Adding an extensive feedback loop for the creation and validation of functional requirements of BIM models

Florian M. Becker[1] and Daniel Napps[1]
[1]Chair of Computing in Engineering, Ruhr-University Bochum, Universitätsstraße 150, 44801 Bochum, Germany
E-mail(s): info@florianbecker.eu, daniel.napps@rub.de

Abstract: During the planning, design and permit process of buildings, various regulations must be adhered to in order to obtain a compliant building permit. Therefore formal requirements for buildings and technical rules in practice are examined and can be checked. For this purpose the open specification for checking functional requirements of a digital building model OpenBIMRL can be used. Architects or engineers may verify their BIM model with a pre-defined set of rules or entire rule-sets. This may be a set of rules for specific countries or usage (housing, commercial). In OpenBIMRL, functional requirements are formulated using a web-based platform separated from the actual rule checking engine. These rules consist of two main components: information extraction through predefined building blocks called $Nodes$, followed by the application of logical operations and comparisons. Previously, testing and verification of these rules on BIM models required a separate, non-portable program. To address this limitation, we conceived a redesign of the engine into a pure dependency structure devoid of user interaction. A server-client architecture was chosen with a RESTful endpoint. This change enables rule creators and end-users to view the outputs of defined rules and their respective $Nodes$ directly in a concise format and within the 3D model itself.
Keywords: Building Information Modeling (BIM), functional requirements, automation

Erschienen in Tagungsband 35. Forum Bauinformatik 2024, Hamburg, Deutschland, DOI: 10.15480/882.13550

1 Introduction

Securing approval for construction projects involves navigating a complex environment of regulations that must be strictly adhered to. These regulations are essential for ensuring that the final construction is safe, legal, and fits its intended purpose. The consequences of non-compliance can be significant, ranging from delays in schedule delays to potential legal implications. Building Information Modeling (BIM) is a powerful tool that is now being used by engineers, architects, and government agencies. In recent years the German federal government has begun to introduce $DigitalerBauantrag$ (digital building permit) where BIM is used extensively in the planning process. In the long term, the goal is to replace conventional paperwork completely [1].

