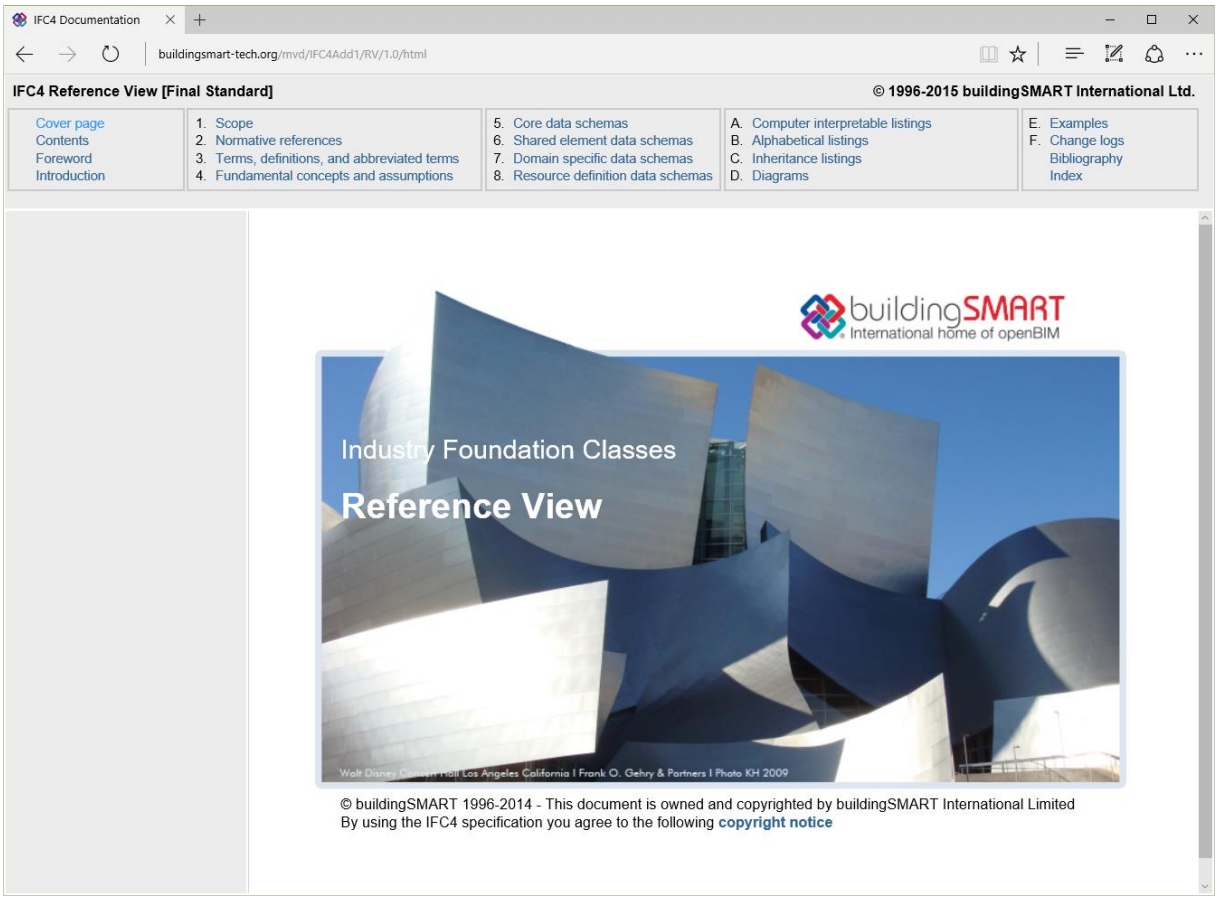


Figure 8: HTML documentation for the Reference View of IFC4 generated with help of the ifcDoc tool and mvdXML

Depending on the usage of mvdXML (documentation or model checking) the definition has a different focus. However, the definition of an MVD is always similar. Main elements of each MVD are:

- **ModelView**:
  one or more of those elements are normally included in an mvdXML file. It is part of the *View* element and is the main container for exchange requirements and root concepts.
- **ExchangeRequirement**:
  represents the data that is relevant for a use case, either for import, export or both.
- **ConceptRoot**:
  represents a class of objects for which the same constraints apply. They are normally linked to entities that are derived from *IfcRoot*, i.e. being a main testable element of an IFC model.
- **Concept**:
  is part of a root concept and defines a constraint on applicable objects and how it is used in exchange requirements.
- **ConceptTemplate**:
  defines a unit of functionality that is used and configured by *ConceptRoot* and *Concept* elements. It is a selection and basic configuration of IFC definitions that are required to implement a specific functionality such as support of property sets, material layer definition or more complex data like BREP geometry.

Each of those elements is able to carry additional meta-data and descriptive text including multilingual support.

Concept templates and their configuration

An important design feature of mvdXML is to reduce the maintenance effort, which is based on the use of configurable concept templates. A concept template defines one or more applicable entities and includes a set of rules that each specifies a sub graph of instantiable attributes. Such sub graph is defined by attribute and entity rules and always starts with an attribute of the applicable entity.

The concept template shown in the code example below is defined for all instances of *IfcRoot* entities and contains two rules for the attributes *Name* and *Description*. Both rules define an additional (optional) rule identifier (*RuleID*), which is a unique name used for further configuration. The code example also shows that an *AttributeRule* is followed by (one or more) *EntityRule* that expand the sub graph.

The rule identifier is later used as a parameter in a logical expression to check existence, values, types, size of sets or uniqueness. Accordingly, above shown example enables to configure both attributes, for instance to check for a specific name or existence of a description. However, logical expressions in mvdXML are limited in their expressiveness in order to be as clear as possible both for definition and processing.

```
<ConceptTemplate uuid="c19ec186-9cfd-47fc-a4d4-9fb35008d04a" name="User Identity"
  applicableSchema="IFC4" applicableEntity="IfcRoot">
    <Definitions>
        <Definition>
             <Body><![CDATA[Code 020- ...]]></Body>
         </Definition>
    </Definitions>
    <Rules>
         <AttributeRule RuleID="Name" AttributeName="Name">
              <EntityRules>
                   <EntityRule EntityName="IfcLabel"/>
              </EntityRules>
         </AttributeRule>
         <AttributeRule RuleID="Description" AttributeName="Description">
              <EntityRules>
                   <EntityRule EntityName="IfcText" />
              </EntityRules>
         </AttributeRule>
    </Rules>
</ConceptTemplate>
```

Figure 9: Definition of a reusable ConceptTemplate for “Name” and “Description” attributes

Checking of exchange requirements

The principle for defining constraints is based on IF THEN statements. The IF-part is defined in *ConceptRoot* nodes and determines the applicability to instances. In general IF-THEN statements put the IF ‘ASE’ THEN R. Applicability narrows down the relevance of a rule. Selection expands it. Exceptions override the Requirement. The THEN-part is defined by Concept nodes and defines the constraints that shall be applied to all applicable instances. In addition to a “selection by type” (through the *applicableEntity* field) it is possible to define additional constraints. For instance if all load bearing walls shall be checked then all instances of *IfcWall* with a property *Pset_WallCommon.LoadBearing = TRUE* must be checked. Such additional constraints are defined in the *<Applicability>* section of *ConceptRoot*. The mvdXML snippet shown below is checking instances of *IfcBeam* with the Name “Beam-206”. It is configuring the concept template shown in the previous snippet.

```
<ConceptRoot uuid="00000035-0000-0000-2000-000000067001" name=" Beam-206"
  applicableRootEntity="IfcBeam">
    <Applicability>
        <Template ref="c19ec186-9cfd-47fc-a4d4-9fb35008d04a"/>
        <TemplateRules operator="and">
              <TemplateRule Parameters="Name[Value]='Beam-206'"/>
```

```
        </TemplateRules>
    </Applicability>
```

Figure 10: Configuration of "User Identity" for the selection of an IfcBeam instance.

The configuration of constraints works in a similar way; a concept refers to a concept template using its *uuid*. The <*Requirements*> section then defines the link to exchange requirements and its expected usage. The configuration of rule identifiers starts thereafter, which may be using nested statements logically combined by Boolean operators. The mvdXML snippet below shows the configuration of a mandatory space property where only the two values "A1" and "A3" are allowed.

```
<Concept uuid="00000003-0000-0000-0000-000000349910" name="Accessibility Labels">
    <Template ref="00000000-0000-0000-0001-000000000001"/>
    <Requirements>
         <Requirement applicability="import"
           exchangeRequirement="00000003-0000-0000-0000-000000000105"
           requirement="mandatory"/>
    </Requirements>
    <TemplateRules operator="and">
        <TemplateRules operator="or">
            <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND
                                      Property[Value]='AccessSecurity' AND
                                      Value[Value]='A1'"/>
            <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND
                                      Property[Value]='AccessSecurity' AND
                                      Value[Value]='A2'"/>
        </TemplateRules>
    </TemplateRules>
</Concept>
```

Figure 11: Definition of constraints for the "Accessibility Labels" defined by the PoR for spaces.

3.4 Design rules

The design rules developed in task 5.3 and WP6 are a way to translate and store different sources of knowledge (e.g. expert knowledge, culture preferences, climatic preferences, personal preferences, client preferences, building regulations or best practises) as relationships between two objects (see basic rule structure below). The rules are encoded as a XML file and are used by the Early Design Configurator to develop a room layout for the early design phase of a project. Accordingly, encoded design rules are actively used to generate design solutions whereas an exchange requirement as developed in WP5 is giving feedback if all required data is included in a particular design solution.

Object X with **Attribute Z** has **relation I** with **Object Y** with **Attribute Z**

Similar to mvdXML, design rules are captured in a separate file or database (independent from the tool). They are typically used to fulfil some design activity and therefore are not incorporated in the information exchange. Only dimensional and geographical relationships are bounded in the information exchange between tools. However, design rules can and should be re-used in other (hospital) projects. If design rules are available in an open format, they can be used as an additional evaluation resource in a later design phase, for instance to check the consistency of a design option regarding the original requirements. In addition to the EDC and similar to the xBIM tool discussed in chapter 5.2, the Design Validator developed in WP6 offers such neutral evaluation option that basically enables to rate a design solution.

4. Captured exchange knowledge

This chapter describes the requirements that need to be checked for the exchange scenarios defined in chapter 2.1, namely data coming from the processes of room programming, early design and energy simulation. It will also explain the tools that are used to define, consume and check the data.

4.1 Program of Requirements exchange requirement

The Label approach

The Program of Requirements (PoR) is developed to capture the requirements of a design. In the early design stage not all information is available so that STREAMER has developed the label methodology. It means that requirements are translated into more generic information containers: the labels. These labels represent semantic information and are used by different tools. Accordingly, the meaning of the labels must be formalised and has to be static throughout the design process.

The “comfort class” label for instance represents requirements for light, daylight, ventilation and design temperatures. This label is first used to describe the wishes from the client regarding those aspects. Secondly they are used as a way to produce a layout possibly by placing the rooms with daylight requirements next to the outer wall. A next step is an early energy simulation where the simulation tool uses the design temperatures of the “comfort class” label as an input for calculating the heating and cooling requirements. For all those different purposes the label (value) needs to represent the same semantic information and must be used in the same way in each use case. Therefore the labels are fixed and static throughout STREAMER.

One exception is suggested; that of an empty layer class. However, this label class is not yet included in the predefined structure or EDC. But in some designs a free label category, not used by other programs, is preferable. A realistic use of the “empty layer class” label is for instance when designing an Islamic health care district, where there is need for a men/women separation within the hospital. The empty label class can be assigned for this men/woman separation and used in design rules to group the rooms allowed for men only or women respectively. The labels represent this semantic information only on that specific project and therefore cannot be interpreted by other tools.

Required data

The PoR needs to contain the area of a room that is to be translated by the EDC into a physical object. The realised floor area of an object is often not the exact area given in the PoR, but is within a certain range. For the application of design rules as described in chapter 3.4 the naming of labels, room types and functional areas need to be consistent. The design rules refer to the attributes of a room described in the PoR. Accordingly, it is recommended to use the STREAMER language next to the labels also for the room types and functional areas (FA). Currently the room types and functional areas do not contain any other semantic information than a description. Therefore, it is possible that an experienced user could make customised names for FAs and room types, if this is also used in the design rules. This approach could be useful in cases that do not fit into the standard “jacket” of STREAMER.

Another semantic use of the room types is the mapping of default labels to default room types. For fast development and standardisation of the PoR, a default mapping is done for room types and labels in the D1.6. This default mapping can be used to fill in a PoR as for instance done in the Rijnstate example with help of the software tool Briefbuilder. For every room type default labels are given. Accordingly, when developing a PoR it is not needed anymore to fill in all requirements for every room. This doesn’t mean the labels do not need a check or that a change of settings is not possible. It is meant as a default setting that fits to most of the projects but not being a hard or generic requirement.

The current STREAMER PoR contains the following objects and restrictions, where the bold text are the column names that need to be used to be processed by the EDC (agreements will be reported in the deliverable D1.6):

1. **RoomName**: any text (string)
2. **RoomType**: list of allowed values (enumeration, see D1.6)
3. **Amount**: any integer value
4. **Area**: any real value
5. **FunctionalAreaType**: list of allowed values (enumeration, see D1.6)
6. **BouwcollegeLayer**: list of allowed values (enumeration), HF, H, O, I
7. **HygienicClass**: list of allowed values (enumeration), H1, H2, H3, H4, H5
8. **AccessSecurity**: list of allowed values (enumeration), A1, A2, A3, A4, A5
9. **UserProfile**: list of allowed values (enumeration), U1, U2, U3, U4
10. **Equipment**: list of allowed values (enumeration), EQ1, EQ2, EQ3, EQ4, EQ5, EQ6
11. **Construction**: list of allowed values (enumeration), C1, C2, C3, C4, C5, C6
12. **ComfortClass**: list of allowed values (enumeration), CT1, CT2, CT3, CT4, CT5, CT6, CT7, CT8

Definition and exchange of PoR

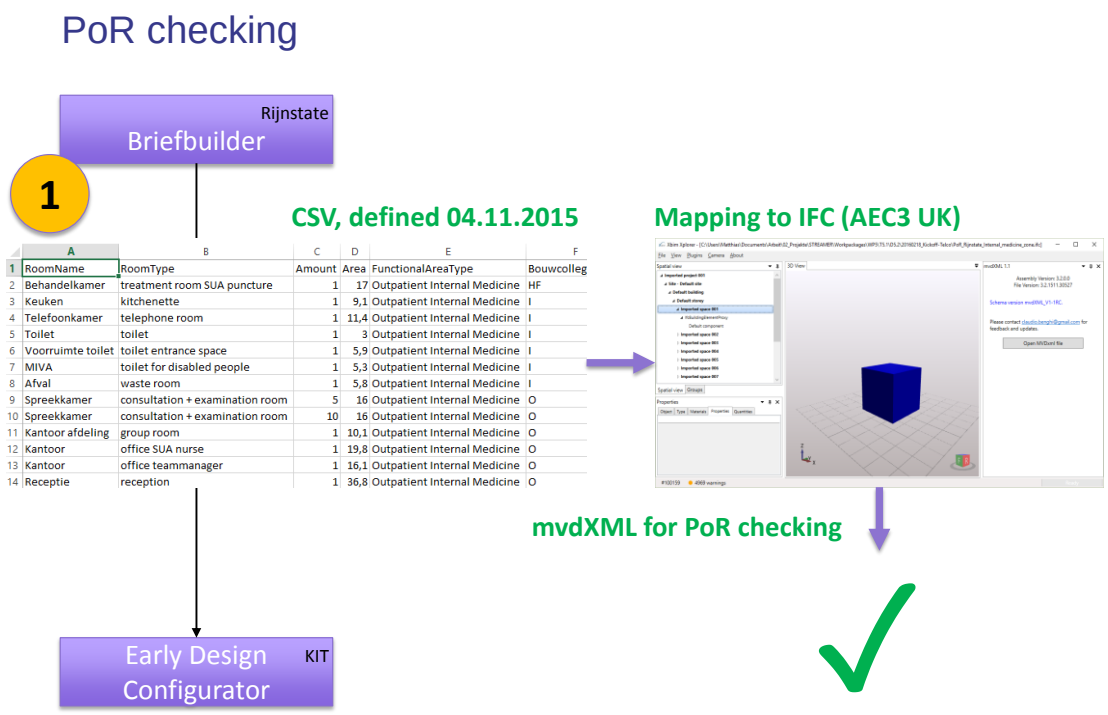
PoR data is captured in a table format. Accordingly, the Comma Separated Values format (CSV) is used to export and share the PoR data. Please note that in order to achieve syntactical compliance (see chapter 2.2) it is necessary to use a point as a decimal separation for any float value, in our case relevant for the required area of a room. If the CSV has a comma as cell separator and the amount is written with a comma, the decimals are processed in another cell according the importer. The “amount” requirement is expected to be an integer value and thus is not critical for the CSV export.

PoR data can be defined in different environments, for example Briefbuilder, dRofus, Excel, Google sheets or any other spreadsheet tool or even in a normal text editor. The main requirement is the use of consistent column names and values in order to be compliant with the agreed table structure. Additionally, the tool must enable to export to a CSV format. For tools like Briefbuilder it is possible to create export templates to meet the required structure of a STREAMER PoR. For clients who do not have access to Briefbuilder a Google sheet environment has been developed. In this template sheet default mappings of labels and rooms are also available. Next to the default setting, an internal library is used. The attribute “room name” has no predefined enumeration values and thus can be used as a project specific name (any text is allowed). However, it is also possible to use an internal library in the Google sheet environment to link the room name to a STREAMER standardised room type. If the room name matches a term in the internal library, then default label settings can be used. In such case it is only necessary to define the room name, amount, area and functional area.

PoR data checking

PoR model checking can be done either directly in the Early Design Configurator, which is using the PoR data to develop the room layout (see import checking in chapter 5.4), or by a separate and neutral model checking tool based on IFC and mvdXML, for instance the XBIM tool as described in chapter 5.2. While the EDC can import the CSV format directly, the mvdXML-based checking option needs to convert the CSV data to an IFC model first, which a general challenge is being also relevant in other data sharing scenarios. This checking scenario is shown in Figure 12 that in particular is relevant for checking the PoR data export by the client who may not own an EDC tool licence. A data export check that can be done by the client enables to avoid additional change requests if missing data is

later identified by the architect when trying to import the data for doing the early design configuration. Accordingly, completeness and consistency is ideally already checked on data export to avoid further delays due to incomplete data sets and additional change cycles.



