



# Applying rule-based model-checking to construction site layout planning tasks

Kevin Schwabe, Jochen Teizer\*, Markus König

Ruhr-Universität Bochum, Universitätsstraße 150, 44801 Bochum, Germany

## ARTICLE INFO

### Keywords:

Building information modeling (BIM)  
Business rule management system (BRMS)  
Construction site layout planning  
Drools  
Offset geometry  
Rule language, engine and checking

## ABSTRACT

Building Information Modeling (BIM) is a widely established method in the architecture, engineering, construction, and facilities management (AEC/FM) industry. Although BIM focuses on processes throughout the lifecycle of the built environment, the applications in the planning phase, e.g. the generation of construction site layouts, have not reached their full potential yet. One important example herein is the allocation and dimensioning of resources (e.g., building materials and equipment) which is typically carried out by humans according to clearly defined rules and best practices. This paper presents model-based rule checking for the planning of construction site layouts. We demonstrate that existing Business Rule Management Systems (BRMS), such as the open-source rule engine Drools, can be used. We combine Drools with the Industry Foundation Classes (IFC) to retrieve data from a building information model and use the information within the rule engine. We define general model requirements and implement a sample set of prototype rules. We also introduce the concept of offset geometry for rules that, for example, demand a known safety distance between temporary construction site elements. The developed approaches are explained and evaluated in field-realistic, practical case studies. Finally, we present a discussion how the application of the developed rule-based system may assist human decision making in tasks such as safe construction sites layout planning.

## 1. Introduction

Efforts to digitize processes in the various industrial sectors are progressing steadily. In the construction industry, digitization is closely linked to the use of Building Information Modeling (BIM). BIM describes an integrated model-based view on a facility's lifecycle which includes design, planning, construction, and operation and maintenance. As the implementation of BIM in the construction industry is evolving, many of its potential use cases are still to be investigated. For example, during the phase of construction site layout planning (CSLP) only very limited intelligent computer-based tools are available that support these complex tasks [1]. The experts often have very short time available to complete their tasks, e.g. during the tendering phase. CSLP also requires extensive knowledge of different interdisciplinary fields, e.g. construction scheduling, resource allocation, equipment optimization, material logistics, project geometry, and many more. Among other important criteria, the experience of the engineers leads to important decisions that have direct impact on cost, schedule and safety. Detailed CSLP is typically performed just before the start of construction and, as a result, provides, among other things, detailed information about the placement of construction equipment (e.g., cranes) and temporary

facilities (e.g., containers, power sources, storage areas) [2]. CSLP intends to put these temporary resources to most effective, efficient, and safe use during operation.

Despite its importance, information applied to site layout planning is kept in many organizations as internal knowledge [3]. Competitiveness further restricts access to company best practices. While these have to meet or surpass official rules and regulations (e.g., safety), the iterative process [4] from conceptual to detailed CSLP is often sub-optimal [5]. For example, the high priority of many organizations on meeting safety standards would require checking hundreds or more of potentially complex rules. This is a key reason why only few elements (e.g., cranes, temporary roads, material storage, and offices) are considered on construction site layout plans.

For above reasons, potential synergy exists by applying modern processes and technologies. The process of model checking in BIM, for example, is such a task that is often seen as a measurement for model and design quality [6]. Model checking is used for validating predefined model requirements or regulatory code compliance – often referred to as *rule checking*. CSLP also follows domain-specific rules and best-practices for the positioning and dimensioning of construction site equipment and temporary facilities.

\* Corresponding author.

E-mail addresses: [kevin.schwabe@rub.de](mailto:kevin.schwabe@rub.de) (K. Schwabe), [jochen.teizer@rub.de](mailto:jochen.teizer@rub.de) (J. Teizer), [koenig@inf.bi.rub.de](mailto:koenig@inf.bi.rub.de) (M. König).

<https://doi.org/10.1016/j.autcon.2018.10.012>

Received 11 November 2017; Received in revised form 9 October 2018; Accepted 18 October 2018

Available online 17 November 2018

0926-5805/ © 2018 Elsevier B.V. All rights reserved.

Our work proposes the application of automated rule checking to CSLP. We combine Industry Foundation Classes (IFC) [7] with a business rule management system (BRMS). The scope of this work is to investigate whether a BRMS is capable of meeting the requirements of geometric and semantic relationships of BIM-models. Furthermore, a methodology for the creation of rules for CSLP is introduced and general model requirements are determined. As for geometry-related rules, we introduce the concept of offset geometry. Rule-checking algorithms are prototypically implemented for the evaluation on a realistic building project. Our findings are discussed, and an outlook of future work is presented.

## 2. Related work

The idea of digitally assisted construction site layout planning is not new. However, rule-based approaches have to date not been considered in practice. One of the most looked upon topics in CSLP is the optimization of one or multiple tower crane positions in order to improve the workflow and logistics on construction sites. Starting with optimization algorithms in 2D in early approaches [4,8,9], further academic research considered 3D and even 4D environments later on [10–15]. The weakness of these approaches is that the equipment to be optimized was preselected. Only the equipment's position including its orientation and articulation was of interest. This means that the first and basic step of dimensioning the equipment was assumed as completed. Some of the research found that the choice and dimensioning of equipment is more dependent on time-constraints [12,14,15] than it is just a geometric problem. An example is the consideration of material peak consumption by work crews. Still, the process of dimensioning site equipment follows specific rules and best-practices. Many additional rules that also exist and which consider geometric relationships, such as keeping safe distances between other site elements and/or to neighboring buildings, have received little attention. These additional rules are either found in written form by means of regulatory texts, such as standards or codes, or in non-written form by means of expert knowledge.

The process of defining and systematically evaluating rules is called *rule checking* or *code compliance checking* [16]. Although some work on safety-rule checking shows promising results [17], the automated process of rule-based model checking is still in the early stages of research. An obstacle for widespread implementation seems to be the transcription of existing, mostly textual rules, standards and best-practices into an algorithmically readable form [18]. The uniform definition and use of rules requires a consistent approach in the form of a *rule language*. A rule language that enables users to intuitively define and check rules in their 3D model does not exist to date. Thus, the need for a designated rule language has been highlighted in several studies in the field of automated rule checking [16,17,19–21]. In order to successfully implement rule languages among the various commercially-existing software products of vendors, it is necessary to use open standard language definitions. This way the Open BIM mindset – the key idea of BIM – will be respected. For this reason, the existing language-driven approaches are evaluated in this research.

In this section, we first give an overview on traditional and modern rule checking. We then introduce Business Rule Management Systems (BRMS) that can manage rules for automatic rule checking. In order to evaluate rules, a BRMS needs to communicate with the building model. For this, the review focuses on query languages that are capable of querying complex geometric relationships between building elements. BRMS can be used to reason about the data and infer the desired information from the building model with the help of a rule language. An overview of existing rule languages is given.

### 2.1. Rule checking

Until now, rule checking in the field of BIM is mainly performed during the process of model quality and collision checking. In general,

rule checking is a subsequent procedure. It is performed after a design is complete. If the results show any rule conflicts, the design will have to be revised. For example, in commercial model checking software (e.g., Solibri Model Checker [22]) the building model can be analyzed regarding geometrical clashes, completeness of properties or relational integrity (e.g., does a column touch the floor and ceiling?). However, these rules are predefined, hard-coded and have a few editable parameters [23]. This means, that the user can only use the predefined rules for special applications. However, with such model checking software it is not possible to create new custom rule types that would offer creative functionality.

The traditional approach of rule checking, as described by Eastman et al. [16], can be structured in the four steps: (1) rule interpretation, (2) building model preparation, (3) rule execution and (4) reporting of checking results. At first, the existing, mostly textual rules are transcribed in a form that can be interpreted by computers. Furthermore, the model to be checked needs to meet certain modeling requirements, so that the given rules can be executed. This includes both geometry and property related requirements. When both steps are complete, the rules can be executed and checked against the model. Afterwards potential conflicts have to be reported. This feature needs to have a textual and a visual component, so that the user can intuitively handle the conflict. Although this procedure is generic, the individual manual execution of each step is not predefined. Depending on the application, individual problems occur that need to be solved manually.

A modern and advanced approach of rule checking, as described by Solihin et al. [24], tries to automate each of the four traditional steps. The approach describes a combination of various efforts in the field of automatic rule checking. In general, five components can be identified:

- *Step 1: distilling the rule content from the written text:* In order to derive the actual rule content from the rule document, the open standard projects of LegalDocML and LegalRuleML have been developed in the course of the Akoma Ntoso (Architecture for Knowledge-Oriented Management of African Normative Texts using Open Standards and Ontologies) project [25]. LegalDocML aims to cover the syntactical and structural part of the document, while LegalRuleML aims to represent the semantical and logical part of the rule. LegalRuleML is based on RuleML [26], which is an open standard mark-up language developed to enable the exchange of rule content. This structured rule content can be mapped into other structured rule languages, which can then be processed by the respective rule engines. Another method to systematically capture a rule's content in the AEC industry is the RASE (Requirement, Applicability, Selection and Exception) [27] mark-up. RASE classifies information in the rule document into the four classes, which make up the abbreviation. Similar to the LegalRuleML approach, the structured information can be mapped into other existing rule languages for further processing.
- *Step 2: transcribing the rule content into a rule language:* After the rule content has been identified, it needs to be transcribed into a language that then can be used by a rule engine. Some rule engines, such as Drools [28], require a custom rule language, while languages, such as Prolog [29] can be interpreted by several inference engines. Current trends in this domain consider Semantic Web technologies, such as SWRL (Semantic Web Rule Language) [30], which enables reasoning about ontology-based knowledge representations. Further information about rule languages will be given in a later section.
- *Step 3: transforming the existing model data into the required knowledge representation:* In general, the building model data is delivered using the IFC schema. This can be seen as a knowledge representation. However, IFC is not meant to be a schema that enables effective computerized reasoning. Therefore, the IFC data needs to be transformed into a knowledge representation fit for a rule engine. For example, when using the SWRL, the knowledge has to be

represented in an ontology that uses the Ontology Web Language (OWL). For this scenario, ifcOWL has been developed by Beetz et al. [31]. Still, the approach using ifcOWL is in early stages of research. A difficulty in this context is that the IFC schema does not exhaustively describe a building's elements and relationships. As a result, there are no detailed requirements, such as relationships in the knowledge representation. This lead to the idea of semantic enrichment, as proposed by Belsky and Sacks [32], where missing facts could be added to the model subsequently.