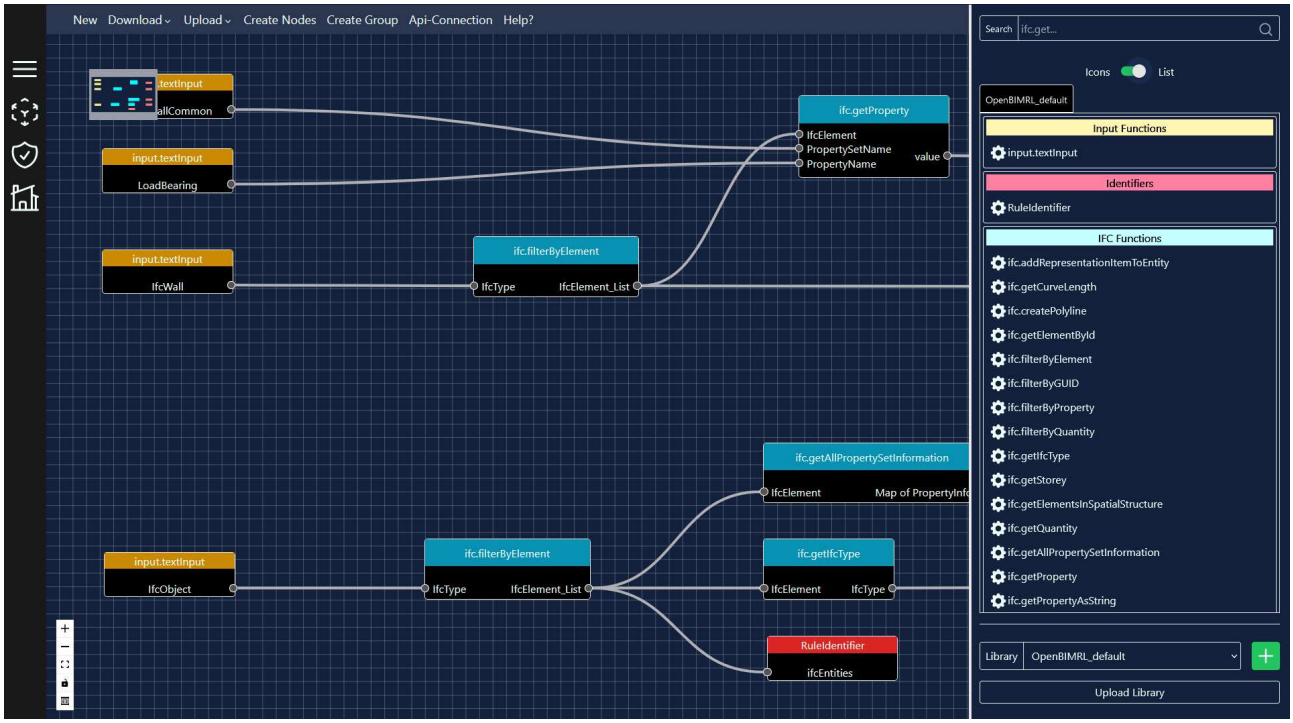

Figure 1: Example of OpenBIMRLs' graph structure.

Permitting agencies still need to verify and review the digital building model for compliance with the specific regulations. Many of these vary from state to state or country to country. BIM Rule Language (BIMRL) aims to address this complexity by defining a Domain-Specific Language (DSL) for rule checking [2]. This automated compliance checking tool is the basis of OpenBIMRL, an open format for code compliance checking of complex functional requirements which takes a different approach with a graph-based DSL [3]. OpenBIMRL, utilizing a graph-based language (see Figure 1), is considered as a low-code or no-code solution. While technical knowledge is still essential, the barrier to entry for staff without prior programming expertise (non-technical staff) is significantly reduced. The implementation was designed as a prototype or proof of concept and therefore did not take usability into account. Therefore, we decided to redesign it in order to support digitization in this area in the future.

2 State of the art and related works

The construction industry has long been confronted with the challenge of ensuring compliance with a multitude of regulations, standards, and building codes. The intricacy of these regulations, coupled with the dynamic nature of construction projects, necessitates the development of robust and efficient rule-checking mechanisms. Over the years, various approaches have been devised and refined in order to address these challenges. In recent times, the advent of digital technology has had a profound impact on the field of rule checking within the context of the construction industry. BIM has emerged as a crucial tool, allowing for the integration of regulatory requirements directly into the design and construction process. The research project MBO2BIM focused on preparing BIM-based inspections and creating a unified foundation for the automated verification of building code requirements using BIM-based inspection tools [4]. Another research project about occupational safety knowledge

for workplace planning deals with the digitization of regulations and the establishment of rules for workplaces e.g. checking escape routes [5]. Additional a study that examines a BIM-based method for automated code-checking of fire safety compliance in Canadian timber buildings, focusing on the National Building Code of Canada [6]. These and many other related works highlight the importance of digital code compliance checking for BIM models and the construction industry. Inspecting related works concerning rule checking BIMRL Simplified Schema (BIMRLSS) can be found in [7]. While promising some of the same aspects as OpenBIMRL and with BIMRL as both their basis (BIMRLSS and BIMRL are from the same authors) the approach to their specific DSL is different from the start. OpenBIMRL was initially designed as a graph-based language, and BIMRL(SS) has considered using a visual programming language, but there are no open source implementations yet. [8]. Their program structure also differs. As for the state of the art the comparison of rule-based checking systems by Solihin, Dimyadi, and Lee [8] highlights several key factors that distinguish different approaches to BIM-based automated rule checking. Their analysis, includes criteria such as language openness, practical applicability, and integration capabilities. According to their findings, OpenBIMRL stands out for its modular design and ease of integration with other software systems (Table 1). As to be explained in detail, our revised solution can be integrated into existing code bases in a variety of ways.

Table 1: Eleven-criteria metrics assessment. Modified according to [8].

Language or tool	**1. Formal or standard schema for the rule definition**	**2. Language expressiveness**	**3. Ease of defining rules**	**4. Minimum logic or programming constructs**	**5. Minimum domain knowledge required**	**6. Support for complex rules**	**7. Integrated geometry engine**	**8. Performance indication**	**9. Openness**	**10. Interface to other languages and systems**	**11. Level of maturity**
Legend: - Low; O Moderate; + High; N.A. Not Applicable; ? Unknown											
NLP	N.A.	N.A.	N.A.	-	-	-	N.A.	?	-	-	-
[...]											
BIMRL	N.A.	+	+	+	+	O	O	+	+	O	O
[...]											
Dynamo + Revit	N.A.	-	+	O	-	O	O	O	O	-	+
[...]											
[a]Solibri Model Checker (SMC)	N.A.	N.A.	N.A.	N.A.	O	O	+	+	N.A.	N.A.	+
OpenBIMRL	+	O	+	+	-	+	+	-	+	+	O
[b]BIM.Permit	O	-	+	O	-	+	+	+	+	O	+
[a]SMC is included here due its popularity even though it is not strictly language-based rule checking system											
[b]BIM.Permit uses the OpenBIMRL											

