OpenBIMRL –
An open format for code compliance checking of complex functional requirements from a regulation to the BIM model

Marcel Stepien[0000-0001-8970-5442], André Vonthron[0000-0001-7055-8200]
and Markus König[0000-0002-2729-7743]
Ruhr-Universität Bochum, Universitätsstraße 150, 44801 Bochum,
Germany Chair of Computing in Engineering
marcel.stepien@ruhr-uni-bochum.de

Abstract. In digital design methods of civil engineering projects, such as in Building Information Modeling (BIM) approaches, the procedure of validating specifications and regulations against a building model is commonly referred to as code compliance checking. In recent developments, initiatives and research projects focusing on the progressive harmonization and digitization of smart standards, open data formats have been increasingly investigated for enabling exchange of requirements between clients and contractors. The application of open formats for model checking can help to simplify and speed up certain processes, such as improving communication among stakeholders prior to the submission of building permits by providing the ability to semi-automatically check the quality of the data exchanged. However, the potentials of available standards are currently not utilized in full. Firstly, known standards are primarily checking semantics only, insufficiently utilizing available geometric definitions. Secondly, checking a model for complex functional requirements, such as finding escape routes in buildings, quickly exceeds the limitations of known open formats. This paper presents OpenBIMRL, an open format and potential approach to define and validate functional requirements to comply with the constraints and requirements in civil engineering.

1. Introduction

Modern and digital planning approaches using the Building Information Modelling (BIM) method incorporate 3D-based and semantic enriched building models. These models have a great potential to undergo automatic checking procedures, such as for checking regulatory requirements from building codes, but also to validate individual semantic and geometric requirements expected by the contractor. Therefore, software on the market provides functions to check information requirements and to apply complex and geometric checks on models, which can rapidly increase the model quality. However, these rules are stored in a proprietary and not humanly readable format. Moreover, when the need for adjustments of individual rulesets arises, it is often necessary to use programming interfaces. To make exchange and automatic checking of regulatory requirements work in a sustainable manner, an open and easy-to-use data format is required.

The standardization in terms of defining and validating models is greatly advancing. The latest standard, the Level of information need (LOIN) (DIN, 2021) allows to describe BIM requirements in a standardized way, including level of information (LOI), level of geometry (LOG) and documentation (DOC). Some of these requirements are directly interdependent and rely on each other across norms and regulations. Basic constraints can be provided in a technical language, from which concrete rules can be derived in a conditional notation ("if..., then..."). A specification is applied to a BIM model by translating the textual form of the requirements into a collection of explicit properties and property sets that specify terms, values, and definitions for specific elements within the model, or by creating geometric representations constrained by specific criteria such as position or dimensions of the elements. However, specific terms with the same terminology can have different meaning in specific contexts and lead to different implications. They cannot be applied directly to a BIM model. The reason for this is that the regulatory and administrative formulation of the technical requirements provide a lot of freedom for interpretation. That can be solved by creating individual modelling guidelines to supplement the original documents and prepare the verbalization for technical implementation on the model. A modelling guideline contain specific use-case related property set definitions for specific model elements. Also, requirements to the geometric representation can be defined, such a as a specific LOG.

To exchange BIM models in an open and standardized format, the Industry Foundation Classes (IFC) are used. In this format the LOI is realized as element properties and the LOG refers to the assigned geometric representation. That allows to have a holistic model with clearly defined semantic parts and geometry information as a basis for an integrated data exchange. Because the information delivery depends on user requirements, a conformity check of the BIM model is necessary, which can be performed with code compliance checking.

Depending on the complexity and profundity of applicable requirements, variants of rules and rulesets for code compliance checking can be defined. A requirement can be called complex if it includes the necessity of interpretation of specific information, such as concluding if a certain geometry is classified on a specific LOG. The profundity is referring to the depths of entangled information that either build upon each other or must be processed in junction, such as determining the building classification first to calculate requirements for fire resistance properties. Common code compliance checking methods struggle to allow these depths of interconnected requirements and their interpretations to be systematically processed.

When executing code compliance checking, two checking stages can be derived. On the one hand, a formal rule checking is performed, which validates the LOI against the definitions from a modelling guideline if required properties exist and have admissible values and data types. On the other hand, a technical rule checking is performed, including the investigation of sophisticated model relations, and using geometric operations to calculate complex measures.

To define formal model rules, the Model View Definitions (MVD) allow to define explicit requirements to a specific filtered entity in IFC models. However, MVD turned out to be very complex to implement and, therefore, to be not well accepted. Consequently, the Information Delivery Specification (IDS) has been developed by buildingSMART International to provide a simpler data format, which currently is in pre-standardization phase and already is becoming implemented by many software vendors.

In terms of technical rule checking, the need for an open format for a more simplistic and methodical approach to check technical requirements. Solving the technical requirements of norms and regulations by machine-readable rules is a necessary consequence in the implementation of the BIM method to enable a simpler exchange of regulatory requirements. This includes the processing of model geometries or the creation and calculation of additional geometric annotations (e.g. showing escape routes or measuring areas). Currently, that can only be partly realised by adopting and combining predefined rulesets in existing rule checking software. The user can create custom rule checking logic by interacting and changing parameters in the development interfaces but still depends on a black-boxed checking procedure. The use of proprietary systems can impose difficulties on users, such as limiting software interoperability and creating requirements that cannot be exchanged.

To deliver an open and standardizable format, this paper proposes the OpenBIM Rule Language (abbr. OpenBIMRL), an XML-based open data structure for defining technical requirements and performing automated code compliance checking on IFC models. It provides a framework, to investigate BIM models by code compliance checking and pre-calculation on the models provided information, such as extract and calculate intermediate results, and subsequently solving the overall requirement applying propositional logic. Pre-calculations are defined by using a Visual Programming approach. The graph-based formalization allows users to define model-based workflows consisting of complex calculations and functional subflows. The generated results are resolved and terminated by a concatenation of logical operators. This enables a straightforward and clear arranged understanding and sustainable digitalisation of complex of regulations. In this paper, the applicability is demonstrated and evaluated on two examples from the German Template building code.