- *Step 4: executing the computable rules within a rule engine:* When both the rules and the facts are available in the correct form, a rule engine can reason about this combined knowledge. These rule engines form the basis of BRMS, such as Drools.
- *Step 5: generating new knowledge from reasoning about existing knowledge via the expressed rules:* After the execution of the rule checking process the results have to be analyzed. The results (pass/fail) can be seen as a form of new knowledge. This knowledge needs to be reported to the end user in visual and/or textual form, as described by Eastman et al. [16]. For this, a custom knowledge representation needs to be designed. Besides the simple pass/fail results of the rules, additional semantic knowledge can be generated by the rule engine via inferencing. For example, the rules can be designed in a way that a selection of elements (e.g. a crane) is chosen as being most fit for a given situation. The process of inferencing is one of the big advantages of rule engines over hard-coded approaches.

As for the rule interpretation and implementation, Solihin and Eastman [21] propose a classification of rule complexity. They identified four major levels of complexity within rules and classified them accordingly:

- Class 1: entities and attributes are queried and checked against a single value
- Class 2: additional values are calculated (e.g. distance) and checked
- Class 3: additional geometry is created, in order to calculate spatial relationships
- Class 4: problem solutions are calculated, and new model data/objects are created

In this paper, we also test the complexity of the developed rule-checking system in a case study. The complexity reaches up to class three. Class four, being the most complex rule class, has not yet been comprehensively implemented. An early approach of this rule class has been implemented by Zhang et al. [17] and Wang et al. [33] for the calculation of fall hazards in buildings and the automatic creation of necessary fall protection systems at the respective locations. Although the approach is well-suited in the mentioned use case, the underlying concept cannot be adopted as a general concept for any other rule scenario of class four.

## 2.2. Business rule management systems

Apart from the transcription of expert knowledge into a designated rule language, there should be a system that is capable of interpreting the transcribed rules. This so-called *inference machine* is the core of the software genre of *Business Rule Management Systems* (BRMS). BRMS belong to the field of knowledge-based methods/systems and thus, to the field of artificial intelligence (AI) [34]. AI knowledge consists of two main types: (1) logic and (2) data. In BRMS the logical knowledge about business processes is described by general business rules, while the case-specific data is referred to as facts [35]. The general framework of BRMS can be seen in Fig. 1. Although the rules can be changed autonomously, they are generally fixed. Facts, on the other hand, depend on the given situation, change frequently and can even be manipulated by the rules. In the example of CSLP, the business rules are

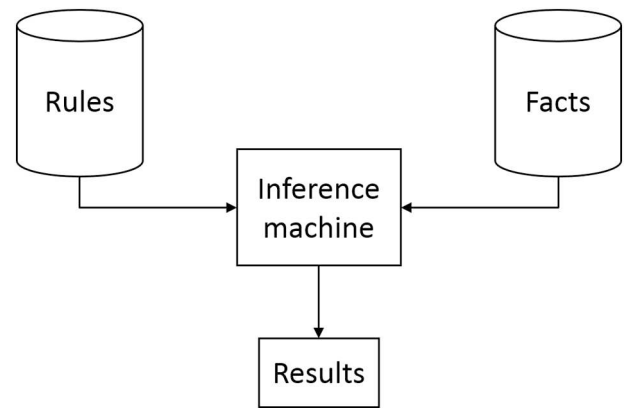


Fig. 1. Framework of business rule management systems. [Modified after 36].

predominantly represented by the existing dimensioning and safety-related rules for the site equipment. The facts are represented by the site layout model for a given project. The advantages of using BRMS instead of hard-coding a rule checking software are the following [35]:

- Separating logic from data allows manipulating both independently,
- Rules can be declared by users,
- Rule languages are better to understand than plain programming languages,
- Users are often domain experts, but not necessarily programmers,
- Inference machines use efficient solving algorithms, and
- So-called *expert systems* can propose solutions to a problem.

In general, most of the rule languages used in different BRMS follow the same principle. They have a conditional part and a consequence that applies, if all conditions are fulfilled. For this research we used the open source BRMS project Drools [28]. The syntax of the rule language used in Drools is depicted in Fig. 2. The general structure is “When A, Then B”. This structure is different from the if-clause known from regular programming languages, which is “If A, Then B, Else C”. The missing *Else* may lead to inconvenient rule declarations, but is an inevitable circumstance dictated by the inference machine. In the when-part of the Drools rule syntax the MVFLEX Expression Language (MVEL [37]) is used. It can be compared with a kind of query language that queries the wanted fact from the fact container (*knowledge session* in Drools). The facts queried in the when-part can be stored in internal variables, which can be used in the then-part. Plain Java code is used in the then-part of the rule syntax to implement a consequence.

## 2.3. Query languages

Query languages are a well-established technology in the field of database systems. They are used to efficiently query the knowledge stored in a database. Database query languages such as SQL [39] are characterized by an easy-to-understand syntax, which enables users with moderate programming skills to formulate queries and receive data from a database. However, in the context of BIM, query languages are still a matter of research. The main reason is that the data structures in databases are often kept simple. Its inherent knowledge can be exhaustively described. This leaves the knowledge querying to be a mere data reading process. In BIM-models on the other hand, a significant amount of knowledge is not stored in the data structure itself. Therefore, the semantic knowledge would have to be derived or calculated, respectively. For example, the adjacency relationship ‘an object stands on the ground’ is not stored in the data structure of an IFC-model but could be derived with the help of geometric algorithms.

Recent research deals with the querying of these semantic

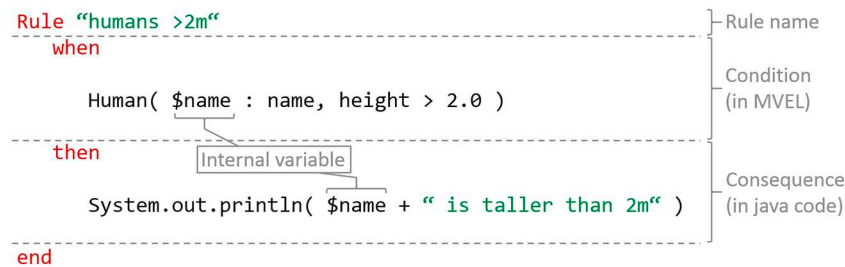


Fig. 2. Rule syntax in Drools.  
[Modified after 38].

relationships. The combination of the semantic enrichment of a data model with rule-based approaches has been investigated by Belsky and Sacks [32]. The idea behind their research is, that semantic relationships between objects follow specific patterns that can be described by rules. This way, the IFC model could be semantically enriched after the import, so that the information does not have to be stored in the IFC model directly. However, this topic is still at the early stages of research.

Another approach to combine query languages and BIM is the BIM Query Language (BIMQL) [40]. This query language could be combined with a rule engine, where it can be used in the conditional part for querying data from the model. Basic geometric concepts for topological relationships in building models have been studied by Borrmann and Rank [41]. Another concept proposed by Preidel and Borrmann [23] for easy-to-use access to data is the adoption of visual programming languages. This way, the data queries could be formulated and connected by visual components in a graph. In summary, a comprehensive data query language for BIM-models that is able to query complex semantic relationships does not exist to date.

#### 2.4. Rule languages

A rule language can be structured into two parts: condition and consequence. In the conditional part, the knowledge is queried for the existence of certain facts. If these facts are true, the defined consequence will be performed. Nevertheless, there are significant differences in existing rule languages. The most significant difference is the format in which the knowledge is represented. The buildingSMART Regulatory Room [42] examined a selection of major open standards for rule languages. It concludes that the concept of using open rule languages in the context of BIM rule checking is of significant relevance. Existing research is still focusing on finding a comprehensive process from rule interpretation to execution within building models.

While searching for an appropriate rule language, Dimyadi et al. [43], for example, evaluated numerous existing query and rule languages for their applicability in BIM. They conclude that existing rule and query languages are unsuitable. Instead they use the concept of these languages to develop a new *BIM Rule Language* (BIMRL) [44]. They demonstrate BIMRL by applying it to the field of compliant design audit processes and state that it is “powerful and efficient” [43] for a wide range of rules. However, the scope of BIMRL has not been validated exhaustively. Other approaches to domain specific languages, such as Building Environment Rule and Analysis (BERA), attempt “to deal with building information models in an intuitive way” [45]. Again, exhaustive testing and improving is still in progress.

The BRMS Drools uses the Drools Rule Language (DRL). As a knowledge representation, Drools uses an object-oriented approach. In the *knowledge session* the facts are represented by objects that can have any number of attributes. Similar to IFC, relationships between objects need to be represented by either a separate object, or by adding the reference of object as an attribute. This way, the transcription from IFC entities to Drools facts can be performed easily. In the past, a similar approach of using Drools for code compliance in the field of

sustainability has shown that Drools can successfully be utilized for the transcription of regulations in this domain [46]. However, they trivialize the need for additional functions that cannot be covered by the respective rule language. Especially semantic and geometric relationships not included in the original knowledge representation cannot simply be queried in the rule condition. This is why there is a need for a requirements analysis of functions, which are not covered by the rule language's logical operators, which is addressed in this paper.

Other rule languages use a different type of knowledge representation, namely an ontology. In this context, the Semantic Web Rule Language (SWRL) [30] is the most prominent. SWRL is able to reason about facts in an ontology that is formulated with the help of the Web Ontology Language (OWL) [47]. OWL enhances the original language for the formulation of ontologies, namely the Resource Description Framework (RDF) [48] with additional functionalities for the Semantic Web. In RDF, the facts are represented as *triples*. Each triple describes a fact as a sentence in the form of *subject* (‘A student’), *predicate* (‘writes’) and *object* (‘a paper’). This way, two elements and their respective relationship can be described. Another form is the SPARQL Protocol And RDF Query Language (SPARQL) [49] in combination with the SPARQL Inferencing Notation (SPIN) [50]. This combination of a query language and an inference framework allows for the reasoning about knowledge represented in RDF. These Semantic Web technologies have been tested in the AEC domain using the ifcOWL [31]. Beach et al. [51] transcribe certain building regulations concerning sustainability aspects with the help of SWRL and the building data in ifcOWL. They state, that their methodology “has successfully and accurately performed compliance checking on building data”. However, the promising approach needs to be further tested in future research, such as Zhang et al. [52] have done in automated safety planning for Job Hazard Analysis (JHA).

None of these approaches have evolved into practical application yet [53]. One reason eventually is that not every rule found in natural language can easily be transcribed into a desired rule language and be executed. In most cases, especially if complex geometric relationships are involved, additional functions need to be hard-coded and provided by the software. A simple example is the check for a safety distance of two elements. In the rule a function *calculateDistance* must be available. Otherwise, the mathematical algorithm for finding the shortest distance between two geometric objects would have to be implemented in the rule.

