AUTOMATED MODEL CHECKING FOR TOPOLOGICALLY COMPLEX CODE REQUIREMENTS – SECURITY ROOM CASE STUDY

Tanya Bloch[1], Meir Katz[1], Raz Yosef[1] and Rafael Sacks[1]
[1]Virtual Construction Laboratory, Faculty of Civil and Environmental Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel

Abstract

A review of the state of the art in the field of automated code compliance checking revealed that existing applications are limited in the scope of clauses they are able to check. While some building code clauses are straightforward, requiring direct checking of parameter values, others depend on the topological relationships among objects, making automated checking of BIM models more challenging. Moreover, existing applications require the user to extensively preprocess the model in preparation for checking. We propose applying semantic enrichment for pre-processing the BIM models. The goal of a semantic enrichment process in support of automated code compliance checking is to derive the needed clause test values automatically and to represent them explicitly. A successful semantic enrichment process can therefore widen the scope of requirements that can be checked automatically. This work demonstrates such a process for checking code clauses involving topologically complex requirements. Although semantic enrichment proved to be successful for several purposes in previous research, dealing with complex topologies involves different types of semantic enrichment tasks. We explore the subject through a test case of requirements from the Israeli code for security rooms.

Introduction

In a visionary paper describing a computerized Building Design System (BDS) published in 1975, Eastman predicted that "*Designing would consist of interactively defining elements…Thus BDS will act as design coordinator and analyzer, providing a single integrated database for visual and quantitative analyses, for testing spatial conflicts and for drafting. … Later, one can conceive of a BDS supporting automated building code checking in city hall or the architect's office"* (Eastman, 1975). While all the other specific capabilities described in that paper have been realized with modern BIM systems (Sacks et al., 2018), automated building code checking remains limited to a narrow class of building code requirements, restricted almost entirely to those that impose numerical constraints on explicitly defined parameters of model objects. Furthermore, the discipline-specific nature of most commercial BIM systems means that their internal data schemas are specific to the discipline they serve. When models are exported to the ISO 16739:2013 Industry Foundation Class (IFC) open file format (ISO, 2013), much of the semantics remain implicit, and thus inaccessible to generic model review systems which require explicitly defined parameters, aggregations, connections and other topological structures. Semantic enrichment offers an automated interface to represent the implicit information in an explicit form, as well as supplement BIM models with missing information for a specific application or need (Belsky et al., 2016).

IFC for automated code checking and the need for Semantic Enrichment

In the 1980s and 1990s, researchers envisaged future automated building design review systems based on two developing technologies: building product modeling (BPM) and artificial intelligence (AI). Building Product Modeling is concerned with development of the object-oriented data schema considered essential for explicit digital representation of the form and function of buildings and their constituent components and systems (Eastman, 1999). The 'General AEC Reference Model' (Gielingh, 1988) was among the early building product models. Bjork laid out the guiding principles for modeling spaces, space boundaries and the building envelopes (Björk, 1992). The RATAS project model (Björk, 1994), a model developed in the EU COMBINE project (Augenbroe, 1994) and CIMsteel (Crowley & Watson, 1997) all followed. This development culminated in the Industry Foundation Classes (IFC), an open building model schema based on the ISO STEP standard (ISO 10303). The current version of IFC is IFC4 Add 2 and it has become an international standard (ISO, 2013).

Existing applications for automated code compliance checking rely mostly on information extracted from the model in conformance with the IFC schema. Dealing with complex code clauses in an automated code checking process requires higher levels of semantic information to be explicitly represented in the models (Solihin et al., 2004). Retrieval of required information in its correct representation is one of the challenges in every existing automated code checking application (Eastman et al., 2009; Preidel & Borrmann, 2015). To overcome this problem, a manual pre-processing stage, often called 'normalization', is required for most rule based code checking routines (Eastman et al., 2009). In addition

DOI:10.35490/EC3.2019.157

to being labour intensive and prone to errors, a manual pre-processing stage limits the degree of automation we can achieve in code checking. A semantic enrichment process is an alternative approach that offers to automate the interface. As the definition of semantic enrichment is the use of domain expert knowledge to infer the semantics of a given model, dealing with topologically complex code clauses requires an extensive semantic enrichment process to reach specific clause test values corresponding to the clause requirements.