2. Related Research

Rule languages provide the ability to formalize requirements to a BIM-based model, which can be then automatically checked. However, the available languages and formats do not equally cover all the criteria that are considered required for practical implementation and application of rule checking. Therefore, Solihin et al. (2019) proposed eleven distinct criteria. Those criteria contain classifications, e.g. for the availability of a standards schema, language expressiveness, the ease of defining rules or the support of complex rules. Furthermore, several studies introduce types of checks by distinguishing and dividing between approaches (Ismail et al., 2017; Solihin and Eastman, 2015). Those are crediting the increasing complexity and solution orientation of rule checking approaches as a key driver for this distinction. In general, rule-checking techniques can be divided into multiple categories, such as *coded rule-checking*, *rule-checking by querying* and *dedicated rule languages*, with researchers comparing strengths and weaknesses (Pauwels and Zhang, 2015). No clear favourite could be identified from the research, as this depends mainly on the user's preferences and use case, but there is a clear trend towards *dedicated rule languages*, which tend to present the most flexible of solutions. To enable open exchange of rules, it is necessary to utilize the potentials of open formats and standards.

For the definition of formal rule checks on IFC models, some open formats are used in state-of-the-art approaches (Jaud and Muhič, 2022; Olsson et al., 2018). The Model View Definitions (MVD) (Liebich et al., 2021) and the upcoming Information Delivery Specification (IDS) (buildingSMART Technical, 2022; van Berlo et al., 2019) present approaches to query and check IFC-based models. In particular, IDS has been identified as an enabler for checking requirements in future developments of upcoming releases of IFC (Berlo et al., 2021).

For the definition of technical rule checks, numerous approaches have been proposed. First, BIMQL (Mazairac and Beetz, 2013) is a query language that specializes in IFC model data and can formalize rules in a syntax similar to SQL. Second, GraphQL can formalize descriptive paths within a database as JSON-based queries, which can be used to resolve linked building data sets (Hartig and Pérez, 2018; Werbrouck et al., 2019). Also, a whole portfolio of technologies are available under the RuleML terminology (Boley et al., 2010) for querying and validating XML-based data sets. These are only conceptual query languages that also require a programming-like formalization of specifications. Another approach is the use of graph-based data structures, which provide more freedom to represent complex data.

The Semantic Web technologies are presenting such a graph-based structure, on which code compliance checking can be performed, using the SPARQL Query Language for RDF (SPARQL, 2018) and Shapes Constraint Language (SHACL, 2018). These technologies cannot handle BIM-based models directly and require a conversion to the Resource Description Framework (RDF), an XML-based data store consisting of triples in the form of subject-predicate-object expressions (RDF 1.1 Concepts and Abstract Syntax, 2018). An RDF structure can be formalized by utilizing the Web-Ontology Language (OWL) (Bechhofer et al., 2004), a data model serving as a descriptor for RDF data. Yurchyshyna and Zarli presented an ontology-based method for the formalisation and application of construction conformance requirements for effective code checking by utilizing sematic web technologies (Yurchyshyna and Zarli, 2009) and Beach et al. presented a semantic framework to automatically check regulatory compliance (Beach et al., 2015). Using semantic Web Technologies for BIM-based rule checking, however, provide some challenges. When converting to RDF, large data sizes are produced and very complicated substructures are created. Thus, some research focuses on implementing shortcuts functions, such as realized in BIMSPARQL (Zhang et al., 2018). Also, replacement of certain IFC elements into alternative representations are performed. Such is the case for geometries, which can partially be replaced by Well-Known Text (WKT) representations (Guo et al., 2021; O'Donovan et al., 2019; Pauwels et al., 2017). Regardless of these challenges, the use of semantic web technologies is the most discussed approach for implementation of code compliance checking.

Instead of considering a generalised approach by using semantic web technologies, in research the idea of using a graph-based rules for specifically checking building code had already been conceived. The KBimCode approach (Kim et al., 2019; Song et al., 2019) introduces the KBVL (KBim Visual Language), a graph based rule checking tool that evaluated the approach for the Koran building code. The KBVL definitions are translated to script that formalizes the graph by a set of conditions. The rules are subsequently executed by a scripting language that operates in combination to the KBVL.

In addition to being graph based, the Visual Code Checking Language (VCCL) (Preidel, 2020; Preidel and Borrmann, 2015; Preidel and Borrmann, 2016) demonstrates that a visual-driven rule language can allow for a more simpler definition of technical requirements. The VCCL is designed to perform Automated Code Compliance Checking (ACCC) explicitly for the formal description of checking for conformance against norm and building codes, i.e. for checking formal requirements. Individual nodes in a VCCL rule check represent atomic functions, which also recognizes the possibility that more complex requirements can be checked and still be presented in a way that is understandable to the user. This is enabled, for example, through the implementation of geometry operators. The VCCL is the closest concept for open rule-based compliance checking of requirements from norm and regulations.

3. Methodology

The methodology is based on a clear distinction between the checking of formal and technical requirements. Both require different approaches for implementation and checking of rules. This paper focuses on checking technical requirements, however, checking the formal requirements is considered as a prerequisite. This separation, in formal and technical rule checking, is a fundamental idea behind the framework that enables technical requirements to be checked.

The framework is divided into four sequential steps (Figure 1), with data handover processes located between those steps. This data contains filtered elements and reports to rule checks. The first step is part of the pre-calculation, which performs graph-based computations of complex algorithms using model data as parametric input. It incorporates functions, such as element filters and mathematical operators as well as the creation of temporary helper-geometries (e.g. polygons of required areas).