	A	B	C	D	E	F
1	RoomName	RoomType	Amount	Area	FunctionalAreaType	Bouwcolleg
2	Behandelkamer	treatment room SUA puncture	1	17	Outpatient Internal Medicine	HF
3	Keuken	kitchenette	1	9,1	Outpatient Internal Medicine	I
4	Telefoonkamer	telephone room	1	11,4	Outpatient Internal Medicine	I
5	Toilet	toilet	1	3	Outpatient Internal Medicine	I
6	Voorruimte toilet	toilet entrance space	1	5,9	Outpatient Internal Medicine	I
7	MIVA	toilet for disabled people	1	5,3	Outpatient Internal Medicine	I
8	Afval	waste room	1	5,8	Outpatient Internal Medicine	I
9	Spreekkamer	consultation + examination room	5	16	Outpatient Internal Medicine	O
10	Spreekkamer	consultation + examination room	10	16	Outpatient Internal Medicine	O
11	Kantoor afdeling	group room	1	10,1	Outpatient Internal Medicine	O
12	Kantoor	office SUA nurse	1	19,8	Outpatient Internal Medicine	O
13	Kantoor	office teammanager	1	16,1	Outpatient Internal Medicine	O
14	Receptie	reception	1	36,8	Outpatient Internal Medicine	O

Figure 12: IFC- and mvdXML based model checking of PoR data.

Much of the design development information is today shared through informal spreadsheets. This is the case even if the information is developed using formal applications with well-structured databases. In our case the PoR was prepared using Briefbuilder and the information was shared as mentioned as a simple CSV file (Table 1).

Table 1: PoR headings and example row

RoomName	Receptie
RoomType	reception
Amount	1
Area	36.8
FunctionalAreaType	Outpatient Internal Medicine
BouwcollegeLayer	O
HygienicClass	H4
AccessSecurity	A1
UserProfile	U2
Equipment	EQ2
Construction	C1
ComfortClass	CT2

In order to make this information available for formal checking prior to incorporation into the design process, it is necessary to add the semantic meaning of the individual rows and columns. This was achieved through the use of

the AEC3 BimServices Transform1 utility (Nisbet 2010). This utility accepts IFC, XML, XHTML, text and spreadsheet inputs and apply a regular XSLT transformation with the result being output as IFC, XML, XHTML, text and spreadsheet as appropriate. For the current need, a library transformation “fromSpreadsheet” was applied to the PoR CSV. The semantic meaning of the rows is by default unknown. The transformation takes a single extra parameter ‘topic’ which identifies the semantic object represented by the rows. The choices include ‘project’, ‘site’, ‘building’, ‘storey’, ‘zone, ‘component’, ‘system’, ‘type’ or in this case ‘space’. The transformation then creates a complete IFC model with the minimum number of other objects necessary to give context for the objects. So by identifying the content as ‘space’, one of all the other object types are created, and one space object for each row found. All the expected relationships and attributes are also created, including Name, Description, ObjectType, and Owner History. Unique identifiers are assigned to the objects. The transformation can draw on a ‘local dictionary’ (Figure 13) which is maintained from any other project models available, to discover the preferred name, English or Dutch description and unique identifiers for the project, site, building and so on.

```
<concept type="object">
       <term context="IFC">IfcBuilding</term>
       <term context="local"> Rijnstate Hospital</term>
       <term context="global">2QkBWKvhH9_S6PYEmI9mJ0</term>
       <term context="en-GB">Rijnstate Hospital</term>
       <term context="du-NL"> Rijnstate Hospital </term>
</concept>
```

Figure 13: Fragment from the local dictionary.

The semantic meaning of the columns is by default unknown. Each field is mapped to a property grouped in a default property set ‘Default_SpaceProperties’. Each property is taken as a simple text property. However, the transformation makes use of a second ‘global dictionary’ which contains hints which can add value to the outcome by associating the column headers (in whatever language) to specific IFC attributes (Figure 14).

```
<concept type="property">
       <term context="BriefBuilder">Room type</term>
       <term context="PoR">RoomType</term>
       <term context="IFC">ObjectType</term>
       <term context="en-GB">Space or Component Type</term>
</concept>
```

Figure 14: Fragment from the global dictionary.

The global dictionary (Figure 15) can also hold pointers to the expected parent, for example a property set, any synonyms, and any expected values. In the STREAMER project many slight variations in these names were proposed before finally being agreed.

```
<concept type="property">
       <term context="BriefBuilder">7. Types of space</term>
       <term context="STREAMER">BouwcollegeLayer</term>
       <term context="IFC"
       parent="STREAMER_SpatialStructureLabels">
       BouwcollegeLayer<
       </term>
       <term context="en-GB">
       Four way classification of hospital spaces by activity
       </term>
</concept>
```

Figure 15: Fragment from the global dictionary.

By default the objects created are arranged spatially in a regular grid pattern appropriate to the topic. Sites, buildings and spaces are laid out on a horizontal plane (Figure 16), storeys in a vertical stack, and components in a 3D pattern to ensure that every component is visible from all sides.

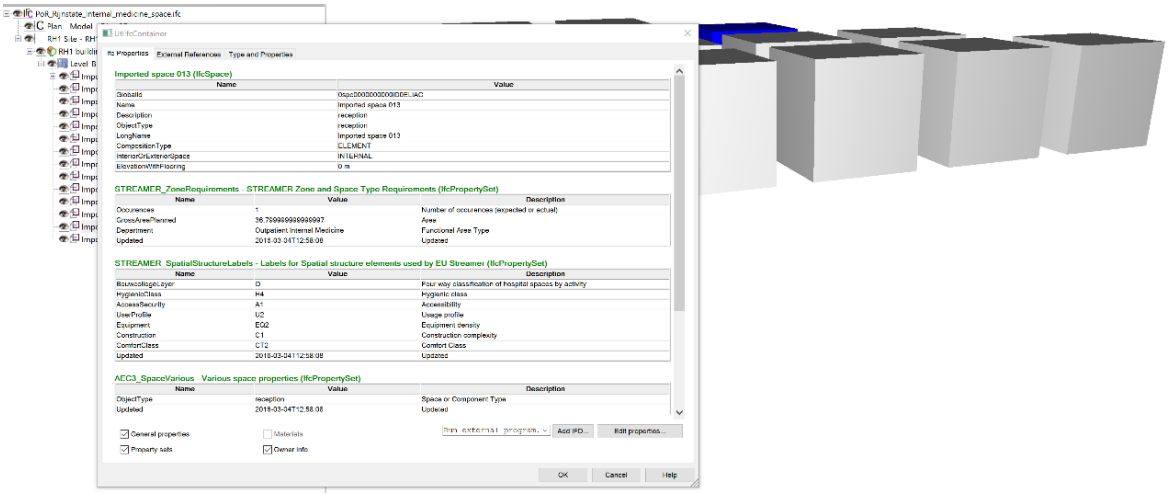


Figure 16: The outcome is a semantically rich IFC model (DDS Viewer)

The outcome is a valid and complete IFC model, ready for review, checking and federation with other sub-models. Note that the specified mvdXML and the EDC validate the PoR on names based on the given vocabulary and structure. If the name does not occur in the CSV file loaded into the EDC, the PoR is rejected and an error will pop up. If other or additional names than agreed within STREAMER shall be used they have to added to the exchange requirements to be properly checked. Otherwise the PoR cannot be processed by the EDC.

4.2 EDC to Energy simulation

The Early Design Configurator produces IFC2x3 files following the agreements of the Coordination View 2.0, more specifically the “SpaceBoundary1stLevelAddOnView”. It is thus fully compatible with available IFC 2x3 files produced by todays CAD applications.

The data given by the PoR is converted into an IFC data model that was selected as the best solution produced by the genetic algorithm of the EDC considering the imported design rules that are used for design validation. The following data is produced and exported by the EDC (for further details see also chapter 5.4) :

- *IfcProject* instance with all basic settings (units and geometric representation context)
- Spatial structure of the building (including *IfcSite*, *IfcBuilding*, *IfcBuildingStorey* and *IfcSpace*)
- Position and orientation of the building
- Space instances with all label properties given by the PoR
- 1st level space boundaries of all spaces
- Base quantities
- Physical bounding elements like walls (*IfcWallStandardCase*) and slabs/roofs (*IfcSlab*)
- Glazing property for walls
- Default material information for bounding elements
- Default instances for each space for ventilation and lighting (*IfcFlowTerminal*)
- Zones being a group of spaces
- Ventilation and lighting systems being a group of *IfcFlowTerminal* instances

Currently not exported are:

- Physical bounding elements like window and door
- Wall connection

This data needs to be imported by the used energy simulation tool and must be processed and enriched by missing data. Main challenges in general are to deal with windows and doors, to include local weather data and to may convert 1st level into 2nd level space boundaries. A specific challenge for STREAMER is to evaluate the given labels, which means to derive relevant room settings like set temperature or occupancy and to group spaces into thermal zones. More details how IFC-based design data are imported by the different simulation tools namely the CENtool, VABI Elements, Trnsys, SBEM and Energy+/Simergy are described below.

CENtool

The CENtool consists of the recently developed standards EN ISO 52016-1:2016, energy needs for heating and cooling, internal temperatures and sensible and latent heat loads, and EN ISO 52010-1:2016, external climatic conditions, which are part of the new EU wide EPB calculation. The original implementation developed for CEN, as validation of these new standards, is reused and extended by supporting IFC input for Streamer purpose.

The IFC file produced by the EDC is used as import by the CENtool. It uses the Streamer labels to add detailed information related to occupancy and set point temperatures of individual rooms. Currently the tool calculates each individual room due to the missing of zoning details according to the CEN EPB methodology described in EN ISO 52000-1:2016. But still, additional input data is needed before the simulation can be started. This includes:

- An external climate file depending on the geo-position of the building; the needed data is imported as a file that is to be placed in the data folder and is using a comma separated value (csv), currently using a “RAY” extension. The file should have values for temperature, humidity, wind speed and direction, irradiance (direct normal, diffuse on horizontal surface, global solar) and ground solar reflectivity
- Some default assumptions are done for properties on walls, windows, floors and roofs which are not yet present in the IFC file from the EDC. However these values do relate to materials, but the conversion would require a detailed material database describing properties of the materials.
- System configurator is defined in an external TXT file (tab separated value format) and includes:
    - Ventilation recirculation factor;
    - Ventilation heat recovery efficiency;
    - total system efficiency (emission + distribution + generation) for heating, cooling (and domestic hot water not yet used).

VABI Elements

Vabi Elements is a commercial design tool for computing energy analysis simulations based on a 3D model. This tool was selected because of its strong position in the Dutch market. It combines several types of energy simulation relevant in the Netherlands and is also able to import IFC files. Other data formats such as gbXML are currently not supported and therefore must be converted to IFC first.

Vabi Elements is able to handle 3D geometrical information of elements, but does not evaluate further parameters such as for instance the U-value of a wall. It also allows to visualize imported IFC data as shown on right side of Figure 17.

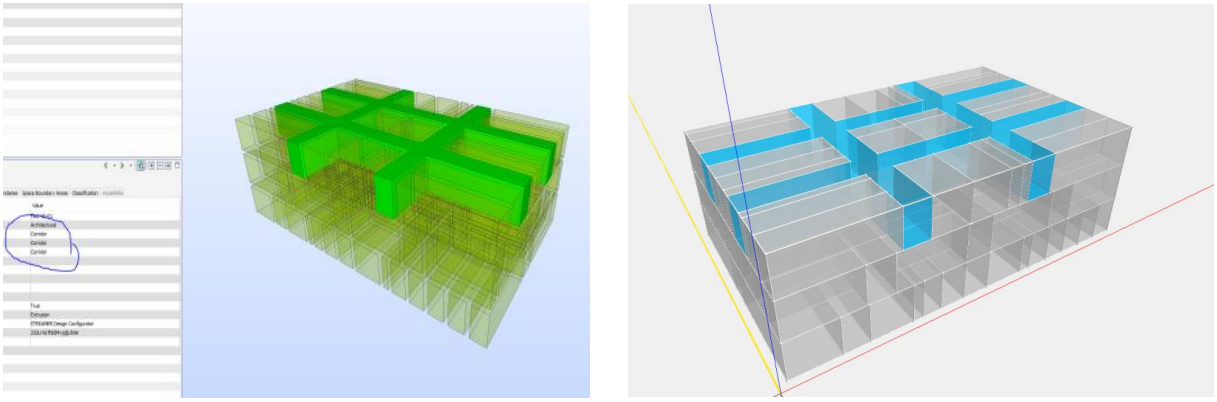


Figure 17: IFC file visualized in Solibri (left) and VABI Elements (right)

Based on the proposed layout and geometry, VABI Elements can do different simulations and calculations, for instance:

- Peak requirements for heating and cooling; both on room and building level
- Heating and cooling demand; both on room and building level
- Thermal comfort calculation on room level

To do energy simulation, VABI Elements requires additional parameters, like occupancy, heating and cooling system, opening times and so on. This data should be derived from labels given by the PoR. Unfortunately, VABI Elements cannot extract these parameters directly from an IFC file exported by the EDC. Thus, a work around has been developed to deal with that issue. It consists of a VABI library that is defining used STREAMER labels. Those labels are enriched with adequate settings for missing parameters. Based on this extension it is possible to calculate energy consumption and thermal comfort.

Data import has been tested with EDC output. At the time of this writing two issues remain unresolved:

- There are no windows and doors, that in particular are relevant for external walls
- Internal and external walls do have the same properties so that they are recognized by VABI as similar walls.

Trnsys

STREAMER labels represent generic data for ordering requirements. Unfortunately, there is no “understandable values” by simulation tools for each label. So, we decided to allow the user to fix some corresponding values for some labels by user graphical interface and to recover the bare minimum data from IFC file. Here are presented the data required by evebim-Trnsys:

- An *IfcSpace* by room (with each *IfcRelSpaceBoundary* - level 2)
- The “BaseQuantity” for *IfcSpace*:
    - NetVolume,
    - NetFloorArea
- The property “IsExternal” must be filled for *IfcWall, IfcSlab* and *IfcDoor*

SBEM

AEC3 already had a mapping from IFC into SBEM input format, based on the mapping of traditional (detailed) models. For STREAMER this was developed to ensure that:

- Unless specified, for example a CHP (combined heat and power) unit, a generic HVAC was assumed

- Unless specified, a generic HWS (hot water system) was assumed
- If systems were present, these were considered in preference to any constituent terminals
- If zones were present, these were considered in preference to any constituent spaces
- BouwCollege semantic labels mapped to the SBEM specific activity codes for the zones. This was achieved through an independent dictionary. The activity code determines the casual and equipment codes, occupancy and lighting requirements.
- SBEM and its results

SBEM is a computer program that provides an analysis of a building's energy consumption. SBEM estimates the monthly energy use and carbon emissions of a building when given a description of the building's geometry, construction, and its building services equipment. SBEM was originally based on the Dutch methodology NEN 2916:1998, Energy Performance of Non-Residential Buildings. It has been developed in order to comply with the emerging CEN standards, where a full description of the methodology implemented in SBEM can be found. The purpose of the program is to aid in the analysis of energy usage in buildings through a simplified calculation that maintains simplicity and consistency for the user.

ENERGY+/SIMERGY/DESIGNBUILDER

The Italian case assumes a retrofitting scenario for which the EDC is less equipped at the moment. Accordingly, a different approach was chosen based on the tools ArchiCAD from Graphisoft and Simplebim from Datacubist. The PoR is organised by classes of labels for the requirements that should ideally be met for each function. In an existing hospital these requirements are not necessarily met. The difference between theoretical requirements and real performance can be analysed as a measure of under- or over performance (Deliverable 2.3). A large number of rooms is defined, each with a number of labels containing crucial information for the purpose of the energy simulation.

The IFC model of the existing building originating in ArchiCAD is exported based on the Concept Design Model View with base quantities, materials (with material code assigned to each) and common properties. ArchiCAD does not correctly export space boundaries omitting surfaces between adjacent rooms and not recognizing adjacencies correctly. Two simulation softwares have been used, both are interfaces of the Energy+ simulation engine. Simergy Pro specifically features a complete importation of the IFC format based on the Concept Design BIM 2010 Model View, used in the beta version however errors were encountered with the larger models. In a parallel track simulation was performed with Designbuilder which does not import IFC but only gbXML, in this case 2nd level space boundaries have been added in Revit.

The exported IFC model of the existing building is checked in Simplebim before import in the energy simulation software. The process of checking the model for minimum required object classes, completeness of labels and allowed values of Streamer labels are set in an Excel template file based on the template supplied with Simplebim. In STREAMER the corridor does not have any label values so that the checking tool should omit checking label values for spaces of type “corridor”.

Rules for Text Property Values

Object or Group [+]	Text Property [+]	Rule for Text Property	Case Sensitive	Allow Empty	Separator	Value [+]
Space	Access security	Must be one of the Values	No	Yes	;	A1;A2;A3;A4;A5
Space	Bouwcollege layer	Must be one of the Values	No	Yes	;	H;HF;O;I
Space	Comfort class	Must be one of the Values	No	Yes	;	CT1;CT2;CT3;CT4;CT5;CT6;CT7;CT8
Space	Construction	Must be one of the Values	No	Yes	;	C1;C2;C3;C4;C5;C6
Space	Equipment	Must be one of the Values	No	Yes	;	EQ1;EQ2;EQ3;EQ4;EQ5;EQ6
Space	HygienicClass	Must be one of the Values	No	Yes	;	H1;H2;H3;H4;H5
Space	User profile	Must be one of the Values	No	Yes	;	U1;U2;U3;U4
Space	Equipment	Must be one of the Values	No	Yes	;	EQ1;EQ2;EQ3;EQ4;EQ5;EQ6

Figure 18: Validation of Streamer labels in Simplebim

The energy simulation software we tested (Simergy Pro, Designbuilder, IDA ICE) do not import the labels defined in the STREAMER process. Therefore the labels that affect energy performance are used to enrich the file with required physical values for properties belonging to the IFC schema. First, in Simplebim the property is assigned to all objects belonging to the IfcSpace.

