Automation in Construction 53 (2015) 69–82

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Classification of rules for automated BIM rule checking development

W. Solihin *, C. Eastman

Georgia Institute of Technology, United States

ARTICLE INFO

Article history:
Received 21 October 2013
Received in revised form 19 December 2014
Accepted 3 March 2015
Available online 20 March 2015

Keywords:
BIM
Building codes
Code checking
Rule checking

ABSTRACT

Automated rule checking has been identified as potentially providing significant value to the AEC industry from both regulatory and industry perspectives. Key challenges to a successful rule checking implementation are the complexities inherent in the rules themselves and the breadth of conditions to which they need to apply. Due to the large number of building codes and theoretically infinite number of rules that can be defined, it is critical to systematize the rules to make the task of rule checking tractable. This paper is a first step towards classifying automated rule checking. In this paper, the authors draw upon their extensive international exposure with various building codes and rule checking areas in the AEC domain to introduce a general classification of rules across application domains using criteria that apply to all known aspects of automated rule checking. The exposure covers both academic research as well as actual production implementation done with CORENET ePlanCheck project in Singapore, and also various pilot implementations in the US and other countries. The authors offer a survey of representative examples of different classes of rules, their key concepts, potential tools required to implement them, and at least one possible solution of each example rule in each category. The examples are drawn from both research domains and existing commercial applications. The aim of this paper is to provide an initial guide for creating a framework for rule classification that articulates both the process challenges and the technology needs to address rule automation.

1. Introduction

BIM is increasingly used in building projects in many parts of the world. Users who have overcome their initial learning curve are beginning to realize the power of modelling the building using one or more BIM platforms, supporting design, fabrication, contracting, and other processes. As people have become more familiar with the BIM tools and the opportunities they offer, we have witnessed the production of ever more complex and detailed building models. It is no longer practical for users to rely on visual inspection to ensure BIM models are of good quality and adhere to requirements. The requirements do not only deal with properties, but also with more abstract and semantically rich information that are not always present in drawings. Automation of checking, where well-defined rules can be applied automatically with minimum user intervention, are increasingly needed [40]. As easier rules are automated, the role of user experts will move towards the higher end of the value chain with rules or issues that are difficult to define and automate.

The role of automated rule checking has been recognized well before the advent of BIM. The topic has been researched as early as the 1960s. Fenves [20] introduced a decision table method for rule checking. Eastman [19] and Nawari [40] covered the brief history of efforts in automating building code-checking. Some progress has been made in the past few years since Eastman surveyed rule-based checking systems in 2009. A survey of research papers has shown a few tracks of research activity in this area. Some have focused on the interpretation of rules into computable forms. Examples of work in this direction are efforts using a RASE (Requirement, Applicability, Selection, Exception) tagging mechanism similar to the technique applied in the SmartCodes project and frame representation like format [29,30], using the Natural Language processing technique [48,59], and using semantic web methods [47]. Others have put more focus into the implementation of computable rules using certain standard techniques or rule engines. Beach et al. extended the RASE mechanism and used the DROOLS open source rule engine to implement the rules [7]. Zong and Ding used the semantic web OWL and SWRL technique and applied the rules into the JESS rule engine [61]. Nawari proposed the use of the SmartCode tagging mechanism and LINQ programmatic query language as the implementation vehicle [41]. Choi took a similar approach using SmartCode tagging but used XML derived from STEP P21 as the means to compare between the rules and the model [15], and Park used a rule-based system called CODE-MAVEN [46]. Work was also done in rule-checking for very specific problem domains. Some examples include: Zhang et al. applied OSHA rules into safety checking in building during construction [60], Martins developed an automated code-checking application for water distribution systems [39], while Liu used it for HVAC system [38]. Tan focused on

* Corresponding author. Tel.: +1 404 448 1165, +65 96738091.
E-mail address: wsolihin3@gatech.edu (W. Solihin).

http://dx.doi.org/10.1016/j.autcon.2015.03.003
0926-5805/

a code-checking application for building envelop designs [56], Lee implemented checking rules over a range of design phases in relation to the US courthouse design [37], and the BERA language was developed for spatial programme and circulation requirements related to the US courthouse design [36]. On the commercial application front, Solibri Model Checker continues to add more capabilities [52]. FORNAX™ on the other hand has focused more on regulatory authority for large scale turnkey implementation. It has recently extended its application in Saudi Arabia [44]. These are practically the only two commercial applications that are available in the market today that cover a reasonable set of automatic checking capabilities.

The progress towards realization of these efforts in real-world use has been accelerated with the wide acceptance of a standard BIM format developed by BuildingSMART, Industry Foundation Classes (IFC). Practically all recent efforts in building automated rule checking systems are based on the IFC format at the BIM data level. The most notable effort is Singapore's CORENET ePlanCheck project that was and is still the most serious and comprehensive attempt to date to automate code checking at the national level [34,43,54]. Based on these experiences, it is envisaged that at present the role of automated rule checking will serve as a decision support system, where some user involvement may be necessary. However, the final aim of the automated rule checking should be a fully automated system that will free experts to focus on what really matters for buildings, such as safety, sustainability and high environmental performance. The existence of fully automated rule checking will increase the quality of these aspects and yet allow designers to innovate without sacrificing these qualities.

Because the role of BIM itself changes throughout the building lifecycle, rule checking covers a broad scope within the AEC Industry and its practices. Eastman [19] suggested that besides building regulatory code-checking, more specialized types of rule checking such as client's requirements and requirements for specific building types are also emerging. In general, we see that the scope of the rules fall into the following categories:

1. Checks for well-formedness of a building model. This group of rules concerns primarily syntactic aspects according to the set of standards or prior agreed set of required conditions for the IFC or for other model views.
2. Building regulatory code checking. This focuses on compliance to well-defined or usually prescriptive building codes or regulations.
3. Specific client requirements. Examples include requirements for hospital design or GSA courthouse design.
4. Constructability and other contractor requirements. These often involve temporary objects or conditions present only during pre-construction process and during construction, such as formwork and shoring.
5. Safety and other rules with possible programmed corrective actions. These support decisions and help automate the search for potential dangers to workers during construction and also maintenance staff during operation.
6. Warrantee approvals. The post-construction model is checked for issues that may affect the warrantee or cost to maintain. These requirements may need to be combined with actual site inspection. Roofing systems are an example. The tool however will provide assistance for the inspector to focus on potential issues of the design and construction.
7. BIM data completeness for handover to the facilities management (FM). Since the role of BIM changes during the lifecycle of the construction process, downstream requirements such as FM are often not considered earlier in the process to the detriment of the facilities manager. Rules to check such completeness at the end of different contract phases will be increasingly required such as requirements defined by COBie and the other families of Information Exchange (IE).

2. Re-use of BIM rule checks

Looking at the above range of potential uses, it is apparent that rule checking has an extensive range of applicability. The methods used for specific building code checking may also be applicable to safety or constructability checking. Significant benefits accrue if rule checking can be organized generically according to the type of checking software infrastructure required, rather than by application area. This more abstract approach allows researchers to target critical needs across application areas and to identify issues benefitting the whole rule-checking area. The key principles that guide us in organizing rule-checking scope are motivated by the following ideas:

1. Potential re-use of rule structures and clauses. Patterns exist in many rule structures and clauses even though they entail detail variations. The extraction of such patterns is challenging because it requires experience with different rule applications and inductive reasoning. This usually happens at the rule interpretation step [19], where rule experts analyse the rule and discover hidden assumptions, dependencies, ambiguities and exceptions to define the exact rule requirements suitable for software development. Re-use of such patterns significantly improves coverage of the range of rules that can be implemented. This will in turn reduce cost of such implementation efforts and add to the logical structure that the rules can be based on.
2. Existence of best practices in specific areas especially in the areas where rules are not well-defined, but commonly accepted best practices exist in the industry. Many examples exist in geometric modelling, for example walls must touch slab below, externally exposed slab and internal slab should be modelled as separate slabs, all parts of the building should be "filled" with space objects for programmatic and floor area assessment, all spaces should be connected except for small spaces used for service ducts [6,37,52].
3. Strategies for closing the gap between the terms used in rules to be checked and the information explicitly represented in the target building model. The gap involves deriving new data that is represented implicitly in the provided model. It also involves the development of terminology for types of rules and their classification using a publicly available open standard such as IFC. IFC has been steadily accepted as a standard in the industry and is the only open and relatively mature standard supported today by major BIM applications. A standard way for representing building model data is crucial in developing any stable rule checking application.

3. Level of development (LOD) and model views (MVD)

There are two major parts that a rule checking system must deal with. The first is the building model, and the second is the rule definitions. Building models are large datasets, even for medium-scale buildings. There is no rule or class of rules that applies to the entire set of building model data. Each rule or class of rules applies to a subset of the data. Development of model views that represent the appropriate subset of data required for an exchange is critical in defining the exact rule checking requirements that the populated models satisfy. Model views are a standard methodology proposed by Hitanen, and BuildingSMART has adopted it as a standard methodology in the implementation of IFC based projects [13,26,27]. Model views serve as "contract" documentation for both the modeller and the implementer. It is a kind of handshake protocol that allows expected meaningful requirements, implementation and results. Explicit definition of model views relies on the inferred rules that are associated with the model view. It typically includes the types of geometry desired by the receivers, the critical variables for the use case, and the restrictions of entity subtypes of relevance in the use case. Such rules are sometimes referred as implementer's agreements.