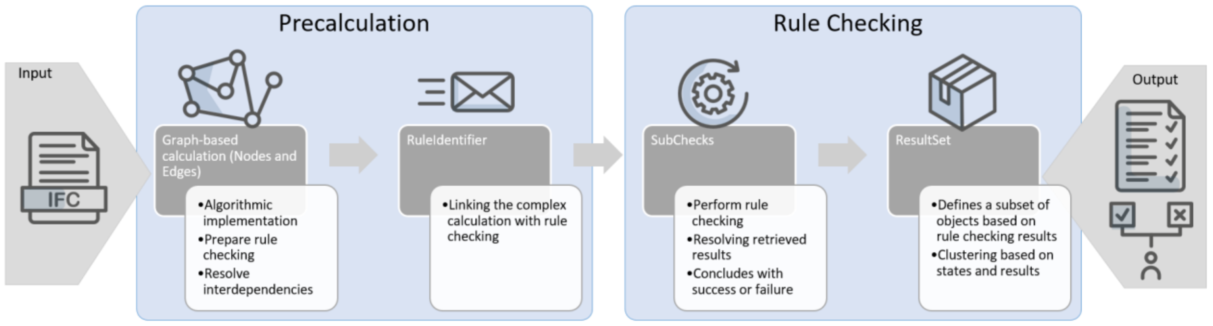

Input
IFC
Precalculation
Graph-based calculation (Nodes and Edges)
•Algorithmic implementation
•Prepare rule checking
•Resolve interdependencies
RuleIdentifier
•Linking the complex calculation with rule checking
Rule Checking
SubChecks
•Perform rule checking
•Resolving retrieved results
•Concludes with success or failure
ResultSet
•Defines a subset of objects based on rule checking results
•Clustering based on states and results
Output

Figure 1 Framework of creating a format for checking technical requirements

This part of the framework (the pre-calculation) is similar to the VCCL and KBVL concepts previously introduced. In fact, in can be argued that this step is interchangeable with those and similar approaches. The generalised and open nature of the OpenBIMRL graph-schema allows for approaches like VCCL and KBVL to be represented and mapped on data structure, be it by class inheritance, by name and type classification or schema alignment. However, a complete rule check should conclude with a clear statement (true/success or false/failure), a necessary step that must follows up the pre-calculation. Therefore, in OpenBIMRL, results of the pre-calculation are linked and resolved by a rule checking interface.

Specifically, the results of this pre-calculation are favourable data structures, such as lists and single values, that are then marked as rule identifiers, to be accessed in the subsequent checking procedure. That ensures a systematic transfer of model data into a holistic rule checking approach. The rule checking itself consists of nested rules (e.g. sub checks), which helps resolving dependencies between sets of information and to conclude results into an overall statement. A statement is either true or false depending on whether the rules could be verified. Finally, these results (e.g. result sets) can be interpreted as rule checking reports that are reflecting the state of information provided in the BIM model. These results then enable a visualization of affected model elements.