Add Property to Object Class or Group

Object or Group [+]	Property [+]	Property Type	Single/List	Unit Type	Unit Symbol
Space	SpaceTemperatureMin	Measure	Single	Temperature	C
Space	SpaceTemperatureMax	Measure	Single	Temperature	C
Space	LightingRequirement	Text	Single		
Space	Illuminance	Measure	Single	Other	lux
Space	MechanicalVentilationRate	Measure	Single	Volumetric Flowrate	m3/s
Space	OccupancyTimePerDay	Measure	Single	Time	h

Figure 19: Adding properties to spaces in Simplebim

The example of Figure 20 shows how the required physical values are assigned based on the Comfort class label value. While this effort is made to preserve the room data (for the large amount of rooms in a hospital PoR), the energy consultant BEQ choses to assign MEP systems and building envelope manually within the simulation software. A more elaborate description of the process can be found in deliverable 3.4 Energy Simulation Tools, paragraph 4.3.

Property Name or Key	Comfort class	SpaceTemperatureMin	SpaceTemperatureMax	LightingRequirement	Illuminance	MechanicalVentilationRate
Operator	Match = equals	Set	Set	Set	Set	Set
	CT1	<no value>	<no value>	NOTDEFINED	<no value>	<no value>
	CT2	<no value>	<no value>	DIRECT DAYLIGHT	150	<no value>
	CT3	20	<no value>	DIRECT DAYLIGHT	500	10
	CT4	20	24	DIRECT DAYLIGHT	500	10
	CT5	20	24	DIRECT DAYLIGHT	500	10
	CT6	18	24	NOTDEFINED	1000	18
	CT7	18	24	NOTDEFINED	1000	60
	CT8	<no value>	<no value>	NOTDEFINED	<no value>	<no value>

Figure 20: Assigning values to properties based on Streamer label values

4.3 Energy simulation to Decision support tool

Ideally, all results are reported in the same way, preferable as properties added to the IFC model and may even be completed by added energy simulation assumptions. This should include the global annual energy demand and consumption, ideally given for heating, cooling, hot water, lighting and equipment. An mvdXML-based checking specification could be based on the previous EDC export check by extending agreed property definitions for the *IfcBuilding* instance.

However, experiences using various commercial energy simulation tools are somehow different and may require further data merging processes or evaluation of proprietary results formats. The next section describes how results are submitted to the Decision support tool.

CENtool

The CEN tool simulation calculates the following results:

- Total building and each individual space Energy demand for heating and cooling
    - Using the layout plan with glazing given by the EDC
    - Using solar irradiance on surfaces
    - Using a replaceable climate data file (data that must match with the geo-position of the building)
    - Using ventilation heat loses (reduced by recirculation and heat recovery units)
- Building Energy consumption for heating and cooling
    - Using a efficiency per system (combination of emission, distribution and generation)

The results are exported to two files:

- ThermalZone[space number].csv

  Detailed hourly report of the calculated demands and internal temperatures in a proprietary CSV format for each thermal zone, currently this equals each space.
- [original name]_energyCalc.ifc

  A new version of the IFC file extended with the results of the calculation (per zone and building). Results are exported to same properties as Trnsys (see Figure 24).

VABI Elements

VABI Elemens supports different types of energy calculations (see chapter 4.2). Accordingly, many results attached to building and space instances can be exported, preferable in a table format as shown in Figure 21. The tool also enables to colour code spaces showing for instance heat losses within the building (see Figure 22).
Besides the Excel table format results can be exported to an IFC file. They then include basic 3D geometry information and result data (see Figure 23).

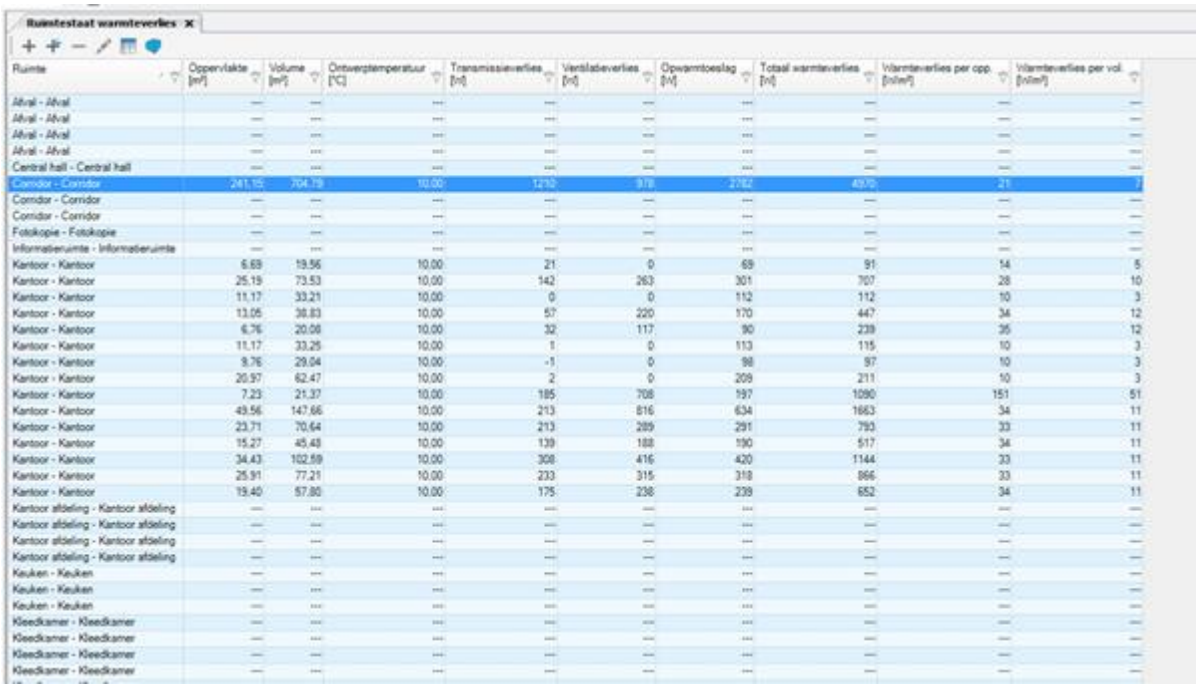


Figure 21: Energy results for spaces produced by VABI Elements

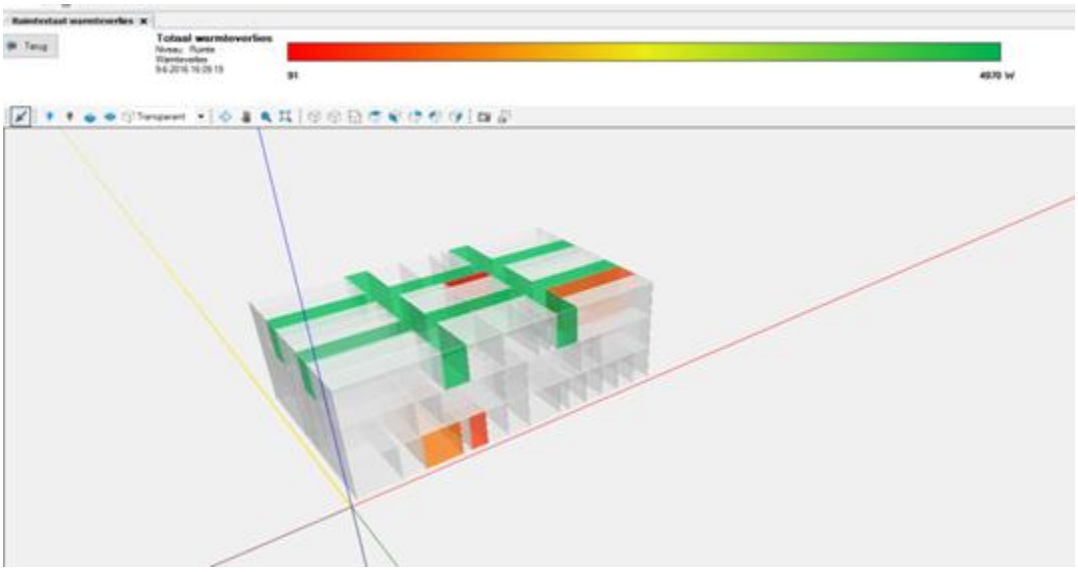


Figure 22: Visualization of results

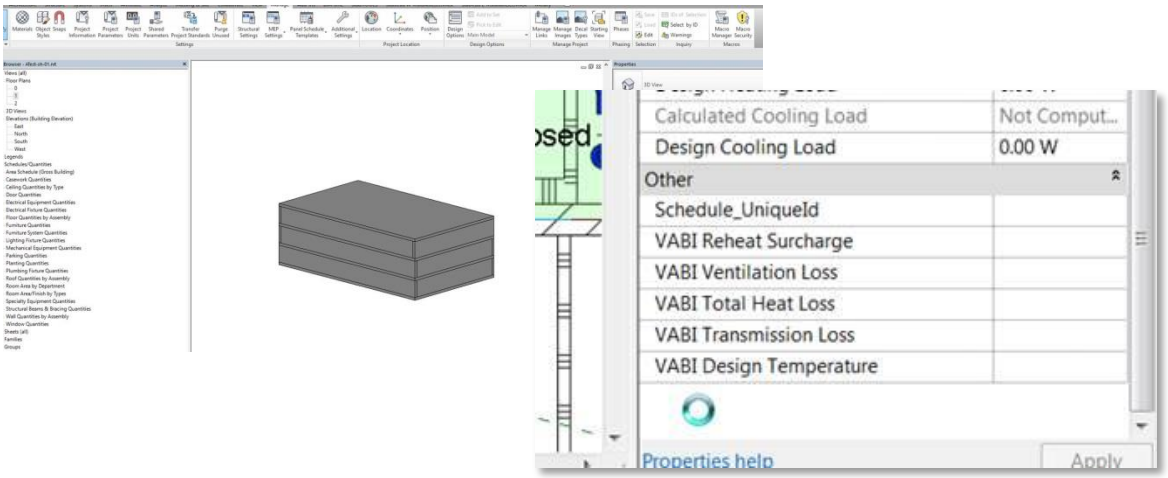


Figure 23: IFC export including basic geometry and result values.

Trnsys

To facilitate the communication of simulations results to the Support Decision Tool, it has been decided to use the IFC format by placing the results into STREAMER property set linked to the building. We decided to use the same IFC file as the one describing the IFC building rather than having a specific IFC for the results to avoid having to copy all unique ID for IfcProject, IfcSite and IfcBuilding to keep the spatial structure. The energy simulations were performed in 2 steps:

- The heat and cold demand: the calculation are stored in the Pset Heat and Cold demand.
- The energy consumption of heating and cooling system are stored in the Energy Consumption Pset (see Figure 24).

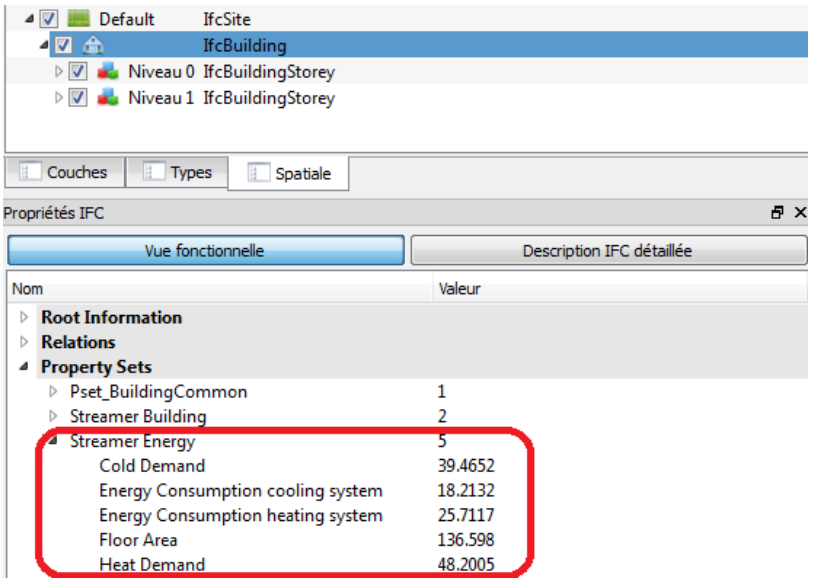


Figure 24: Psets used to store energy simulations results in IFC

SBEM

SBEM produces a single Building Energy Rating (for comparison against a Target Energy Rating). Other results are reported in a CSV format. These results include monthly breakdowns (in kWh and MJ) for each demand (heating, lighting, and equipment, auxiliary) and each consumption (gas, electricity, other fuels). Breakdown is also available for each individual zone, including also occupancy gains. These results were mapped to make an IFC submodel and then merged with the input IFC to create a complete outcome, holding the campus, zones and the chosen system upgrades with area, cost and energy values.

Table 2: SBEM_AnnualEnergyDemand_UK - REPORT- Energy consumption by End Uses in MJ and stored in an IfcPropertySet

Name	Value	Description
HeatingAnnualEnergyDemand	675609 MJ	Heating energy demand
CoolingAnnualEnergyDemand	0 MJ	Cooling energy demand
AuxiliaryAnnualEnergyDemand	138790 MJ	Auxiliary energy demand
LightingAnnualEnergyDemand	33003.8 MJ	Lighting energy demand
HotWaterAnnualEnergyDemand	336095 MJ	Hot water energy demand
EquipmentAnnualEnergyDemand	419656 MJ	Equipment energy demand

Table 3: SBEM_AnnualEnergyConsumption_UK - REPORT- Energy consumption by Fuel Type in MJ and stored in an IfcPropertySet

Name	Value	Description
NaturalGasAnnualEnergyConsumption	1.0117E+006 MJ	Natural gas energy consumption
LiquidPetroleumGasAnnualEnergyConsumption	0 MJ	Liquid Petroleum Gas energy consumption
BioGasAnnualEnergyConsumption	0 MJ	BioGas energy consumption
OilAnnualEnergyConsumption	0 MJ	Oil energy consumption

CoalAnnualEnergyConsumption	0 MJ	Coal energy consumption
AnthraciteAnnualEnergyConsumption	0 MJ	Anthracite-ECF
SmokelessAnnualEnergyConsumption	0 MJ	Smokeless energy consumption
DualFuelAnnualEnergyConsumption	0 MJ	Dual fuel energy consumption
BiomassAnnualEnergyConsumption	0 MJ	Biomass energy consumption
GridSupplyAnnualEnergyConsumption	171794 MJ	Grid Supply Electricity energy consumptions
WasteHeatAnnualEnergyConsumption	0 MJ	Waste Heat energy consumption
DistrictHeatingAnnualEnergyConsumption	0 MJ	District heating energy consumption
Displaced-ECF	0 MJ	Displaced-ECF

The SBEM results CSV file is transformed into an IFC sub-model and then merged back into the main IFC model. These two processes are both achieved by converting the inputs into XML (Spreadsheet XML 2003 and ifcXML, applying an XSLT transformation and then converting the resulting ifcXML into IFC.

The transformation of the SBEM results CSV notes the headings found in the file. A local dictionary file contains the correspondence between each heading and the desired IFC property name and IFC property set name. This ensures that the pipeline can match any IFC definition or STREAMER requirement.

```
<concept type="property">
     <term context="SBEM">NatGas-ECF</term>
     <term context="en-GB">Natural gas energy consumption</term>
     <term context="IFC">NaturalGasAnnualEnergyConsumption</term>
</concept>
```

Figure 25: Example of the global dictionary holding SBEM, en-GB and IFC terms.

ENERGY+/SIMERGY/DESIGNBUILDER

Simulation data is exported in a CSV file, which is then merged with the IFC file through a separate tool called Simplebim. This additional post-process enables to include all required data in a single IFC file that can be evaluated by the Decision Support Tool (DST). Results are exported as property sets and include energy consumption as well as heating and cooling demands (see result tables below). The carbon emission KPI required in the DST is also a direct output from the energy simulation. The life cycle cost KPI is calculated directly within the DST.

Table 4: Annual consumption by end use

	Electricity [kWh]	Natural Gas [kWh]	Additional Fuel [kWh]	District Cooling [kWh]	District Heating [kWh]	Water [m3]
Heating	0	0	0	0	154684	0
Cooling	0	0	0	112937	0	0
Interior lighting	359326	0	0	0	0	0
Interior equipment	30879	0	0	0	0	0
Water systems	0	0	0	0	663206	10385

Total end uses	390205	0	0	112937	817890	10385

Table 5 : Annual Energy consumption for the Carreggi hospital (preliminary data)

Description	Yearly consumption (MWh)
Room Electricity	30879
Illumination	359326
Heating (Other)	181900
Cooling (Electricity)	60204
Cooling (Other source)	22203
ACS (Other)	780242

4.4 Decision support tools to Modelling tool

IFC data produced by the EDC and may enriched by the energy simulation and decision support tool shall be used in standard CAD authoring tools to improve and extend the current design proposal. Missing elements like windows, doors, HVAC equipment etc. must be added and properly connected to existing data. If some data is not properly imported or lost, it has to be re-entered or otherwise imported and merged to the BIM data. This can be time-consuming and error-prone and thus shall be avoided. This chapter describes experiences with using the EDC-IFC file in commercial CAD tools like Revit and ArchiCAD.

Further data processing with Revit

Autodesk Revit 2016 is able to import IFC files. The IFC file that is generated by the current version of the EDC (July 01, 2016) does contain all the information that is collected through the process, besides the applied design rules. *IfcSpaces* are translated to the room family in Revit. The semantic data attached to the *IfcSpace* are stored and linked to the Revit room family. This data includes the labels, functional area, room name, number of occurrences (amount) and room type. It does not include the required area (see Figure 26 and Figure 27). The Area (value is 16.0) under STREAMER PoR of the IFC file cannot be traced back in the room properties in Revit. This is probably due the fact that Revit does not import the structure of the IFC file and placing all parameters under.