Semantic Enrichment with rule inferencing and Machine Learning

Current automated code compliance checking applications such as Solibri and SMARTreview (SMARTreview APR, 2017; Solibri, 2017) use hardcoded rule sets that represent specific code clauses. Each rule set requires an explicit representation of the relevant information to support evaluation of the clause test value. Sacks et al. laid out a spectrum for code checking strategies and the required pre-processing for each strategy (Sacks et al., 2019). The strategies for checking range from traditional rule inferencing (Eastman et al., 2009), at one end of the spectrum, to the use of deep learning algorithms to classify building models, at the other end. Hypothetically, deep learning might eliminate the need for semantic enrichment to support automated compliance checking, because the patterns linking a model's topology and a classification of pass or fail for any given clause may be identified by the system. However, this is still largely unexplored territory in terms of research to date, and there are major unsolved challenges in its application to building models.

Research on semantic enrichment to date has focused on inferring meaningful information that was implied in a BIM model and explicitly adding its representation in the IFC file. The 'SEEBIM' system has been used successfully to demonstrate application of rule-inferencing to enrichment tasks such as classification of building objects, aggregation of building systems, unique identification and numbering, and reconstruction of occluded objects (in the case of models compiled from point cloud data) (Sacks et al., 2017). More recent work (Bloch & Sacks, 2018) has shown that some SE tasks can be performed more efficiently using supervised machine learning techniques. These experiments led to the conclusion that there are several types of semantic enrichment tasks with distinct characteristics and that the approach for SE should be chosen based on the nature of the problem.

To make progress in the field of automated code compliance checking, it is important to make progress in the field of semantic enrichment. While previous research focused on solving specific semantic enrichment tasks for different purposes, we strive to understand the structure and characteristics of semantic enrichment problems typically required to support an automated code checking process. The general idea is that a better understanding and characterization of the problem will lead to discovery of suitable solutions. The approach is to explore a range of test cases corresponding to a variety of regulatory codes and requirements. The expected result is a general framework for SE to support automated code checking. In this paper, we describe one such test case that involves complex topology.

The paper begins with an explanation of the code clauses and identification of specific semantic enrichment tasks to support an automated compliance check of models for conformance to them. We then describe each semantic enrichment task, detailing its implementation and results. The next section describes the compliance checking process using the results obtained in the semantic enrichment step. Finally, we report an experiment conducted using a model of an eight story residential building to validate the rule-sets compiled to implement semantic enrichment and code checking for the test case.

Security Room Code Requirements

A security room is a reinforced concrete space designed to protect its occupants from shrapnel and blast impacts from conventional weapons and from chemical weapons. Every residence built in Israel is required to have a security room, or alternatively, a larger security room must be provided to serve all apartments on a floor. The design requirements for security rooms are defined in a design code (Home Front Command, 2010). The code includes both simple geometric clauses, such as minimum wall thicknesses, and clauses that depend on topologically complex relationships between objects, such as requirements for intervening concrete walls to protect security room doors from line-of-sight exposure to windows.

Checking the supported area of walls

In this paper, we treat the requirement that security rooms must be stacked one on top of the other to form vertical shafts, and that at least 70% of any given security room's walls must be continuous and reach the structural foundations.

Figure 1 illustrates the general requirement and provides examples of pass and fail cases. The structural walls at the foundation level support a security shaft containing three security rooms, one on each floor. The rooms on the first and second floors pass, since more than 70% of the walls are continuously supported and reach the foundation. For the security room above the third floor, the north wall is missing and the east wall is discontinued on the first floor, meaning only 50% of the walls are continuously supported, and it is therefore non-compliant.