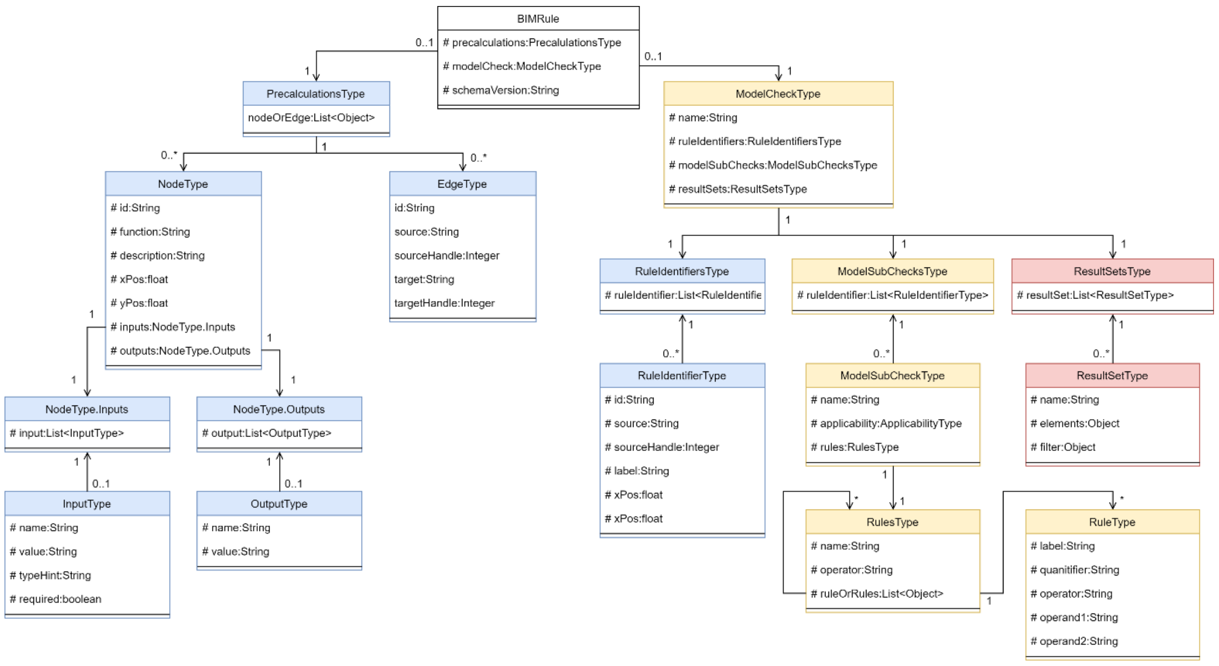


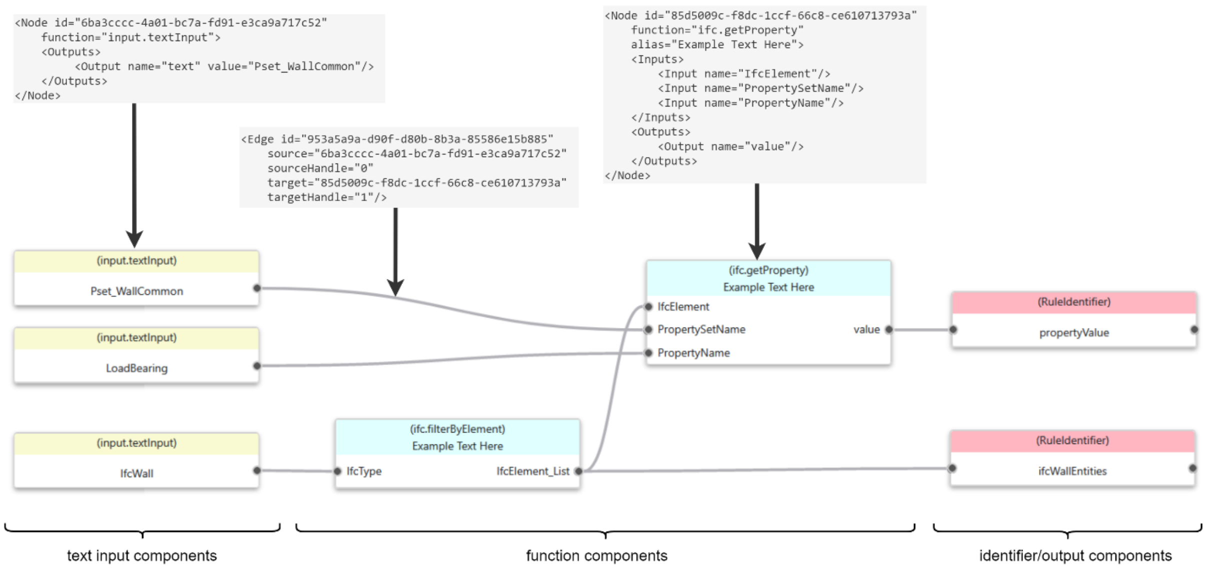
Figure 2 Overview of the OpenBIMRL data schema as UML class diagram, showing related structures for graph pre-calculations (blue), rule checking (yellow) and the retrieval of result sets (red)

As the data model of the methodology, the OpenBIM Rule Language (OpenBIMRL) is introduced, an open format implementation of the described framework (Figure 2). The format is a solution for the current need of defining technical requirements for IFC based models in an open format. The language uses a directed graph-based approach for defining sets of rules, so that arbitrary functions can be linked as components and can be subsequently executed in a cascaded manner to retrieve geometric and semantic model information. The schema contains templates for defining generic functions (nodes) that are required to be implemented by rule checking applications to solve the specifics of the ruleset. Therefore, rulesets can be defined completely independent from any software implementation.

The pre-calculation graph consists of three distinct types of node components, which are required for creating technical requirements.

1. *Function Component*, which are nodes that perform specific operations on incoming data. Each NodeType element defines a function component.
2. *Identifier Component*, which serves as a placeholder that transfer data from the graph-based calculation to the rule checking. These are referenced by RuleIdentifierTypes.
3. *Input Component*, which are nodes that allow direct data input, such as texts or numbers. These are designed to allow user interaction and are a special case of function components, therefore, also defined using NodeType elements.

All nodes are connected by edges forming a directed acyclic graph (Figure 3). The edges represent the dataflow between nodes. An edge connects a specific output of a node (source) with an input of another node (target). Depending on the functionality, specific in- or output can be left open, meaning they are either optional or assumed to be default a certain value. The *number of in- and outputs*, their *labels*, the *function name* and *namespace* are essentially the signature of the node, which alone serves to distinguish between different implementations of certain functions. In addition, the nodes are provided with a unique identifier (e.g., a Universally Unique Identifier, UUID) that enables explicit identification and tracking of the implemented functional nodes. Standardization and unification of these functions will reinforce the reliability of the implemented rules in all applications that use the format.



```
<Node id="6ba3cccc-4a01-bc7a-fd91-e3ca9a717c52"
    function="input.textInput">
    <Outputs>
         <Output name="text" value="Pset_WallCommon"/>
    </Outputs>
</Node>
```

```
<Edge id="953a5a9a-d90f-d80b-8b3a-85586e15b885"
    source="6ba3cccc-4a01-bc7a-fd91-e3ca9a717c52"
    sourceHandle="0"
    target="85d5009c-f8dc-1ccf-66c8-ce610713793a"
    targetHandle="1"/>
```

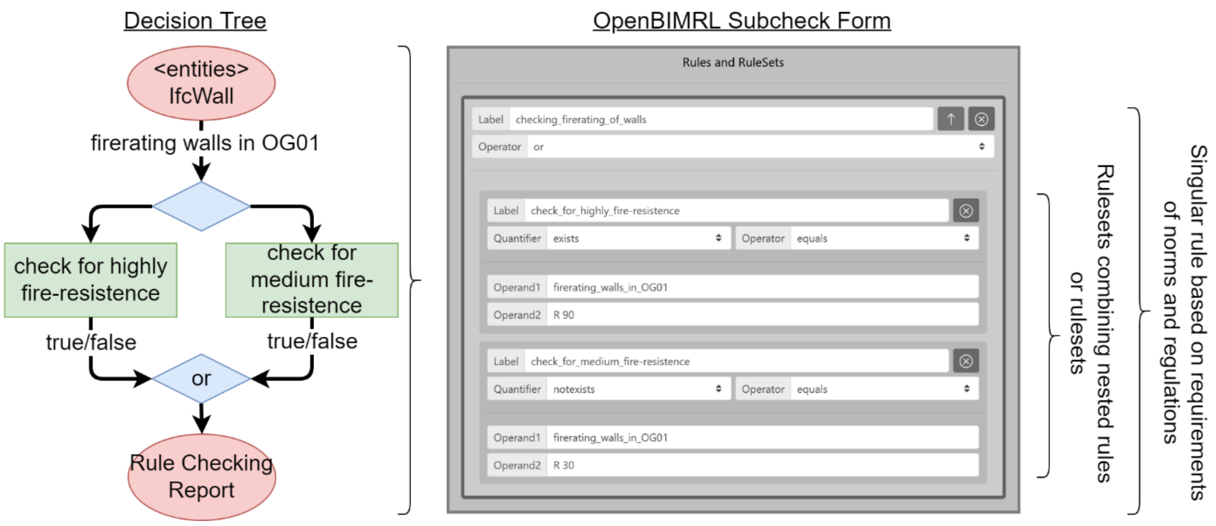
```
<Node id="85d5009c-f8dc-1ccf-66c8-ce610713793a"
    function="ifc.getProperty"
    alias="Example Text Here">
    <Inputs>
        <Input name="IfcElement"/>
        <Input name="PropertySetName"/>
        <Input name="PropertyName"/>
    </Inputs>
    <Outputs>
        <Output name="value"/>
    </Outputs>
</Node>
```