Another important factor affecting building models is the level of development of the building model, or its LOD. The LOD was developed by BIMForum as a specification to articulate with clarity the content and reliability of BIM at different levels of development [9]. Different objects and features have different LODs during the phases of a project. Each object's LOD monotonically progresses, but at varied rates. For example, the rebar detailing and layout progresses at a slower rate than the formwork because the formwork defines the constraints that determine the reinforcing layout. Conversely, the phases of the project lifecycle have different LODs, which identify the structure and detail that rules can apply meaningfully. The higher the LOD is, the more detailed information within the model is expected. There are currently six levels of LOD: LOD 100 mainly requires objects in graphical representation, LOD 200 adds approximate quantities, shape, location and orientation with possibly non-graphic information attached, LOD 300 requires more specific systems, objects or assembly in term of quantity, size, shape, location and orientation with possibly non-graphic information attached, LOD 350 adds requirements on interfaces with other building systems, LOD 400 contains more detailed information required for fabrication, assembly and installation, and LOD 500 is a field verified representation. For building models at the design development phase, which is the most typical stage where a building model is complete enough for code compliance submission, LOD 300 or LOD 350 is generally sufficient. For this reason, we expect that initially LOD 300 will be assumed, but over time rules will evolve to be able to work at least partially with a lower level of LOD. Fabrication issues will apply to higher level LODs. Requirements for rule checking may work harmoniously with systems at varied LOD. For our uses, LOD should be at the lowest level adequate for rule checking.

Table 1
Example of ambiguity and implicit knowledge or requirements in building codes.

Clause ref.	Description	Questions during interpretation process
Reg. 44(1) BCA	Protection of staircase and staircase landing Every staircase or staircase landing shall be protected on any side overlooking an air-well, courtyard, void or external open space by either a railing, parapet or balustrade capable of resisting the lateral loading as specified in the Table 4 of the fourth Schedule.	1. What are the criteria when the protection starts to be required? What is the height of the protection? How about the shape of the protection (especially when there are gaps)? 2. What exactly defined as "overlooking"? Is a full glass wall considered "overlooking" and therefore requires additional protection, or the glass wall itself can be considered a protection? 3. How far a protection, e.g. railing, can be from the edge before it is no longer considered a protection to that edge? 4. If the edge has a gap to the adjacent edge, how large a gap is allowed before protection is needed?
Code of practice on sewerage and sanitary works; Part 3.1 sanitary drainage systems	3.1.3.6 Inspection chamber and ventilation 3.1.3.6 (b) i) The first inspection chamber shall be ventilated except when there is a discharge/ventilating stack of not less than 100 mm diameter or where provision of the ventilation stack would cause odour nuisance to the surroundings.	1. How do we determine the first inspection chamber? 2. What qualifies as "ventilated"? 3. In the case of a stack, where can it be located relative to the chamber? 4. What defines when a ventilation stack would cause odour nuisance?
BCA lighting requirements (F)	The aggregate light transmitting area for each room must not be less than 10% of the floor area of the room to be lighted.	1. Unstated requirement: There is a need to consider the indirect light coming through window from adjacent "buffer" space, such as veranda, balcony, wash areas, terrace, or corridor.

The second major part of rule checking is the rule definitions. Today, the rules are typically written in human-oriented languages that require significant domain knowledge in order to "interpret" them into a machine interpretable manner. There are many ways of approaching the interpretation, as mentioned earlier, but most rule checking studies focus merely on the language representation of syntax and grammar of the rules. In practice, expert knowledge is often required to interpret the meaning or semantics of the rules: the intent, base and hidden assumptions, assumed general knowledge of the subjects, and dependencies with other rules. Experience in CORENET ePlanCheck, which used a logic-based interpretation method [19], shows that the interpretation is crucial in transforming the rules that are often ambiguous into more precise definitions, thereby removing the ambiguity and clarifying checking concepts or principles. This is usually achieved by asking a series of questions. During the process of interpretation, implicit assumptions or expectations are discovered that help to complete the understanding of what needs to be checked. Too often this fact is not rigorously considered due to the practical challenges in looking through all possible rules and in anticipating the complexity in terms of their implementability. This usually leads to a sampling of relatively low complexity rules and using them to prove the approach would work before extrapolating it to the rest. This approach may lead to gross underestimations of the complexity involved in the implementation of more complex rules, as suffered by the CORENET implementation team [53]. To illustrate the issue, three examples in Table 1 show the existence of implicit assumptions that will need expert knowledge in order to interpret them:

The problem is how to address the domain expertise assumed in the interpretation of the rules — this can only be done manually today. Rule interpretation is a significant step in the process of rule checking. In CORENET, it took approximately 20–30% of the overall effort. This was not a small effort but this process allowed a one-time investment for good rule checking that led to automation of the manual effort. The same conclusion was observed in phase 1 of the AutoCode project that Fiatech undertook: manual rule checking is largely inconsistent because of human judgement that fills in the ambiguities, incorporating experience and unwritten local adaptation of the rules [22]. The phase II of rule checking is to generate consistent, precise and quantifiable conditions and constraints for each rule [21]. A step in the right direction are the recent trends using Ontology and Semantic Web approaches to systematically capture domain knowledge embedded into rule definitions that transform the rules into a consistent representation suitable for the software engineers [28,42,47,48,58,59,61]. However, both traditional logic based interpretation and Ontology based interpretation do not remove the complexity in terms of actual checking requirements represented as predicates or functions that need to be codified.

In the actual codification of rules, one has to consider the trade-off between requiring most of the required data to be carried inside the model and entered by the user and using the computer programme and logic to derive the new information. On one side, requiring the data to be in the model often dictates significant user input effort. This process will overburden the designer and it will be prone to errors and inconsistencies. On the other end deriving the information from

basic BIM data inside the rule implementation will significantly increase the complexity in terms of the implementation [53]. For example, calculating the area and volume of a Residential Unit that includes a group of spaces and everything else in between. It involves many tasks that start with the spaces, finding all objects (mainly walls) bordering the spaces through the space boundary relationship, clipping the walls if they extend beyond the unit boundaries, and including the area of the holes in the spaces (for example, columns). Even with derived information, the responsibility of the designer to input minimum information may still be required, which should be consistent as the derivation rules in the model views.

4. Dealing with arity issues

Rule checking systems are hardly a simple one question one answer condition, even in its simplest form. Rule conditions can be satisfied with exactly one criterion, at least one of many possible criteria, or multiple/all criteria at once. Without consideration of the arity issue, a rule checking system will suffer from a large number of false negatives, which will become noise in reporting the exact checking results. This needs to be carefully avoided because even a single false negative for one rule usually will be multiplied to many; the object being checked inside the building model typically numbers in the hundreds if not thousands for reasonably sized buildings. The rule checking system must be smart enough to report only the real negatives according to the appropriate criteria defined for the rule.

5. Dealing with combinatorial issues

As rule checking progressively moves towards more complex requirements, it no longer simply deals with cases of returning true/false conditions on a straightforward parameter vs. value check, but rather evaluating combinatorial issues. It needs to deal with multiple possibilities where each possibility may lead to a different solution path. In a large building model, the combinatorial problems may easily grow into a large and hard-to-manage scope, often based, for example on space occupancy and type of structure. Clearly defining boundaries that make sense in narrowing down potentially large combinatorial conditions becomes critical, especially in the more advanced classes of rules that will be described later. Domain expertise and experience in dealing with rule checking will be extremely important in such cases. A simple example of combinatorial issues that rule checking needs to deal with is given in the appropriate example under Class-2 rule classification for building code from IBC 1008.1.8.

6. Rule classification

In the end result, all the data in IFC reduces down to attribute values. These may be material type, reference to a geometric placement, or x,y,z point locations. Considering this point, the fundamental operation is the reading of values in the model structure. The model structure can be simple or complex, such as when geometry is concerned. Based on the above discussion, this paper classifies various rules according to the complexity of the rules processing. It also assumes in the examples the processing of an IFC model with the LOD 300, and using the "standard case" property sets and structure according to the basic MVD appropriate to the examples being cited. In many of the cases, they follow the BuildingSMART's Coordination View 2.0 [12] and its extension, code-checking MVD that is used for CORENET ePlanCheck [51].

We find it useful to distinguish four general classes of rules, described in more details below. The description is organized by first explaining the definition, typical uses of the rules, followed by an explanation of the complexity and suitability of the tools or techniques required. When appropriate, we present one or two examples of rules to illustrate how the rule checking is typically processed. The examples given in this paper represent one possible solution to address the specific rule discussed. It is not the scope of this paper to go into the algorithmic structure of the rules. The examples rather are presented to highlight the complexity involved and to clarify the basis of classification of the rules.

At the end of this section we present a listing of several other representative examples in Table 3, drawn from various sources commonly applied in the AEC industry. In Table 4 we summarize the list of applications both from research domains and commercial products that fit into solutions that match the classes.

6.1. Class 1 — Rules that require a single or small number of explicit data

This class of rules checks explicit attributes and entity references that exist inside the BIM dataset (Fig. 1). Typical uses of this class of rules are:

a. The required attribute settings of entities for correctness checking.
b. Some simple building code checking, for example "Egress doors shall be of the pivoted or side-hinged swinging type" (IBC 1008.1.2) [33], and "every building comprising 5 or more storeys above the ground level shall be provided with one or more passenger lifts" (BCA Reg 53(1)) [49].
c. Some model view business rules in MVDs restricting options allowed, for example Name and LongName attributes, which are optional in the IFC schema for IfcBuilding, are mandatory in the Basic FM Handover MVD [14].
d. Attributes and attribute values that are essential for consistent derived values required in other classes of rules. For example, to be able to derive a concept of an apartment unit, all spaces within the unit must be members of an IfcZone and the IfcZone must be named and classified according to the standardized naming convention or classification.