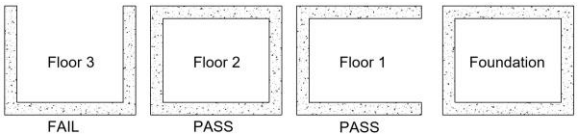


Figure 1 explanation of the requirement that 70% of the security room walls must continue to reach the structural foundation.

Semantic enrichment tasks

The first step to implementation of an automated code checking routine is to analyse the code at hand to recognize all objects, attributes and relationships that might be used during checking. The concepts are compiled and detailed in the format of a formal Model View Definition (MVD). Once we identify all needed information, we move on to compose specific semantic enrichment tasks. Since we aim to enrich a model to a point we can automatically check its compliance to the code, the exact requirements for enrichment have to be defined. Thus, we first compose a checking routine for a specific code clause and draw all required concepts for enrichment from that routine.

To explore the abilities of the semantic enrichment process in the case of topologically complex code requirements, we aim to fully enrich the model. Thus, the checking process will include a single logical rule applied on an explicit clause test value we provide during enrichment. In this case we aim to enrich the model so that each security room explicitly contains the value of total supported walls percentage. This essentially dictated all the requirements for semantic enrichment, which consists of the following steps:

a) identify all enclosed spaces (rooms) in a model, and where objects representing the spaces are missing, create them;
b) uniquely identify the security rooms and classify them;
c) explicitly associate all of the walls belonging to a security room with it;
d) identify any walls or foundations that support each wall and associate them with it;
e) for walls supported by foundations, identify the portion of their cross section that is supported, and store this supported area;
f) for walls supported by walls below them, identify the portion of their cross section area that is supported by the lower wall's supported area, and store it;
g) calculate the value of the total supported area of walls for each security room.

This final test value needs to be stored as an attribute of each security room. These SE tasks are explained in detail in the next section of the paper.

Semantic Enrichment Steps

Classification tasks

Bloch et al. (2018) demonstrated that classification of room types in residential apartments is a task better solved with a ML approach than with rule inferencing. From this work we can conclude that dealing with classification of physical elements with distinct geometry, like bridge elements (Ma et al., 2016) is different than classification of abstract elements with a similar box-like geometry. Since we are only interested in security rooms for the checking of the code at hand, we defined a less complex task of a two-class classification. In other words, we classify all spaces in the apartment as 'security room' or 'not security room'. Although this specific ask can be solved by both rule inferencing and machine learning, we continue to treat problems of room type classifications with a machine learning approach. Using a two class decision forest algorithm we reach 96.6% accuracy in classification on a cross validation set.

Association tasks

Once all spaces in the model are classified and the security rooms are tagged, we proceed to associate the walls bounding each security room to the security room. To make this association explicit, we do not create relationships between the walls and the spaces but tag the walls with the IDs of the spaces they bound. During the association, we also compute and store each wall's initial cross section area as an attribute of the security room.

IF *object 1 is an element of type IFCSPACE* AND
object 1 description is "Security Room" AND
object 2 is an element of type
    IFCWALLSTANDARDCASE AND
object 2 centroid elevation is between object 1
    minimum elevation and object 1 maximum
    elevation
THEN
object 2 description is "Security Room Wall" AND
object 2 tag := object 1 tag AND
object 1 user field "Area" := object 2 Area

To enable the next step, in which the supported area for each wall is computed, we need to compare every two walls that are in contact but above one another to find the overlap between them (the calculation itself is detailed in the next section). To form those pairs of walls we need to take into account the fact that every pair of walls is separated by a slab that can vary in thickness. This makes it difficult to use an operator which tests for "contact" between two walls. To overcome this problem, we explicitly associate the slabs to each security room and use them to find the correct pairs of objects for the calculation.