Figure 3 A visual example of the component graph performing complex calculation prior to rule checking, presenting input components (yellow), function components (blue) and identifier components (red)

Depending on the requirement to be checked, a complete graph can quickly reach high complexity. However, node clustering and nested graph execution can help structuring the rule checking and mitigating the rising complexity.

The results of the individual function nodes can be stored in identifier components, which can be directly accessed from sub-checks of the rule checking structure (Figure 4). In OpenBIMRL, a sub-check is defined as a nested collection of ruleset and rule. A ruleset requires an operator to solve the nested requirements. A singular rule additionally requires a quantifier, describing the strategy in case a list of values will be iterated upon. Such a singular rule is based on a constraint or requirement formalized in norms and regulations.

The labels of identifier components are used as operands in a rule, binding the stored result to the rule left- or right-side operand. Each check is concluded with a statement (true or false), concluding the whole collection of rulesets and rules in a sub-check. Multiple sub-checks can be defined and processed simultaneously if a defined applicability allows for variants of approaches and alternative solutions.


Decision Tree
<entities>
IfcWall
firerating walls in OG01
check for highly fire-resistence
check for medium fire-resistence
true/false
true/false
or
Rule Checking Report
OpenBIMRL Subcheck Form
Rules and RuleSets
Rulesets combining nested rules or rulesets
Singular rule based on requirements of norms and regulations

Label	checking_firerating_of_walls
Operator	or

Label	check_for_highly_fire-resistence		
Quantifier	exists	Operator	equals
Operand1	firerating_walls_in_OG01		
Operand2	R 90		

Label	check_for_medium_fire-resistence		
Quantifier	notexists	Operator	equals
Operand1	firerating_walls_in_OG01		
Operand2	R 30		

Figure 4 Defining the rule checking on precalculated results, as formal decision tree (left) and implemented data structure form in OpenBIMRL (right)

4. Case Study

To demonstrate the applicability of the proposed approach, two cases from the German Template building code (abbr. in German as MBO) were implemented. These are the checks of the building classification (MBO §2) and the escape routes (MBO §33). However, the OpenBIMRL approach is not limited to check requirements of the BIM domain, but also allows for integration of GIS specific constraint. Additional domains can simply be addressed by considering specialised node libraries, that handle the multi-model integration.

The implementation itself contains of a *creator-tool*, which provides visual web-environment that enables the drag and drop creation of the graph and rulesets defining an OpenBIMRL document. The modelling of those graphs using such a visual modelling tool helps reviewing the rule checking results and provides visual feedback. Interacting with such an editor is an essential part of the user experience while creating or deriving technical requirements into rules. Secondly, an *engine* contains a set of implemented function nodes and enables the execution of the OpenBIMRL document against an IFC model. As a prerequisite, the conducted BIM models are expected to have already passed a preliminary formal check.

4.1 Checking the Building Classification

The MBO comprehensively relies on the building classification information defined in §2. Most of the relevant model requirements are related to the building classification, such as permissible values for fire resistance properties of building elements. The MBO distinguishes between five building classifications, which each requires a specific building height, maximum gross areas and maximum counts of building units. Additionally, information about the location and environment are also required to specific classifications. If the values comply with defined thresholds, a building classification can be determined, which is used to further investigate the provided properties in the model.

The calculation that needs to be performed requires the filtering of model elements and the examining of geometries. Initially, the gross areas are solved grouped by the individual building units and types. Then the required building height is measured starting from the average surface height of terrain model and the finish floor hight of the highest level of human stay. Finally, functions for counting, sorting, splitting and joining lists of elements are necessary to process the calculations to determine the data for the subsequent rule checking.

The pre-calculation graph can quickly reach a high level of complexity for such a technical requirement. However, the graph can be divided into functional routines, each of which solves parts of the prerequisites for the rule check. For example, the sub-routine that solves the calculation of the building height requires a combination of *addition*-, *subtraction*-, *filterBy*- and *get*-functions components (Figure 5). The resulting building height is compared against the requirements of the building code to find the valid building classification. Those requirements can, for example, be formalised using a Nassi-Shneiderman diagram and then can be translated into OpenBIMRL compliant rules (Figure 6).