The inconvenience of having the need of partially hard-coded functions has been neglected in current research on open rule languages for BIM. This might be one of the reasons, why fully hard-coded rule checking systems, such as Solibri Model Checker, dominate commercial applications. In this paper, we illustrate a selection of additional functions for construction site layout planning, which are able to complete the neglected part.

### 3. Problem definition

There are several problems that need to be addressed in order to effectively use rule checking for construction site layout planning. A



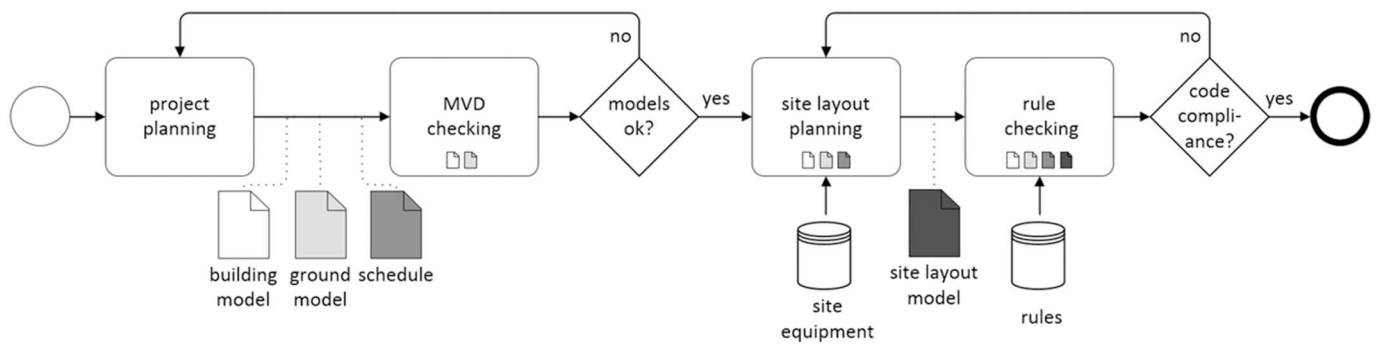


Fig. 3. General process of rule checking for CSLP.

few are explained in more detail.

- Modeling standards:** At the current stage, the BIM lifecycle offers optimal conditions for the creation of building models. On the other hand, the modeling of temporary site equipment or facilities is hardly performed due to the required manual effort. An additional reason is the absence of general modeling standards for these element types. For example, while columns, walls or beams can be found out of the box in standard modeling software, site equipment, such as cranes, or excavators require a custom design. Although there are some site equipment vendors (e.g., Liebherr cranes [54]) that offer very detailed, proprietary models, a specification of level of geometry and especially a level of information is missing. Therefore, an analysis of model requirements has to be performed. This process strongly correlates with the definition of algorithmic rule functionality.
- IFC entities:** There are hardly any construction site elements represented in the IFC-schema. This is essential for the classification of model elements. The absence of these entities complicates the systematical querying of site elements, so that a manual identification needs to be performed, for example, by using *IfcBuildingElementProxy* and property-based identifiers. This approach is not recommended due to its high potential of errors. However, as long as the entities do not exist, the definition of classification features for site equipment and facilities is inevitable.
- Rule language:** As described earlier there is no designated rule-checking language for BIM-related rules yet. However, extending existing languages to enable the processing of complex geometric relationships will be a vital step towards the development of such a designated rule language. Therefore, a requirements analysis needs to be performed. This way the necessary algorithmic functionality and additional querying and comparative operators can be defined (e.g., geometric intersection). Moreover, the rule language should be defined in a way, that engineers with low or moderate programming skills will be able to use it effectively.
- Information acquisition:** Formulating rules for the dimensioning of site equipment, such as cranes, is strongly dependent on construction schedule-constraints. These time-constraints can be derived from the project's schedule information, which is not automatically stored in a building model [55]. Schedule data is kept separately in a schedule file and can be connected to a 4D-model in specialized software. Nevertheless, both the model and schedule data have to be considered. Other resource information, e.g., the number of workers, has to be considered as well. Thus, it is necessary to define a consistent strategy for the acquisition of data in a multi model container [56].
- Conflict handling:** As mentioned earlier rule-based systems can either be used for simple conflict detection or for logical reasoning and automatic problem solving. The latter is not recommended for the beginning, because it is a highly complex task and a uniform strategy for conflict handling needs to be developed. Thus, the first

step is to formulate the rules in the conditional part of the rule, so that a consistent conflict handling by means of true/false status report can be assured in the rule consequence.

This list of issues with rule-checking is not seen as exhaustive. For example, new rule sets are likely to appear that end users will need to add. The following proposes and evaluates a novel methodology that allows the progressive adaptation of new rules.

#### 4. Proposed concept

We focus on the research question, whether BRMS and their respective rule languages can be used in the field of BIM-based CSLP. The chosen Drools rule engine is just one option. With the help of Drools the procedure is explained and a practical evaluation is made. Drools can be replaced by other BRMS. Furthermore, we focus on the methodology of defining rules with a given rule language. This means that we do not want to transcribe every rule that exists, but instead we provide a framework for experts in the domain of CSLP to formulate their own customized rules.

The general concept of rule checking in the domain of CSLP is depicted in Fig. 3. In the proposed process, we assume that a detailed BIM model suitable for construction exists. Additionally, a digital work schedule linked with the BIM model and the ground model must be available. The models need to meet the model requirements dictated by the site layout planner. These requirements have to be represented in a Model View Definition (MVD) [57]. Checking whether the requirements are met is a very important task.

However, based on the functionality of MVD, it only measures whether specific entities and/or attributes exist. Currently, it is not possible to check whether certain geometric modeling requirements are fulfilled. This has to be performed manually based on prescribed geometric restrictions.

Subsequently, the required entities and attributes can be transferred from the checked IFC model into the respective knowledge representation that is needed by the rule engine. As already mentioned, semantic and geometric relationships which are not represented in the IFC schema can be derived subsequently. For this, semantically rich query languages, such as BIMQL/BIMRL, can be used in a rule's condition, as described by Solihin et al. [24]. Another aspect described by Solihin et al. [24] that needs to be taken into consideration is that incoming data must be simplified and large data sets should be reduced.

The actual site layout planning is always performed by an expert. This person uses a rich object catalog that contains the company's owned, rented and/or leased site equipment for the creation of the site layout model. After the design of the site layout, the building and ground models as well as the linked work schedule are imported into the rule checking software. The rules are written in the specific rule language and saved as rule files. If the given site layout encounters any conflict during rule checking, the conflict is presented to the expert in a way that a redesign of the layout can be initiated. If no conflict appears,

the site layout complies with all rules.

For the purpose of plausibility, a lot of tasks and responsibilities rest with the site layout planner. This is a reason why rule checking will enhance the quality of work and make it more competitive. The rule checking software only has a supporting function. The tasks can be grouped into the following:

Site layout planners:

- Formulate rules with rule language,
- Develop site equipment catalog, and
- Provide model requirements.

Supporting rule checking software:

- Implement helper functions for rule language,
- Compile rules from external rule files, and
- Provide conflict report after rule checking.

First of all, the task of formulating the rules is performed by the site layout planner. This way, the expert can directly express and reproduce the required knowledge. Another advantage is that changes or additions of existing rules can instantly be updated whenever they occur. Furthermore, the development of the site equipment catalog is a responsibility of the planner to represent the company's inventory. The planner will provide model requirements based on the rules and the catalog. On the other side, the supporting rule checking software needs certain helper functions. These functions can then be used by the planner to formulate the rules. This feature is important, because the domain expert in CSLP does not necessarily have knowledge about geometric modeling and the corresponding complex mathematical problems that need to be solved. For example, the calculation of a geometric intersection between two known objects is a complex mathematical algorithm that cannot be coded within a simple rule statement.

## 5. Implementation

### 5.1. Model requirements

In general, model requirements consist of two types: geometry and attributes. When using the concept of Level of Development (LOD) [58] within a BIM project these types of model requirements are represented by Level of Geometry (LoG) and Level of Information (LoI). For the process of site layout planning certain model requirements need to be established.

#### 5.1.1. Level of information

As mentioned earlier, if a classification system and corresponding IFC-entities for the site elements are used, these elements can easily be queried within the rule. If not, the elements require a property that enables a unique classification subsequently. The property and its values have to be formulated as model requirements. Other important attributes have to be established as model requirements as well. A list of possible attributes can be found in Table 1. The table focusses only on tower cranes, storage and personnel containers, storage areas and excavation pits.

All model requirements are derived from existing rules for site layout planning [59]. Simultaneously, some of the model requirements depend on a specific algorithm that is used for checking the rule. For example, checking for safety distance can be performed via direct distance calculation of element's edges or via offset geometry. Both algorithms have different model requirements.

Apart from the site elements, model requirements also have to be established for the elements of the building. For example, if a crane's maximum load must not be exceeded, the element to be lifted must have a property *mass*. If this parameter does not exist, the mass can be

**Table 1**  
Model requirements: attributes.

Element	Required attribute
Tower crane	Type (top panning, trolley jib etc.)
	Tower height
	Block height
	Jib length
	Maximum torque (read from a lifting capacities table [38])
	Maximum load (read from a lifting capacities table [38])
Containers	Mass
	Type (material, tools, office, sanitary etc.)
	Connection type (water, electricity etc.)
	Stacking possible?
Storage areas	Mass
	Type (bricks, timber processing etc.)
	Type (sloped, vertical etc.)
	Soil quality (grown, non-cohesive etc.)
Excavation pits	Dewatering system

calculated by the product of the properties *volume* and *density* of the element's *material*. For the correct calculation of the elements' volume, the geometry of the element must be specified in detail. The LOD and LoG, respectively, has to be preset early and allow modification.

#### 5.1.2. Level of geometry

CSLP is often performed when contractors either bid a project or plan its execution. At both stages, the choice of design of the building elements' geometry is theoretically complete [5]. A high LOD can be assumed. In addition, the construction process, logically represented by the work schedule, is divided into smaller construction sequences. As the assembly sequences progress, the site layout can change, which consecutively leads to position changes of site elements or even complete equipment replacements. As proposed by Cheng and Kumar [12], site layout planning can be performed at any given point of time during construction. On the other hand, handling every site layout planning task individually may lead to negative results, because it could violate vital interdependencies between the construction sequences. In brief, the dynamic nature of the construction process must be considered.