3 Methodology

Since this project employs an open format and implementation, the entire development process is accessible and can be reviewed on GitHub[1].

3.1 The reference Implementation

The reference implementation consisted of a non open source Industry Foundation Classes (IFC) viewer called *apstex IFC Framework* which as of the time of writing did not receive an update in over four years and does not comply to the latest IFC standards, specifically $IFC4.3ADD2$ [9], [10]. Portability was also not considered for the initial draft and users with different Operating Systems (OS) experienced difficulties using the software in their respective environments.

3.2 General Design

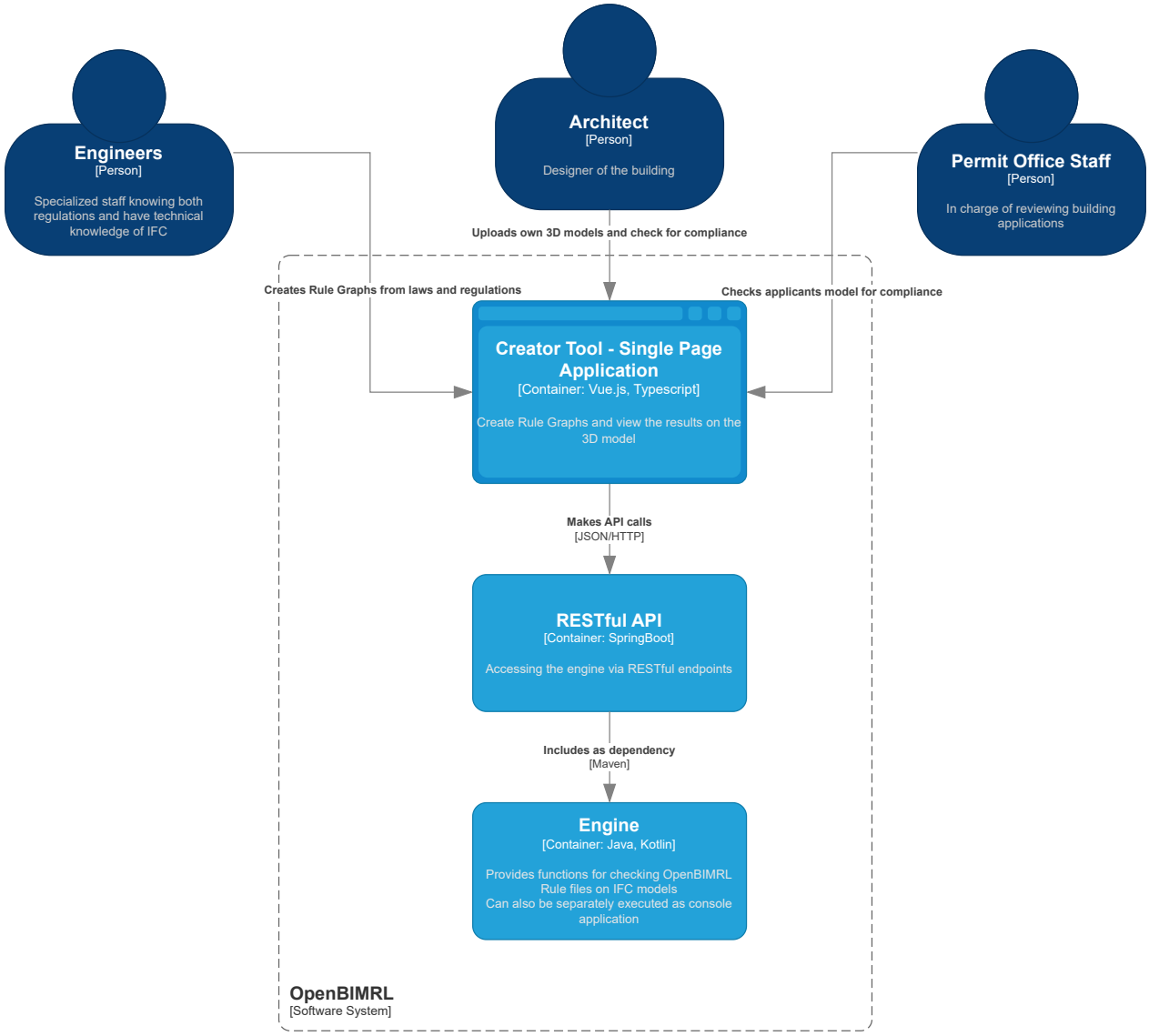

Figure 2: C4 Container diagram of OpenBIMRL.

[1] https://github.com/OpenBimRL/

We have opted for a containerized application with a web-frontend (Creator-Tool) shown in Figure 2, for two main reasons: First, containerization addresses the previously mentioned issue of operating system differences by encapsulating the application and its dependencies within containers, so that a consistent runtime environment across various OS is ensured. This guarantees that our OpenBIMRL implementation operates uniformly regardless the underlying OS. Second, containerization significantly enhances reproducibility. Packaging the application along with its complete environment in containers ensures that the exact setup can be recreated across different systems and by different user(-groups) regardless the previous configuration of their system. This approach not only enhances the reliability but also supports a broader audience by providing an instantaneous repeatable and verifiable environment. For Graphical User Interface (GUI) we chose a web-frontend (client) which is a common choice when using a RESTful API on the backend (server). There