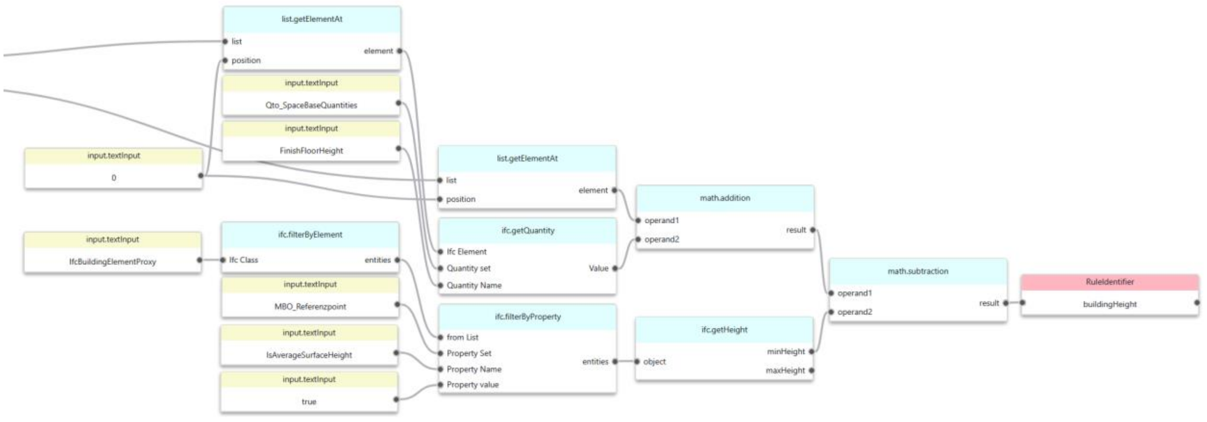
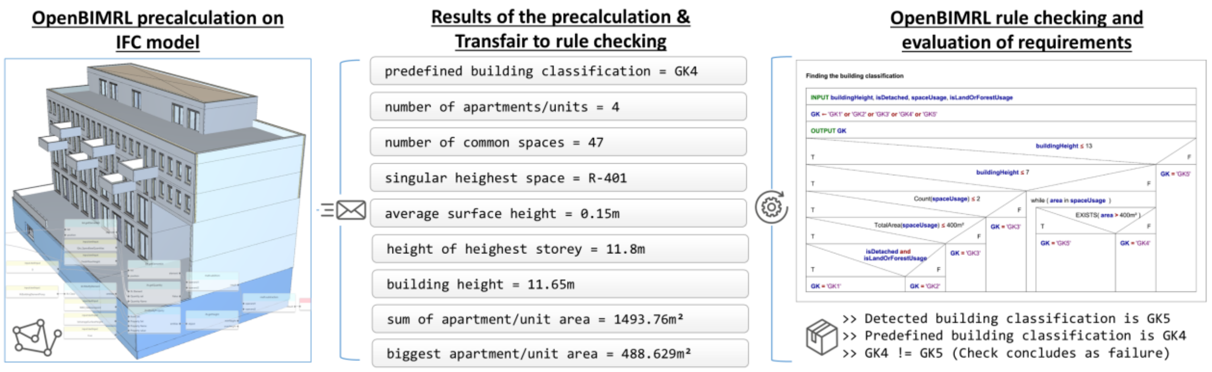


Figure 5 OpenBIMRL sub-routine for calculating the building height based on an average surface height and finish floor height


OpenBIMRL precalculation on IFC model
Results of the precalculation & Transfair to rule checking
OpenBIMRL rule checking and evaluation of requirements
Finding the building classification
INPUT buildingHeight, isDetached, spaceUsage, isLandOrForestUsage
GK ← 'GK1' or 'GK2' or 'GK3' or 'GK4' or 'GK5'
OUTPUT GK
buildingHeight ≤ 13
T
F
buildingHeight ≤ 7
GK = 'GK5'
Count(spaceUsage) ≤ 2
while (area in spaceUsage)
TotalArea(spaceUsage) ≤ 400m²
GK = 'GK3'
EXISTS(area > 400m²)
isDetached and isLandOrForestUsage
GK = 'GK3'
GK = 'GK5'
GK = 'GK4'
GK = 'GK1'
GK = 'GK2'

```
predefined building classification = GK4
number of apartments/units = 4
number of common spaces = 47
singular heighest space = R-401
average surface height = 0.15m
height of heighest storey = 11.8m
building height = 11.65m
sum of apartment/unit area = 1493.76m²
biggest apartment/unit area = 488.629m²
```

```
>> Detected building classification is GK5
>> Predefined building classification is GK4
>> GK4 != GK5 (Check concludes as failure)
```

Figure 6 Checking the building classification in OpenBIMRL, exemplified by the process of the rule checking framework. The processed IFC model (left) results into a set of input values (middle) that are required for rule checking (right).

4.2 Escape Route Check

The second example are the escape routes defined in MBO §33. It describes that the first and second escape routes are dependent on the exits and room constellation of a storey. Specifically, an escape route must start at the farthest corner within a building unit (e.g. an apartment) that considers human stay. The first escape route either must end to a door leading out of the building or to a door leading into a save area, such as a main stairwell. Second escape routes, however, can also end to a ladder able window that is accessible by firefighters. Furthermore, the lengths of these routes are restricted to a maximal allowed length of 35 meters.

For the application of this technical requirement, several geometric dependencies must be solved. For example, an escape route is planned by going around wall-corners and through doors. Some elements and room-objects must be attributed with properties that mark them as endpoints, obstacles or relevancy to pass through. Accordingly, a set of filter operation will be required to find these elements and their data. Since the possibilities for calculating escape routes allow for various approaches, in this case, a graph-theoretical approach has been performed. Therefore, a network of positions and paths is created across an entire storey to determine possible routes. The focus of the rule checking, then, is to verify the lengths of routes.