The geometry of the site elements does not need to be as detailed as the elements of the building. Although the boundary dimensions must be precise, the inner body of a site element (e.g., truss members of a crane) can be approximate (e.g., cuboid) or even missing. For example, the footing of a tower crane model must have precise dimensions, because it is involved in a geometry-based rule. Too much of detail may add significant computation time to calculating geometric relationships. Similar decisions on geometric requirements can be dictated for containers and storage areas. Suggested is precise boundary dimensions, but an approximate inner body, unless retrieving information directly is important (e.g., mass). Otherwise it requires additional computing (e.g., as explained before using *volume* and *density*).

As for excavation pits, the geometric requirements are more challenging and indefinite. The reason for this is that the excavation pit itself mainly consists of empty space. However, the boundaries of this void are determined by other elements, for example slopes. These elements form the border between the pit and the surrounding terrain. Therefore, we propose three types of elements for the definition of an excavation pit: (a) ground model, (b) void elements, and (c) bordering elements (see Fig. 4). The ground model represents the surrounding terrain and includes information about soil quality. The void elements cut the ground model vertically. Bordering elements are added to the cutting edges to represent for example slopes or pit walls such as Berlin-type pit lining.

This approach enables the simple calculation of the volume of the excavated soil by subtracting the total volume of the bordering elements from the total volume of the void elements. Furthermore, this approach enables the query of an objects relation to the pit void, by

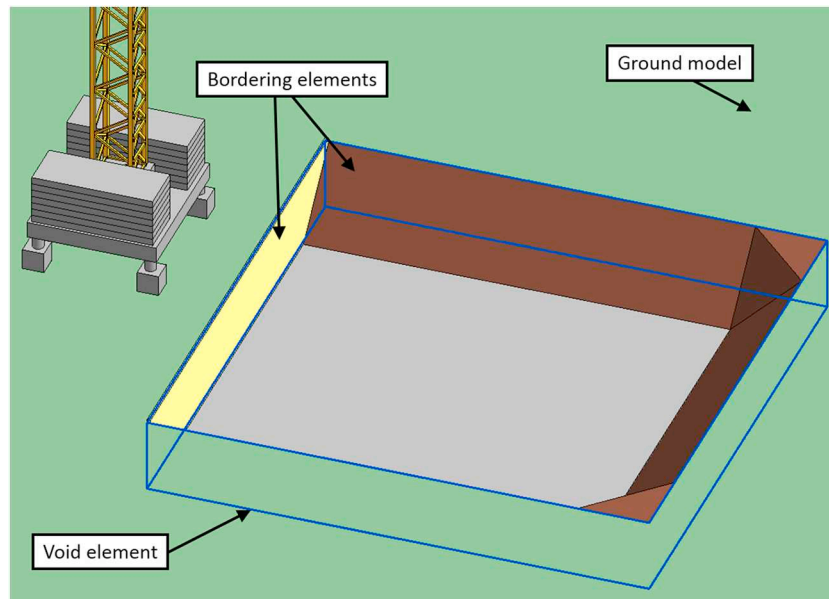


Fig. 4. Model elements of an excavation pit.

calculating the intersection between both objects. For example, if a crane intersects the void element, it is safe to say, that the crane stands inside of the pit. When the excavation pit is involved in a specific rule, the model elements have to be queried.

### 5.2. Algorithmic functionality

A requirement analysis was performed in order to grasp the general scope of the algorithmic functionality the rule checking application in the field of site layout planning needs to offer. We analyzed an academic handbook for CSLP [59]. It structures a variety of CSLP rules and best-practices. Furthermore, we asked for opinions of experts in the field of CSLP. Their feedback concerned their experience on applied processes and rules in the field. The following listing in Table 2 presents an example of the identified issues concerning resource related rules that are important to practitioners. The scope was limited thereafter on CSLP rules for tower cranes, containers, and storage areas. Note that in addition to geometry, functions may include numbers of workers or schedule information.

The result of the requirement analysis is that several geometric relationships between objects have to be calculated, because they cannot be derived directly from the model data (and neither are included in the IFC schema). Thus, the rule language needs to include the given geometric operations and a suited data structure in order to query complex geometric relationships efficiently. Table 3 shows the required geometric algorithms for the calculation of additional semantic relationships. Every example presents a known functionality, its rule, and a visualization.

### 5.3. Offset geometry

The result of a rule can be achieved by various approaches. One

example are rules concerning safety distances between objects. In the following, two different approaches for the checking of safety distances are presented. In both cases, the geometry representation of B-REP (boundary representation), as commonly used for 3D visualization, is assumed. An approach is to calculate the *shortest distance* between the objects' edges (see Fig. 5a) or faces, respectively. If this distance is smaller than a specific value dictated in the rule, the safety distance is violated. Another approach is the calculation and creation of a so-called *offset geometry*. It describes the geometry in which any single point has the exact same distance from the original geometry (see Fig. 5b). If this distance is set to the specific value dictated in the rule, the safety distance is represented by an individual object. An object that intersects the offset geometry consequently violates the demanded safety distance.

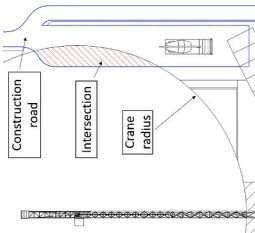
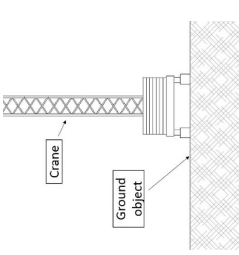
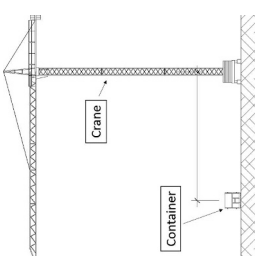
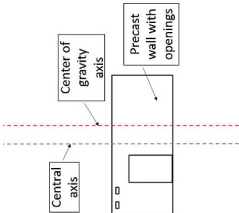
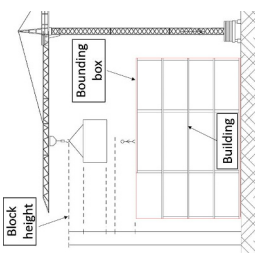
Both approaches have several advantages and disadvantages. In contrast to the offset geometry, the first approach directly calculates the distance between two objects. This value can be used for further analysis. The offset geometry merely validates whether the safety distance is violated. An advantage is that the offset geometry can be visualized in the viewer. This would enable the software to show the safety distance before an object is positioned. This way, a benefit is generated for the user who could intuitively see where a crane is allowed to stand.

A significant disadvantage is that the creation of an offset geometry is mathematically highly complex. Several publications about this topic try to interpolate between the vertex normal vectors of a triangular mesh (simplest form of B-REP geometry) [61]. An interpolated vector is used for the translation of the original vertex. This approach leads to an approximation of the offset geometry. The precision of the approximation has been increased in further research [62] but is still not the exact offset geometry. A different approach which calculates a highly accurate offset geometry uses multiple surface points [63,64]. First multiple points are created on the surface of the original geometry.

**Table 2**  
Resource related functions.

Example rule	Functionality
Best-practice for in-situ concreting work: 1 crane manages to serve 15 workers	Get from resources: number of workers
Best-practice for cranes: 1 crane manages to produce 1500 m <sup>3</sup> gross volume of a building per month	Calculate gross volume per month, e.g., by summing up space volumes (from schedule and model)
Best-practice for masonry work: store bricks for 1 week; 2 m <sup>3</sup> bricks per 1 m <sup>2</sup> storage area	Calculate masonry volume per week (from schedule and model)

**Table 3**  
Geometric functions.

Example	1	2	3	4	5
Rule	Temporary construction road must partially be within range of the tower crane.	Soil quality beneath a crane must comply with specific criteria [60].	Certain building components must be lifted by a tower crane (dependent on the distance to the crane's rotational axis).	A building element (e.g., precast reinforced concrete wall segment) must be lifted by a tower crane (dependent on the center of gravity).	Crane block/hook height > length of rope slings + load height + required safety distance (1.0 m) + minimum working space (2.5 m) + building height Calculate bounding box of multiple elements.
Function	Create an object: crane radius (cylinder) and calculate geometric Boolean operators (intersection, difference, union).	Calculate adjacency relationships (i.e., lateral, underneath, above) of objects	Calculate distance between objects' axes.	Calculate center of gravity of building element.	
Visual					

These points are translated along the normal vector of the original geometry's triangle by the given offset factor. To get closed round edges, additional points are added in a circle (sphere in 3D) around the original geometry's vertices. Afterwards, the boundary points of the entire point set are calculated which eliminates points inside the boundary. Meshing the resulting boundary point set provides the offset geometry.

For implementation purposes we simplified the described approach for convex geometry. This way the calculation of the boundary points can be achieved by generating the convex hull of the entire point set [65]. Furthermore, the creation of additional points on the edges/faces is unnecessary. Fig. 6a and Fig. 6b show that the simplified approach works for convex geometries, in contrast to concave geometries seen in Fig. 6c.

When using the simplified approach, a model requirement is to only use convex solids for the modeling of the elements affected by the given rule. For the implementation we focused on safety distances between cranes and excavation pits. The found related rule is divided into sub-categories depending on the given situation. Table 4 shows the corresponding pit type in the first column and the required safety distance in the second column.

## 6. Application and results

We show the prototypical application of the developed BRMS in a BIM environment. It utilizes the offset geometry approach for geometry-related rules for CSLP. We chose to combine the BRMS Drolls [28] with the open source project Open IFC Tools [66]. Both frameworks are Java-based, which enables the combination into a single Java-based application. Open IFC Tools is an IFC-Viewer [7], which can be extended with additional functionality by using its API for custom purposes. A custom communication between the IFC-model and the rule management system can be established in order to query model information in the rule condition and manipulate model data in the rule consequence, respectively. In addition, Open IFC Tools uses the class library Java3D [67] for the creation and visualization of the model geometry. With adequate knowledge in Java3D, its rich set of geometry operations can be used for the analysis and manipulation of the model geometry.

For implementation purposes we focused on three rule scenarios:

- 1) Is a tower crane placed too close to an excavation pit?
- 2) Can the tower cranes reach and lift all building elements?
- 3) Does the material transport from a storage area to the building footprint collide with paths or presence of workers (e.g. in containers/or on pedestrian walkways)?

The complexity of these rules corresponds to class two and three from the rule classification proposed by Solihin and Eastman [21].