IF *object 1 is an element of type IFCFOOTING* AND
object 2 is an element of type IFCWALLSTANDARDCASE AND
object 2 is made of "concrete" AND
object 1 is in contact with object 2
THEN *create IFCRELAGGREGATES object 1 "supports" object 2*

IF *object 1 is an element of type IFCSLAB* AND
object 2 is an element of type IFCWALLSTANDARDCASE AND
object 2 is made of "concrete" AND
object 1 is in contact with object 2 AND
object 1 is above object 2
THEN *create IFCRELAGGREGATES object 2 "supports" object 1*

IF *object 1 is an element of type IFCSLAB* AND
object 2 is an element of type IFCWALLSTANDARDCASE AND
object 2 is made of "concrete" AND
object 1 is in contact with object 2 AND
object 2 is above object 1
THEN *create IFCRELAGGREGATES object 1 "supports" object 2*

Notice that if a non-structural wall bounds a security room, which clearly denotes a modelling or design mistake, the wall will be associated to the room but will not be used to calculate the walls' overlapping areas, which is the correct interpretation.

Computation tasks

The next step is the calculation of overlapping area between every pair of supported objects. Starting from the overlap between the structural foundation walls to the room bounding walls that rest on them, we first compute the bounding box representing the overlapping area between each pair of walls (or pairs of slabs and walls) where one is immediately above the other. The same computation is implemented until we reach the top of the building. The procedure is illustrated in Figure 2 below. This computation requires association of all pairs of walls (or pairs of slabs and walls) where one is immediately above the other.

Once the supported area of all security room walls is computed, we compute the overall percentage of support for each room. This is the sum of the supported areas divided by the sum of the gross areas for the set of walls belonging to any given room. This value is stored in a user-defined 'walls_to_foundation_ratio' field for each security room.

IF *object 1 Aggregates("supports") object 2* AND
object 1 longest dimension is parallel to object 2 longest dimension
THEN *object 2 user field "Support Area" := calculate_supported_area of object 1 and object 2.*

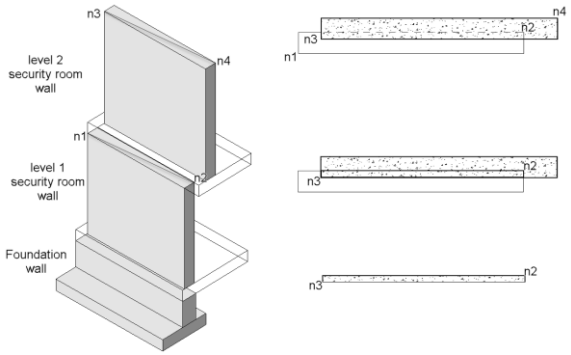


Figure 2 calculation of overlap between walls using their bounding boxes

Compliance Checking Step

Since we have enriched the model to contain an explicit clause test value, the checking itself consists of a single rule to check if the final calculated test value is greater than 70%.

IF *object 1 is an element of type IFCSPACE* AND
object 1 description is "Security room" AND
Object 1 walls_to_foundation_ratio >= 70%
THEN *object 1 user field support_clause_flag := TRUE*

Once the checking is complete, the results are stored in the IFC file. This enables reporting in various forms, including viewing of a colour-coded model.

Validation

To validate the results, the entire process of enrichment and checking was applied to a construction project of a residential building in Tel Aviv. The building contains eight floors with six apartments on each floor, and three underground floors for parking. Every floor is designed to contain a public security room that serves all six apartments on that floor. Figure 3 shows a typical floor plan.

Although the security rooms are similar on every typical floor, there are some significant differences on the ground floor and the underground levels, where the walls have openings to facilitate parking spaces. These openings are most significant for the code check at hand, adding geometric complexity to the topological complexity of the minimum foundation support clause case. Figure 4 (a) illustrates the configuration of a typical security room, and (b) the configuration of the security room's walls with the openings on level -3. Notice that there is an opening in the north wall and in the east wall, meaning the walls are not entirely continuous to the structural foundations.