For validating the current design with the original requirements it would be preferable to have the original requirements attached to the room or linked by a link to the original file. In Briefbuilder for example it is possible to include a URL of the room typology, so that at any time the original requirements can be found and validated.

PropertySets from entity	
STREAMER Room	
Length	5.6666666666667
Area	17.6
Width	3.1058823529412
STREAMER Labels PoR	
ComfortClass	CT3
Construction	C1
BouwcollegeLayer	O
HygienicClass	H2
AccessSecurity	A4
UserProfile	U2
Equipment	EQ2
STREAMER PoR	
FunctionalAreaType	Admission
RoomType	Office
Area	16.0
Amount	48
STREAMER Labels MEP	
Ventilation	unknown
BaseQuantities	
NominalHeight	3.25
GrossWallArea	57.021568627451
GrossFloorArea	17.6
GrossCeilingArea	17.6
GrossVolume	57.2
GrossPerimeter	17.5450980392157

Figure 26: Properties of IfcSpace object seen in the FZK viewer

Since the latest version of the EDC, the IFC export includes walls that are processed by Revit into a Revit wall family with actual geometry. Some of the semantic data produced in the EDC is lost however. For instance the distinction between an outer and inner wall cannot be found in the wall properties within Revit (see Figure 26 and Figure 28 for comparison between the IFC file and the same file imported into Revit). This could be a risk as some data is re-entered and could thereby be different than the original source. If every tool uses other assumptions the performance simulation could be off. This is crucial within a STREAMER workflow as a design is not only simulated and validated on energy performance but also on LCC and Quality as well.

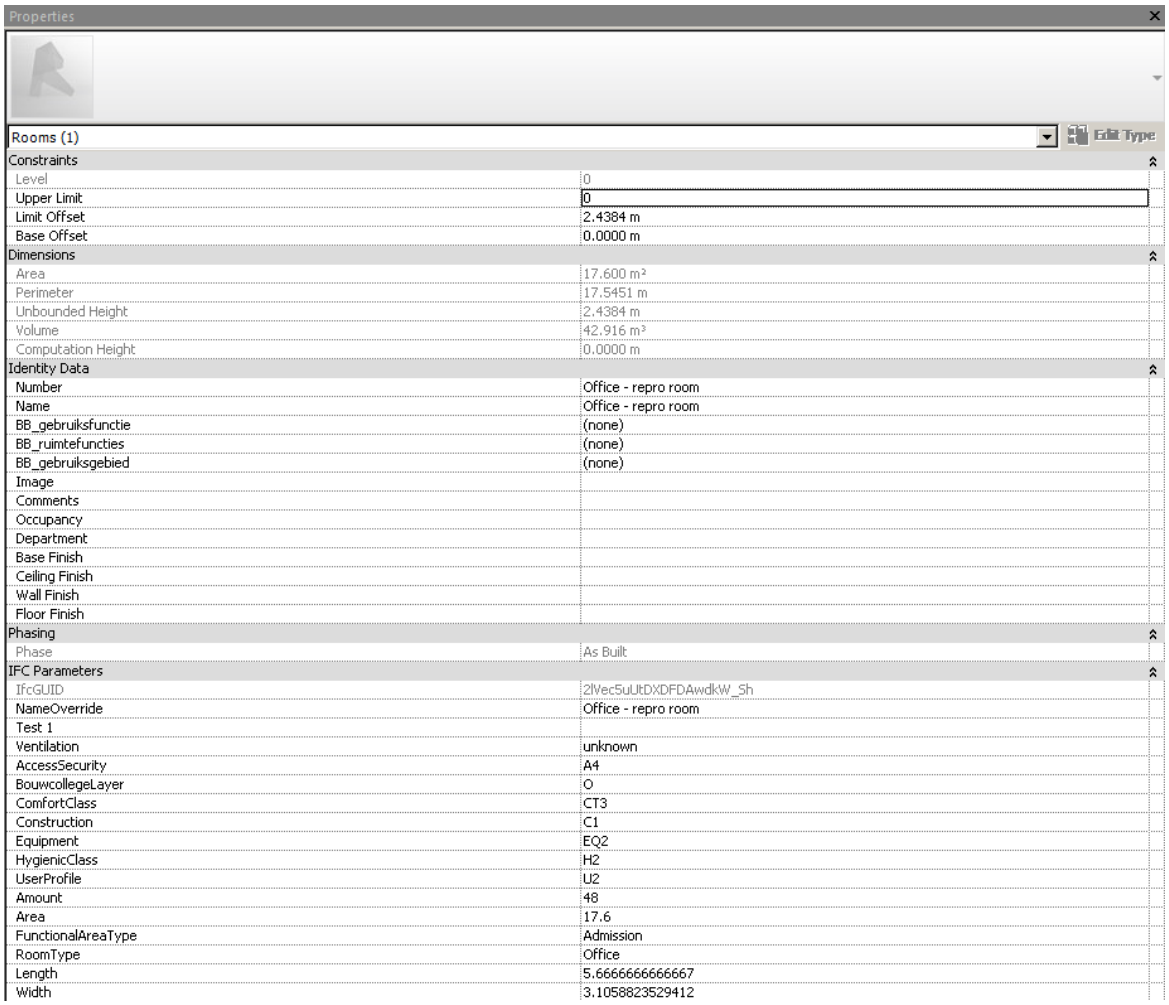


Figure 27 Properties of IfcSpace object seen in Revit

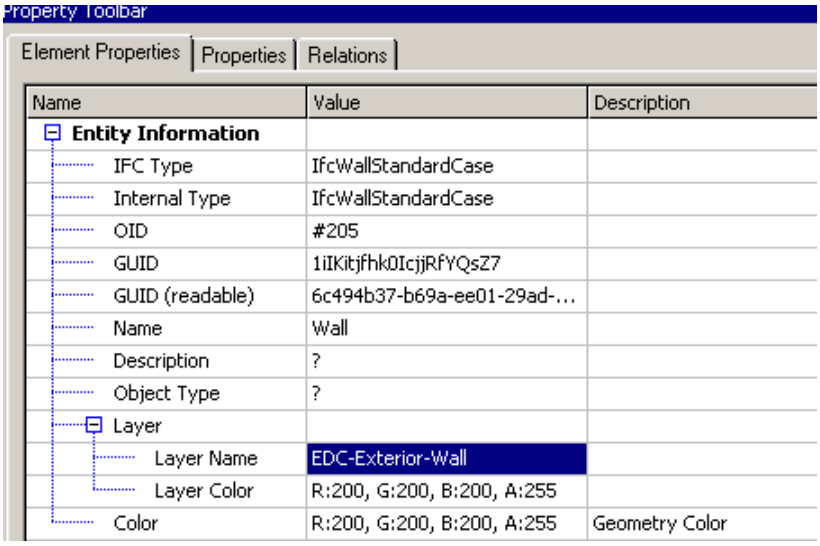


Figure 28: Properties of IfcWall object imported into FZK Viewer.
Highlighted is the exterior wall parameter which is not present in Revit.

IFC Parameters	
IfcGUID	1iIKitjfhk0IcjjRfYQsZ7
NameOverride	Wall

Figure 29: Properties of IfcWall object imported into Revit

Some additional steps probably need to be taken to transfer the IFC parameter values to “natural” Revit parameters. There are several ways to do this more or less automatically. An option is making a room schedule in Revit with the IFC parameter in one column and the Revit parameter in another. Then copy paste the content of the IFC parameter into the native Revit parameter. A more automatically driven option is by using the plugin Dynamo in Revit, which is a visual scripting tool based on Python.

ArchiCAD – Careggi

The Carreggi hospital is seeking to extend its SACS database to include STREAMER results. In the Carreggi case, both the existing hospital and intervention proposals are modelled in ArchiCAD. Due to known limitations of the IFC format as not having been intended for round tripping geometry, the best results will be obtained by importing numerical information from different data sources (energy data from simulation software, KPI’s from the Decision Support Tool) back into the original ArchiCAD file. To guarantee a correct coupling of the data both the file as well as all the rooms should have a unique ID.

Once inserted into the ArchiCAD file the simulation results are static information belonging to a single design phase which will not update automatically as the design is being developed.

5. Prototype Implementation

5.1 Requirements Capturing with BIM-Q

Need for a shared, web-enabled requirements management tool

As outlined in chapter 3.2 exchange requirements are a means for communication and thus need to be agreed and shared between involved participants. Also, many requirements are applicable for several processes so that a lot of definitions can and should be reused.

Today, exchange requirements are typically captured in a spreadsheet format. For each physical or conceptual thing it captures relevant properties, its meaning and use in design processes (IDM). It is simple and straight forward but the more information is captured and shared, the more difficult it is to keep consistency and maintain the content. There are also limitations to evaluate and export requirements, in particular for generating various reports and producing an mvdXML file for checking purposes. Accordingly, there is a need for better tool support leading to the web-based BIM-Q solution.

Before collecting exchange requirements an initial set-up of the database is necessary. The first step is to define a template guideline that shall group all definitions. This might later be used to configure project requirements. Next to this, the selection of involved stakeholders, covered stages and processes as well as relevant mappings is necessary. Mappings include links to classification systems, translations to other languages and the representation in data structures like IFC. In this initial step it means to set-up the boundaries for the discussed use cases in terms of definitions and standards that become relevant to clarify the meaning of terms and to be used for data exchange. Each of those settings can be changed or extended in later stages, but it defines the starting point for defining relevant terms, which is the first main step of capturing domain knowledge.

Set-up of reusable concepts

Definition of exchange requirements follows the object-oriented modelling principle, but with less restrictive rules. Everything is a concept. Each concept can be described, typed, mapped to other definitions and arranged to each other in order to form more complex concept definitions. A concept can for instance represent a class of beam objects whereas another concept represents a simple datatype property for fire rating.

An exchange requirement is typically defined for a property of some object class. A fire safety calculation may require the fire rating property for all loadbearing building elements. It is a simple and natural way of expressing requirements that can be defined by non-IT experts.

Experiences have shown that a lot of concepts are reused for requirement definitions, in particular in case of generic properties. This is leading to a lot of copied content that is later difficult to maintain. Therefore, the first step is to collect reusable concept definitions that can be arranged in any level of complexity. In that way, a pool of concepts is defined that later can be arranged to any requirement setting that needs to be described. Each reusable concept is linked to default definitions, such as a description or the mapping to IFC, which however can be overridden in a requirement setting if necessary.

The pool of reusable concepts can be organized according to own preferences. Our recommendation based on experiences is to organize similar concepts in groups like classes, properties and geometry. STREAMER is using a labelling approach and thus is using the structure as shown in Figure 30. Further subgroups are recommended, but should be kept as simple as possible. If properly arranged it later helps to find the right concept and to configure the requirement settings.

Figure 30: Reusable concepts as defined in the STREAMER project.

Configuration of exchange requirements

The next step is to link objects with properties in order to express requirements. This is done by dragging reusable concepts to a new requirements tree as shown in Figure 31. Both trees provide independent search capabilities so that concepts can easily be found and arranged in the requirements tree. In order to speed-up the set-up process it is also possible to drag and drop a concept with all child elements. If reusable concepts are properly arranged it supports an easy and fast set-up process.

Differently to reusable concepts there are some constraints regarding the organization of the requirements tree. Those constraints exist mainly due to the fact that some meaningful reports or an mvdXML file shall be generated out of this tree. By following the idea of having a property of some object class the structure should follow the rule of having a property concept, marked as a simple datatype, always as a child element of an object concept. In between there might additional group concepts for better organization of requirements, which are ignored for later model checking. There are special solutions for enumeration datatypes having allowed values as child concepts, which however do not break described general rule. Nevertheless, a risk of configuring a requirements tree that cannot be properly exported to mvdXML checking file remains so that this step should carefully done.

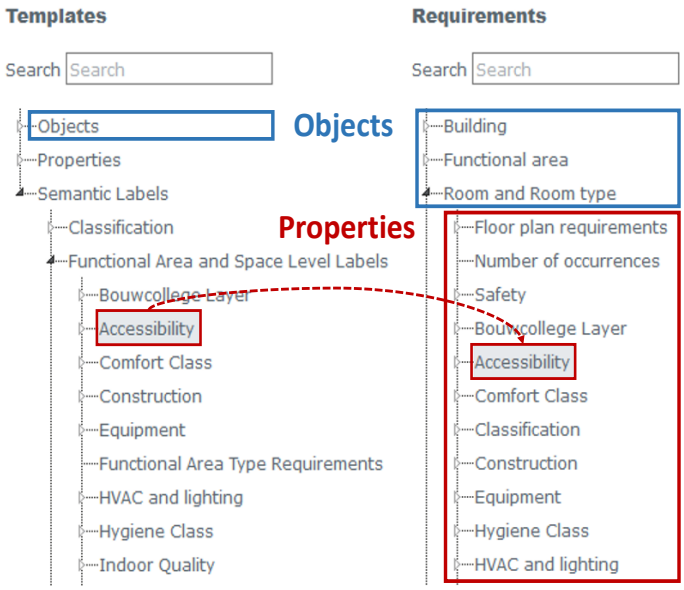


Figure 31: Set-up of the requirements tree by dragging reusable concept from templates (left) to the requirements tree.

Once the requirements tree is defined the usage settings for the different processes can be configured. It basically means to make a decision what data is required, optional or not allowed. Additionally, an owner of a data concept has to be defined who is responsible to deliver that information (Figure 31).

IFC mapping definitions

Each concept can have any number of mapping definitions to whatever data structure is of interest. In our case the focus is on the open IFC-BIM format that can be formalized by mvdXML definitions.

There are basically two types of mapping definitions:

- **Object concept mappings**:

  For mvdXML it means to configure a *ConceptRoot* element comprising of the selection of an IFC entity (*applicableRootEntity*) and, optionally, additional *Applicability* settings.
- **Property concept mappings**:

  This requires the configuration of a *Concept* element, which needs to identify and configure an appropriate *ConceptTemplate*.

The BIM-Q tool supports a simple syntax to easily configure most frequently needed mapping definitions. An object concept for instance maps either 1:1 to an IFC entity, or is additionally restricted by the *PredefinedType* attribute or some property values. The expression *IfcWall.IfcWallTypeEnum.SHEAR* is for instance applicable for all *IfcWall* instances having the *PredefinedType* attribute set to “Shear”. Similar solutions are available for property concepts, where for instance the configuration of properties and quantities is often needed. Uncommon mapping definitions have to go through a more complex configuration process. This however shouldn’t be a problem as this step has to be done by an IFC expert who is familiar with the IFC specification and available mvdXML concept templates.

Concept Definition	Owner	ER1-PoR	ER2-EDC
Building	Architect	-	MAN
Functional area	-	MAN	MAN
Room and Room type	Architect	MAN	MAN
Accessibility	Building owner	MAN	-
Bouwcollege Layer	Building owner	MAN	-
Classification	Architect	MAN	MAN
Comfort Class	Building owner	MAN	-
Construction	Building owner	MAN	MAN
Containment	Architect	-	MAN
Equipment	Building owner	MAN	MAN
Geometric representation	Architect	-	MAN
HVAC and lighting	Building owner	OPT	-
Hygiene Class	Building owner	MAN	MAN
Indoor Quality	Building owner	OPT	-
Number of occurrences	Building owner	MAN	NOT

Figure 32: Definition of usage settings and assignment to a concept owner

Reporting and mvdXML export

The final step in the requirements capture process is to produce some sort of evaluable result. This might be a specific PDF report that could act as a contract annex, an mvdXML file for checking purposes or some template documents. In case of mvdXML it is possible to export all settings to a single file. Alternatively, it is also possible to export settings of specific processes or a single owner only. The contractual content can also be presented as a BIM Guideline compliant to the ISO 12911 framework, presenting the strategic goals, management objectives and implementation requirements.

The export feature itself is translating the used mapping syntax to an mvdXML, which for instance in case of properties expands to a check of properties on occurrences and properties on types. At the time of this writing there is no consistency check against the IFC specification so that spelling errors are not identified. However, testing a valid file should quickly show wrong mapping definitions.

5.2 mvdXML plugin for XBIM

In order to test the adoption of mvdXML-based requirement specifications against the model data exchanged between different stakeholders of the STREAMER project, an implementation of the validation features of mvdXML 1.1 has been developed using the infrastructure offered by the open source xBIM toolkit.

The implementation is mainly designed to allow individual stakeholders to independently verify the conformity of received and produced IFC models against the agreed exchange requirements and concept roots in a user friendly visual 3D environment.

To maximize the reusability of the developed components in other validation scenarios the implementation has been divided into two software components:

1. the mvdXML validation library (mvdLib) is a .NET dynamic link library providing validation capabilities that can be consumed in multiple deployment scenarios (e.g. Xplorer UI, web services, cloud environments, command line applications, etc.)
2. the XbimXplorer mvdXML Plugin (mvdUi) is an extension plugin for the pre-existing XbimXplorer IFC viewer that provides the User Interface for interactive validation of models against specification files.

Both modules have further development activities planned in response to feedback from the STREAMER project as well as from scheduled innovations in the underlying xBIM toolkit.

User interface development and collaboration workflow

To enable a complete collaboration workflow between stakeholders of the established IDM processes the mvdUI component has been designed to allow the interactive analysis of models according to arbitrary combinations of exchange requirements, concept roots and IFC classes, the UI allows immediate feedback on the validation status of selected elements as well as whole models; this filtering strategy also helps to improve the responsiveness of the application which can become relevant if thousands of requirements need to be checked for large IFC models. Visual color coding styles have been developed to allow rapid traffic-light model inspection in the 3D viewer of passing and failing requirements.

The development of features for the semi-automatic production of validation reports in the BIM collaboration format (BCF) have required the redesign of the XbimXplorer plugin API in order to allow integration of the MVD plugin with the existing BCF plugin; the designed features allow stake-holders to exchange communication threads on the result of validation tests across different BIM plat-forms while retaining complete reference of the involved IDM, MVD and IFC background.