The rule to the first scenario is explained in Table 4. We focus on a real case that includes a slope of grown soil. In this case the safety distance of objects near an excavated pit must be located at least 2.0 m away from the pit boundary. In order to investigate the offset geometry approach, we used the offset-algorithm explained earlier. For this, certain model requirements must be met. Firstly, the excavation pit must be modeled as seen in Fig. 4. Secondly, each bordering element (e.g., slope) must be modeled strictly convexly (see Fig. 7a and Fig. 7b). The reason is that the implemented algorithm for the creation of the offset geometry is simplified for convex geometry only. As for determining the attribute requirements, the slope elements need to have the property *soil quality*. Additionally, the property *type* with value 'slope' must be included for classification purposes. The tower crane has been modeled using simple geometry, as seen in Fig. 7c.

For the creation of the model we used the software Autodesk Revit 2017 [68]. For the implementation of the simplified approach (the calculation of the offset geometry), we used an open source library with



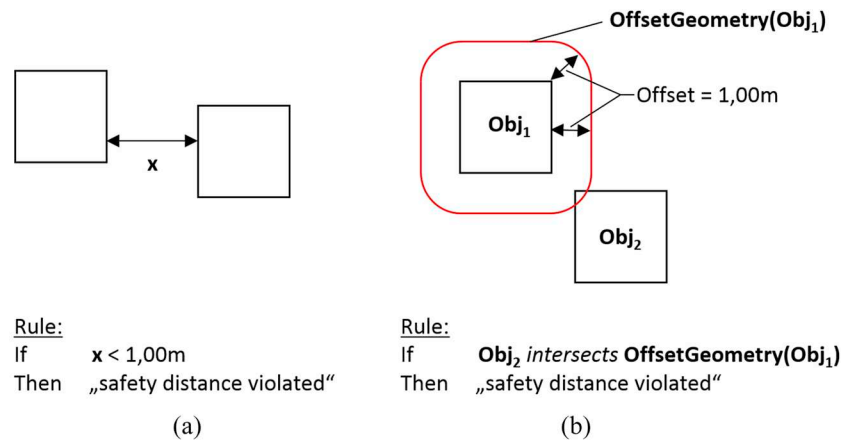


Fig. 5. Safety distance between objects in 2D: (a) distance between edges, (b) intersection of offset geometry.

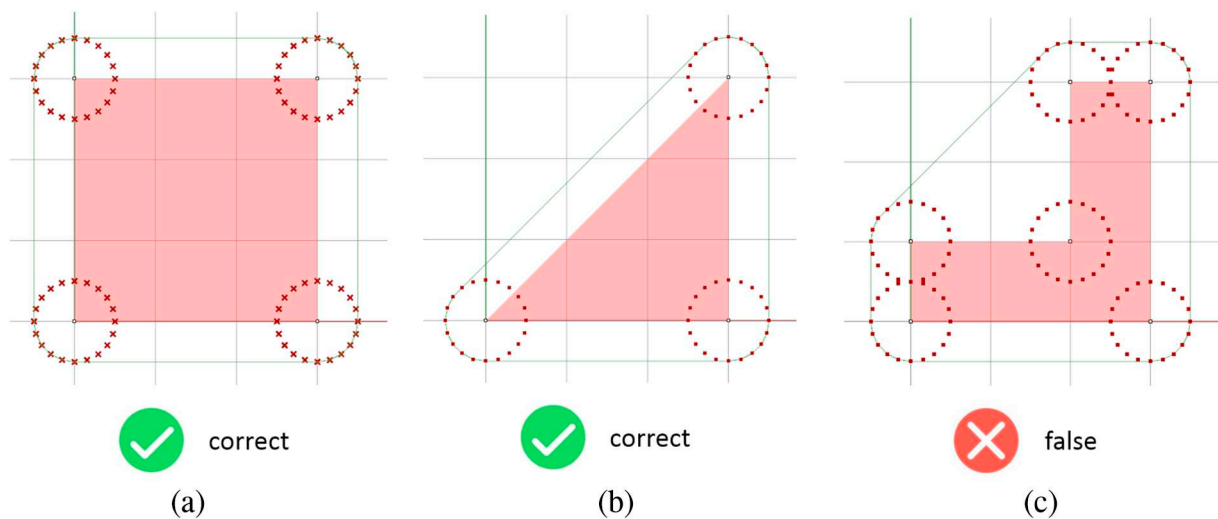


Fig. 6. Offset geometry via convex hull: a first (a) and second (b) example of a convex body and (c) a concave body.

**Table 4**  
Rules for the safety distance between excavation pits and tower cranes ( $x$  = required minimum distance).

Elements	Criteria
Sloped pits	$x > 0.5\text{ m}$ from lower slope edge for a crane within the excavated pit $x > 2 \cdot h$ (height of pit) $> 2.0\text{ m}$ from upper slope edge for a crane outside of the excavated pit with non-cohesive soils $x > h$ (height of pit) $> 2.0\text{ m}$ from upper slope edge for a crane outside of excavated pit with grown soils
Vertical pits with excavation support system	$x > 0.6\text{ m}$ for crane weight $< 12\text{ tons}$ $x > 1.0\text{ m}$ for crane weight $> 12\text{ tons}$

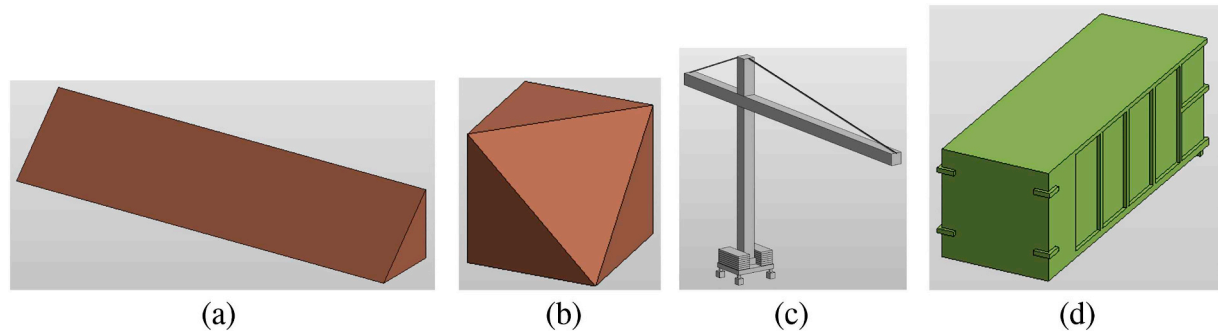


Fig. 7. Element geometry: (a) slope along an excavated pit's straight edge, (b) slope corner, (c) tower crane, (d) container.

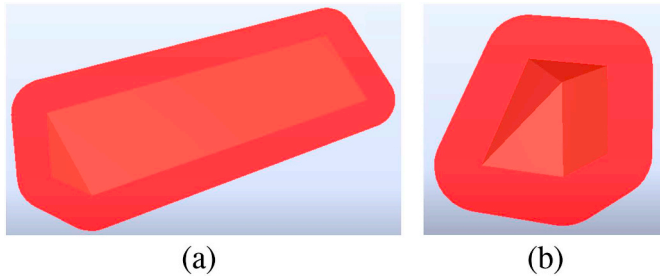


Fig. 8. Offset geometry: (a) slope along an excavated pit's straight edge, (b) slope corner.

a convex hull algorithm [69]. The results of the offset calculation of Fig. 7a and Fig. 7b is illustrated in Fig. 8a and Fig. 8b. The Intersection is calculated by the Boolean Modeller [70] of the Open IFC Tools API.

For the third scenario, we added two elements, a storage area being modeled as a simple cuboid and a container based on a standard family of Autodesk Revit 2017 (see Fig. 7d). The complexity of this rule lies in defining the activities involved in a hoist (e.g., material moved from a temporary laydown yard to an elevated platform). In order to check if the material transport intersects with an object during the hoist, it needs to be represented by a geometry. In our example, precast columns are unloaded and stored at a specific storage area. They are lifted by a tower crane to the certain position in the building. This geometry can be assumed in various ways [71]. We propose the concept of modeling the transport geometry as the convex hull of the storage area itself and the list of building elements placed in storage area within the range of the tower crane. Consequently, every possible linear movement of the precast columns will take place within the convex hull.

In summary, the following functions were implemented for the example:

```
rule "Classification: tower crane"
  when
    $dummy : DummyCadObject( $n : name contains "crane", $m :
      findProperty("Mass"), $j : findProperty("Jib length") )
  then
    TowerCrane $crane = new TowerCrane( $n, StringToDouble($m),
      StringToDouble($j), $dummy );
    insert( $crane );
    retract( $dummy );
  end
```

- *FindProperty* – returns the value of a given property of an IFC-object,
- *StringToDouble* – returns the floating-point number of a given string,
- *OffsetGeometry* – calculates the offset geometry of a given object,
- *VisualizeOffset* – shows the offset geometry in the viewer,
- *MaterialTransportGeometry* – calculates a geometry approximating the material transport from a storage area to building using a crane, and
- *Intersects* – calculates the intersection of two objects.

At this point all necessary requirements to describe the rules in Drools are provided. However, in order to query the model elements, a classification must be established. The classification can directly be performed in the rule file for Drools. This classification process involves two steps: (1) type declaration and (2) mapping using certain

properties. This way, the classified elements can be reused without querying the model for the properties for every rule. For our example we specified four objects: (1) *TowerCrane*, (2) *Slope*, (3) *PitWall* and (4) *VoidElement*.

The Codeblock 1 shows the syntax for the type declaration of the tower crane. The other objects are declared identically. The properties for the classification are either *name* or *type*. Additional properties are declared afterwards. Lastly, the assigned CAD object from the IFC model (in this example named 'DummyCadObject') is added to the object, so that further calculations based on the object can be performed. As described earlier, Drools uses a container of facts that is checked against the rules. In this container all CAD objects of the IFC model are included as facts.

#### Codeblock 1: Drools rule file: type declaration

```
declare TowerCrane
  name : String
  mass : double
  jibLength : double
  dummy: DummyCadObject
end
```

During the rule-based classification, the *DummyCadObjects* are queried and exchanged by the classified objects (see Codeblock 2). The rule searches for a *DummyCadObject* (*\$dummy*). For each received object it checks whether the required properties are included. Then, a new object of the desired type (*\$crane*) is created and filled with the corresponding properties. Afterwards, the new object is inserted into the fact container and the *DummyCadObject* is retracted from the facts. The classification rule for the other objects is formulated identically.