6.1.1. Degree of complexity/required tool

At this level, information is explicitly available from the model either directly from the entities or its associated properties with other entities using the explicit relationship entities. Frequent tools used in supporting this level of rule checking are IFC toolkits that are available both commercially (for example EDM IFC toolkit from EPM Technology http://www.epmtech.jotne.com/, EuroSTEP toolkit, etc.) and various open source alternatives (a list is available from http://www.ifcwiki.org/index.php/Open_Source).

One example on how this type of rule can be processed:

"Fire Walls Must Have Correct Wall, Door, and Window Types" [SMC — Building Codes rules] [52].

To process this rule, the model is queried for well-defined entities (wall and its components, door, window) and their relationship. The fire rating property to be checked is obtained using explicit relationships from each of the entities being processed. All entities and relationships are usually explicit in the model and the rule is able to follow the links in a straightforward manner.

6.2. Class 2 — Rules that require simple derived attribute values

Checks are based on a single value or a small set of derived values. This class of rule checks derive attributes or values but does not generate new data structures (Fig. 2). This class of rule involves a trade-off between requiring the user to derive the data vs. rule checking-derived data. The rule checking-derived data has a clear benefit to the industry, which needs the work done once across all applications instead of including similar data derivations in each BIM platform.

6.2.1. Degree of complexity/required tool

At this level, implicit relationships are often required to fulfil the checking requirements. BIM models do not usually capture this explicitly and the programme needs to derive the value from the basic BIM model data and relationship. Certain arithmetic or trigonometric calculations, such as finding the straight line distance between two points or determining if a point is inside an enclosed polygon, may be involved in rules of this class. Many IFC supporting tools are already capable of trigonometrically creating simple geometries that can be used to support such calculations without the need of a sophisticated 3D solid modeller.

We take ICC IBC 2009 as an example to illustrate the process to check this type of rule:

> *"1008.1.8 Door arrangement. Space between two doors in a series shall be 48 inches (1219 mm) minimum plus the width of a door swinging into the space. Doors in a series shall swing either in the same direction or away from the space between the doors."*

This rule has two sub-rules that need to be tested independently: a rule to check whether the gap distance between the two doors are not less than 48″ (1219 mm) after reducing it with the width of the door leaf opening into the space, and a rule to check whether the doors in series open in the same direction or if both open away from the space.

The first step in solving this rule requires an analysis of what is implicit in this code, which involves the definition of what constitutes doors in series. The illustrations given for the clause focuses only on the obvious (Fig. 3) that assumes implicitly that a human expert will interpret the data by visual inspection to identify the cases of doors in series. Computer programmes on the other hand require a precise definition to evaluate and distinguish the model from any number of possible configurations. Fig. 4 shows several possibilities of different configurations of doors in series. This is not an exhaustive list of configurations, but it highlights the need to have a generic algorithm that can assess all the possibilities and constrain them into a reasonable definition of what makes doors in series. To illustrate the example we select one particular configuration and describe the generic algorithm that can be used to handle this issue as described in the diagram in Fig. 5.

* *Illustration is reproduced based on IBC 2003 Commentary* [32]

The example above shows the generic algorithm that is applied to a particular case such as shown in Fig. 5(A). It can be applied to the cases illustrated in Fig. 4 since it deals with these key checking items:

- Determining candidate doors in series by pairing all doors adjacent to the space being checked (steps 2.0 and 3.0). These steps will require trigonometric calculations to find the path and to determine whether a straight line distance of 48″ (1219 mm) is sufficient or the need of defining a box of 30″ × 48″ (762 mm × 1219 mm) is required (Fig. 3).
- Eliminating pairs that cannot be defined as doors in series, i.e. they are too separated and do not serve the purpose of doors in series (steps 4.0 and 5.0). This step will eliminate a case illustrated in Fig. 4(E) and other spaces with similar conditions.
- Checking sub-rule no. 1 requires doors to be either opening in the same direction or both opening away from the space (steps 6.0 and 7.0). This can be achieved using the accepted convention of +Y-axis direction as the door swing direction and simple trigonometric calculation to determine that the point in +Y-axis is inside or outside of the Space.
- Checking sub-rule no. 2 that requires that any doors in series must have a gap of 48″ (1219 mm) between the doors discounting distance taken by the door leaf that opens into the space or there is a clear area in a box of 30″ × 48″ (762 mm × 1219 mm) in front of the door (steps 8.0 and 9.0), making use of information obtained in the steps 2.0 and 3.0 above. An additional information often inferred and necessary to limit the scope of search for the solution space is the maximum distance between doors that can be reasonably considered as candidates for the doors in series. In this example 9′–10′ (3 m) is deemed to be a reasonable distance that sufficiently covers the rule requirements and to filter out unlikely candidates from the search space.

In the specific example selected (Fig. 5(A)), the rule starts with evaluation of candidate space (step 1.0). The space returns three doors (D1, D2, D3) as its boundaries in step 2.0. With three doors, 3 pairs of doors and 2 directions need to be evaluated (step 3.0). We will describe this combinatorial issue more detail in the next section. Since there is no door pair that has distance longer than specified, no door-door path is eliminated (Steps 4.0 and 5.0). Using the sub-rule no. 1, i.e. Doors in series must swing in the same direction or both swing away from the space (steps 6.0 and 7.0). P2 connecting D2 and D3 will fail this test for both directions. P1 and P3 will pass both sub-rule no. 1 and sub-rule no. 2 that specifies the distance between the two doors minus the leaf door that opens into the space (steps 8.0 and 9.0).

6.2.2. Example of combinatorial issue

The example of the door in series above (Fig. 5(A)) illustrates the need for a rule checking system to deal with combinatorial issues. The space being checked offers six combinatorial configurations to be evaluated and there can be any number of the possible configurations that meet or fail the requirement. Table 2 below describes the six conditions:

Depending on the exact rule that is being applied, the combination may be reduced. The above rule (IBC 1008.1.8) is in the context of Path of Egress. In this case only one path direction is relevant depending on the direction of the Egress, which leaves only either cases 1, 3, 5 or 2, 4, 6 to be relevant. While the combination is reduced in this case, the result is unaffected, i.e. Path 2 (P_2 between D_2 and D_3) fails the checking criteria.

6.3. Class 3 — Rules that require extended data structure

This class of rule requires an extension to the data structure that encapsulates higher level semantic conditions of building data (Fig. 6). The main idea is to be able to "compute once, use many" since such information typically requires extensive computation often involving geometry operations. Multiple data structure construction strategies may apply. We offer one of the possibilities. Typical use of this class of rules is building code checking that involves complex requirements.

Table 2
Six combinatorial cases in rule check for IBC 1008.1.8.

Case	Potential path	Door swing direction	Condition to satisfy [Swing (D_i) = S && Swing (D_j) = S′] or [Swing (D_i) = S′ && Swing (D_j) = S′]
1	P_1 (D_1, D_3)	Swing (D_1) = S′ Swing (D_3) = S	Pass
2	P_1 (D_3, D_1)	Swing (D_3) = S Swing (D_1) = S′	Pass
3	P_3 (D_1, D_2)	Swing (D_1) = S′ Swing (D_2) = S	Pass
4	P_3 (D_2, D_1)	Swing (D_2) = S Swing (D_1) = S′	Pass
5	P_2 (D_2, D_3)	Swing (D_3) = S Swing (D_2) = S	Fail
6	P_2 (D_3, D_2)	Swing (D_3) = S Swing (D_2) = S	Fail

Where $P_i(D_i, Dj)$ path i connecting doors D_i and D_j in the direction of $D_i \rightarrow D_j$; $Swing(D_i) = S$ or S' swing direction of door I, either into Space (S) or away from space (S').

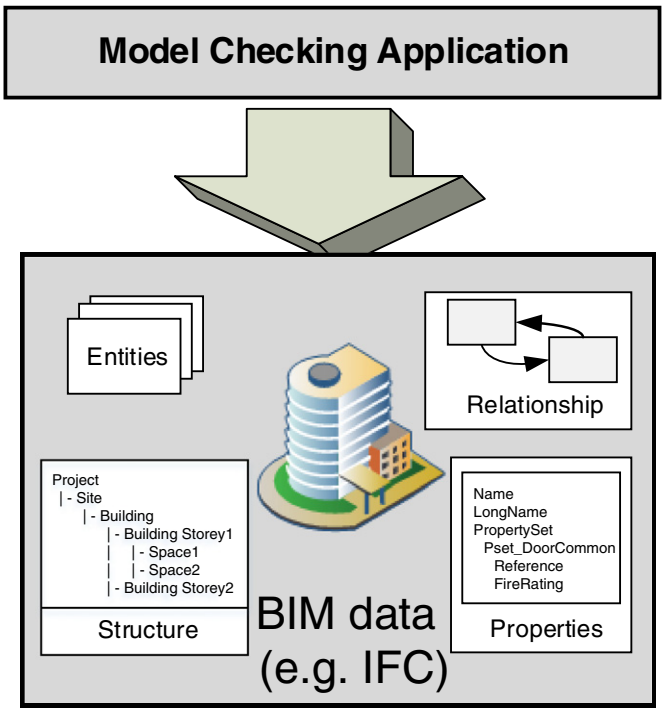

Fig. 1. Diagram for typical application implementing Class-1 rules.