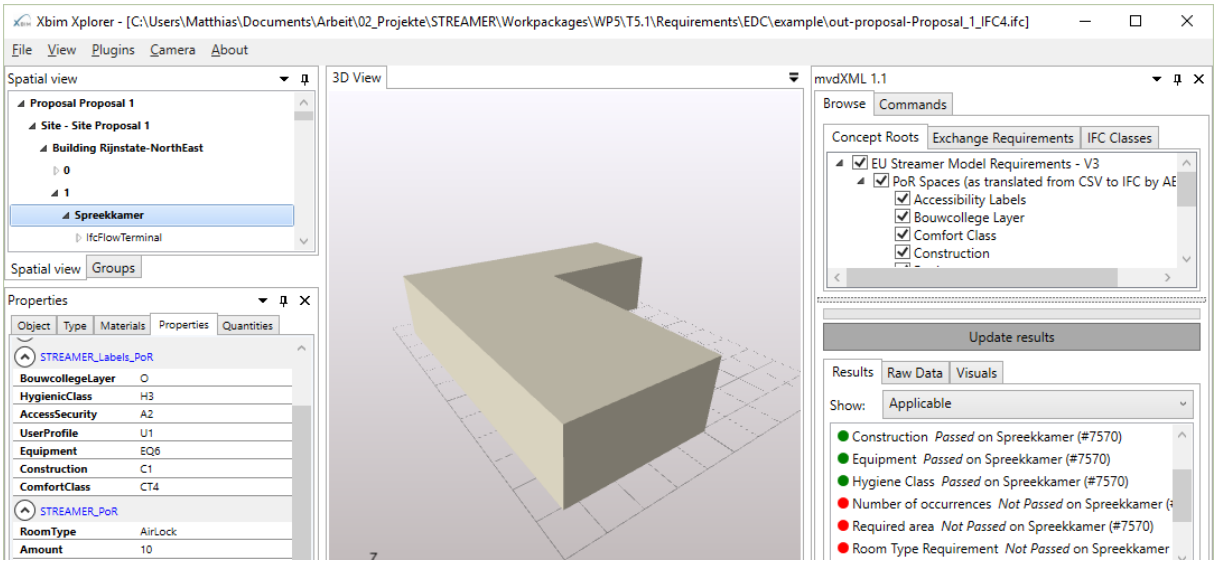


Figure 33: Checking result of an example space layout generated from space requirements. Project details on the left, results on the right

5.3 eveBIM

The snapshot shown in Figure 34 is an example of BCF workflow within eveBIM. On the left you can see the data deposited on the document server: IFC file and the related BCF annotation. The corresponding file is opened in the centred 3D view. On the right the detail of the corresponding annotation is available: author, date, title, description, snapshot, related IFC file and BIM objects. The click in the annotation snapshot repositions the building in the same view point.

The workflow proposed her is to open IFC file from the CSTB document server, to analyse it, to apposite an annotation if needed and to link this annotation with the IFC file (as a related file) on the document server.

At any time anyone could retrieve this IFC file from the server and its related annotations and can answer to it or upload a new IFC file to correct the model according to the annotation.

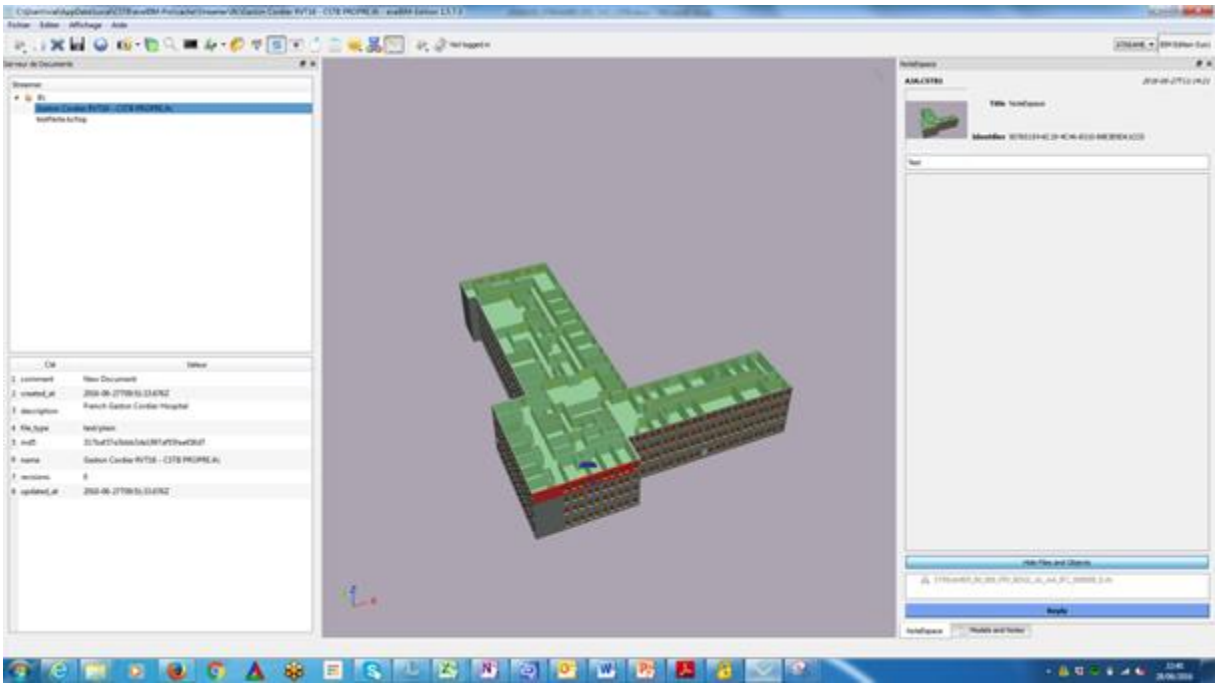


Figure 34: Screenshot of the eveBIM viewer

5.4 Checking in the Eearly Design Configurator

Import and checking inside the EDC

Currently the Early Design Configurator (EDC) imports two data sets:

- the Program of Requirements (PoR)
- and the Design Rules

The PoR is a CSV (Comma Separated Value) file, which is generated by the software BriefBuilder or by a Microsoft Excel Template. The separator of the CSV file is not predefined, but can be redefined in the settings dialog of the EFC. All values of the CSV, which are not a free text, must follow the specification of the STREAMER vocabulary [DiGiulio2015]. The title of the columns is also predefined. According to the requirements describe in chapter 4.1 the EDC checks the following titles and values (see also chapter 4.1 and D1.6):

- **RoomName**: Free text -> no further check
- **RoomType, FunctionalAreaType, BouwcollegeLayer, HygienicClass, AccessSecurity, UserProfile, Equipment, Construction, ComfortClass**: Predefined text from the STREAMER vocabulary
- **Amount**: Integer > 0 (a positive value is expected)
- **Area**: Real number > 0 (with area unit)

All names and values are case sensitive. The import does not validate the semantic of the requirements, e.g. if two labels values can be combined.

The Design Rules describe geometric relation of rooms and are stored in an XML format. Each rule contained in a rule file contains a rule name, priority, a applicability expression and a relation. The name is a free text. The priority

determines which rule has a higher probability to be fulfilled. The expression selects one or two groups of room to which the relation is applied. The relation defines which geometric relation should be applied to the rooms.
The defined relations are:

- **SameStorey**: must be on same storey as
- **DifferentStorey**: must be on different storey as
- **ContainedInStorey**: must be contained in storey
- **Clustered**: must be clustered horizontally and vertically
- **TravellingDistance**: Travelling distance between two Rooms on the same floor
- **TravellingDistanceRtoR_Patient**: must have travelling distance RoomToRoom Patient of
- **TravellingDistanceRtoR_Staff**: must have travelling distance RoomToRoom Staff of
- **TravellingDistanceFAtoFA_Patient**: must have travelling distance FAToFA Patient of
- **TravellingDistanceFAtoFA_Staff**: must have travelling distance FAToFA Staff of
- **TravellingDistanceRtoFA_Patient**: must have travelling distance RoomToFA Patient of (optional)
- **TravellingDistanceRtoFA_Staff**: must have travelling distance RoomToFA Staff of (optional)
- **TravellingDistanceFAtoR_Patient**: must have travelling distance FAtoRoom Patient of (optional)
- **TravellingDistanceFAtoR_Staff**: must have travelling distance FAtoRoom Staff of (optional)
- **DirectlyAbove**: must be directly above
- **DirectlyBelow**: must be directly below
- **PartlyAbove**: must be partly above
- **PartlyBelow**: must be partly below
- **SeparationRtoR**: must have separation RoomToRoom of
- **SeparationFAtoFA**: must have separation FAToFA of
- **SeparationFAtoR**: must have separation FAToRoom of
- **SeparationRtoFA**: must have separation RoomToFA of
- **SeparationRtoOB**: must have separation RoomToOuterBoundary of
- **SeparationFAtoOB**: must have separation FAToOuterBoundary of

While importing the file, the EDC checks the file against the defined XML schema [Sleiman2015].

Export

The Early Design Configurator (EDC) is able to export the results as a BIM model (IFC) or as a GIS model (CityGML). In the case of the BIM model all available details of the building are exported, while in the GIS case only the outer shell separated in wall -, roof – and ground surfaces is exported (see Figure 35).

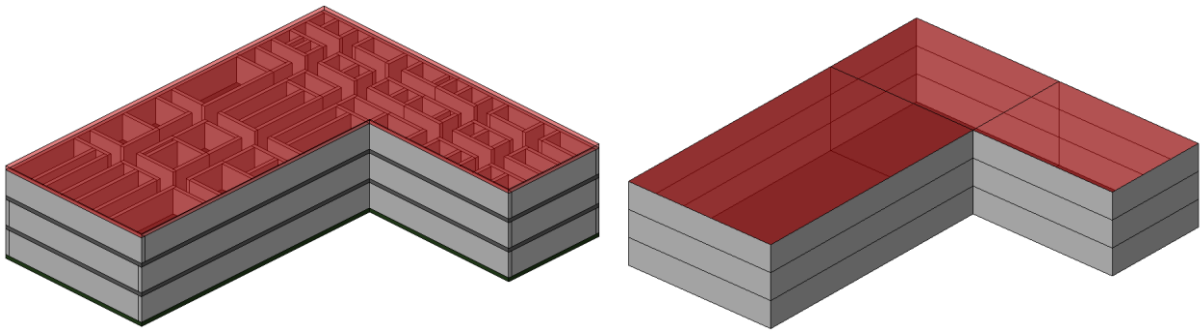


Figure 35: The BIM (IFC) model on the left side and the GIS (CityGML) model on the right side

Both models are located on the position (including orientation), which was selected in the EDC. While IFC 2x3 has only one possibility to place the model in the world by adding the longitude/latitude values to the site, CityGML allows different kind of coordinate reference systems. In the case of the EDC, the models are exported in UMT coordinate systems. In the case of Rijnstate hospital, the EPSG:25831 (UTM Zone 31N) was exported. The correct geo reference of the buildings is important to access the corresponding climate data for energy calculation.
The CityGML model is only intended for a rough energy estimation by using a single zone model with generalised default values for thermal transmittance, occupant behaviour etc.
The IFC model can be used for a detailed simulation on space level. Beside the geometry of all building elements, the space boundaries of each room are exported (see Figure 36).

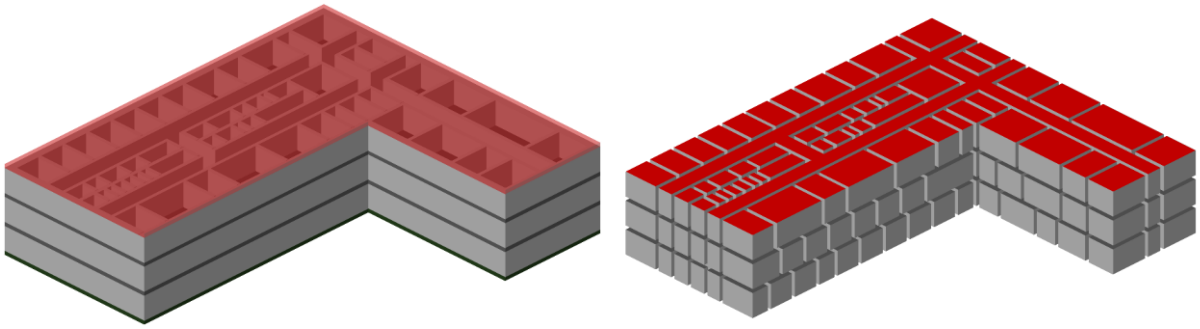


Figure 36: The generate building elements (walls, slabs, roof, left side) and the corresponding space boundaries (right side)

Although only generating 1st Level space boundaries, these boundaries can be useful as input for energy calculation. The generated spaces contain all information from the Program of Requirement (PoR) and are extended by base quantities (volume, area, perimeter and nominal height). If not calculated by the simulation tools, these base quantities are relevant parameters for the energy calculation. All spaces are grouped according to the functional areas they belong to and the grouping is exported as IFC zones. It has to be evaluated, if these zones are corresponding to thermal zones. If there is a relation, these zones can be used by the simulation system. If not, they are useless for energy calculation.
In order to model the building construction only two IFC element type are used: *IfcWallStandardCase* and *IfcSlab*. The *IfcWallStandardCase* is used to model both intern and external walls. By using the predefined type of *IfcSlab* (*Floor*, *Roof* and *BaseSlab*), the base floor, the floor slabs and the roof of the building are distinguished. By generating the topology of the wall, the wall axis and the connectivity path can be calculated and exported. Besides the base quantities for each building element, the available common properties are exported. For each class of building elements (exterior walls, interior wall, base slab, floor slab and roof slab) a material layer set can be assigned (see Figure 37 left). The material information will be exported in IFC and will be relevant for the energy calculation.
Besides the building elements for the building construction for each space, two HVAC terminals are created, one with the entity type *IfcLightFixtureType* and one with the type *IfcAirTerminalType*. The terminals have no geometry but they are grouped in corresponding systems (*IfcSystem*). As most of the CAAD systems don't import IFC building elements without geometry, and the benefit for of such entity for energy calculation has not been demonstrated, it is unclear if such entities should be exported in the early design stage.
Currently, no doors and windows are generated. In order to get a rough estimation about the minimum area of glazing, each space connected to an exterior wall get and property with the minimum required glazing surface (in m^2) (see Figure 37 right).

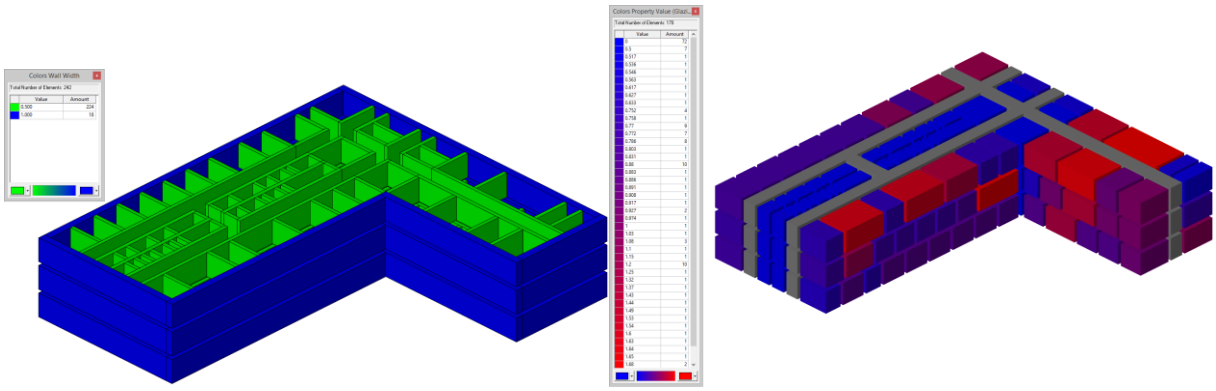


Figure 37: Created walls coloured by wall thickness (left) and created spaces coloured by the minimum glazing area (right).

As the exported BIM model is not only intended for energy simulation but also for further design processes, the exported IFC models must also be suitable for further processing in CAAD tools. In order to test this process the IFC model exported from the EDC was imported into four CAAD systems: ArchiCAD 19, REVIT 2015, Allplan 2016 and Autodesk Architecture 2015 (see Figure 38 and Figure 39). Basically the model was accepted by all systems. However, all systems ignored the HVAC terminal, as they do not have geometry. ArchiCAD 19 and REVIT 2015 created corresponding space labelling (see Figure 38), while Allplan and Autodesk Architecture 2015 do not create space labels by default (see Figure 39).

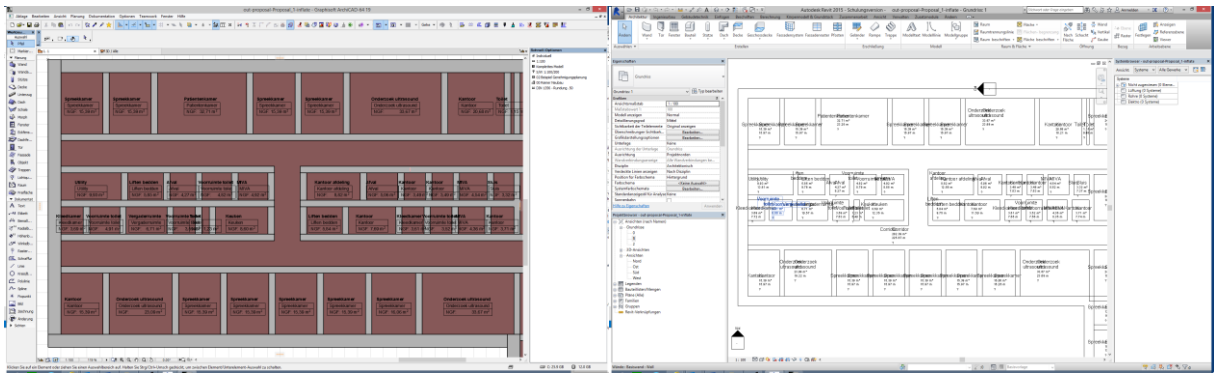


Figure 38: Import of the IFC model generated from the EDC into ArchiCAD 19 (left) and REVIT 2015 (right)

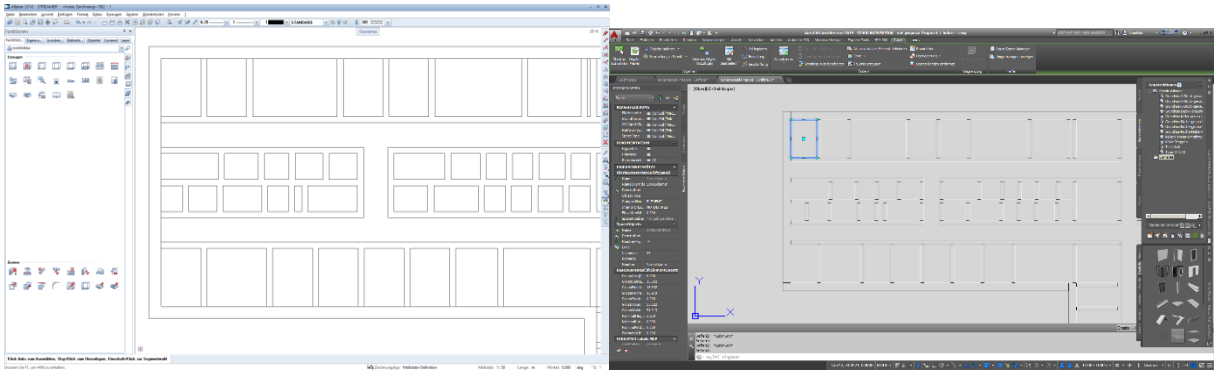


Figure 39: Import of the IFC model generated from the EDC into Allplan 2016 (left) and Autodesk Architecture 2015 (right)

In order to proof if the generated models can be treated like native models, more test must be carried out. In one of the next steps, type information for every building element will be generated. This might be interesting especially for systems, which are internal working with types (families, muster etc.).