To find escape routes, the essential sub-routine is the creation of a graph-based network that is the basis for the application of a path search algorithm, which in this case, the A*-algorithm for shortest paths in a non-directed graph is used. This sub-routine requires a set of geometries to be processed, such as the representations of IfcSpaces and IfcDoors to create the points of the network and of IfcWalls to filter and remove intersecting paths (e.g. edges as line geometry). These geometric dependencies are solved by primarily utilizing a *checkLinecasts*-function node on the created and collected edges against the IfcWall-entities, solving the intersection between geometries and removing all edges which lead into a wall structure. Several collection function nodes, such as *flattenCollection* and *joinCollection*, are then utilized to prepare and sort function node results. A visual representation of the network, on top of calculated distances between points, is visualized in Figure 7. The precision of results, specifically of the calculated distances and found paths, is dependent on the parametrics of the graph-based network, which can be manipulated to align for optimal constraints (e.g. see *textInput*-nodes in). This way, the flexibility afforded by the norms and regulations is integrated into the rules giving the user the freedom to adjust certain variables.

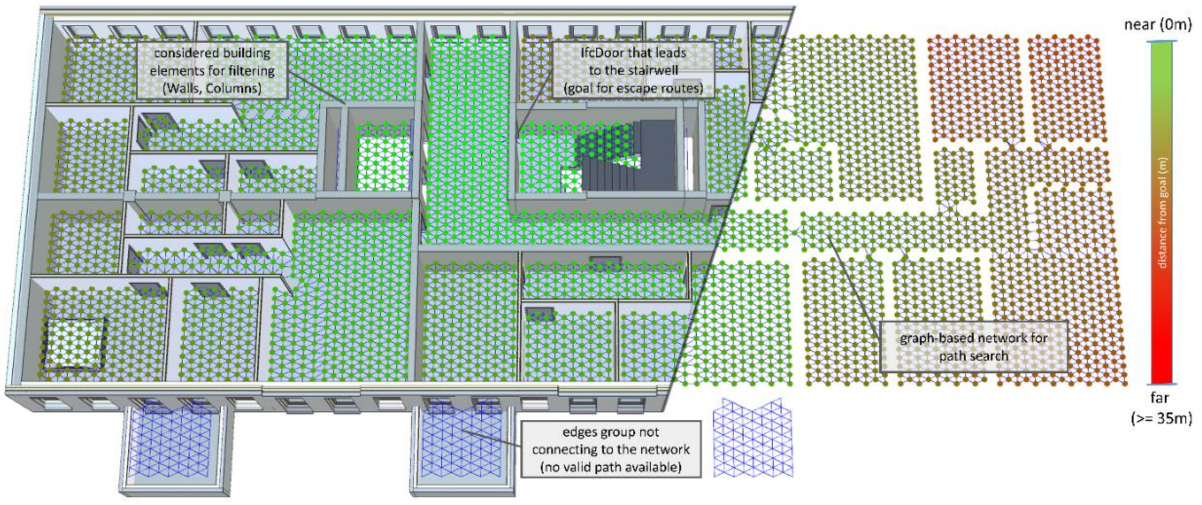

considered building elements for filtering (Walls, Columns)
IfcDoor that leads to the stairwell (goal for escape routes)
near (0m)
distance from goal (m)
graph-based network for path search
far (>= 35m)
edges group not connecting to the network (no valid path available)

Figure 7 The created network of positions and paths, with filtered out wall structures. Marking distances as far (in red) and near (in green) towards a potential escape through the door leading into a stairwell.

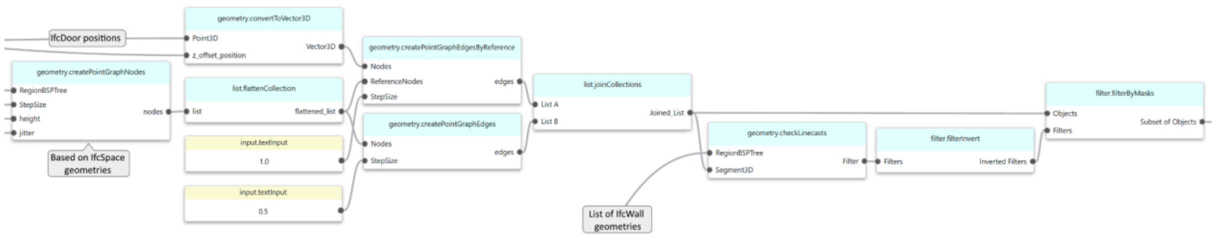


Figure 8 OpenBIMRL sub-routine to create the network for finding potential escape routes that will be further processed in the rule checking.

5. Conclusion

Because of the need for an open and transparent exchange of information, methods for a semi-automated checking of building data are becoming essential. The examination of formal requirements is already handled and standardized by several technologies and formats, such as the Model View Definitions (MVD) and the upcoming Information Delivery Specification (IDS). However, current technologies and formats either do not sufficiently support the validation of technical requirements, or their formalization requires explicit programming skills. In addition, potential rule languages must translate BIM-based models and documents into compliant data structures for the implementation of technical requirements. This is a time-consuming and mostly error-prone process, which can be mitigated by directly incorporating IFC-based data into the rule checking process, such as performed in the formal rule checking.

Building on a conceptualized framework and the distinction between formal and technical rule checking, this paper proposes the open format OpenBIMRL. This format allows formalizing technical requirements, by processing model dependencies using functional nodes in a graph-based structure. The resulting insights lead to decisive statements on the quality of information delivery. OpenBIMRL simplifies the rule definition approach of rather complex requirements by systematically utilizing the advantages of visual programming (e.g. Graph to Model Execution), providing an easier entry barrier for BIM users. Furthermore, the OpenBIMRL format is generalised in such a way, that known concepts for graph-based rule checking can easily adapt the data model, aiming to create a basis for code compliance checking going forward. The interpretation of terminologies and requirements in norms and regulations is resolved, by processing the results of those graph-based calculations in an examination similar to formal rule checking approaches.

The method is currently only evaluated on the German building code. Consequently, formalising any type of construction rule and regulation to be applied on BIM models require large scale in-depth formalisation of rules from all kinds of regulations. In future developments, standardization of individual components can help to implement a distinct rule language and formalization of the technical requirements. The understanding of complex correlations in the model can be greatly improved, if fundamental functions are defined and utilized for computation that are able to systematically validate dependent technical requirements. To enable the transition to a full rule language, the OpenBIMRL data model must be challenged against more and diverse sets of requirements. Their formalisations will help shaping the languages specifics.