6.3.1. Degree of complexity/required tool

To be able to solve rules in this class, software relying on geometrical, topological and other properties and algorithms is often required. Spatial relationships may be important. The spatial relationship of object geometry has been a subject of research especially by Borrmann [10]. To effectively deal with complex geometrical and spatial operations, a Solid modelling library may be required to perform complex geometric operations. Derivation of topological graphs may be required as well and algorithms such as shortest path may be involved. The most mature implementation that does these is FORNAX™, which is the base implementation of CORENET ePlanCheck [34,53,54]. Due to Intellectual Property issues, we are not able to publish many technical details of FORNAX™ implementation. Fig. 7 shows FORNAX™ system architecture, which makes use of a solid modeller as its geometry engine to perform advanced geometry operations.

FORNAX™ extends IFC model data with a new structure that defines higher level abstractions of the building model [54,55]. One example implementation described is:

- *Singapore Fire Safety and Shelter Department Fire Code 2.3.5* (Fig. 8) *(as implemented in CORENET IBP)*
  Basement Exit Staircase

  a) *Any exit staircase which serves a basement storey of a building shall comply with all the applicable provisions for exit staircase, and*
  b) *Such exit staircase shall not be made continuous with any other exit staircase which serves a non-basement storey of the building, and*

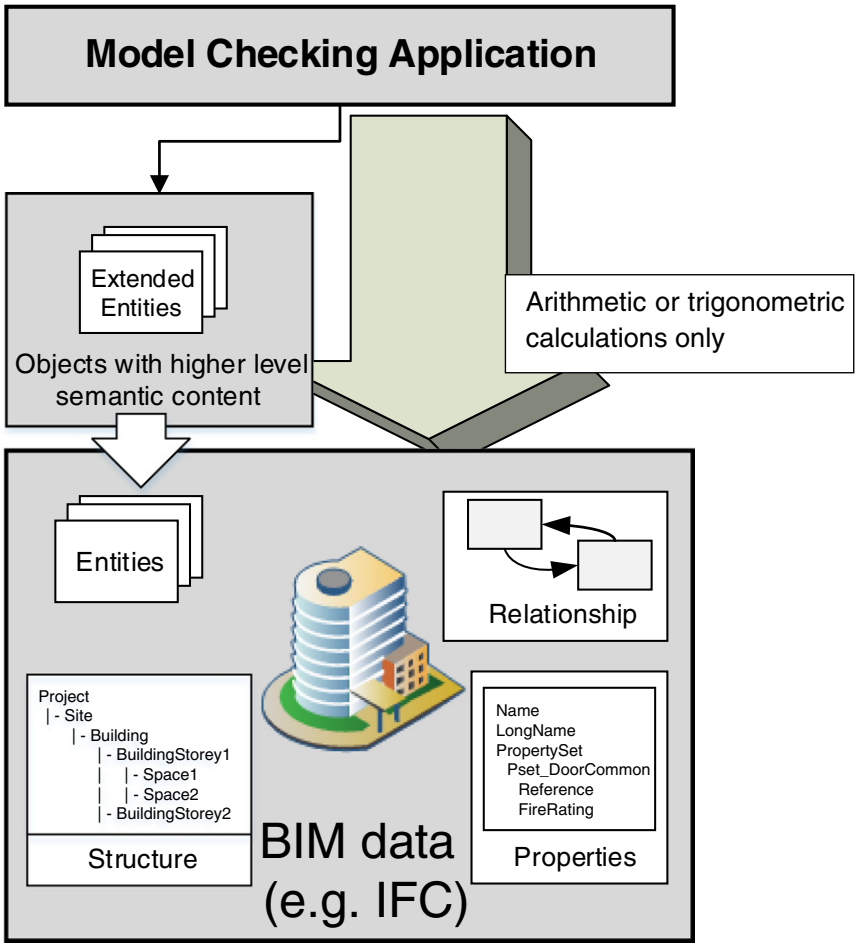

Fig. 2. Diagram for typical application implementing Class-2 rules.

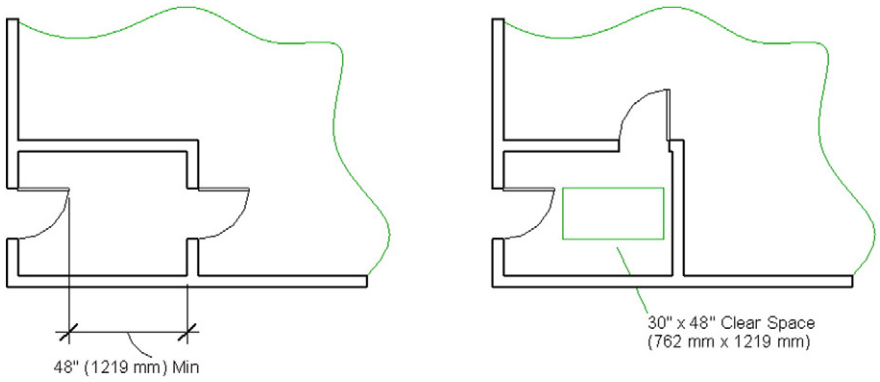

Fig. 3. Diagrams illustration of doors in series for IBC 1008.1.8.

c) *Basement exit staircases which are vertically aligned with the exit staircases of non-basement storeys shall be separated from such other exit staircases by construction having fire resistance for a minimum period equal to that required for the enclosure.*

This rule requires new concepts to be introduced:

- Vertical Shaft, that will be useful not only to connect exit staircase spaces to form exit path through the staircase, but also useful for other fire safety requirements, e.g. fire rating of the shaft or checking requirement for pressurization, which requires the compartment to be watertight.
- Discharge, which is access that is directly opened to the ground level oriented towards outside of the building.
- Separation of the discharge space from the upper level and the basement if it is modelled as one space separated by other objects such as railing.

Here is another example in this category:

- IBS CP10 installation and servicing of electrical fire alarm systems, Clause 4.3 spacing and locations of detector
  Clause 4.3.3
  Spacing between detectors.

1. *The distance between detectors for flat ceiling shall not exceed:*
   a) *7 m for areas other than corridors and*
   b) *10 m for corridors* (*See* Fig. 9).

This rule requires a sophisticated geometry engine to create a coverage map using distance triangulation between detectors [23]. This allows identification of the nearest neighbour detectors and hence the distance between the detectors can be checked (Fig. 10). This example shows the need to "compute once use many" for generating the triangulated network, in that it can be re-used in many other rules related to the placements of detectors. The same technique is also required in other similar clauses applicable to sprinkler systems.

6.4. Class 4 — Rules that require a "proof of solution"

This class of rules is a collection of rules that do not strictly ask for compliance or non-compliance, but rather requires a "proof of solution". The rules usually focus more on how the building model proves compliance rather than merely fulfilling prescribed criteria (Fig. 11). It generally represents performance based codes or other similar rules. However, it can also be applied to rules that can be modified to satisfy the compliance

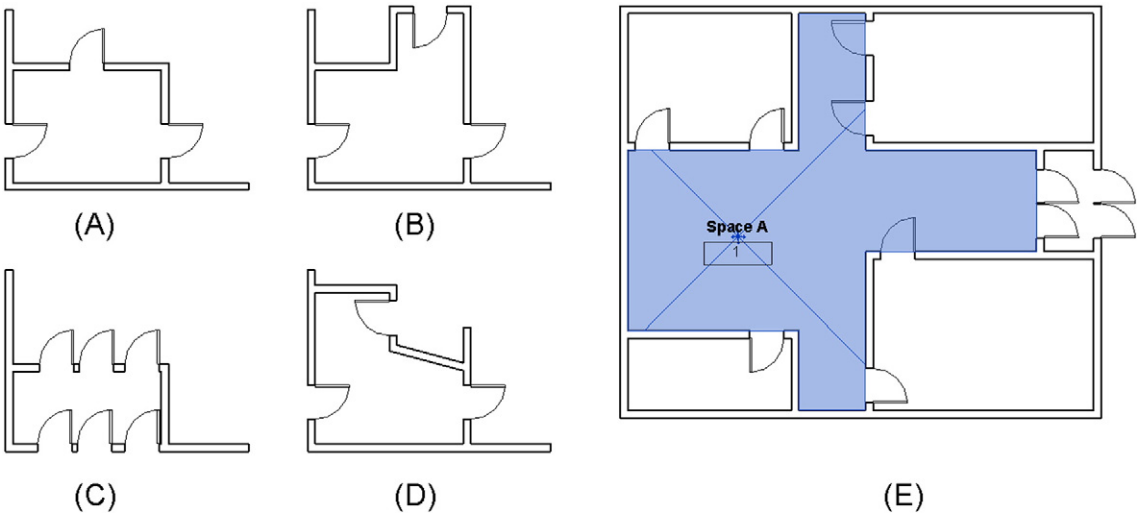

Fig. 4. Several possibilities that should be considered when evaluating a space for doors in series. In (E) the focus is on Space A.