6. Conclusion

The work presented in this deliverable is dealing with BIM information management and quality control. It introduces a layered approach that differentiates between different kinds of model checking. Each layer requires own checking knowledge and is typical managed by different authors or standardization bodies. It starts with generic low-level requirements about the file format and the data structure and is adding more and more domain specific or even project-dependent knowledge that ends with the client requirements (PoR). The presented approach is reusing existing standards and specifications and thus more easily enables to integrate available BIM tools. It represents a generic solution and is based on principles adopted from the IDM/MVD methodology defined by buildingSMART. Thus, it contributes to the use and improvement of the underlying mvdXML specification that was introduced for documentation purposes and definition of so called model views.

The framework presented in this deliverable describes a universal but abstract work- and data flow relevant for energy-efficient early design including the retrofitting scenario. It also shows relationships to other STREAMER tasks and work packages that are relevant for BIM information management and quality checks. They mainly deliver relevant domain knowledge and are doing further tests.

The focus of T5.1 activities is on collaboration support, more specifically on checking data exchange requirements that support information sharing between stakeholders. A main aspect is completeness of information derived from specific needs of design processes, such as the early design configuration or energy simulation. Other kinds of BIM knowledge such as on the data format layer (syntactic, structural compliance) or needed to carry out specific processes such as space layout (design rules) or other kinds of compliance checks (general consistency, regulations, program of requirements) are out-of-scope for T5.1 developments. The solutions developed in task 5.1 are based on agreements about the data flow between stakeholders. Both, the sender and receiver of information will be able to check if all agreements are fulfilled or if some data is missing. They can use the same specification and neutral checking tools to validate results, ideally before submitting requested data to the receiver. This will help to reduce “requests for information” due to incomplete datasets.

From a technical point of view the main research questions of the developed approach have been: (1) how to do neutral model checking based on open BIM and (2) how to capture and manage exchange requirements. It was decided to use and extend the existing mvdXML specification. mvdXML is a neutral standard released by buildingSMART, which is already used in the specification work of the IFC BIM standard. So far, the main focus of this standard has been on documentation purposes. STREAMER started to use mvdXML for model checking and was contributing to the release of the mvdXML 1.1 in spring 2016. Accordingly, the chosen approach is already embedded in buildingSMART developments thus leading to better acceptance by the industry.

An important feature of mvdXML is the use of configurable concept templates. They enable to specify generic units of functionality that on one hand support software implementation and on the other hand enable to configure specific needs of data exchange scenarios. This concept simplifies the definition and management of exchange requirements because they can be based on a set of configurable software functionalities.

While mvdXML is a technical solution to encode agreed requirements for IFC-based data exchange another challenge is to adequately capture domain knowledge that is normally discussed using the language of domain experts and thus needs to be translated into mvdXML or other IT specifications. STREAMER decided to capture and manage both types of specifications in order to support later maintenance where it might be necessary to revise and update requirements. Additionally, the same type of information might be relevant in different data exchange scenarios. The link to processes adds another configuration layer to exchange requirements. Conceptually, these aspects are supported by applying the IDM/MVD methodology.

Data exchange discussed in this deliverable is covering three main steps, comprising of: (1) client requirements for developing an initial design (PoR data), (2) basic room layout data to be used for energy simulation and (3) energy simulation results to be used for design evaluation and further design detailing. While this sequence is in principle relevant for new design as well as retrofitting scenarios the latter is more complex due to fact that there are more design constraints given by the shape of the existing building. Such constraints may need to be imported into the Early Design Configurator as an initial design layout that can be used as a starting point by the genetic algorithm for finding a better design solution. Additionally, exchange requirements are still influenced by used energy simulations tools that may not be able to deal with STREAMER specific properties such as the introduced labels. A similar situation is reported about energy results that may even not exported back to IFC and thus need a data post-processing to be comparable with results from other tools. Experiences with different tools show existing differences and the need for harmonization.

Several prototypes have been used to validate the developed approach and to show existing variety of model checking. The web-based BIM-Q tool has been used to capture exchange requirements and to generate adequate mvdXML specifications. mvdXML-based model checking has been implemented as a plug-in for the XBIM viewer offering a neutral solution for testing exchange requirements. It also supports the use of the BCF format to discuss and manage identified issues. Model checking has also been implemented in the EDC tool offering an import check of PoR data and design rule settings. Additionally, data export of the EDC already ensures a certain level of quality trying to fulfil data requirements for energy simulation. Accordingly, the architect who is using the EDC may fully rely on built-in checking capabilities of the EDC without seeing a need for doing further checks. However, if the EDC shall be replaced by another tool or if other stakeholders need to check their data then a neutral mvdXML-based checking solution that may even be used as a legal agreement for design collaboration offers an adequate solution.

Results of task 5.1 show how definition and checking of exchange requirements can be done. The developed approach is in line with buildingSMART developments and provides a sound basis for further specification work. Presented mvdXML specifications are still under revision and will be extended to reflect changes and extensions of the data flow. Ideally, requirements for the different energy simulation tools as well as expected result set are fully harmonized and thus independent from used tools. Such neutral specifications could then be input to further standardization activities.

References

Literature and Standards

BCF (2015) buildingSMART: BCF intro, http://www.buildingsmart-tech.org/specifications/bcf-releases, last access January 2015

BIM protocol for collaboration (in Dutch): http://www.pioneering.nl/SiteFiles/1/files/13010o10552__BIM%20Protocol%202_0%20%20v5.pdf

Böhms M. et al. 2008. D23: The SWOP Semantic Product Modelling Approach; With PMO – The SWOP Product Modelling Ontology. Deliverable of the SWOP project (STRP 016972).

CB-NL presentation: http://www.bouwinformatieraad.nl/wp-content/uploads/2015/08/IASS2015-CB-NL-Dik-Spekkink-20150819.pdf

Chipman, T. Liebich, T. & Weise, M. 2016. mvdXML – Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules. buildingSMART International Ltd. 15.02.2016.

Coins presentation (BIM exchange standard): http://www.bouwinformatieraad.nl/wp-content/uploads/2015/08/IASS2015-COINS-Renzo-van-Rijswijk-21050819.pdf

Deliverable 1.4: Multi-scale and multi-stage scenarios for energy-efficiency retrofitting, STREAMER Deliverable of work package 1, (to be released).

Deliverable 2.3: New design solutions of integrated EeB solutions for MEP and energy systems, STREAMER Deliverable of work package 2, (to be released).

Deliverable 2.6: New design solutions of energy-efficient building envelope and spatial configurations, STREAMER Deliverable of work package 2, (to be released).

Deliverable 3.6: Design decision-support and lifecycle validation tool, STREAMER Deliverable of work package 3, (to be released).

Di Giulio, R.: D1.2 Semantic typology model of existing buildings and districts, STREAMER Deliverable of work package 1, 3rd of September 2015

Dutch Revit standards: http://www.revitstandards.org/downloads/

DutchGovernmental buildings BIM Standard: http://www.rijksvastgoedbedrijf.nl/binaries/central-government-real-estate-agency/documents/publication/2014/07/08/rgd-bim-standard-v1.0.1-en-v1.0_2/Rgd_BIM_Standard_v1_0_1_EN_v1_0__2_.pdf

Explanation Dutch BIM standards: http://www.bouwinformatieraad.nl/wp-content/uploads/2015/05/kaart02-eng-v4.pdf

Hilaire, B. et al.: D.3.4 Simulation tool of the energy performance for newly designed and retrofitted buildings, STREAMER Deliverable of work package 3, Draft version August 2016

Juif, T. et al. (2015): D4.1 Framework for management of information flow, design actors and collaboration in virtual design and construction, STREAMER Deliverable of work package 4, 7th of Sept. 2015.

Nisbet, N. (2010) “BimServices – Command-line and Interface utilities for BIM”, http://www.aec3.com/en/6/6_04.htm (accessed 01-Jan-2016), including Transform1 for asset schema interoperability

Presentation Semantic Web: http://www.bouwinformatieraad.nl/wp-content/uploads/2015/08/IASS2015-SemanticWebTechForCIV2-Hans-Schevers-20150819.pdf

See, R.: Concept Design BIM 2010, IFC Model View Definition Diagram, 1st February 2010 http://www.blis-project.org/IAI-MVD/Snapshots/GSA-005_MVD_IFC2x3_Concept_Design_BIM_2010.pdf

Sleiman, H., Hempel, S.: D6.1 Configurator of workflow and building process requirements, STREAMER Deliverable of work package 6, 27th of August 2015

Terzaghi, F.: D7.5 Real case in Italy, Description and outlined design plan, STREAMER Deliverable of work package 7, 26th of February 2015

Weise M. et al.: D5.1 State-of-the-art review of advancements and challenges in ontology research, STREAMER Deliverable of work package 5, 28th of February 2015

Werensteijn, D., Di Guilio, R.: D1.6 Semantic baseline design model for new energy-efficient healthcare districts, STREAMER Deliverable of work package 1, 29th of August 2016 (Draft)

BIM-Q: Database accessible at (with user login): http://85.10.201.48:4570/

xBIM: Latest version available at: https://dl.dropboxusercontent.com/u/38347707/ReleaseV4.zip

CENtool:

Energy+: https://energyplus.net/

DesignBuilder: http://designbuilder.co.uk/

VABI Elements: https://www.vabi.nl/producten/vabi-elements/

SBEM: http://www.uk-ncm.org.uk/

Simergy: http://www.digitalalchemypro.com/html/products/DAProducts_Simergy.html

SimpleBIM: http://www.datacubist.com/

Trnsys: http://www.trnsys.com/

Appendix

mvdXML for PoR

The following mvdXML is checking existence of rooms and labels as described in chapter 4.1. It is based on a minimum set of concept templates (see highlighted `<Templates>` tag) that have been used in the first mvdXML export of BIM-Q. It was developed for the IFC4 release, and later adjusted to IFC2X3 for checking PoR data included in PoR-IFC files but also EDC-IFC files. Please note that specification in the appendix still shows the IFC4 release (please see `applicableSchema` entries).

The specification has been manually extended by checking all allowed enumeration values. Furthermore, it not only checks proper use of predefined label values but is also testing if requested properties are either attached to space instances (*IfcSpace*) or space type instances (*IfcSpaceType*). Whereas both ways to attach properties are supported, it was finally decided to map room requirements to *IfcSpace* instances only so that this feature is not really necessary for checking our STREAMER example data.

The presented mvdXML file includes a single Exchange Requirement only (see `<ExchangeRequirement>` tag). Checking of required label properties are defined by concepts being part of a `<ConceptRoot>` element. They are linked to the `<ExchangeRequirement>` element to specify whether a property is mandatory, optional or not required.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Template for strict export -->
<mvdXML xmlns="http://buildingsmart-tech.org/mvd/XML/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation="http://buildingsmart-
tech.org/mvd/XML/1.1 http://www.buildingsmart-tech.org/mvd/XML/1.1/mvdXML_V1.1_add1.xsd"
uuid="00000013-0000-0000-0000-000000000013" name="AEC3 Requirements Database Manager">
  <Templates>
    <ConceptTemplate uuid="00000000-0000-0000-0001-000000000001" name="ProductConceptTemplate"
applicableSchema="IFC4" applicableEntity="IfcProduct">
      <Definitions>
        <Definition>
          <Body lang="en"><![CDATA[Concept Template for any Product]]></Body>
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule RuleID="Name" AttributeName="Name"/>
        <AttributeRule RuleID="Description" AttributeName="Description"/>
        <AttributeRule RuleID="PredefinedType" AttributeName="PredefinedType"/>
        <AttributeRule AttributeName="IsDefinedBy">
          <EntityRules>
            <EntityRule EntityName="IfcRelDefinesByProperties">
              <References>
                <Template ref="10000000-0000-0000-0001-000000000001"/>
              </References>
            </EntityRule>
            <EntityRule EntityName="IfcRelDefinesByType">
              <References IdPrefix="T_">
                <Template ref="10000000-0000-0000-0001-000000000002"/>
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="HasAssociations">
          <EntityRules>
            <EntityRule EntityName="IfcRelAssociatesClassification">
              <References>
                <Template ref="10000000-0000-0000-0001-000000000004"/>
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
```

```
    <ConceptTemplate uuid="a322fdd7-cd28-4ea7-8797-f6cf124ab3d6" name="Partial Templates" ap-
plicableSchema="IFC4">
      <Definitions>
        <Definition>
          <Body lang="en"><![CDATA[Partial concept templates are described herein to indicate
usage of common data types, which are then incorporated into other templates.]]></Body>
        </Definition>
      </Definitions>
      <SubTemplates>
        <ConceptTemplate uuid="88b4aaa9-0925-447c-b009-fe357b7c754e" name="Properties" code=""
applicableSchema="IFC4" applicableEntity="IfcSimpleProperty">
          <Definitions>
            <Definition>
              <Body lang="en"><![CDATA[Properties may contain user-defined data, where data
types are open-ended.]]></Body>
            </Definition>
          </Definitions>
          <Rules>
            <AttributeRule RuleID="Property" AttributeName="Name">
              <EntityRules>
                <EntityRule EntityName="IfcIdentifier"/>
              </EntityRules>
            </AttributeRule>
            <AttributeRule AttributeName="Description">
              <EntityRules>
                <EntityRule EntityName="IfcText"/>
              </EntityRules>
            </AttributeRule>
          </Rules>
          <SubTemplates>
            <ConceptTemplate uuid="6655f6d0-29a8-47b8-8f3d-c9fce9c9a620" name="Single Value"
applicableSchema="IFC4" applicableEntity="IfcPropertySingleValue">
              <Rules>
                <AttributeRule RuleID="Property" AttributeName="Name">
                  <EntityRules>
                    <EntityRule EntityName="IfcIdentifier"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule AttributeName="Description">
                  <EntityRules>
                    <EntityRule EntityName="IfcText"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="Value" AttributeName="NominalValue">
                  <EntityRules>
                    <EntityRule EntityName="IfcValue"/>
                  </EntityRules>
                </AttributeRule>
              </Rules>
            </ConceptTemplate>
            <ConceptTemplate uuid="c148a099-c351-43a8-9266-5f3de0b45a95" name="Enumerated Va-
lue" applicableSchema="IFC4" applicableEntity="IfcPropertyEnumeratedValue">
              <Rules>
                <AttributeRule RuleID="Property" AttributeName="Name">
                  <EntityRules>
                    <EntityRule EntityName="IfcIdentifier"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule AttributeName="Description">
                  <EntityRules>
                    <EntityRule EntityName="IfcText"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="Value" AttributeName="EnumerationValues">
                  <EntityRules>
                    <EntityRule EntityName="IfcValue"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule AttributeName="EnumerationReference">
                  <EntityRules>
                    <EntityRule EntityName="IfcPropertyEnumeration">
                      <AttributeRules>
                        <AttributeRule RuleID="Property" AttributeName="Name">
                          <EntityRules>
                            <EntityRule EntityName="IfcLabel"/>
                          </EntityRules>
                        </AttributeRule>
```

```
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </Rules>
          </ConceptTemplate>
        </SubTemplates>
      </ConceptTemplate>
      <ConceptTemplate uuid="10000000-0000-0000-0001-000000000001" name="IfcRelDefinesBy-
Properties" applicableSchema="" applicableEntity="IfcRelDefinesByProperties">
        <Rules>
          <AttributeRule AttributeName="RelatingPropertyDefinition">
            <EntityRules>
              <EntityRule EntityName="IfcPropertySet">
                <References>
                  <Template ref="10000000-0000-0000-0001-000000000007"/>
                </References>
              </EntityRule>
              <EntityRule EntityName="IfcElementQuantity">
                <References>
                  <Template ref="10000000-0000-0000-0001-000000000008"/>
                </References>
              </EntityRule>
            </EntityRules>
          </AttributeRule>
        </Rules>
      </ConceptTemplate>
      <ConceptTemplate uuid="10000000-0000-0000-0001-000000000008" name="IfcElementQuantity"
applicableSchema="IFC4" applicableEntity="IfcElementQuantity">
        <Rules>
          <AttributeRule RuleID="Set" AttributeName="Name">
            <EntityRules>
              <EntityRule EntityName="IfcLabel"/>
            </EntityRules>
          </AttributeRule>
          <AttributeRule AttributeName="Description">
            <EntityRules>
              <EntityRule EntityName="IfcText"/>
            </EntityRules>
          </AttributeRule>
          <AttributeRule AttributeName="Quantities">
            <EntityRules>
              <EntityRule EntityName="IfcPhysicalSimpleQuantity">
                <AttributeRules>
                  <AttributeRule RuleID="Property" AttributeName="Name">
                    <EntityRules>
                      <EntityRule EntityName="IfcLabel"/>
                    </EntityRules>
                  </AttributeRule>
                  <AttributeRule RuleID="Value" AttributeName="CountValue"/>
                  <AttributeRule RuleID="Value" AttributeName="LengthValue"/>
                  <AttributeRule RuleID="Value" AttributeName="AreaValue"/>
                  <AttributeRule RuleID="Value" AttributeName="VolumeValue"/>
                  <AttributeRule RuleID="Value" AttributeName="WeightValue"/>
                  <AttributeRule RuleID="Value" AttributeName="TimeValue"/>
                </AttributeRules>
              </EntityRule>
            </EntityRules>
          </AttributeRule>
        </Rules>
      </ConceptTemplate>
      <ConceptTemplate uuid="10000000-0000-0000-0001-000000000007" name="Property Sets" ap-
plicableSchema="IFC4" applicableEntity="IfcPropertySet">
        <Rules>
          <AttributeRule RuleID="Set" AttributeName="Name">
            <EntityRules>
              <EntityRule EntityName="IfcLabel"/>
            </EntityRules>
          </AttributeRule>
          <AttributeRule AttributeName="Description">
            <EntityRules>
              <EntityRule EntityName="IfcText"/>
            </EntityRules>
          </AttributeRule>
          <AttributeRule AttributeName="HasProperties">
            <EntityRules>
              <EntityRule EntityName="IfcSimpleProperty">
```