#### Codeblock 2: Drools rule file: classification mapping as a rule

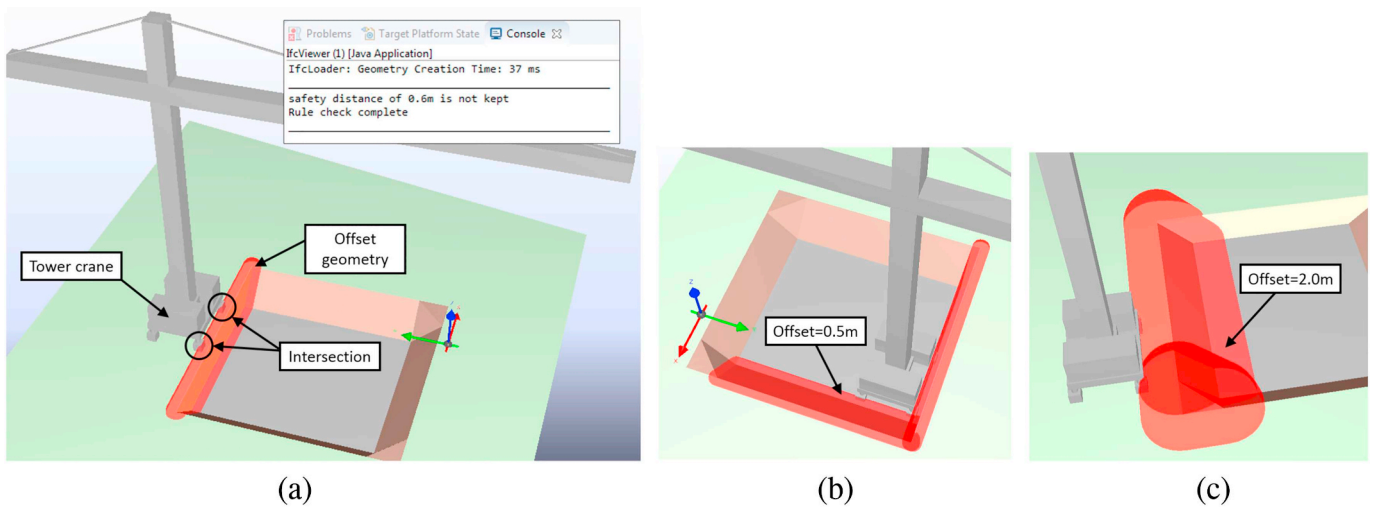
The rule for the first scenario is shown in Codeblock 3. In the first part, the rule queries the relevant objects from the fact container. For the three elements *TowerCrane*, *PitWall* and *VoidElement* the corresponding *DummyCadObjects* are stored in three temporary variables (*\$dt*, *\$dp*, *\$dv*). The first *eval* command evaluates whether the crane is outside of the excavation pit looking for an intersection between the crane and the pit's void element. If there is no intersection, the second *eval* command evaluates whether the offset geometry of the pit element intersects the crane. The offset geometry is created by the *OffsetGeom* function with an offset value of 0.6 m. When all conditional clauses are true, which means that the crane is too close to the excavation pit, the offset geometry is visualized. Furthermore, detailed information (a text and the respective object) about the conflict is stored for textual and visual reporting of checking results.

**Codeblock 3:** Drools rule file: rule for safety distance

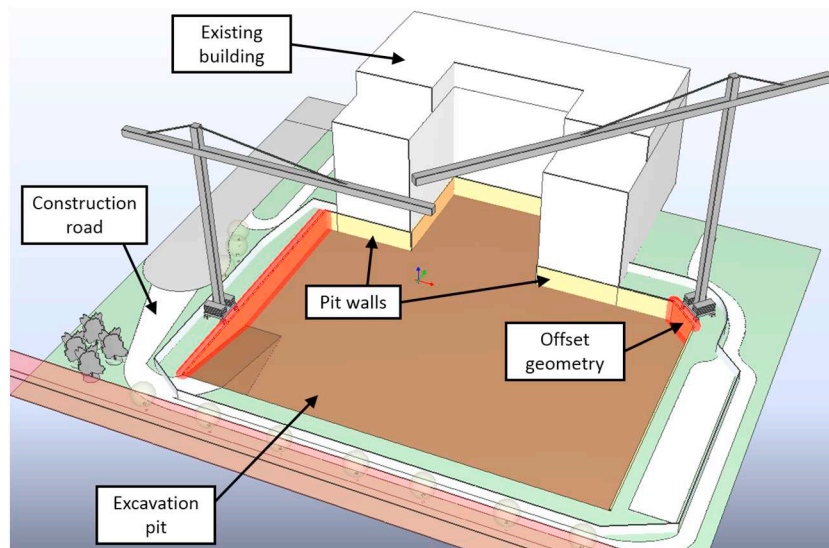
```

rule "crane outside & pit wall"
when
    TowerCrane( $dt : dummy, mass < 12 )
    PitWall( $dp : dummy )
    VoidElement ( $dv : dummy )
    eval( Intersects( $dt, $dv ) == false )
    eval( Intersects( $dt, OffsetGeom( $dp, 0.6 ) ) )
then
    $dp.getOffsetGeometry().visualizeOffset();
    insert(new ConflictInformation("safety distance of 1.0m is not kept",
    $dp));
end

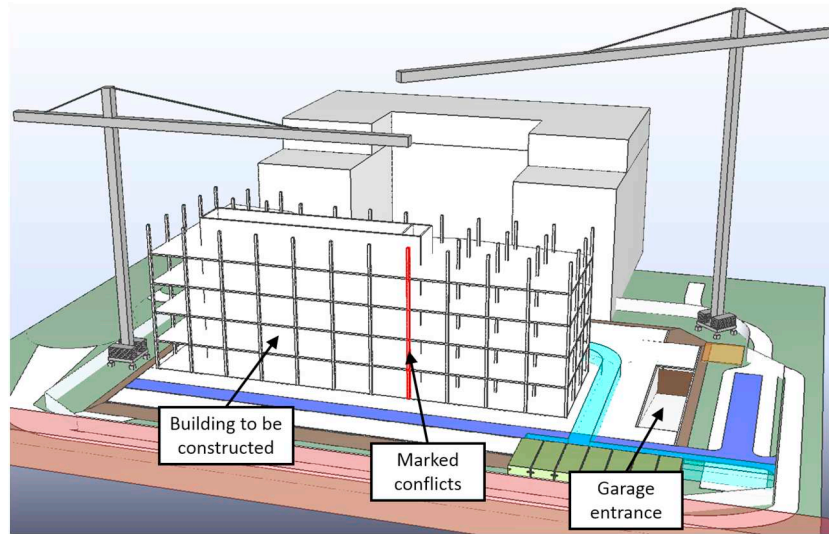
```



**Fig. 9.** Different rule conflicts (indicated by red color in the digital version of the paper): (a) result of the first rule check, (b) result to a manually repositioned crane inside of the pit - in addition to the first rule violation there is a second rule violation (too close to the bottom of the of the sloped excavation edge), and (c) crane above ground too close to the top edge of sloped excavation pit. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**Fig. 10.** Rule checking of a tower crane's proximity against excavated pit walls (offset indicated by red color in the digital version of the paper). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**Fig. 11.** Precast columns and cranes (conflicting columns indicated by red color in the digital version of the paper). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The result of applying this rule is shown in Fig. 9a. If there is a conflict, the site layout should be changed manually until all conflicts are resolved. Other examples and possible rule conflicts are shown in Fig. 9b and Fig. 9c. Also the impact of a dynamic rule-based change of a bigger offset value is addressed.

After testing the rules using a simplified scenario, the next step was to evaluate the rules in field-realistic construction scenarios. The rules are applied to two construction phases of a complex building project. The two phases are depicted in Figs. 10 and 11. The building is a multi-story office building above an underground parking garage. The ex-

which returns the distance in the XY-plane of the locations of two given objects. The function *calculateDistance* queries the placement location of the objects. This requires the crane object to have a rotating axis and the final placement location. In many cases, for example for walls, the placement location does not match the load application point, which is usually the center of gravity. However, we have not implemented this functionality yet. The rule transcribed into the Drools syntax is shown in Codeblock 4.

**Codeblock 4:** Drools rule file: rule for columns to be lifted

---

```

rule "crane can lift column"
when
    Column( $dc : dummy, $v : volume, $d : density )
    $craneList : ArrayList() from collect ( TowerCrane( $dt : dummy, ($v
        * $d) < ReadFromTable("LoadTable_crane", jibLength,
            CalculateDistance($dt, $dc)) ) )
    eval( $craneList.isEmpty() )
then
    markObject($dc);
    insert(new ConflictInformation(("column is too heavy: " + ($v * $d)),
        $dc));
end

```

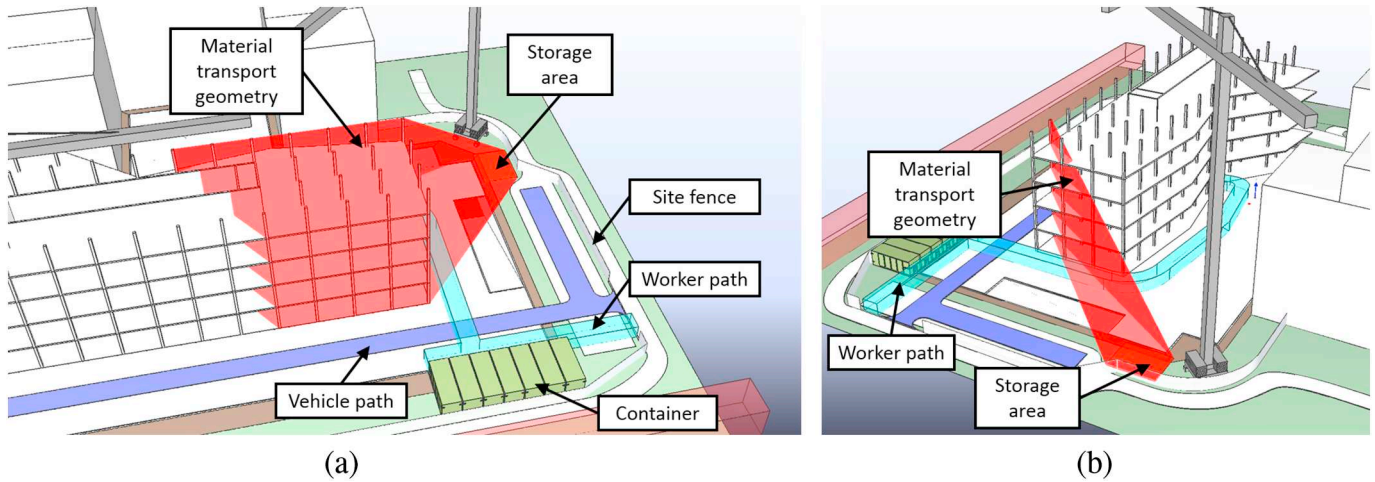
---

cavation pit is secured with pit walls. The shell structure consists of precast reinforced concrete columns and walls as well as in-situ concrete slabs. The pit to crane distance rule can be used as described above. The result of the rule-checking is shown in Fig. 10. The offset geometries of the pit wall elements are visualized, indicating that the cranes are too close to the pit.