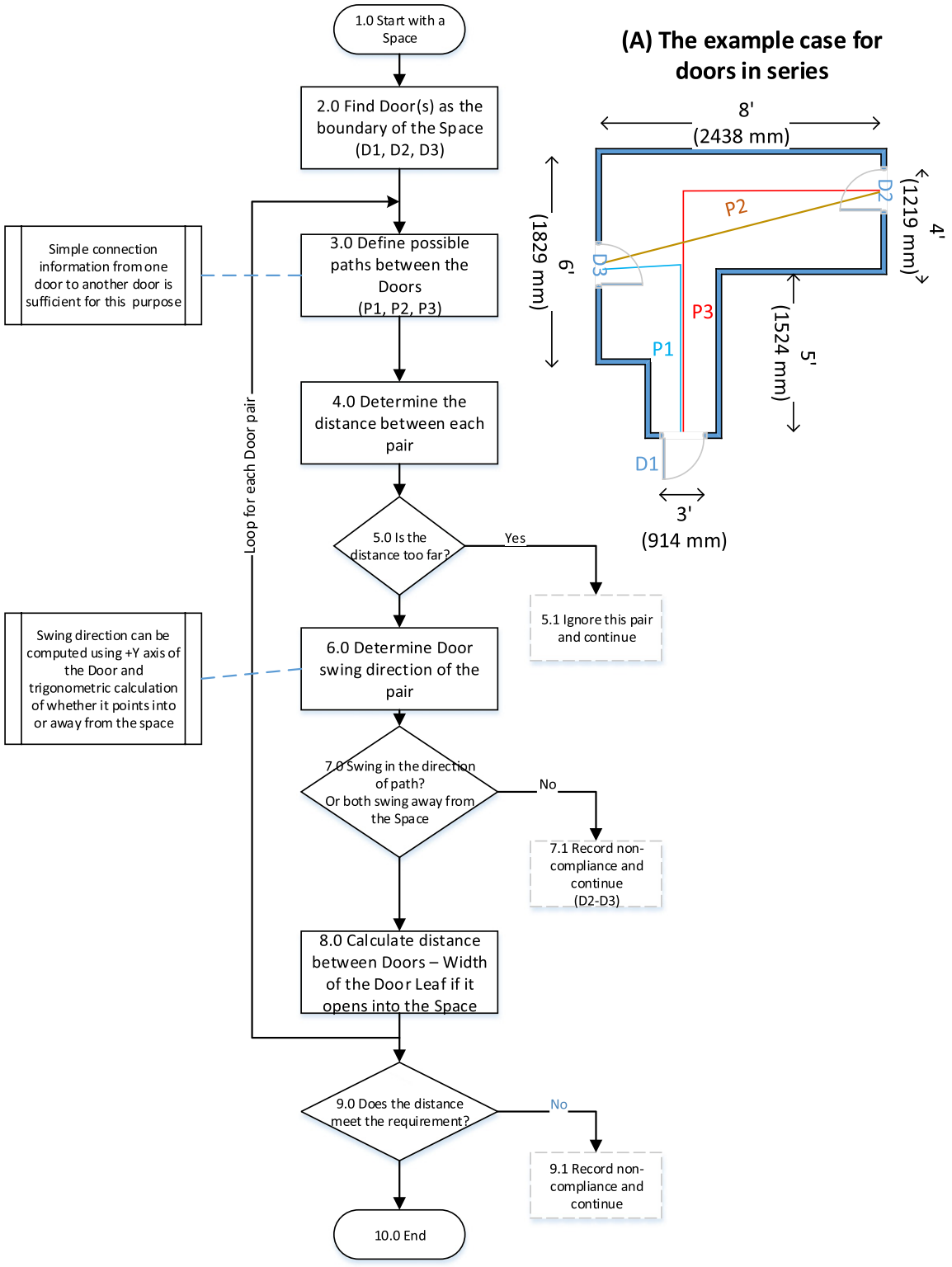

Fig. 5. Checking process for IBC 1008.1.8.

by looking into a solution in the form of additional model data being inserted into the existing one, usually either temporarily and virtually. Generally the application of this class of rules is more interested in the solution, which may have more than one acceptable answer, all of which eventually can be traced to the final answer of whether a design (or the design with additional information added into the design) complies with what the rules expect. The source of the solution is typically a knowledge-based facility that is coded into the system [31,60]. Such knowledge base is typically captured and continuously updated as the new knowledge arises. For example in the case of design that

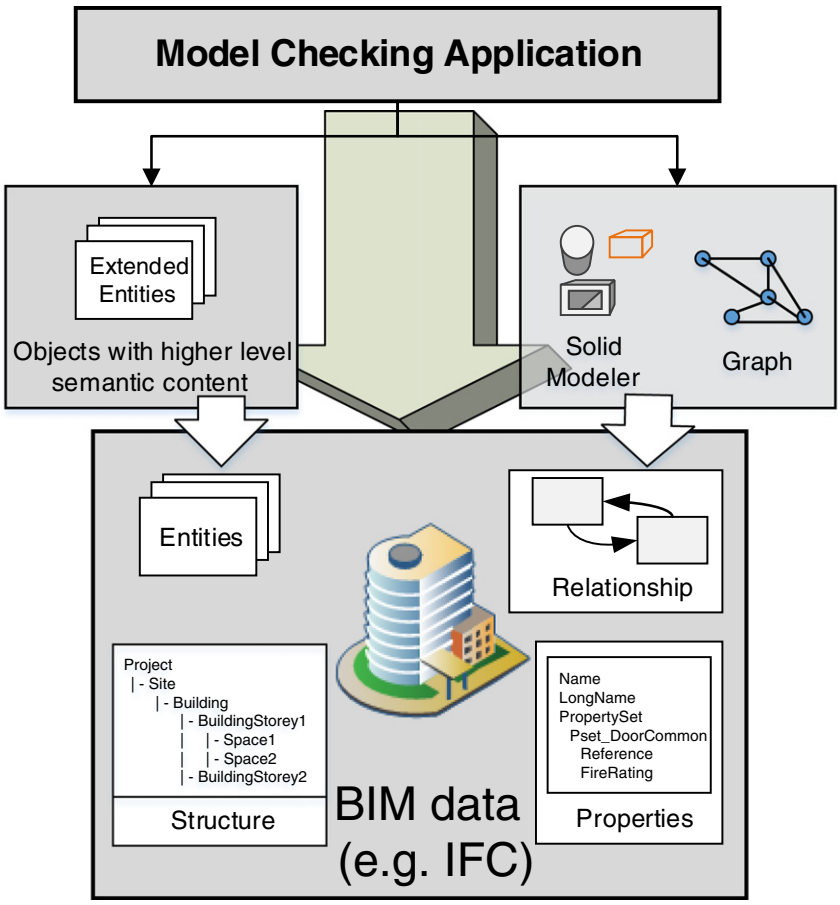

Fig. 6. Diagram for typical application implementing Class-3 rules.

affects fatality in a construction site, design rule such as "Design columns with holes at 21 and 42″ (0.53–1.06 m) above the floor level to provide support locations for lifelines and guardrails". This rule should be added into the knowledge base as the result of statistics that there were relatively high numbers of fatalities caused by this reason [8]. In recent years, with the increasing number of "green" buildings, new types of installation such as photovoltaic (PV) panels on the roof surfaces increases the risk from falling and therefore would require a new rule to define protection around the edges and distance of such installation to the edges [17]. This new knowledge should be easily added into the existing system and automatically extend the coverage of an existing rule checking capability that deals with protection from falling.

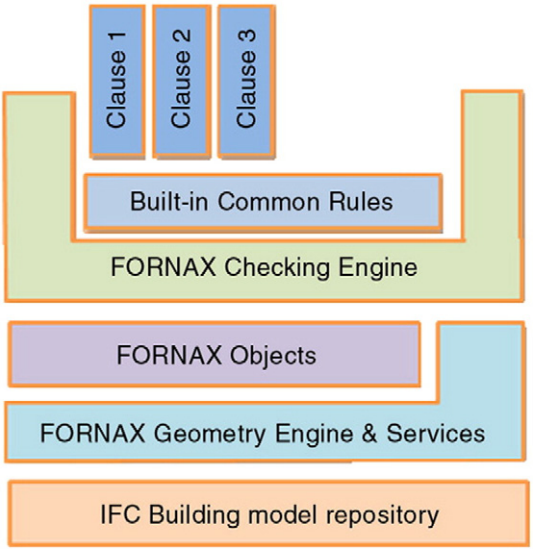

Fig. 7. FORNAX system architecture.

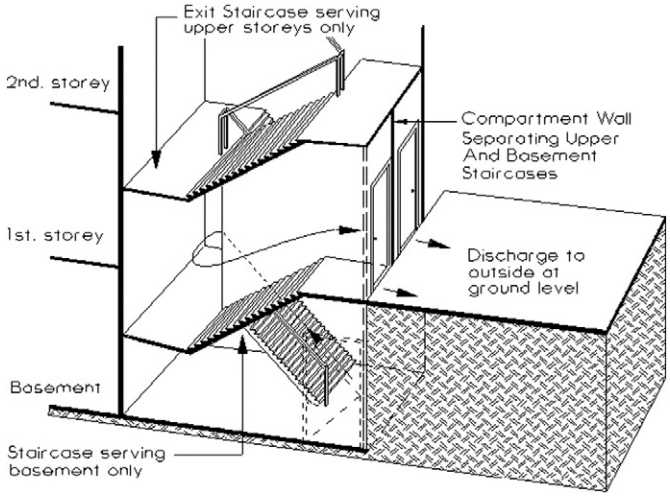

Fig. 8. Illustration for IBP fire code clause 2.3.5 (a, b and c).