```
                <References>
                  <Template ref="88b4aaa9-0925-447c-b009-fe357b7c754e"/>
                </References>
              </EntityRule>
            </EntityRules>
          </AttributeRule>
        </Rules>
      </ConceptTemplate>
       <ConceptTemplate uuid="10000000-0000-0000-0001-000000000002" name="IfcRelDe-
finesByType" applicableSchema="IFC4" applicableEntity="IfcRelDefinesByType">
        <Rules>
          <AttributeRule AttributeName="RelatingType">
            <EntityRules>
              <EntityRule EntityName="IfcTypeObject">
                <AttributeRules>
                  <AttributeRule RuleID="Name" AttributeName="Name"/>
                  <AttributeRule RuleID="Description" AttributeName="Description"/>
                  <AttributeRule RuleID="PredefinedType" AttributeName="PredefinedType"/>
                  <AttributeRule AttributeName="HasPropertySets">
                    <EntityRules>
                      <EntityRule EntityName="IfcPropertySet">
                        <References>
                          <Template ref="10000000-0000-0000-0001-000000000007"/>
                        </References>
                      </EntityRule>
                      <EntityRule EntityName="IfcElementQuantity">
                        <References>
                          <Template ref="10000000-0000-0000-0001-000000000008"/>
                        </References>
                      </EntityRule>
                    </EntityRules>
                  </AttributeRule>
                  <AttributeRule AttributeName="HasAssociations">
                    <EntityRules>
                      <EntityRule EntityName="IfcRelAssociatesClassification">
                        <References>
                          <Template ref="10000000-0000-0000-0001-000000000004"/>
                        </References>
                      </EntityRule>
                    </EntityRules>
                  </AttributeRule>
                </AttributeRules>
              </EntityRule>
            </EntityRules>
          </AttributeRule>
        </Rules>
      </ConceptTemplate>
      <ConceptTemplate uuid="10000000-0000-0000-0001-000000000004" name="IfcRelAssociates-
Classification" applicableSchema="IFC4" applicableEntity="IfcRelAssociatesClassification">
        <Rules>
          <AttributeRule RuleID="Code" AttributeName="Name">
            <EntityRules>
              <EntityRule EntityName="IfcLabel"/>
            </EntityRules>
          </AttributeRule>
          <AttributeRule AttributeName="RelatingClassification">
            <EntityRules>
              <EntityRule EntityName="IfcClassificationReference">
                <AttributeRules>
                  <AttributeRule RuleID="ClassificationReference_ReferencedSource" Attri-
buteName="ReferencedSource">
                    <EntityRules>
                      <EntityRule EntityName="IfcClassification">
                        <AttributeRules>
                          <AttributeRule AttributeName="Source">
                            <EntityRules>
                              <EntityRule EntityName="IfcLabel"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule RuleID="Table" AttributeName="Name">
                            <EntityRules>
                              <EntityRule EntityName="IfcLabel"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule AttributeName="ReferenceTokens">
                            <EntityRules>
                              <EntityRule EntityName="IfcIdentifier"/>
```

```
                          </EntityRules>
                        </AttributeRule>
                      </AttributeRules>
                    </EntityRule>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="Identification" AttributeName="Identification">
                  <EntityRules>
                    <EntityRule EntityName="IfcIdentifier"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="ClassificationReference_Identification" Attribute-
Name="Identification">
                  <EntityRules>
                    <EntityRule EntityName="IfcIdentifier"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="ClassificationReference_Name" AttributeName="Name">
                  <EntityRules>
                    <EntityRule EntityName="IfcLabel"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="ClassificationReference_Description" Attribute-
Name="Description">
                  <EntityRules>
                    <EntityRule EntityName="IfcText"/>
                  </EntityRules>
                </AttributeRule>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </SubTemplates>
</ConceptTemplate>
</Templates>
<Views><ModelView uuid="00000003-0000-0000-0000-000000000003" name="EU Streamer Model Re-
quirements - V3" applicableSchema="IFC4">
  <Definitions>
    <Definition>
      <Body><![CDATA[]]></Body>
    </Definition>
  </Definitions>
  <ExchangeRequirements>
    <ExchangeRequirement uuid="00000003-0000-0000-0000-000000000105" name="S03-P01 Early
Design by Briefbuilder : S01 Early Design" applicability="import">
      <Definitions>
        <Definition>
          <Body><![CDATA[[English]: Programme of Requirements]]></Body>
        </Definition>
      </Definitions>
    </ExchangeRequirement>
  </ExchangeRequirements>
  <Roots>
    <ConceptRoot uuid="00000003-0000-0000-2000-000000001141" name="PoR Spaces (as trans-
lated from CSV to IFC by AEC3 tool)" applicableRootEntity="IfcSpace">
      <Definitions>
        <Definition>
          <Body><![CDATA[Space object]]></Body>
        </Definition>
      </Definitions>
      <Concepts>
        <Concept uuid="00000003-0000-0000-0000-000000349910" name="Accessibility Labels">
          <Definitions>
            <Definition>
              <Body lang="en"><![CDATA[Scale depending on the position and the category of
users allowed to access]]></Body>
            </Definition>
          </Definitions>
          <Template ref="00000000-0000-0000-0001-000000000001"/>
          <Requirements>
            <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
          </Requirements>
          <TemplateRules operator="and">
            <TemplateRules operator="or">
```

```
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='AccessSecurity' AND Value[Value]='A1'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='AccessSecurity' AND Value[Value]='A2'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='AccessSecurity' AND Value[Value]='A3'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='AccessSecurity' AND Value[Value]='A4'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='AccessSecurity' AND Value[Value]='A5'"/>
                <TemplateRules operator="and">
                  <TemplateRules operator="nor">
                    <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='AccessSecurity'"/>
                  </TemplateRules>
                  <TemplateRules operator="or">
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='AccessSecurity' AND T_Value[Exists]='A1'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='AccessSecurity' AND T_Value[Exists]='A2'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='AccessSecurity' AND T_Value[Exists]='A3'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='AccessSecurity' AND T_Value[Exists]='A4'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='AccessSecurity' AND T_Value[Exists]='A5'"/>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000349922" name="Bouwcollege Layer">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='BouwcollegeLayer' AND Value[Value]='H'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='BouwcollegeLayer' AND Value[Value]='HF'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='BouwcollegeLayer' AND Value[Value]='I'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='BouwcollegeLayer' AND Value[Value]='O'"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='BouwcollegeLayer'"/>
                    </TemplateRules>
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='BouwcollegeLayer' AND T_Value[Value]='H'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='BouwcollegeLayer' AND T_Value[Value]='HF'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='BouwcollegeLayer' AND T_Value[Value]='I'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='BouwcollegeLayer' AND T_Value[Value]='O'"/>
                    </TemplateRules>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000349911" name="Comfort Class">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[Scale depending on the expedient to assure the
safety according to activities and functions]]></Body>
                </Definition>
```

```
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT1'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT2'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT3'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT4'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT5'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT6'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT7'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass' AND Value[Value]='CT8'"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='ComfortClass'"/>
                    </TemplateRules>
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT1'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT2'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT3'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT4'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT5'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT6'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT7'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='ComfortClass' AND T_Value[Value]='CT8'"/>
                    </TemplateRules>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000349912" name="Construction">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[Scale depending on requirements related to floor
strength, shielding against radiation, floor height, air tightness]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C1'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C2'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C3'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C4'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C5'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C6'"/>
```

```
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction' AND Value[Value]='C7'"/>
                <TemplateRules operator="and">
                  <TemplateRules operator="nor">
                    <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Construction'"/>
                  </TemplateRules>
                  <TemplateRules operator="or">
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C1'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C2'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C3'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C4'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C5'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C6'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Construction' AND T_Value[Value]='C7'"/>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000014624" name="Equipment">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[Scale depending on the type of function, high elec-
tric power needed, medical gasses, ITC data points]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ1'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ2'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ3'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ4'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ5'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ6'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment' AND Value[Value]='EQ7'"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='Equipment'"/>
                    </TemplateRules>
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ1'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ2'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ3'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ4'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ5'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ6'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='Equipment' AND T_Value[Value]='EQ7'"/>
                    </TemplateRules>
                  </TemplateRules>
```

```
              </TemplateRules>
            </TemplateRules>
          </Concept>
          <Concept uuid="00000003-0000-0000-0000-000000014628" name="Hygiene Class">
            <Definitions>
              <Definition>
                <Body lang="en"><![CDATA[Scale depending on the level of hygienic condition
requested in the different type of spaces]]></Body>
              </Definition>
            </Definitions>
            <Template ref="00000000-0000-0000-0001-000000000001"/>
            <Requirements>
              <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
            </Requirements>
            <TemplateRules operator="and">
              <TemplateRules operator="or">
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='HygienicClass' AND Value[Value]='H1'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='HygienicClass' AND Value[Value]='H2'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='HygienicClass' AND Value[Value]='H3'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='HygienicClass' AND Value[Value]='H4'"/>
                <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='HygienicClass' AND Value[Value]='H5'"/>
                <TemplateRules operator="and">
                  <TemplateRules operator="nor">
                    <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='HygienicClass'"/>
                  </TemplateRules>
                  <TemplateRules operator="or">
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='HygienicClass' AND T_Value[Value]='H1'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='HygienicClass' AND T_Value[Value]='H2'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='HygienicClass' AND T_Value[Value]='H3'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='HygienicClass' AND T_Value[Value]='H4'"/>
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='HygienicClass' AND T_Value[Value]='H5'"/>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </TemplateRules>
          </Concept>
          <Concept uuid="00000003-0000-0000-0000-000000011454" name="Number of occurrences">
            <Definitions>
              <Definition>
                <Body lang="en"><![CDATA[Example: There should be x rooms of a room
type.]]></Body>
              </Definition>
            </Definitions>
            <Template ref="00000000-0000-0000-0001-000000000001"/>
            <Requirements>
              <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
            </Requirements>
            <TemplateRules operator="and">
              <TemplateRules operator="or">
                <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Property[Value]='Oc-
currences' AND Value[Exists]=TRUE"/>
                <TemplateRules operator="and">
                  <TemplateRules operator="nor">
                    <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Prop-
erty[Value]='Occurrences'"/>
                  </TemplateRules>
                  <TemplateRules operator="or">
                    <TemplateRule Parameters="T_Set[Value]='STREAMER_PoR' AND T_Prop-
erty[Value]='Occurrences' AND T_Value[Exists]=TRUE"/>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </TemplateRules>
          </Concept>
```

```
            <Concept uuid="00000003-0000-0000-0000-000000043617" name="Required area">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[should be matched (Stefan 10% less, 30% more are
tolerable)]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Prop-
erty[Value]='GrossAreaPlanned' AND Value[Exists]=TRUE"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Prop-
erty[Value]='GrossAreaPlanned'"/>
                    </TemplateRules>
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_PoR' AND T_Prop-
erty[Value]='GrossAreaPlanned' AND T_Value[Exists]=TRUE"/>
                    </TemplateRules>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000043618" name="Room Type Requirement">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Property[Value]='Ob-
jectType' AND Value[Exists]=TRUE"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Prop-
erty[Value]='ObjectType'"/>
                    </TemplateRules>
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_PoR' AND T_Prop-
erty[Value]='ObjectType' AND T_Value[Exists]=TRUE"/>
                    </TemplateRules>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000343618" name="Functional Area Type">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Prop-
erty[Value]='FunctionalAreaType' AND Value[Exists]=TRUE"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_PoR' AND Prop-
erty[Value]='FunctionalAreaType'"/>
                    </TemplateRules>
```