Another rule that is very relevant for the next construction phase was implemented. The rule evaluates whether all precast columns can be lifted by at least one crane. We chose this rule, because it highlights more complex relationships between the geometry of elements, its attributes, and the involved lifting resources. The maximum load torque of a crane can be found in the load capacity table of the corresponding manufacturer [54]. The manufacture defines the maximum load at a specific distance from the cranes rotation axis. Lifting capacities were added to the cranes by linking the table represented by a CSV-file (character-separated values). The function *readFromTable* was implemented to return the maximum load for a given distance. For the calculation of the distance the function *calculateDistance* was integrated

The first rule (Codeblock 3) queries every combination of pit walls and cranes and evaluates the safety distance. Whereas, the second rule needs to consider all cranes at the same time. Therefore, the *from collect* command is used. This command creates a list of all elements that match the given criteria. In this case all cranes are needed, which can lift the given column. If the list is empty, no crane can lift the column, which means, the element is too heavy. The mass of each column is calculated by the product of the attributes volume and density ( $v$ ,  $d$ ) and the maximum load is extracted from the lifting capacities table. Afterwards, all columns that cannot be lifted by at least one crane are marked and visualized in the model (see Fig. 11). In the given example the volume of the conflicting precast columns is  $0.36 \text{ m}^3$  and the density is  $2500 \text{ kg/m}^3$ , which calculates to a mass of  $900 \text{ kg}$ . The crane's jib length is  $52 \text{ m}$  and its distance towards the conflicting columns is  $49.5 \text{ m}$ . The allowed lifting capacity at this distance is  $800 \text{ kg}$  (read from the crane's lifting capacities table). This is less than the column's mass of  $900 \text{ kg}$ , which leads to the detection of a conflict and thus to the result visualized in Fig. 11. Additionally, a textual rule report is





**Fig. 12.** Material transport geometry (indicated by red color in the digital version of the paper); (a) transport geometry of all reachable columns and (b) transport geometry of a single column (top level) intersecting with a priori-designed pedestrian worker path. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

generated. The task of the construction manager is now to replace the crane or apply other cost-effective alternative approaches, e.g., renting a mobile crane if such elements appear late in the construction schedule when all tower cranes are already demobilized, or when the project contains only few of such building elements.

The result of the third rule, which evaluates whether a worker path intersects the material transport geometry, can be seen in Fig. 12a. The corresponding Drools syntax is shown in Codeblock 5. The rule first

workers is shown in Fig. 12b. If all these statements are true, the material transport geometry is visualized in red and the conflict information is created. Similar rule checks can be executed for more complex scenarios, for example, conflicts of material movement or working crews with vehicle sweep lanes.

**Codeblock 5:** Drools rule file: rule for material transport above worker paths

```
rule "no material Lifting above containers"
when
    StorageArea( $dsa : dummy )
    Walkway ( $dwalk : dummy )
    TowerCrane( $dt : dummy, $j : jibLength > CalculateDistance($dt,
        $dsa )
    $columnList : ArrayList() from collect ( $c : Column( $dc : dummy, $j
        > CalculateDistance($dt, $dc) ) )
    eval( Intersects($dwalk, CreateMaterialTransportGeometry($columnList,
        $dsa ) ) )
then
    $dsa.getMaterialTransportGeometry().visualizeGeometry();
    insert(new ConflictInformation("container is within material
        transportation space", $dcon));
end
```

queries for storage areas, walkways and cranes. If a tower crane can reach a certain storage area, a list of precast columns is created, which are located in the building and can also be reached by the corresponding tower crane. Next, the function *CreateMaterialTransportGeometry* calculates the convex hull of the corner points of the queried columns and the storage area. Additionally, the points are duplicated, and their z-coordinate is set to the lowest level. In result, the material transport geometry covers the entire potential hazardous space, from the actual element location to the ground. This is especially important, when querying a single column. In Fig. 12b, a single column from the top level is queried and the transport geometry covers a space from top level to the lowest level. If the additional points were not added, the transport geometry would not intersect the desired object on the ground, because it would be located above that object. Finally, it is checked whether an intersection exists between the created material transport geometry and the queried object. This object can be any desired CadObject. In this example, a hypothetical walkway of

## 7. Discussion

The case study presented in this paper only covers a few necessary rules for CSLP. Due to the implications for construction site engineering and management, we consider a human-assisted rule-based checking approach tackling simple modes of safety code. The presented scope of research further did not intend to implement all existing rules, rather the focus was on evaluating whether a rule engine and its respective rule language is able to match the requirements for the application of rule based CSLP. If successful, the respective experts (e.g., planners, engineers, safety specialists) would be able to formulate their own rules. The used rule management system Drools is one possibility. Other rule languages have not been tested. However, extending the rule set and comparing the performance to other rule languages can be matters of future research.

The listed problems in CSLP cover a vast range of research topics. We could provide evidence that the use of open rule languages in

combination with the definition of appropriate model requirements have advantages over existing hard coded rule checking software. Additionally, the rule engine can be used to validate whether the model requirements are fulfilled (MVD checking).

Furthermore, we introduced the concept of offset geometry to check safety distance violations. Although it has certain advantages with respect to potential conflict visualization, its mathematically complex algorithm for concave geometry and the long computing time for geometric intersections still need to be optimized (currently the calculation took 5 s for the shown case study using a standard desktop computer). Handling large and complex models in the real-world use may cause additional performance issues that we did not address in this work, but could be resolved using spatially indexed model data servers.

A significant inconvenience during the formulation of the rules in Drools is the missing debugging functionality. The rules are written in external rule files and are not immediate part of the rule checking application. Consequently, the compilation of the rules is performed during runtime and errors in the rule code are hard to find. Helper tools that assist the user to formulate rules are recommended.

The proposed method, as any automated process, relies on following model requirements strictly. In the real-world use this may burden the modeler with limited freedom for model creation, but allows for an automated standardized checking routine. For example, if the excavation pit is not modeled with the three required model elements (i.e. ground model, void and bordering elements), the respective rule might not be able to check the situation correctly. A modeler could still choose a different modeling technique, but subsequently has to alter the respective rule set.

## 8. Conclusion and outlook

Although the BIM method is steadily progressing in the AEC/FM domain, the process of construction site layout planning (CSLP) is hardly considered [3]. This paper tries to address some of the obstacles on the way to an integrated BIM-based CSLP. The application of rule-based algorithms is promising for model-based checking processes, because it captures the logic behind the checking routines. More research is needed until the BIM method can address all the complex relationships between planned construction and logistics as well as as-built construction progress monitoring and reactive process planning [55]. Apart from the application of rule-based CSLP, the proposed method can be applied to other scenarios, such as occupational safety and health [72]. With the proposed method, the process of safety planning follows specific and strict rules. For example, the safety rules can be formulated with the help of open rule languages and the respective model requirements can be determined.

An advantage of using open rule languages is that the knowledge stays with the human experts, because they will formulate the rules. Additionally, the equipment catalog and customized model requirements will be created by the experts themselves. Furthermore, we demonstrate the importance of additional functions that need to be implemented using traditional hard-coding. Additional functions have been neglected by current approaches of automatic rule checking. Our work shows that with the help of BRMS, model-based rule checking is possible. Considering the rule classification of Solihin and Eastman [21], we showed, that even complex rules can be handled with our proposed method. An advantage of our proposed method is the use of offset geometry, which provides an intuitive way to visualize and check for keeping safety distances.

In future research BRMS can be further investigated with respect to backward chaining. This method enables the rule engine to derive problem solution proposals, for example the automatic dimensioning of a crane. These complex solution proposals can be seen as a class four rule [21]. We have shown, that the semantic relationships between model elements are not exhaustively described in the IFC data model. Methods of semantic enrichment with the help of rule-based

algorithms, as proposed by Belsky and Sacks [32], can be addressed in future work. Another concept to simplify the process of formulating rules is the use of visual programming languages. As proposed by Preidel and Borrmann [23], the intuitive handling of this type of programming can help the experts to describe the logic of a rule better and faster.

## Acknowledgement

This research is funded by the German Federal Ministry of Education and Research (BMBF) within the “Innovations for Tomorrow's Production, Services, and Work” Program as well as by the European Social Fund (ESF). The authors are responsible for the content of this publication.