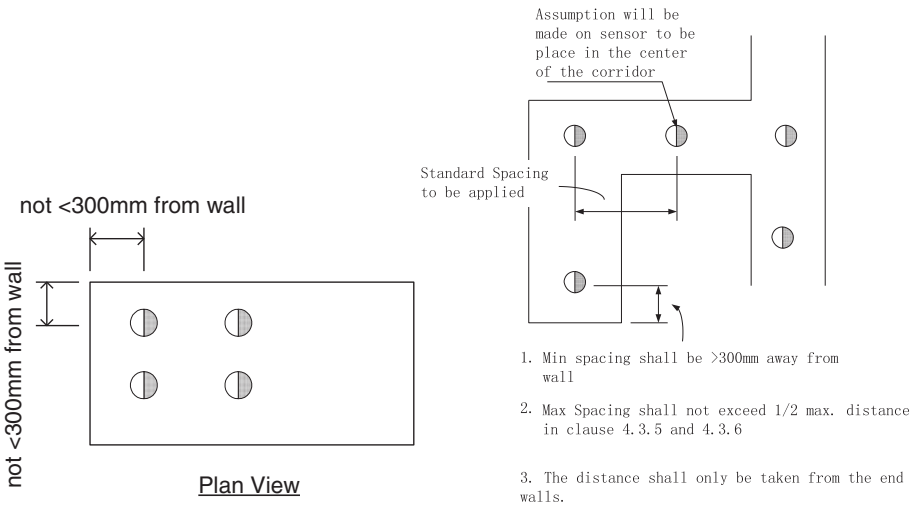

Fig. 9. Illustration for IBS CP10 Clause 4.3.3.

6.4.1. Degree of complexity/required tool

We expect that the degree of complexity does not significantly increase from the other classes, but it has new requirements in order to capture knowledge and present it as a viable solution. This enters the realm of expert systems, though it probably focuses more on domain specific knowledge [16]. Many of the applications currently seem to be suitable for the construction domain where there are many temporal conditions that highlight the need for solutions, such as those related to temporary structures (shoring and temporary structural supports), and also those related to safety issues.

This is a new area of rule checking. There are not many examples available currently. The work on safety from falling done by the Georgia Institute of Technology serves as a good example of an implementation for this class of rules [60].

There is also a possibility that rules that traditionally fall into other classes, especially Class-3, will evolve into Class-4 when one looks at the problem in a different perspective. For instance, the example presented in Class-3 (IBS CP10 4.3.3) can be restated as a rule. If this is so, then the solution would be an input that the designer could use to correct his design when non-compliance is detected in terms of the correct placement of the detectors. In this case an algorithm can be developed using the same concept of triangulation to assist the designer to highlight suggested placements of the detectors that will satisfy the rule by optimizing the distance between the detectors

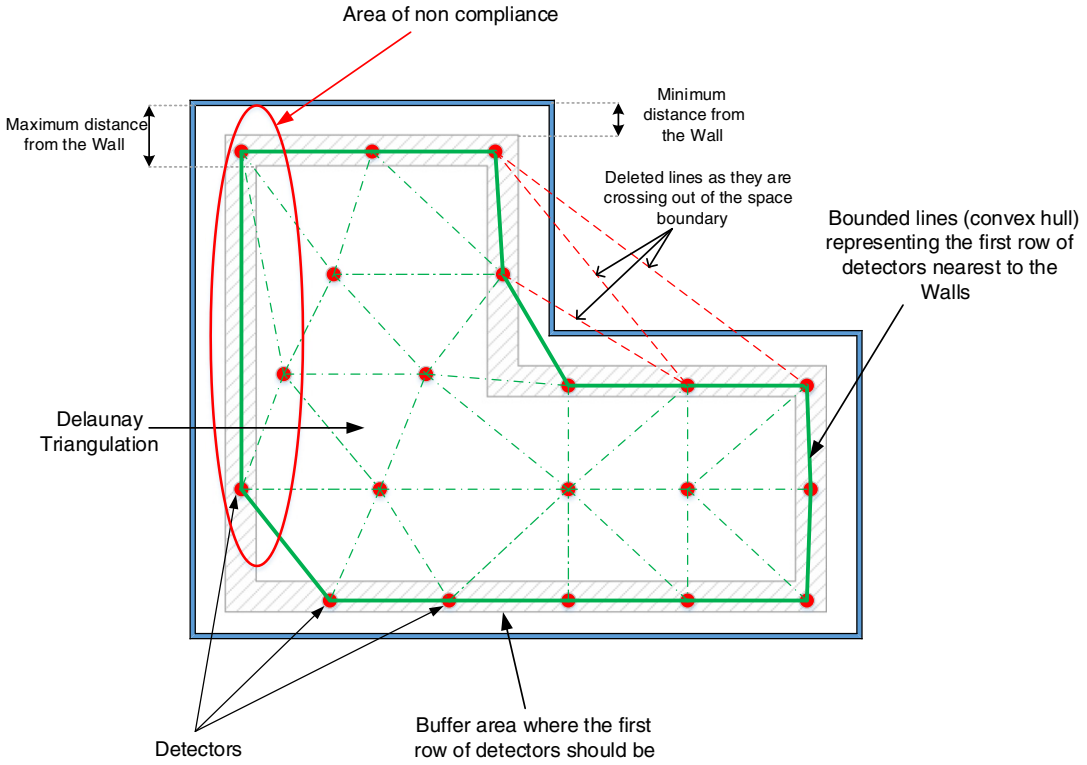

Fig. 10. Coverage map of detectors for detection of distances and identifying the first row.

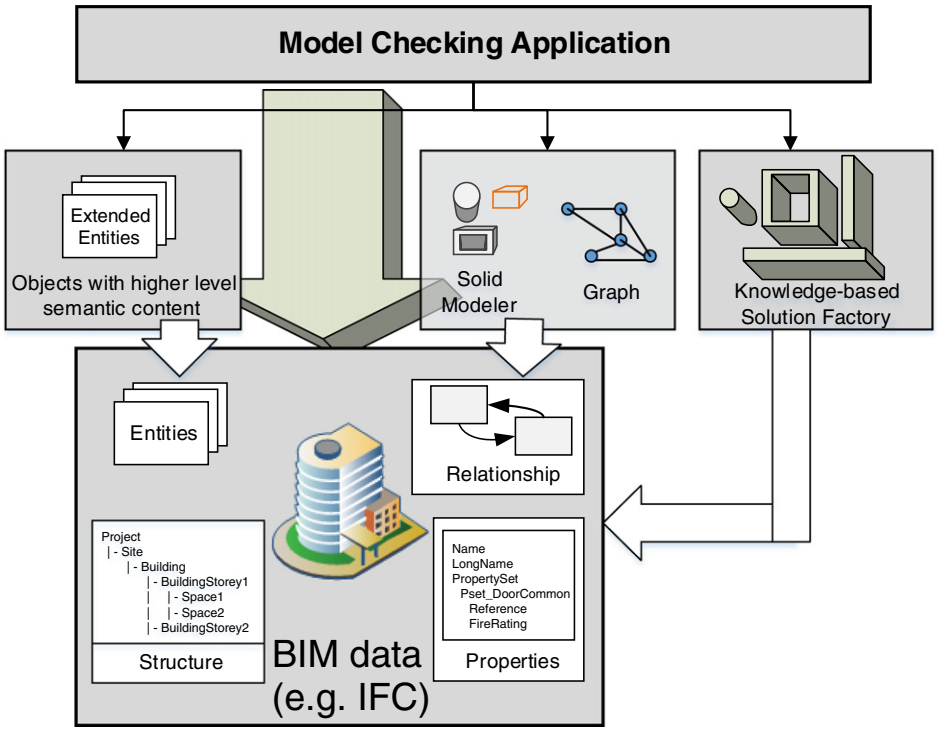

Fig. 11. Diagram for typical application implementing Class-4 rules.

and making sure the first row fits into the buffer space as highlighted in Fig. 10.

7. Discussion

As outlined in [19], rule checking involves:

1. Converting rules written for manual interpretation into a form that is amenable for computer implementation; this is a major issue for legacy systems such as building codes; eventually rule languages may be defined that are relatively semantically unambiguous to both knowledgeable computers and people.
2. Defining the model view needed to support checking, in terms of the object, attributes and relations required to either directly check the data (Case 1), or to allow the checking procedures to derive the data needed for checking (Case 2 or 3).
3. Executing the rule, dealing with the logic of the rule, in terms of nested conditions, combinatorial conditions and the logical issues of existence tests versus universality tests.
4. Reporting back to interested parties the results of the rule checking, in a manner facilitating correction of identified issues.
5. and in the last case, setting up procedures for automatically correcting rule failures.

This list offers a beginning guide to the services that an effective rule checking tool could provide. The paper elaborates the computational capabilities needed to support issues #2 and #3. A geometric modelling toolkit is a clear need in addressing implicit spatial properties. A graph processor has been integrated into checking applications to deal with circulation movements. The geometric algorithms to support building rule checking (based on various approaches) are readily available in the literature [10,25,36,37]. Easy ways to identify decision trees, which can include within them checks of all occurrences of a condition and checks that a required condition exists, are fundamental to rule checking.

8. Conclusions

This paper classifies various rules for application in building models according to their computational complexity and requirements imposed on the rule execution environment. The first class of rules that checks explicit entities and values is relatively straightforward to implement and yet it has enormous potential to help improve building model quality and predictability, especially if there is a consistent and easy way for users to define the rules to check basic information that complies with the definition of MVDs. The second level requires some level of expertise and probably requires planned development efforts. It is in the complex types (cases 3 and 4) where progress is most lacking. The classes of rules do not imply any degree of importance, as they all address valid needs for automated rule checking but with different scopes and uses. Having the classes and scope defined along with the typical or expected tools to support them helps identify the focus for further research and development where the industry will benefit most.