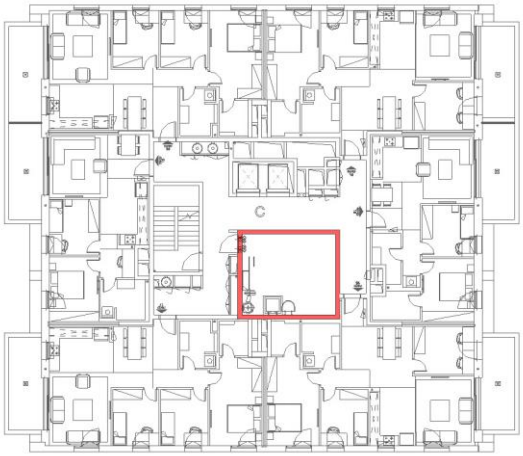
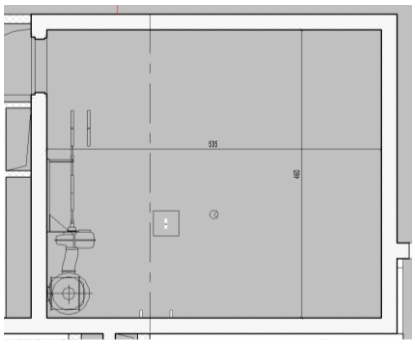
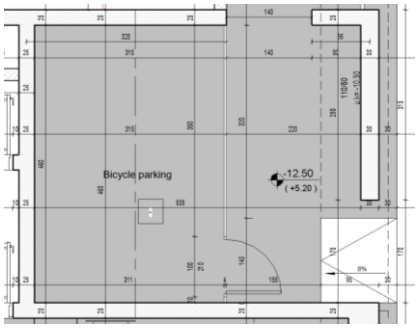


Figure 3 A typical floor plan



(a)



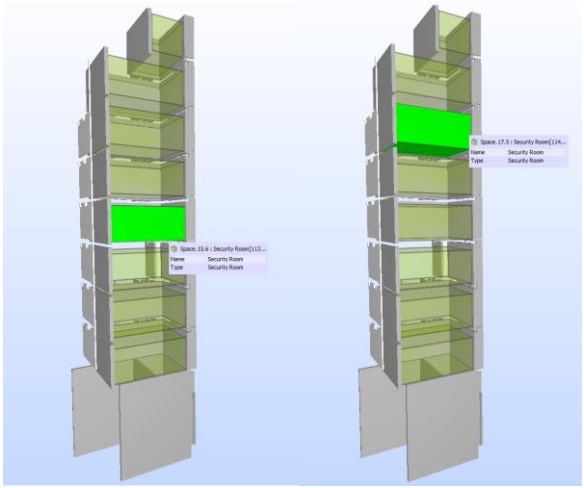
(b)

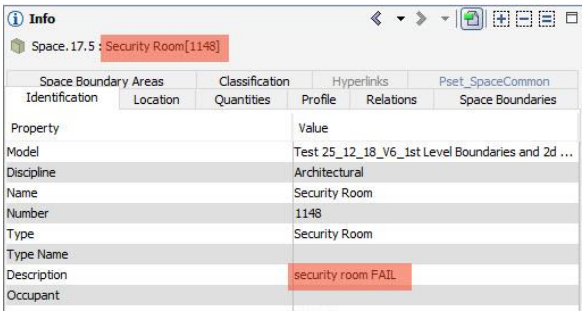
Figure 4. (a) A typical security room. (b) The security room's suporting walls on level -3.

As the building had already been granted a construction permit, we expected the model to be compliant to the code clause, which was indeed the result of the enrichment and checking process in SEEBIM. To further explore and validate the proposed procedure, we introduced several changes to the model's geometry to create cases that were not compliant to the described code clause.

To create and test a case similar to the one illustrated in Figure 1, we introduced two changes to the original model. First, we changed the material of a single wall of the security room on floor four from cast in place concrete to masonry concrete blocks. This should have the same effect on the particular code clause as removing the wall entirely. As can be seen in Table 1, in the row for floor 4, according to manual calculations, this will lead to 70.2% of the security room's walls on floor four being continuous to the structural foundation, which is still compliant.