are many benefits when targeting the browser as GUI due to the long and ongoing process of standardising ECMAScript (also known as JavaScript), HTML and CSS across all platforms (e.g. mobile, PC, Mac, ...). While some inconsistencies remain, the majority of feature are widely available. With these design choices it is possible for any user to use our software without installing anything[2] while the backend may be deployed by organisations without any prior configuration. Per design the website is not tied to the backend with the intention of choosing a compatible backend. Any system that provides the needed endpoint is a valid option as well as any website, calling these endpoints may display and process given data their way, making not only OpenBIMRL an open standard but also not limiting the user to any of our design implementations. Also the actual rule checking engine has a separate code base to the RESTful API. This is by design to minimise the hurdle of integrating OpenBIMRL into an existing project as it may be included via a dependency management system.

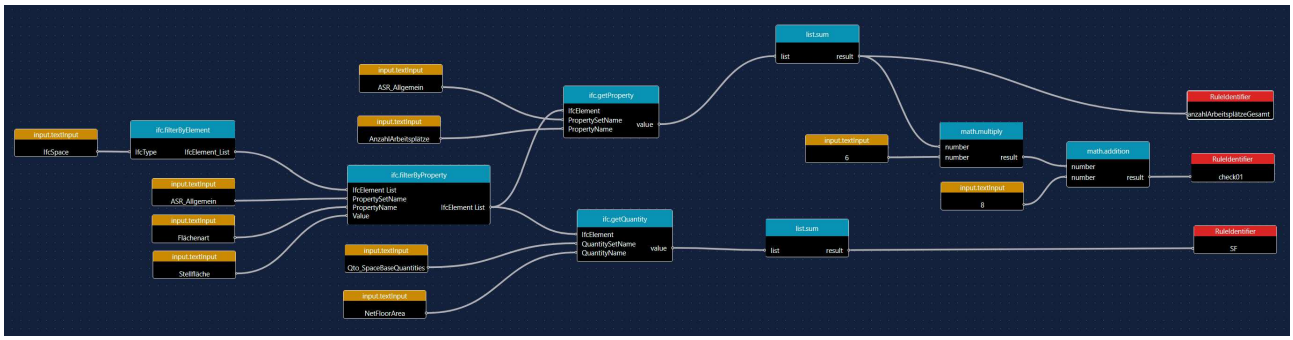

Figure 3: Implementation of a rule set from ASR A1.2 in OpenBIMRL.

When explained, rules can be considered logical but getting to a final result as demonstrated in Figure 3, proved to be a difficult task in terms of debugging. When writing code a so called debugger gives crucial insight into a programs inner workings by exposing changes in values after each operation. The absence of such features can be a significant disadvantage, requiring substantial guesswork, especially when dealing with an unfamiliar DSL. This is where the new implemted feedback loop becomes essential. With the new reference implementation the outputs are displayed ordered by node as a JSON e.g. Algorithm 1. Also the IFC elements can he highlighted in the result viewer.