```
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_PoR' AND T_Prop-
erty[Value]='FunctionalAreaType' AND T_Value[Exists]=TRUE"/>
                    </TemplateRules>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000014664" name="User Profile">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[Scale depending on the usage time of spaces and op-
erating hours]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRules operator="or">
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='UserProfile' AND Value[Value]='U1'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='UserProfile' AND Value[Value]='U2'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='UserProfile' AND Value[Value]='U3'"/>
                  <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='UserProfile' AND Value[Value]='U4'"/>
                  <TemplateRules operator="and">
                    <TemplateRules operator="nor">
                      <TemplateRule Parameters="Set[Value]='STREAMER_Labels_PoR' AND Prop-
erty[Value]='UserProfile'"/>
                    </TemplateRules>
                    <TemplateRules operator="or">
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='UserProfile' AND T_Value[Value]='U1'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='UserProfile' AND T_Value[Value]='U2'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='UserProfile' AND T_Value[Value]='U3'"/>
                      <TemplateRule Parameters="T_Set[Value]='STREAMER_Labels_PoR' AND T_Prop-
erty[Value]='UserProfile' AND T_Value[Value]='U4'"/>
                    </TemplateRules>
                  </TemplateRules>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="00000003-0000-0000-0000-000000014634" name="Name">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[[Id of an element that is typically used by humans
to identify elements.]]></Body>
                </Definition>
              </Definitions>
              <Template ref="00000000-0000-0000-0001-000000000001"/>
              <Requirements>
                <Requirement applicability="import" exchangeRequirement="00000003-0000-0000-
0000-000000000105" requirement="mandatory"/>
              </Requirements>
              <TemplateRules operator="and">
                <TemplateRule Parameters="Name[Exists]=TRUE"/>
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView></Views>
</mvdXML>
```

mvdXML for PoR, EDC and Energy Simulation

The following mvdXML was exported from BIM-Q and specifies all Exchange Requirements in a single mvdXML (see Figure 42). Accordingly, there are three `<ExchangeRequirement>` elements. This example enables to show how concepts can be linked to several Exchange Requirements, which is useful if same data is required in different ERs such as the labels that shall be included not only in the PoR dataset but also in the EDC output. The use of several ERs within the same mvdXML is shown in Figure 40.

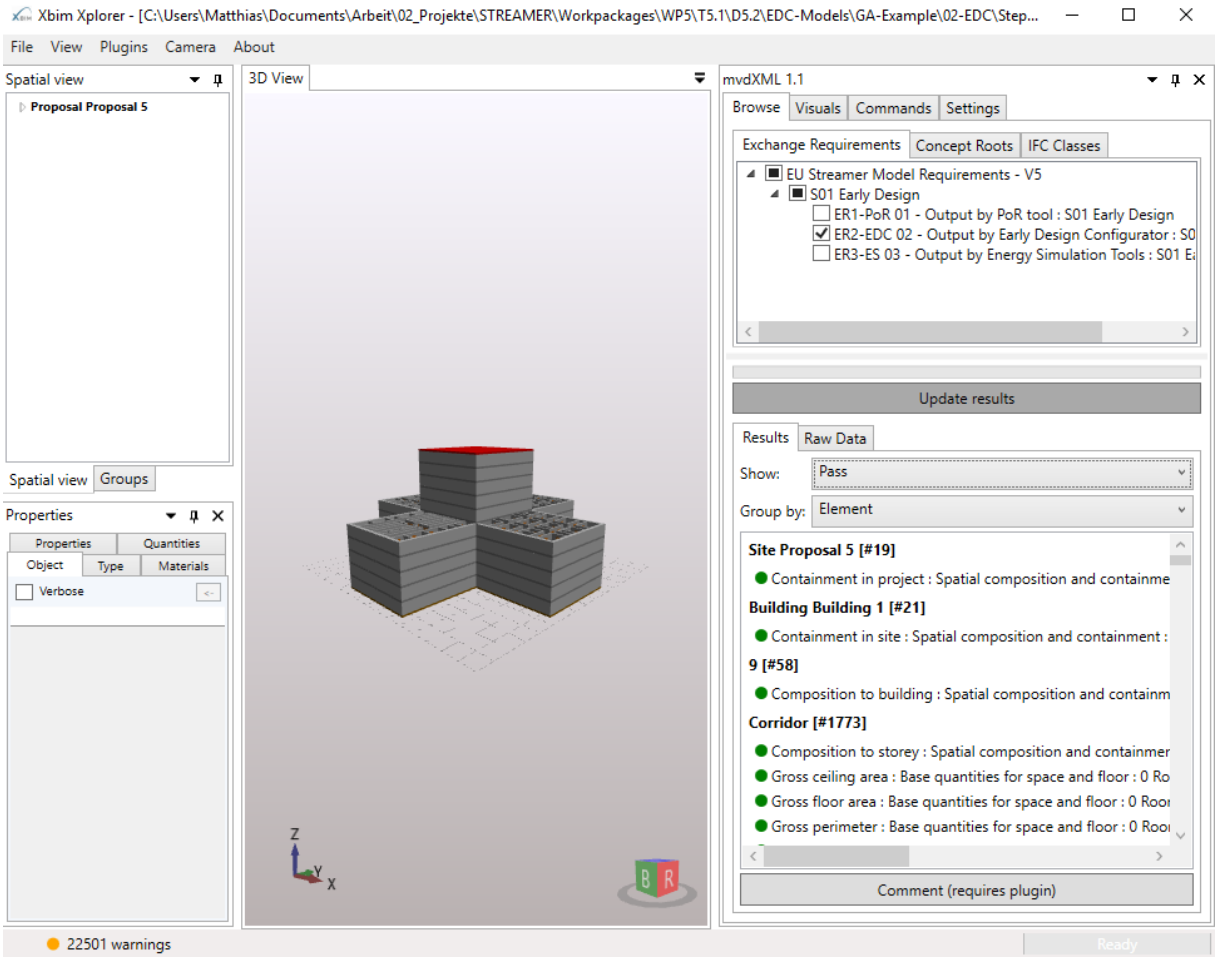


Figure 40: XBIM user interface for selecting one of the three ERs contained in the mvdXML file.

The used ConceptTemplates are derived from the latest IFC4 specification that are used in the buildingSMART software certification started in July 2016. Two ConceptTemplates have been added to deal with STREAMER requirements[5]: (1) a template for checking assignment of spaces to zones and (2) a template for checking existence of space boundaries. Furthermore, for sharing the requirement definition for PoR and EDC output data it was necessary to add an exception to the checking definition of space instances. This means to add a constraint to the applicability definition that excludes checking of “Corridor” space types that are produced by the EDC but do not have any room requirements (see Figure 41).

[5] STREAMER Labels can be defined using the existing property set template.

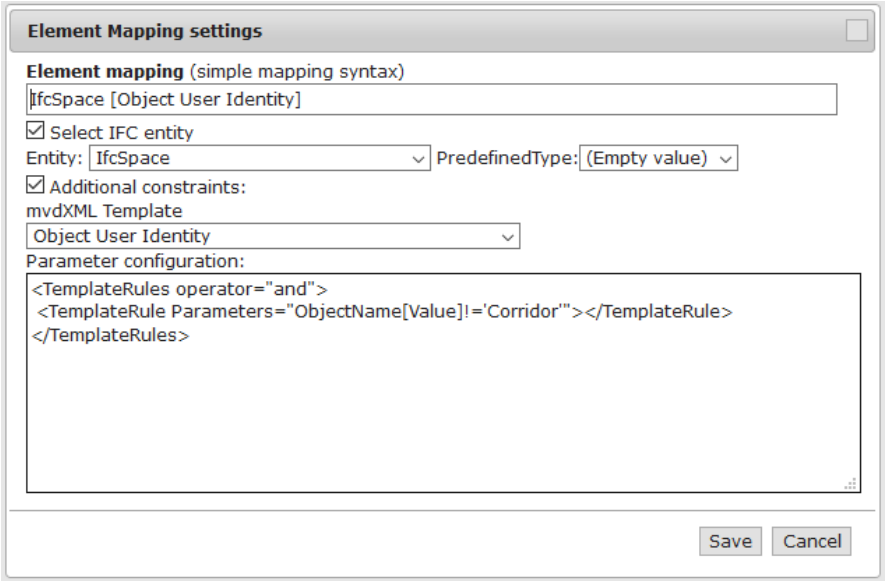


Figure 41: BIM-Q user interface for adding applicability constraints for checking space instances for PoR and EDC output.

The mvdXML export was finally adjusted by changing the schema name to support checking of IFC2X3 files as used in the STREAMER examples.

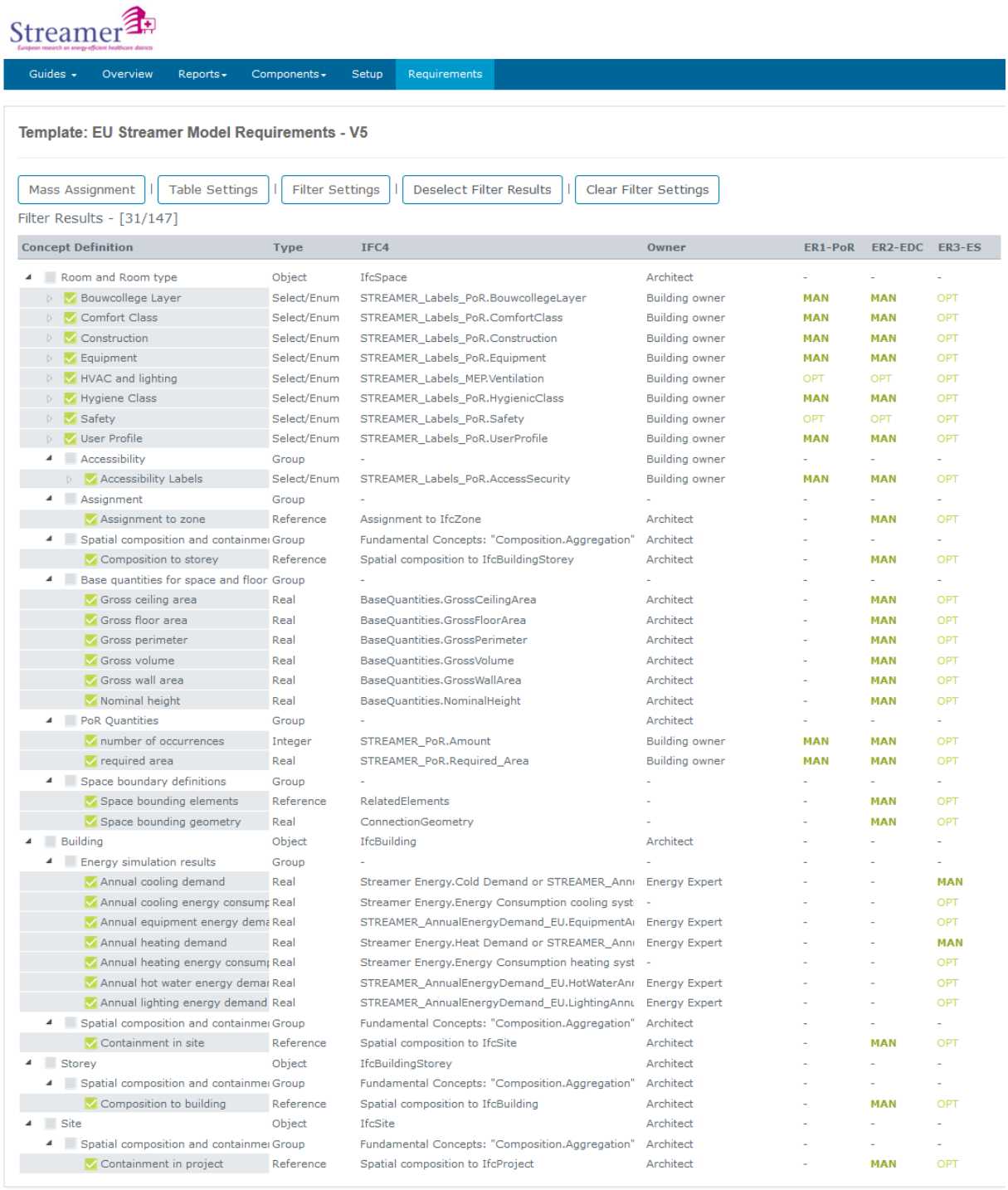


Figure 42: Required (MAN) data for the different Exchange Requirements.

```
<?xml version="1.0" encoding="UTF-8"?>
<mvdXML xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1" uuid="7e3a6d8d-913c-4505-a988-6ab72f6b6b9d" name="" status="sample"
xsi:schemaLocation="http://buildingsmart-tech.org/mvd/XML/1.1 http://www.buildingsmart-
tech.org/mvd/XML/1.1/mvdXML_V1.1_add1.xsd">
  <Templates>
    <ConceptTemplate uuid="805deb96-2684-4bc5-a9ad-3a29199dc023" name="Project Context" sta-
tus="sample" applicableSchema="IFC2X3" applicableEntity="IfcContext">
      <Definitions>
        <Definition>
          <Body><![CDATA]]></Body>
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="LongName">
          <EntityRules>
            <EntityRule EntityName="IfcLabel"/>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="ObjectType">
          <EntityRules>
            <EntityRule EntityName="IfcLabel"/>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Phase">
          <EntityRules>
            <EntityRule EntityName="IfcLabel"/>
          </EntityRules>
        </AttributeRule>
      </Rules>
      <SubTemplates>
        <ConceptTemplate uuid="ab1e2cfd-9f21-4f4a-b6af-cc26d84e45ac" name="Project Declara-
tion" status="sample" applicableSchema="IFC2X3" applicableEntity="IfcContext">
          <Definitions>
            <Definition>
              <Body><![CDATA]]></Body>
            </Definition>
          </Definitions>
          <Rules>
            <AttributeRule AttributeName="Declares">
              <EntityRules>
                <EntityRule EntityName="IfcRelDeclares">
                  <AttributeRules>
                    <AttributeRule RuleID="Type" AttributeName="RelatedDefinitions"/>
                  </AttributeRules>
                </EntityRule>
              </EntityRules>
            </AttributeRule>
            <AttributeRule AttributeName="Phase">
              <EntityRules>
                <EntityRule EntityName="IfcLabel"/>
              </EntityRules>
            </AttributeRule>
            <AttributeRule AttributeName="ObjectType">
              <EntityRules>
                <EntityRule EntityName="IfcLabel"/>
              </EntityRules>
            </AttributeRule>
            <AttributeRule AttributeName="LongName">
              <EntityRules>
                <EntityRule EntityName="IfcLabel"/>
              </EntityRules>
            </AttributeRule>
          </Rules>
          <SubTemplates>
            <ConceptTemplate uuid="634e27f7-7edb-4e22-b8cc-25f1cdc765ce" name="Project Type
Definitions" status="sample" applicableSchema="IFC2X3" applicableEntity="IfcContext">
              <Definitions>
                <Definition>
                  <Body><![CDATA[]]></Body>
                </Definition>
                <Definition>
                  <Body lang="tt"><![CDATA[]]></Body>
                  <Link lang="tt" category="definition" title="AUTOMATIC"
href=""><![CDATA[]]></Link>
                </Definition>
```

```
            </Definitions>
            <Rules>
              <AttributeRule AttributeName="Declares">
                <EntityRules>
                  <EntityRule EntityName="IfcRelDeclares">
                    <AttributeRules>
                      <AttributeRule RuleID="RelatedTypes" AttributeName="RelatedDefini-
tions">
                        <EntityRules>
                          <EntityRule EntityName="IfcTypeObject"/>
                        </EntityRules>
                      </AttributeRule>
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
              <AttributeRule AttributeName="Phase">
                <EntityRules>
                  <EntityRule EntityName="IfcLabel"/>
                </EntityRules>
              </AttributeRule>
              <AttributeRule AttributeName="ObjectType">
                <EntityRules>
                  <EntityRule EntityName="IfcLabel"/>
                </EntityRules>
              </AttributeRule>
              <AttributeRule AttributeName="LongName">
                <EntityRules>
                  <EntityRule EntityName="IfcLabel"/>
                </EntityRules>
              </AttributeRule>
            </Rules>
          </ConceptTemplate>
        </SubTemplates>
      </ConceptTemplate>
      <ConceptTemplate uuid="4ccaac0c-88f8-4c1d-91fd-2214d0e513c4" name="Project Units" sta-
tus="sample" applicableSchema="IFC2X3" applicableEntity="IfcContext">
        <Definitions>
          <Definition>
            <Body><![CDATA[]]></Body>
          </Definition>
          <Definition>
            <Body lang="tt"><![CDATA[]]></Body>
            <Link lang="tt" category="definition" title="AUTOMATIC"
href=""><![CDATA[]]></Link>
          </Definition>
        </Definitions>
        <Rules>
          <AttributeRule RuleID="HasUnits" AttributeName="UnitsInContext">
            <EntityRules>
              <EntityRule EntityName="IfcUnitAssignment">
                <AttributeRules>
                  <AttributeRule AttributeName="Units">
                    <EntityRules>
                      <EntityRule EntityName="IfcDerivedUnit">
                        <AttributeRules>
                          <AttributeRule RuleID="DerivedUnitType" AttributeName="UnitType">
                            <EntityRules>
                              <EntityRule EntityName="IfcDerivedUnitEnum"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule AttributeName="Elements">
                            <EntityRules>
                              <EntityRule EntityName="IfcDerivedUnitElement">
                                <AttributeRules>
                                  <AttributeRule AttributeName="Unit">
                                    <EntityRules>
                                      <EntityRule EntityName="IfcNamedUnit"/>
                                    </EntityRules>
                                  </AttributeRule>
                                </AttributeRules>
                              </EntityRule>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule AttributeName="UserDefinedType">
                            <EntityRules>
                              <EntityRule EntityName="IfcLabel"/>
```

```
        </EntityRules>
      </AttributeRule>
    </AttributeRules>
  </EntityRule>
  <EntityRule EntityName="IfcMonetaryUnit">
    <AttributeRules>
      <AttributeRule AttributeName="Currency">
        <EntityRules>
          <EntityRule EntityName="IfcLabel"/>
        </EntityRules>
      </AttributeRule>
    </AttributeRules>
  </EntityRule>
  <EntityRule EntityName="IfcSIUnit">
    <AttributeRules>
      <AttributeRule RuleID="SIUnitType" AttributeName="UnitType">
        <EntityRules>
          <EntityRule EntityName="IfcUnitEnum"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="Prefix">
        <EntityRules>
          <EntityRule EntityName="IfcSIPrefix"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule RuleID="SIUnitName" AttributeName="Name">
        <EntityRules>
          <EntityRule EntityName="IfcSIUnitName"/>
        </EntityRules>
      </AttributeRule>
    </AttributeRules>
  </EntityRule>
  <EntityRule EntityName="IfcConversionBasedUnit">
    <AttributeRules>
      <AttributeRule AttributeName="Dimensions">
        <EntityRules>
          <EntityRule EntityName="IfcDimensionalExponents"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule RuleID="ConversionUnitType" AttributeName="Unit-
Type">
        <EntityRules>
          <EntityRule EntityName="IfcUnitEnum"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule RuleID="ConversionUnitName" AttributeName="Name">
        <EntityRules>
          <EntityRule EntityName="IfcLabel"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="ConversionFactor">
        <EntityRules>
          <EntityRule EntityName="IfcMeasureWithUnit"/>
        </EntityRules>
      </AttributeRule>
    </AttributeRules>
  </EntityRule>
  <EntityRule EntityName="IfcConversionBasedUnitWithOffset">
    <AttributeRules>
      <AttributeRule AttributeName="Dimensions">
        <EntityRules>
          <EntityRule EntityName="IfcDimensionalExponents"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="UnitType">
        <EntityRules>
          <EntityRule EntityName="IfcUnitEnum"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="Name">
        <EntityRules>
          <EntityRule EntityName="IfcLabel"/>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="ConversionFactor">
        <EntityRules>
          <EntityRule EntityName="IfcMeasureWithUnit"/>
```

```
                          </EntityRules>
                        </AttributeRule>
                        <AttributeRule AttributeName="ConversionOffset">
                          <EntityRules>
                            <EntityRule EntityName="IfcReal"/>
                          </EntityRules>
                        </AttributeRule>
                      </AttributeRules>
                    </EntityRule>
                  </EntityRules>
                </AttributeRule>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Phase">
          <EntityRules>
            <EntityRule EntityName="IfcLabel"/>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="ObjectType">
          <EntityRules>
            <EntityRule EntityName="IfcLabel"/>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="LongName">
          <EntityRules>
            <EntityRule EntityName="IfcLabel"/>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="38dac6f0-997c-4544-9bca-b6326b9a3e4b" name="Project Representa-
tion Context" status="sample" applicableSchema="IFC2X3" applicableEntity="IfcContext">
      <Definitions>
        <Definition>
          <Body><![CDATA[]]></Body>
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="RepresentationContexts">
          <EntityRules>
            <EntityRule EntityName="IfcGeometricRepresentationContext">
              <AttributeRules>
                <AttributeRule RuleID="ContextIdentifier" AttributeName="ContextIdenti-
fier">
                  <EntityRules>
                    <EntityRule EntityName="IfcLabel"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule RuleID="ContextType" AttributeName="ContextType">
                  <EntityRules>
                    <EntityRule EntityName="IfcLabel"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule AttributeName="Precision"/>
                <AttributeRule AttributeName="CoordinateSpaceDimension">
                  <EntityRules>
                    <EntityRule EntityName="IfcDimensionCount"/>
                  </EntityRules>
                </AttributeRule>
                <AttributeRule AttributeName="WorldCoordinateSystem">
                  <EntityRules>
                    <EntityRule EntityName="IfcAxis2Placement3D">
                      <AttributeRules>
                        <AttributeRule AttributeName="Location">
                          <EntityRules>
                            <EntityRule EntityName="IfcCartesianPoint"/>
                          </EntityRules>
                        </AttributeRule>
                        <AttributeRule AttributeName="Axis">
                          <EntityRules>
                            <EntityRule EntityName="IfcDirection"/>
                          </EntityRules>
                        </AttributeRule>
                        <AttributeRule AttributeName="RefDirection">
                          <EntityRules>
```

```
                              <EntityRule EntityName="IfcDirection"/>
                            </EntityRules>
                          </AttributeRule>
                        </AttributeRules>
                      </EntityRule>
                    </EntityRules>
                  </AttributeRule>
                  <AttributeRule AttributeName="TrueNorth">
                    <EntityRules>
                      <EntityRule EntityName="IfcDirection"/>
                    </EntityRules>
                  </AttributeRule>
                  <AttributeRule AttributeName="HasSubContexts">
                    <EntityRules>
                      <EntityRule EntityName="IfcGeometricRepresentationSubContext">
                        <AttributeRules>
                          <AttributeRule RuleID="SubContextIdentifier" AttributeName="Con-
textIdentifier">
                            <EntityRules>
                              <EntityRule EntityName="IfcLabel"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule RuleID="SubContextType" AttributeName="Con-
textType">
                            <EntityRules>
                              <EntityRule EntityName="IfcLabel"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule AttributeName="TargetScale">
                            <EntityRules>
                              <EntityRule EntityName="IfcPositiveRatioMeasure"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule AttributeName="TargetView">
                            <EntityRules>
                              <EntityRule EntityName="IfcGeometricProjectionEnum"/>
                            </EntityRules>
                          </AttributeRule>
                          <AttributeRule AttributeName="UserDefinedTargetView">
                            <EntityRules>
```

(... 90 pages more ...)
The full file can be obtained through the authors.