The second change introduced to the model was to create a 123cm wide opening in the north wall of the security room on flor six. Again, based on the manual calculations shown in Table 1, we expect the result of checking the supported wall area to be only 64.3% for the security room on floor six, making it not compliant to the code requirements. Subsequently, the supported wall areas of security rooms on floors seven and eight will also be 64.3%, meaning those security rooms are not code compliant as well.





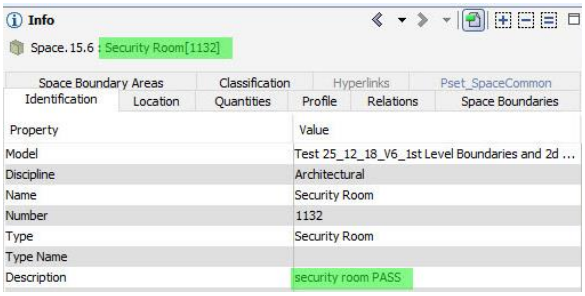


Figure 5. A view of the building model with tags for security rooms that pass and that fail the foundation support clause.

The model with modified geometry was exported as an IFC 4 design transfer view and used as input for the SEEBIM engine. The file size of the original REVIT file was 122 MB; the corresponding IFC 4 reference view file was 45 MB; the run time in SEEBIM was ~3 minutes (although the time to write the export file using the IFC Engine is some 60 minutes). The rule sets detailed above were used in SEEBIM to complete all of the described semantic enrichment tasks and to perform the compliance check. Results received from SEEBIM match the expected results as described in Table 1.

Figure 5 illustrates a view of the model using an IFC file viewer software, in which users can visualize the results of the code compliance check for minimum continuous wall foundation support for security rooms. The illustration shows results for the security room on floor four, which is compliant to the clause and on floor six, which is not compliant. All of the security rooms were evaluated correctly, both in the original building model obtained from the developer, and in the model in which the researchers intervened to create cases in which the security rooms on floors 6-8 failed the code check.

Technical Implementation

As described above, the rules sets were implemented in the SEEBIM 2.0 inferencing engine. The interface is illustrated in Figure 6.

The rules were compiled directly in the user interface, without any need for programming. Rules can be compiled directly in the user interface, by selecting from a set of pre-coded operators to apply to every pair of IFC objects in the model (see Figure 7).

Users can upload any IFC file and apply any of the existing rules to that model. Rules can also be aggregated into rule sets, in this case including all the enrichment and checking rules needed for checking the compliance of the security rooms wall supported area. Once the user runs a set of rules on a specific file, a new, enriched IFC file is created, which can be downloaded and opened in any compatible platform.

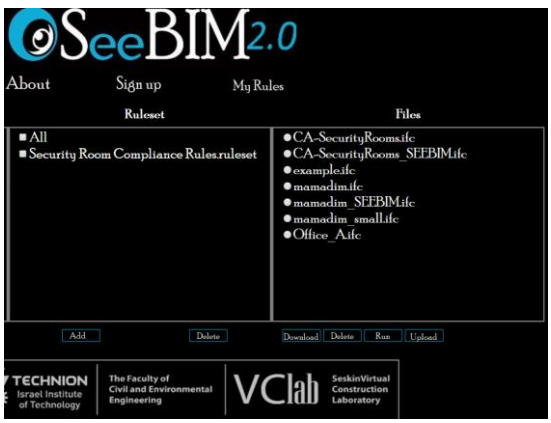


Figure 6 The SEEBIM 2.0 user interface

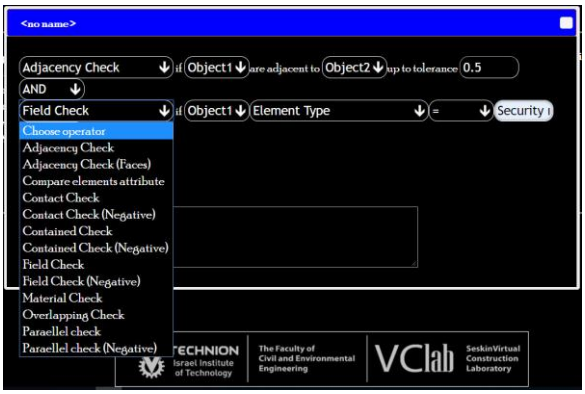


Figure 7 Compiling new rules directly in the user interface

During the validation experiments, a number of challenges arose which shed light on the process as a whole. These included:

a) Unexpected and non-uniform modelling practices on the part of the design team. For example, some walls were left 'unconnected' to the slabs above them, so that they protruded through and above the slabs (apparently a result of changes to the levels of slabs during design); structural foundations were modelled as floors.
b) Omission of objects and technical errors in the IFC files exported by Revit 2018: walls with openings were absent from IFC4 reference view export.
c) Unforeseen design situations. For example, in some places in the parking levels, designers modelled floor to ceiling openings in concrete walls by placing two wall sections rather than one section with an opening. This meant that the cardinality of the 'structural support' relationship between walls was m:n rather than 1:1, as had been implicitly assumed (such situations were not contemplated in the simple lab models used when compiling the rules).

Discussion and Conclusion

A set of rules was compiled and implemented for checking compliance of a building model to a building code clause that requires evaluation of complex geometry and topology. The rules were evaluated using the SEEBIM forward chaining inference engine. The successful application of the rule sets demonstrates the feasibility of replacing normalization - the manual process for pre-processing a model to populate it with the specific information concepts needed for code compliance checking - with an automated procedure. By defining a specific clause test value and the corresponding concepts in an MVD, we can define semantic enrichment tasks that will lead to explicit representation of that test value in the BIM model. Once all semantic enrichment tasks are complete, simple logical rules can perform the

compliance check itself. The overall development procedure can be broken down into three stages: a) establish the interim concepts and define them as targets in an MVD; b) compose the SE rule sets and implement them in the inference engine or in ML processes, c) test and validate the resulting system.

Through this experiment, we demonstrate that it is feasible to compose a set of rules that chain together to fulfil the semantic enrichment that is needed. The rules progressively add the concepts that are needed, such as 'supported area' for each wall, as the rule inferencing engine is cyclical, repeating all of the rules on each and every pair of objects on the model, until such time as no new information is added in a full cycle. This is an effective method for enrichment of concepts that involve complex geometry as they require a progressive chain of calculations based on the relevant geometric features.

Once the SE process is complete, the rule checking reduces to a simple IF THEN rule evaluation for each space. We assume that this is the case for all code clauses with a single test case value that can be represented explicitly in the BIM model either directly in the design phase or through semantic enrichment. The result emphasizes the importance of progress in the field of semantic enrichment.

In general, the selection of a rule-based or a machine learning approach will depend on the nature of the problem and the amount of effort that is needed for compiling the code checking solution. In ongoing research, we are testing the machine learning approach to the same clause, with a view to learning more about alternative approaches.

Table 1. Validation test case values.

Floor	Supported Wall Areas [m^2]					Ratio of supported walls
	North	East	South	West	Total	
Whole room	1.41	1.21	1.41	1.21	5.24	100.0%
-4	1.41	0.41	1.41	1.21	4.43	84.6%
-3	1.06	0.41	1.41	1.21	4.08	78.0%
-2	1.06	0.41	1.41	1.21	4.08	78.0%
-1	1.06	0.41	1.41	1.21	4.08	78.0%
G	1.06	0.41	1.41	1.21	4.08	78.0%
1	1.06	0.41	1.41	1.21	4.08	78.0%
2	1.06	0.41	1.41	1.21	4.08	78.0%
3	1.06	0.41	1.41	1.21	4.08	78.0%
4	1.06	0.00	1.41	1.21	3.68	70.2%
5	1.06	0.00	1.41	1.21	3.68	70.2%
6	0.75	0.00	1.41	1.21	3.37	64.3%
7	0.75	0.00	1.41	1.21	3.37	64.3%
8	0.75	0.00	1.41	1.21	3.37	64.3%

The unexpected modelling conditions encountered in the validation underline the need for effective semantic enrichment. Despite being modelled by a highly skilled and professional design team, the model still contained object compositions that were not foreseen during SE rule compilation. The essentially unlimited diversity of possible modelling practice raises a question concerning the feasibility of a purely rule-based approach to SE. Machine-learning may be superior in dealing with diverse object compositions. The same can be said for unforeseen design situations.