[2]browser not included but is pre-installed on most devices

Algorithm 1: Output of a rule check.

```
{
  nodes:{
    ifc.getAllPropertySetInformation:{...},
    input.textInput:{...},
    ifc.getProperty:{...},
    ifc.filterByElement:{
      inputs:[...],
      outputs:[
        0:[
          0:{
            type:"IfcSite",
            guid:"20FpTZCqJy2vhVJYtjuIce",
            properties:{...},
            quantities:{...},
          },
          1:{...},
          ...,
        }
      },
    ...
```

4 Empirical Study

To see where OpenBIMRL is best used is to look at the application of Germanys' "Technische Regeln für Arbeitsstätten" (technical regulations for workplaces) or ASR for short. For example the rule for checking the ASR A1.2 which regulates the size of office spaces for a given amount of work spaces BAuA [11], may be automated with the graph visualized in Figure 3. While looking intimidating at first, this graph boils down to four steps:

1. Fetching all IfcSpace elements from the BIM model and filtering them for allocated office spaces.
2. At this point the graph splits into two paths. One retrieving the property $Anzahl Arbeitspl\ddot{u}tze$ (number of work spaces) in the property group $ASR\_Allgemein$ (ASR general) which is a IfcProperty specially set for ASR A1.2.
3. The second path retrieves the IfcQuantity $NetFloorArea$ in the quantity group $Qto\_SpaceBaseQuantities$ which is a quantity defined in the Ifc specification [12, ch. 5.4.5.6].
4. At the end there are some calculations done to match the space requirements of given work spaces.

The graph is able to perform these steps. Once the graph is done executing there are checks done to verify the results. In this case the check in Figure 4 compares each workplaces' space requirement against their actual floor area and produces a list of Boolean values (e.g.: [true, false, ...]) which are then again reduced to a final check result. This is done by defining the underlying group check

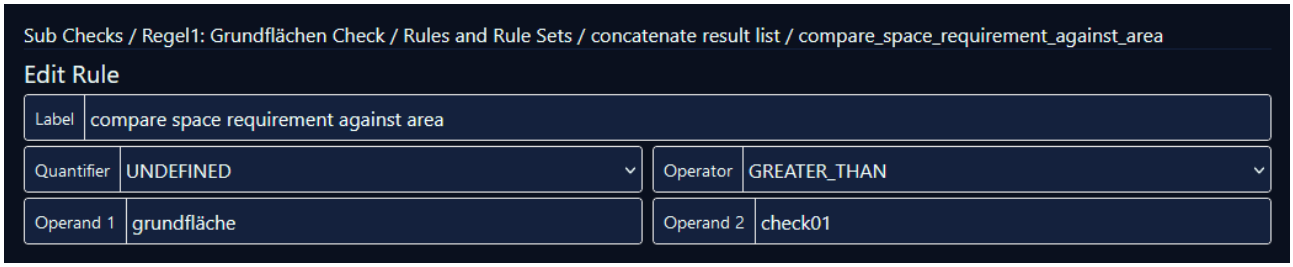

Figure 4: Checking the result of check01 against the floor area.

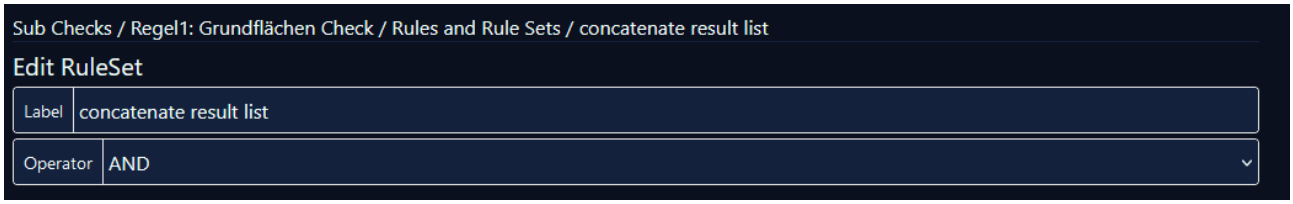

Figure 5: Defining logical operator for result sets.

and setting the concatenation operator to AND as seen in Figure 5 which logically applies the AND operation to all elements of the list.