6. References

Beach, T. H., Rezgui, Y., Li, H. and Kasim, T. (2015) ‘A rule-based semantic approach for automated regulatory compliance in the construction sector’, *Expert Systems with Applications,* vol. 42, no. 12, pp. 5219–5231.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L. A. and others (2004) ‘OWL web ontology language reference’, *W3C recommendation,* vol. 10, no. 2, pp. 1–53.

Berlo, L., Tauscher, H., Liebich, T., Kranenburg, A. and Paasiala, P. (2021) ‘Future of the Industry Foundation Classes: towards IFC 5’, *Proc. of the Conference CIB W78,* pp. 11–15.

Boley, H., Paschke, A. and Shafiq, O. (2010) ‘RuleML 1.0: the overarching specification of web rules’, *International Workshop on Rules and Rule Markup Languages for the Semantic Web,* pp. 162–178.

buildingSMART Technical (2022) *Information Delivery Specification IDS - buildingSMART Technical* [Online]. Available at https://technical.buildingsmart.org/projects/information-delivery-specification-ids/ (Accessed 16 November 2022).

DIN (2021) *17412-1: Building Information Modelling - Level of Information Need - Part 1: Concepts and principles*: Beuth Verlag.

Guo, D., Onstein, E. and La Rosa, A. D. (2021) ‘An Improved Approach for Effective Describing Geometric Data in ifcOWL through WKT High Order Expressions’, *An Improved Approach for Effective Describing Geometric Data in ifcOWL through WKT High Order Expressions*.

Ismail, A. S., Ali, K. N. and Iahad, N. A. (2017) ‘A review on BIM-based automated code compliance checking system’, *2017 International Conference on Research and Innovation in Information Systems (ICRIIS),* pp. 1–6.

Jaud, Š. and Muhič, S. (2022) ‘Checking IFC with MVD Rules in Infrastructure: A Case Study’, *Engineering Proceedings,* vol. 17, no. 1, p. 33.

Kim, H., Lee, J.-K., Shin, J. and Choi, J. (2019) ‘Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking’, *Journal of Computational Design and Engineering,* vol. 6, no. 2, pp. 143–148.

Liebich, T., Chipman, T., Weise, M., Geiger, A. and Muhic, S. (2021) *mvdXML - V1.2: Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules* [Online]. Available at https://raw.githubusercontent.com/buildingSMART/mvdXML/master/mvdXML1.2/mvdXML_V1-2-Draft9.pdf.

Mazairac, W. and Beetz, J. (2013) ‘BIMQL-An open query language for building information models’, *Advanced Engineering Informatics,* vol. 27, no. 4, pp. 444–456.

O’Donovan, J., O’Sullivan, D. and McGlinn, K. (2019) ‘A method for converting IFC geometric data into GeoSPARQL’, *CEUR Workshop Proceedings,* pp. 7–20.

Olsson, P.-O., Axelsson, J., Hooper, M. and Harrie, L. (2018) ‘Automation of building permission by integration of BIM and geospatial data’, *ISPRS International Journal of Geo-Information*, vol. 7, no. 8, p. 307.

Pauwels, P., Krijnen, T., Terkaj, W. and Beetz, J. (2017) ‘Enhancing the ifcOWL ontology with an alternative representation for geometric data’, *Automation in construction*, vol. 80, pp. 77–94 [Online]. DOI: 10.1016/j.autcon.2017.03.001.

Pauwels, P. and Zhang, S. (2015) ‘Semantic Rule-checking for Regulation Compliance Checking: An Overview of Strategies and Approaches’.

Preidel, C. (2020) *Automated compliance checking of digital building models regarding applicable standards and guidelines using a visual programming language,* Technische Universität München [Online]. Available at https://mediatum.ub.tum.de/1534486.

Preidel, C. and Borrmann, A. (2015) ‘Automated code compliance checking based on a visual language and building information modeling’, *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, p. 1.

Preidel, C. and Borrmann, A. (2016) ‘Towards Code Compliance Checking on the basis of a Visual Programming Language’, *ITcon*, vol. 21, pp. 402–421.

RDF 1.1 Concepts and Abstract Syntax (2018) [Online]. Available at https://www.w3.org/TR/rdf11-concepts/ (Accessed 16 November 2022).

Shapes Constraint Language (SHACL) (2018) [Online]. Available at https://www.w3.org/TR/shacl/ (Accessed 16 November 2022).

Solihin, W. and Eastman, C. (2015) ‘Classification of rules for automated BIM rule checking development’, *Automation in construction*, vol. 53, pp. 69–82.

Song, J., Kim, J. and Lee, J. K. (2019) ‘Converting KBImCode into an executable code for the automated design rule checking system’, *Intelligent and Informed-Proceedings of the 24th International Conference on Computer-Aided Architectural Design Research in Asia*, pp. 795–804.

SPARQL Query Language for RDF (2018) [Online]. Available at https://www.w3.org/TR/rdf-sparql-query/ (Accessed 16 November 2022).

van Berlo, L., Willems, P. and Pauwels, P. (2019) ‘Creating information delivery specifications using linked data’, *Advances in ICT in Design, Construction and Management in Architecture, Engineering, Construction and Operations (AECO) : Proceedings of the 36th CIB W78 2019 Conference*, pp. 647–660.

Yurchyshyna, A. and Zarli, A. (2009) ‘An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction’, *Automation in construction*, vol. 18, no. 8, pp. 1084–1098 [Online]. DOI: 10.1016/j.autcon.2009.07.008.

Zhang, C., Beetz, J. and Vries, B. de (2018) ‘BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data’, *Semantic Web*, vol. 9, no. 6, pp. 829–855.