System environments are needed to support rule checking on both BIM platforms and IFC development environments. We have tried to outline some of the prerequisites for an effective tool definition environment. We may not see a “one size fits all” type of application, but we hope that this research will help software vendors to focus attention on providing the platform facilities needed to support different classes of rule definition. The survey also identifies patterns that exist across various types of rules that span beyond building regulatory code checking. In fact, the potential benefits to the industry will probably be greater if the automated code checking system is able to check and satisfy many rules within domain specific problems or best practices.

With this paper, the authors hope that further advancement in the area of automated rule checking can become more focused; the benefit of such systems can be immediately applicable to broad base use in the AEC industry. Clearly targeting one or more rule classes will help both the AEC industry and software vendors to provide applications that will be immediately useful to the industry and will spur further innovation in this broad area.

Table 3
Example of rules that fit the four rule classification.

Rule classification	Source	Rule description
Class-1 Checks based on a single or small number of explicit data	Solibri model checker (SMC) rules [52]	General BIM file structure rules *"Model should have components"*, checks: • Explicit entity types • Element's attribute describing construction types, and • Assigned classification name (using IfcClassificationReference) Spatial information rules: *"Spaces must be from agreed list"*, checks: • IfcSpace. Name (space number), LongName (space name), Type (classification)
	International code council IBC 2009 [33]	*"909.20.4 Mechanical ventilation alternative. The provisions of Sections 909.20.4.1 through 909.20.4.4 shall apply to ventilation of smokeproof enclosures by mechanical means.* *909.20.4.1 Vestibule doors. The door assembly from the building into the vestibule shall be a fire door assembly complying with Section 715.4.3. The door assembly from the vestibule to the stairway shall not have less than a 20-minute fire protection rating and meet the requirements for a smoke door assembly in accordance with Section 715.4.3. The door shall be installed in accordance with NFPA 105."* Checks: • Pset_DoorCommon.FireRating *"1008.1.2 Door swing. Egress doors shall be of the pivoted or side-hinged swinging type."* Checks: • IfcDoorStyle.OperationType for value of the SWING type (single or double swing)
	Singapore fire code (prescriptive) Chapter 3 clause 3.2.5.e [50]	*"The fire command centre shall be separated from other parts of the same building by compartment walls and floors having fire resistance of at least 2 h."*, checks: • All space boundary elements (through IfcRelSpaceBoundary) and their Pset_<object>Common.FireRating for value at least 2 h.
	MVD checking	Checks: • Existence of valid entities (IfcObject and its subtypes) • Valid relationship between entities (IfcRelationship and its subtypes) • Their attributes and property sets and possibly their valid values.
Class 2 Checks based on simple derived attribute values	SMC rules [52]	Component properties rules: *"free area in front of components rule"*, checks: • There is no obstruction in front of an object like windows, doors or specified furniture (It can also check both sides of the object). • It requires extra information processing what objects could be in front or behind Door or Window. *"Components must touch other components"*, checks: • Object spatial connectivity by face coincidence other defined objects and how much the area intersection and checks if object has the acceptable height.
	Construction coordination	Clash detection • It requires an object to be aware of any other objects around it.
	Singapore Code of practice on sewerage and sanitary works; Part 3.2 Sanitary plumbing systems [49]	*(42) 3.2.2 Design criteria* *f) The discharge pipe shall not be located in places where it can cause health and safety hazards such as locating the discharge pipe above any portable water storage tank and electrical transformer/switchgear.* Checks: • Discharge Pipe needs to be aware of specific objects underneath it as long as there is no protection.
	Hospital design	*"All patient rooms must be visible from the nurse station"*, checks: • Simple geometry computation that evaluate "line of sight" from the nurse station to entrances for each of patient rooms under supervision of the nurse station.
Class 3 Checks based on extended data structure	Construction best practice	*"There has to be a direct access to MEP control devices that are concealed for the purpose of maintenance through access opening of minimum size of 18″ × 24″ (457 mm × 610 mm). Where there is no direct access from the opening, it is permitted to have indirect access through crawl space that has minimum clear dimension of 22″ × 30″ (559 mm × 762 mm) at the cross section uniformly applied throughout the entire crawl space."* • Requires spatial information of the concealed space, creation of sweep solid to create a required crawl space starting from the nearest access opening to the front of the control device.
	GSA courthouse design [24] — circulation requirements [51]	"*The USDC courtroom should be accessible from the Prisoner HLDG.CELL only through secure circulation (3-14-16)*" "*The judge's chambers are accessed from restricted circulation with convenient access to the courtrooms.*" • These requirements imply the need to have space connection graph to allow evaluation whether a space is accessible from another space though one or multiple intermediate spaces such as circulation spaces. All traversed spaces must be of type 'secure'.
	Singapore Building Control Act (Chapter 29) 1990, Division 5 Staircase [49]	*Regulation 44 (1) Protection of staircase and staircase landing* *Every staircase or staircase landing shall be protected on any side overlooking an air-well, courtyard, void or external open space by either a railing, parapet or balustrade capable of resisting the lateral loading as specified in the Table 4 of the fourth Schedule*, checks: • Side edges of the stair, the number of steps and evaluating existence of protection as well as the depth of fall from the edges Involves implicit requirements: • The protection may start only when the staircase has more than 5 steps. • The danger of falling is defined to be an open edge that has a drop more than 1 m high.
	ICC IBC 2009 [33]	*1005.2 Door encroachment. Doors, when fully opened, and handrails shall not reduce the required means of egress width by more than 7 in. (178 mm). Doors in any position shall not reduce the required width by more than one-half. Other nonstructural projections such as trim and similar decorative features shall be permitted to project into the required width a maximum of 1–1/2 in. (38 mm) on each side.* • This rule becomes more complex when one considers in the real life scenarios where: • Egress path is not simple and straight. • There are more than one door opening into the Egress path.

Table 3 (*continued*)

Rule classification	Source	Rule description
Class 4 Checks and suggests corrective actions or solutions	Occupational Safety and Health Administration (OSHA) Protection from falling in Construction [45]	*OSHA health and safety rule for protection from falling during construction*
	Construction and operation & maintenance related best practice	*"Ensure a footprint of free space for placing a ladder to reach access opening on the concealed ceiling, or MEP control devices"* • While this rule may simply return compliance or non-compliance of access opening or MEP device that do not have enough footprint of free space underneath, it is a good candidate for highlighting how much space it requires fulfilling the requirement. It will help designers to alter the model to satisfy this requirement during the pre-construction coordination process. One possible solution is placing a ladder object underneath the access opening or MEP control device and highlight those either there is no enough free space or the item to access is beyond the reach by standard ladder. *"Find possible accessible path to move a large equipment into a building under construction"*, or similar requirement for Plant: *"Find possible path and placement of crane to lift a plant equipment out of its current position for removal or replacement of the equipment"* • This requirement involves analysis of the building or plant layout to identify the possible paths and clear space for moving the object into the final position or out from its current position. It involves path generation, accessibility analysis on the path possibly using swept solid along the path, n-shortest paths algorithm, and clash detection.

Table 4
List of research works and commercial applications implementing different classes of rules.

Class 1 Checks based on explicit data	Class 2 Checks based on simple derived attribute values	Class 3 Checks based on extended data structure	Class 4 Checks and suggests corrective actions or solutions
Solibri model checker [52]	Solibri model checker [52]	GA Tech GSA courthouse design circulation check [37], which is based on Solibri model checker	GA Tech's automatic safety checking of construction models and schedule [60]
Automated MVD checks: – IFC2x3 CV2.0 certification by BuildingSMART and iabi [11] – Digital Alchemy, that is used in GSA CD BIM 2010 certification [2]	Applications with clash detection capability: – Navisworks [3] – Tekla BIMsight [57]	FORNAX™ as implemented in CORENET ePlanCheck [54]	Various proprietary domain specific implementations[a], e.g.: – Crane Simulation system by JGC using Navisworks. – Autodesk's simulation for Lockheed Martin on risk assessment of moving large objects inside a building based on 3D Studio Max.
Revit model review (subscription add-in) [5]	FORNAX™ that is used as an engine for CORENET ePlanCheck implementation [53]		
Navisworks using search set[b] [4]			
Research project by QUT, Australia [18]			
EDM rule schema based: – Norway pilot project [35] – Xabio [1]			

[a] No public information available, they are based on authors' involvements with the developers of the applications.
[b] IFC support in Navisworks for search set will be limited to entities and properties.