5 Discussion

OpenBIMRL is not at all finished. Limitations include: Currently, OpenBIMRL is not dynamic in terms of graph nodes. All nodes currently available had to be implemented one by one and are static in their behavior.This lack of dynamism limits the flexibility and adaptability of the system, which can be a significant drawback in complex and evolving projects. Large graphs become quite difficult to navigate with no option to group related nodes together.This can lead to confusion and inefficiencies, as users struggle to maintain a clear overview of the project structure. Furthermore, the current system has a critical flaw in data persistence. When the creator page is reloaded, all the work is lost since it is not stored unless manually saved. To improve OpenBIMRL, it is essential to address these issues.

6 Conclusion and future work

In a nutshell the process of adapting new rules has been streamlined and the feedback now provided enables everyone using the new reference implementation of OpenBIMRL to better understand the inner workings of the system. Furthermore, the newly introduced modularity facilitates the seamless integration of OpenBIMRL into other software systems. Addressing the limitations, one thing currently on our bucket list are groups. Much like Unreal Engines Blueprint DSL Nodes may be visually clustered as distinct groups or act as one Node with their inputs and outputs inherited from the grouped Nodes they call Macros [13]. Another limitation - the lack of dynamic behaviour of nodes - can be elevated by allowing the creation of "code nodes". These nodes, ideally sand-boxed, capable of executing arbitrary code in a general-purpose programming language, would allow extendability without modifying the core system's source code.

References

[1] Jesco Denzel for the German federal government, *Planungsbeschleunigung, digitalisierung und wärmewende*, https://www.bmwsb.bund.de/SharedDocs/kurzmeldungen/Webs/BMWSB/DE/2023/03/meseberg.html, 2023.

[2] J. Dimyadi, W. Solihin, C. Eastman, and R. Amor, "Integrating the bim rule language into compliant design audit processes", Nov. 2016.

[3] M. Stepien, A. Vonthron, and M. König, "Openbimrl – an open format for code compliance checking of complex functional requirements from a regulation to the bim model", Jul. 2023.

[4] Chair of Computing in Engineering of the Ruhr-University Bochum, "Digitalisierung der Musterbauordnung (MBO2BIM)", Tech. Rep.

[5] Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA) [Federal Institute for Occupational Safety and Health], "Arbeitsschutzwissen für die Arbeitsstättenplanung - Neue Planungsinstrumente zur Unterstützung einer arbeitsschutzgerechten Gestaltung bei der Bau-Planung von Arbeitsstätten durch eine bauteilorientierte und maschinenausführbare Aufbereitung des Arbeitsstättenrechts", Tech. Rep.

[6] K. Kincelova, C. Boton, P. Blanchet, and C. Dagenais, "Fire safety in tall timber building: A bim-based automated code-checking approach", *Buildings*, vol. 10, no. 7, 2020. DOI: 10.3390/buildings10070121. [Online]. Available: https://www.mdpi.com/2075-5309/10/7/121.

[7] W. Solihin, J. Dimyadi, Y.-C. Lee, C. Eastman, and R. Amor, "Simplified schema queries for supporting bim-based rule-checking applications", *Automation in Construction*, vol. 117, 2020. DOI: 10.1016/j.autcon.2020.103248. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85084290727&doi=10.1016%2fj.autcon.2020.103248&partnerID=40&md5=6aa7679ba4b23d1fe8c745a2ecc99587.

[8] W. Solihin, J. Dimyadi, and Y.-C. Lee, "In search of open and practical language-driven bim-based automated rule checking systems", in *Advances in Informatics and Computing in Civil and Construction Engineering*, I. Mutis and T. Hartmann, editors, Cham: Springer International Publishing, 2019, pp. 577–584.

[9] *apstex IFC Framework*, https://www.apstex.com/.

[10] *IFC schema specifications*, https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/.

[11] Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA) [Federal Institute for Occupational Safety and Health], "Bek. v. 1.3.22, Bekanntmachung von Technischen Regeln; ASR A1.2 „Raumabmessungen und Bewegungsflächen"", *Gemeinsames Ministerialblatt*, p. 241, 2022.

[12] *IFC4x3ADD2 Schema Specifications*, buildingSMART International, Ltd., 2023, ch. 5.4.5.6,

[13] *Unreal engine 4.27 documentation*, Epic Games, Inc., Macro Library UI.