References

Augenbroe, G. (1994) Integrated use of building design tools: results from the COMBINE project. In: *Automation Based Creative Design–Research and Perspectives*. Elsevier, pp.205–218.

Belsky, M., Sacks, R. & Brilakis, I. (2016) Semantic enrichment for building information modeling. *Computer-Aided Civil and Infrastructure Engineering*, 31 (4), pp.261–274.

Björk, B. (1994) RATAS Project—Developing an Infrastructure for Computer-Integrated Construction. *Journal of Computing in Civil Engineering*, 8 (4), pp.401–419.

Björk, B.-C. (1992) A conceptual model of spaces, space boundaries and enclosing structures. *Automation in Construction*, 1 (3), pp.193–214.

Bloch, T. & Sacks, R. (2018) Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models. *Automation in Construction*, 91, pp.256–272.

Crowley, A.J. & Watson, A.S. (1997) Representing engineering information for constructional steelwork. *Computer-Aided Civil and Infrastructure Engineering*, 12 (1), pp.69–81.

Eastman, C. (1975) The use of computers instead of drawings in building design. *AIA Journal*, 63 (3), pp.46–50.

Eastman, C., Lee, Jae-min, Jeong, Y. & Lee, Jin-kook (2009) Automatic rule-based checking of building designs. *Automation in construction*, 18 (8), pp.1011–1033.

Eastman, C.M. (1999) *Building product models: computer environments supporting design and construction*. Boca Raton, FL, USA, CRC press.

Gielingh, W. (1988) General AEC reference model (GARM). In: *ISO TC184/SC4*. CIB Publication, pp.165–178.

Home Front Command (2010) *Specifications for Building Shelters*. Ramle, Israel, Protective structures department, Home Front Command. Available from: <http://www.oref.org.il/SIP_STORAGE/files/6/3196.pdf>.

ISO (2013) ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.

Ma, L., Sacks, R., Kattel, U. & Bloch, T. (2016) 3D Object Classification Using Geometric Features and Pairwise Relationships. *Computer-Aided Civil and Infrastructure Engineering*, 33 (2), pp.152–164.

Preidel, C. & Borrmann, A. (2015) Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vilnius Gediminas Technical University, Department of Construction Economics & Property, p.1.

Sacks, R., Bloch, T., Katz, M. & Yosef, R. (2019) Automating Design Review with Atrificial Intelligence and BIM: State of the Art and Research Framework. In: *Future Cities and Resilient Infrastructures*. Atlanta, GA, USA.

Sacks, R., Eastman, C., Lee, G. & Teicholz, P. (2018) *BIM handbook: A guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers*. Third Edition. Hoboken, NJ, John Wiley & Sons.

Sacks, R., Ma, L., Yosef, R., Borrmann, A., Daum, S. & Kattel, U. (2017) Semantic Enrichment for Building Information Modeling: Procedure for Compiling Inference Rules and Operators for Complex Geometry. *Journal of Computing in Civil Engineering*, 31 (6), p.04017062.

SMARTreview APR (2017) SMARTreview APR [Internet]. Available from: <https://www.smartreview.biz/home> [Accessed 6 July 2017].

Solibri (2017) Solibri Model Checker (SMC) [Internet]. Available from: <https://www.solibri.com/> [Accessed 13 March 2017].

Solihin, W., Shaikh, N., Rong, X. & Poh, K.L. (2004) Beyond interoperatibility of building model: A case for code compliance checking. In: *BP-CAD Workshop*. Carnegie Mellon University.