References

[1] AEC3, Xabio, http://www.aec3.com/en/5/5_010_XABIO.htm 2014 (Retrieved 03/01/2014).
[2] D. Alchemy, IFC BIM validation service, http://www.digitalalchemypro.com/html/services/IfcBimValidationService.html 2014 (Retrieved 03/01/2014).
[3] Autodesk, Navisworks clash detection, http://www.autodesk.com/products/autodesk-navisworks-family/features 2014 (Retrieved 03/01/2014).
[4] Autodesk, Navisworks search set feature, http://www.augi.com/library/power-up-with-search-and-selection-sets 2014 (Retrieved 03/01/2014).
[5] Autodesk, Revit model review, http://wikihelp.autodesk.com/Revit/enu/2013/Help/00005-More_Inf0/0001-Subscrip1/0035-Autodesk35 2014 (Retrieved 03/01/2014).
[6] BCA, Singapore BIM guide, http://www.corenet.gov.sg/integrated_submission/bim/BIM/Singapore%20BIM%20Guide_V2.pdf 2013 (Version 2).
[7] T. Beach, T. Kasim, H. Li, N. Nisbet, Y. Rezgui, Towards automated compliance checking in the construction industry, in: H. Decker, L. Lhotská, S. Link, J. Basl, A. Tjoa (Eds.), Database and Expert Systems Applications, vol. 8055, Springer, Berlin Heidelberg, 2013, pp. 366–380.
[8] M. Behm, Linking construction fatalities to the design for construction safety concept, Saf. Sci. 43 (8) (2005) 589–611, http://dx.doi.org/10.1016/j.ssci.2005.04.002.
[9] BIMForum, Level of development specification — for building information models, http://bimforum.org/wp-content/uploads/2013/08/2013-LOD-Specification.pdf: BIMForum 2013.
[10] A. Borrmann, E. Rank, Topological analysis of 3D building models using a spatial query language, Adv. Eng. Inform. 23 (4) (2009) 370–385.
[11] buildingSMART, Global Testing Documentation Server (GTDS), http://gtds.buildingsmart.com 2014 (Retrieved 03/01/2014).
[12] BuildingSMART, IFC2x3 CV2.0 Certification sub-schema (MVD), http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0/sub-schema.
[13] BuildingSMART, IFC solutions factory — the model view definition site, http://www.blis-project.org/IAI-MVD/.
[14] BuildingSMART, FM basic handover, http://www.buildingsmart-tech.org/specifications/ifc-view-definition/fm-handover-aquarium/fm-basic-handover 2009 (http://www.buildingsmart-tech.org/specifications/ifc-view-definition/fm-handover-aquarium/fm-basic-handover).
[15] J. Choi, J. Choi, I. Kim, Development of BIM-based evacuation regulation checking system for high-rise and complex buildings, Autom. Constr. 46 (2014) 38–49, http://dx.doi.org/10.1016/j.autcon.2013.12.005.
[16] E.A. Delis, A. Delis, Automatic fire-code checking using expert-system technology, J. Comput. Civ. Eng. 9 (2) (1995) 141–156.

[17] K.S. Dewlaney, M.R. Hallowell, B.R. Fortunato III, Safety risk quantification for high performance sustainable building construction, J. Constr. Eng. Manag. 138 (8) (2011) 964–971.
[18] L. Ding, R. Drogemuller, J. Jupp, M. Rosenman, J. Gero, Automated code checking, CRC CI International Conference 2004, 2004.
[19] C. Eastman, J.-m Lee, Y.-s Jeong, J.-k Lee, Automatic rule-based checking of building designs, Autom. Constr. 18 (8) (2009) 1011–1033.
[20] S.J. Fenves, Tabular decision logic for structural design, Proc. Am. Soc. Civ. Eng. 92 (1966) 473–490.
[21] Fiatech, Automated code plan checking tool-proof-of-concept (Phase 2), http://www.fiatech.org/project-management/projects/593-automated-code-plan-checking-tool-proof-of-concept#sthash.RaV9fZnR.dpuf 2014 (Retrieved 03/01/2014).
[22] Fiatech, AutoCodes project: phase 1, proof-of-concept final report, http://www.fiatech.org/images/stories/techprojects/project_deliverables/Updated_project_deliverables/AutoCodesPOCFINALREPORT.pdf 2012.
[23] S. Fortune, Voronoi diagrams and Delaunay triangulations, Computing in Euclidean geometry, 11992. 193–233.
[24] GSA, Courts design guide, http://www.gsa.gov/graphics/pbs/Courts_Design_Guide_07.pdf.
[25] C.S. Han, Computer Models and Methods for a Disabled Access Analysis Design Environment, Stanford University, 2000.
[26] J. Hietanen, S. Final, IFC model view definition format, International Alliance for Interoperability2006.
[27] J. Hietanen, S. Lehtinen, The Useful Minimum, Tampere University of Technology, Tampere, 2006.
[28] E. Hjelseth, Foundation for Development of Computable Rules, Norwegian University of Life Sciences (UMB), Dept. of Mathematical Sciences and Technology, Norway, 2009.
[29] E. Hjelseth, N. Nisbet, Exploring semantic based model checking, Proceedings of the 2010 27th CIB W78 International Conference, 2010.
[30] E. Hjelseth, N. Nisbet, Capturing normative constraints by use of the semantic mark-up (RASE) methodology, CIB W78 2011 28th International Conference-Applications of IT in the AEC Industry, 2011.
[31] Z. Hu, J. Zhang, Z. Deng, Construction Process Simulation and Safety Analysis Based on Building Information Model and 4D Technology, Tsinghua Sci. Technol. 13 (Supplement 1(0)) (2008) 266–272, http://dx.doi.org/10.1016/S1007-0214(08)70160-3.
[32] ICC, ICC IFC 2003 commentary, http://www.constructionbook.com/2003-icc-international-fire-code-ifc-commentary-3410s03/2003-international/ 2003.
[33] ICC, IBC 2009, http://shop.iccsafe.org/codes/2009-international-codes/2009-international-building-code-tab-combo.html 2009.
[34] L. Khemlani, CORENET e-PlanCheck: Singapore's automated code checking system, http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html 2005 (Retrieved 03/01/2014, 2014).
[35] M. Lê, F. Mohus, O. Kvarsvik, M. Lie, The HITOS project—A full scale IFC test, Proceedings of the 2006 ECPPM Conference, 2006.
[36] J.K. Lee, Building Environment Rule and Analysis (BERA) Language and its Application for Evaluating Building Circulation and Spatial Program(PhD) Georgia Institute of Technology, 2011.
[37] J.M. Lee, Automated Checking of Building Requirements on Circulation Over a Range of Design Phases(PhD) Georgia Institute of Technology, 2010.
[38] X. Liu, B. Akinci, M. Bergés, J.H. Garrett Jr., Domain-specific querying formalisms for retrieving information of HVAC systems, J. Comput. Civ. Eng. 28 (1) (2014) 40–49, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000294.
[39] J.P. Martins, A. Monteiro, LicA: a BIM based automated code-checking application for water distribution systems, Autom. Constr. 29 (2013) 12–23, http://dx.doi.org/10.1016/j.autcon.2012.08.008.
[40] N. Nawari, The challenge of computerizing building codes in BIM environment, Proceedings of the 2012 Asce International Conference on Computing in Civil Engineering, Clearwater Beach, Florida; Jun 17–20, 2012, 2012.
[41] N.O. Nawari, Automating codes conformance in structural domain, ASCE International Workshop on Computing in Civil Engineering, Miami, Florida, June 19–22, 2011, 2011.
[42] N.O. Nawari, Automated code checking in BIM environment, 14th International Conference on Computing in Civil and Building Engineering, Moscow, Russia, 27–29 June, 2012, 2012.
[43] novaCITYNETS, novaSPRINT awarded CORENET integrated plan checking system, http://www.novacitynets.com/news.htm 2000 (Retrieved 03/01/2014, 2014).
[44] novaCITYNETS, FORNAX Makes Headway in Kingdom of Saudi Arabia [Press release], 2013.
[45] OSHA, Fall Protection in Construction, OSHA, 1998. 3146.
[46] J. Park, Y. Lee, Developing a Performance-based design system with semantic interoperability, in: C. Stephanidis (Ed.), Hci International 2013 — Posters' Extended Abstracts, vol. 373, Springer, Berlin Heidelberg, 2013, pp. 69–73.
[47] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, J. Van Campenhout, A semantic rule checking environment for building performance checking, Autom. Constr. 20 (5) (2011) 506–518.
[48] D. Salama, N. El-Gohary, Semantic modeling for automated compliance checking, Proc., 2011 ASCE Intl. Workshop on Comput. in Civ. Eng, 2011.
[49] Singapore, Singapore building codes, http://www.corenet.gov.sg/einfo/Index.aspx.
[50] Singapore, Singapore fire code, http://www.scdf.gov.sg/content/scdf_internet/en/building-professionals/publications_and_circulars.html.
[51] B. Singapore, IFC2x2 code checking certification view, http://www.ifcwiki.org/index.php/IFC_Certified_Software 2005.
[52] Solibri, Solibri model checker, http://www.solibri.com/solibri-model-checker/functionality-highlights.html 2014 (Retrieved 03/01/2014).
[53] W. Solihin, Lessons Learned From Experience of Code-checking Implementation in Singapore, 2004.
[54] W. Solihin, N. Shaikh, X. Rong, L.K. Poh, Beyond interoperability of building models: a case for code compliance checking, BP-CAD Workshop, Carnegie Melon University, 2004.
[55] W. Solihin, N.A. Shaikh, Improving application resiliency the face of imperfect information, International Workshop on nD Modelling Road map: A Vision for nD-Enabled Construction, University of Salford, 2005.
[56] X. Tan, A. Hammad, P. Fazio, Automated code compliance checking for building envelope design, J. Comput. Civ. Eng. 24 (2) (2010) 203–211.
[57] Tekla, Tekla BIMsight clash detection feature, http://www.teklabimsight.com/helpcenter/designCoordinationTopic.jsp?topic=dcg_conflict_checks 2014 (Retrieved 03/01/2014).
[58] A. Yurchyshyna, A. Zarli, An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction, Autom. Constr. 18 (8) (2009) 1084–1098.
[59] J. Zhang, N. El-Gohary, Extraction of construction regulatory requirements from textual documents using natural language processing techniques, Proc., Comput. Civ. Eng, 2012, pp. 453–460.
[60] S. Zhang, J. Teizer, J.-K. Lee, C.M. Eastman, M. Venugopal, Building information modeling (BIM) and safety: automatic safety checking of construction models and schedules, Autom. Constr. 29 (2013) 183–195.
[61] B. Zhong, L. Ding, H. Luo, Y. Zhou, Y. Hu, H. Hu, Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking, Autom. Constr. 28 (2012) 58–70.