## References

- [1] B. Akinci, M. Fischer, J. Kunz, R. Levitt, Representing work spaces generically in construction method models, *J. Constr. Eng. Manag.* 128 (2002), [https://doi.org/10.1061/\(ASCE\)0733-9364\(2002\)128:4\(296\)](https://doi.org/10.1061/(ASCE)0733-9364(2002)128:4(296)).
- [2] J. Teizer, J. Melzner, *Sicherheitstechnische Planung von Hoch- und Ingenieurbauprojekten mithilfe von Bauwerksinformationsmodellen (BIM)*, VDI-Bautechnik, Jahressausgabe 2015/16, Bauingenieur, Springer, Heidelberg, 2015, pp. 128–135 (ISSN 0005-6650).
- [3] K. Schwabe, M. König, J. Teizer, BIM applications of rule-based checking in construction site layout planning tasks, The 33rd International Symposium on Automation and Robotics in Construction and Mining, 2016 <https://doi.org/10.22260/ISARC2016/0026>.
- [4] X. Ning, K. Lam, M. Lam, A decision-making system for construction site layout planning, *Autom. Constr.* 20 (2011) 459–473, <https://doi.org/10.1016/j.autcon.2010.11.014>.
- [5] A.M. Khallafallah, Assessing the performance of the non-dominated sorting genetic algorithm in optimizing construction site planning, International Conference on Computing in Civil and Building Engineering, 2014 <https://doi.org/10.1061/9780784413616.156>.
- [6] E. Hjelseth, Classification of BIM-based model checking concepts, *J. Inf. Technol. Constr.* 21 (2016) 354–369 <http://www.itcon.org/2016/23>.
- [7] Industry Foundation Classes, IFC, <http://www.buildingsmart-tech.org/specifications/ifc-overview>, Accessed date: 8 October 2018.
- [8] N. Pradhananga, J. Teizer, Congestion analysis for construction site layout planning using real-time data and cell-based simulation model, International Conference on Computing in Civil and Building Engineering, 2014 <https://doi.org/10.1061/9780784413616.085>.
- [9] M. Yahya, M. Saka, Construction site layout planning using multi-objective artificial bee colony algorithm with Levy flights, *Autom. Constr.* 38 (2014) 14–29, <https://doi.org/10.1016/j.autcon.2013.11.001>.
- [10] M. Andayesh, F. Sadeghpour, The time dimension in site layout planning, *Autom. Constr.* 44 (2014) 129–139, <https://doi.org/10.1016/j.autcon.2014.03.021>.
- [11] H. Astour, V. Franz, BIM-and simulation-based site layout planning, International Conference on Computing in Civil and Building Engineering, 2014 <https://doi.org/10.1061/9780784413616.037>.
- [12] J. Cheng, S. Kumar, A BIM based construction site layout planning framework considering actual travel paths, The 31st International Symposium on Automation and Robotics in Construction and Mining, 2014 <https://doi.org/10.22260/ISARC2014/0060>.
- [13] J. Olearczyk, Z. Lei, B. Ofri, S. Han, M. Al-Hussein, Intelligent crane management algorithm for construction operation (iCrane), The 32nd International Symposium on Automation and Robotics in Construction and Mining, 2015 <https://doi.org/10.22260/ISARC2015/0083>.
- [14] S.H. Han, Z. Lei, A. Bouferguene, M. Al-Hussein, U.R. Hermann, 3D visualization-based motion planning of mobile crane operations in heavy industrial projects, *J. Civ. Eng.* 30 (2016), [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000467](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000467).
- [15] J.K.W. Yeoh, J.H. Wong, L. Peng, Integrating crane information models in BIM for checking the compliance of lifting plan requirements, The 33rd International Symposium on Automation and Robotics in Construction and Mining, 2016 <https://doi.org/10.22260/ISARC2016/0116>.
- [16] C. Eastman, J.-M. Lee, Y.-S. Jeong, J.-K. Lee, Automatic rule-based checking of building designs, *Autom. Constr.* 18 (2009) 1011–1033, <https://doi.org/10.1016/j.autcon.2009.07.002>.
- [17] S. Zhang, J. Teizer, J. Lee, C. Eastman, M. Venugopal, Building information modeling (BIM) and safety: automatic safety checking of construction models and schedules, *Autom. Constr.* 29 (2013) 183–195, <https://doi.org/10.1016/j.autcon.2012.05.006>.
- [18] D.M. Salama, N.M. El-Gohary, Semantic text classification for supporting automated compliance checking in construction, *J. Comput. Civ. Eng.* 30 (1) (2016) 04014106, [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000301](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000301).
- [19] E. Hjelseth, Foundations for BIM-Based Model Checking Systems: Transforming Regulations Into Computable Rules in BIM-Based Model Checking Systems, PhD Thesis [http://ibim.no/Eilif\\_Hjelseth\\_PhD\\_Foundations\\_for\\_BIM-based\\_model\\_checking\\_systems\\_2015-11-06.pdf](http://ibim.no/Eilif_Hjelseth_PhD_Foundations_for_BIM-based_model_checking_systems_2015-11-06.pdf), (2015), Accessed date: 8 October 2018.

- [20] W. Solihin, A Simplified BIM Data Representation Using a Relational Database Schema for An Efficient Rule Checking System and Its Associated Rule Checking Language, PhD Thesis <https://smartech.gatech.edu/handle/1853/54831>, (2016), Accessed date: 8 October 2018.
- [21] W. Solihin, C. Eastman, Classification of rules for automated BIM rule checking development, *Autom. Constr.* 53 (2015) 69–82, <https://doi.org/10.1016/j.autcon.2015.03.003>.
- [22] Solibri Model Checker, <http://www.solibri.com/products/solibri-model-checker/>, Accessed date: 8 October 2018.
- [23] C. Preidel, A. Borrmann, Towards code compliance checking on the basis of a visual programming language, *J. Inf. Technol. Constr.* 21 (2016) 402–421 <http://www.itcon.org/2016/25>.
- [24] W. Solihin, J. Dimyadi, Y.C. Lee, C. Eastman, R. Amor, The Critical Role of Accessible Data for BIM Based Automated Rule Checking System, LC3 2017: Volume I – Proceedings of the Joint Conference on Computing in Construction, <https://doi.org/10.24928/JC3-2017/0161>, (2017).
- [25] OASIS, Akoma Ntoso, [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=legaldocml](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legaldocml), Accessed date: 8 October 2018.
- [26] RuleML, Rule Markup Language, <http://ruleml.org/index.html>, Accessed date: 8 October 2018.
- [27] E. Hjelseth, N. Nisbet, Capturing normative constraints by use of the semantic mark-up (RASE) methodology, *CIBW78 2011 28th International Conference-Applications of IT in the AEC Industry*, 2011.
- [28] JBoss Project, Drools, <https://www.drools.org>, Accessed date: 8 October 2018.
- [29] D.L. Bowen, L. Byrd, F.C.N. Pereira, L.M. Pereira, D.H.D. Warren, DECsystem-10 Prolog User's Manual, <https://www.2.cs.duke.edu/csl/docs/cprolog.html>, (1988), Accessed date: 8 October 2018.
- [30] W3C, SWRL, <https://www.w3.org/Submission/SWRL/>, Accessed date: 8 October 2018.
- [31] buildingSMART, ifcOWL, <http://www.buildingsmart-tech.org/future/linked-data>, Accessed date: 8 October 2018.
- [32] M. Belsky, R. Sacks, Semantic enrichment for building information modeling, *Comput. Aided Civ. Inf. Eng.* 31 (2016) 261–274, <https://doi.org/10.1111/mice.12128>.
- [33] J. Wang, S. Zhang, J. Teizer, Geotechnical and safety protective equipment planning using range point cloud data and rule checking in building information modeling, *Automation in Construction*, vol. 49, Elsevier, 2014, pp. 250–261 <https://doi.org/10.1016/j.autcon.2014.09.002>.
- [34] P. Norvig, S. Russell, *Artificial Intelligence: A Modern Approach*, 3rd edition, Pearson Education, Inc., 978-0136042594, 2009.
- [35] J. Ary, Instant: Drools Starter, Packt Publishing Ltd., 978-1782165545, 2013.
- [36] JBoss, Drools Documentation, <https://www.drools.org/learn/documentation.html>, Accessed date: 8 October 2018.
- [37] MVFLEX Expression Language (MVEL), <http://mvel.documentnode.com>, Accessed date: 8 October 2018.
- [38] M. Bali, Drools JBoss Rules 5.X Developer's Guide, Packt Publishing Ltd., 978-1782161264, 2013.
- [39] SQL, ISO/IEC 9075-1:2016, <https://www.iso.org/standard/63555.html>, Accessed date: 8 October 2018.
- [40] W. Mazairac, J. Beetz, BIMQL – an open query language for building information models, *Adv. Eng. Inform.* 27 (2013) 444–456, <https://doi.org/10.1016/j.aei.2013.06.001>.
- [41] A. Borrmann, E. Rank, Topological analysis of 3D building models using a spatial query language, *Adv. Eng. Inform.* 23 (2009) 370–385, <https://doi.org/10.1016/j.aei.2009.06.001>.
- [42] buildingSMART, Regulatory Room, <https://www.buildingsmart.org/wp-content/uploads/2017/11/17-11-08-Open-Standards-for-Regulation.pdf>, Accessed date: 8 October 2018.
- [43] J. Dimyadi, W. Solihin, C. Eastman, R. Amor, Integrating the BIM rule language into compliant design audit processes, *Proceedings of 33rd CIB W78 Conference: IT in Construction*, 2016 <https://www.semanticscholar.org/paper/Integrating-the-BIM-Rule-Language-into-Compliant-Dimyadi-Dimyadi/bd7f601743be8c4b5bc3609a97b73daa7341be61>, Accessed date: 8 October 2018.
- [44] J. Dimyadi, G. Clifton, M. Spearpoint, R. Amor, Computerizing regulatory knowledge for building engineering design, *J. Comput. Civ. Eng.* 30 (2016), [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000572](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000572).
- [45] J. Lee, C. Eastman, Y. Lee, Implementation of a BIM domain-specific language for the building environment rule and analysis, *J. Intell. Robot. Syst.* 79 (2014) 507–522, <https://doi.org/10.1007/s10846-014-0117-7>.
- [46] T. Beach, T. Kasim, H. Li, N. Nisbet, Y. Rezgui, H. Decker, L. Lhotská, S. Link, J. Basl, Am. Tjoa (Eds.), *Towards Automated Compliance Checking in the Construction Industry, Database and Expert Systems Applications*, vol. 8055, Springer, Berlin Heidelberg, 2013, pp. 366–380 [https://doi.org/10.1007/978-3-642-40285-2\\_32](https://doi.org/10.1007/978-3-642-40285-2_32).
- [47] W3C, OWL, <https://www.w3.org/TR/owl2-overview/>, Accessed date: 8 October 2018.
- [48] W3C, RDF, <https://www.w3.org/RDF/>, Accessed date: 8 October 2018.
- [49] W3C, SPARQL, <https://www.w3.org/TR/rdf-sparql-query/>, Accessed date: 8 October 2018.
- [50] W3C, SPIN, <http://spinrdf.org>, Accessed date: 8 October 2018.
- [51] T. Beach, Y. Rezgui, H. Li, T. Kasim, A rule-based semantic approach for automated regulatory compliance in the construction sector, *Expert Syst. Appl.* 42 (2015) 5219–5231, <https://doi.org/10.1016/j.eswa.2015.02.029>.
- [52] S. Zhang, F. Boukamp, J. Teizer, Ontology-based semantic modeling of construction safety knowledge: towards automated safety planning for job hazard analysis (JHA), *Autom. Constr.* 52 (2015) 29–41, <https://doi.org/10.1016/j.autcon.2015.02.005>.
- [53] J. Dimyadi, P. Pauwels, R. Amor, Modelling and accessing regulatory knowledge for computer-assisted compliance audit, *J. Inf. Technol. Constr.* 21 (2016) 317–336 <http://www.itcon.org/2016/21>.
- [54] Liebherr-International AG, <https://www.liebherr.com/en>, Accessed date: 8 October 2018.
- [55] J. Teizer, J. Melzner, M. Wolf, O. Golovina, M. König, *Automatisierte 4D-Bauablaufvisualisierung und Ist-Datenerfassung zur Planung und Steuerung von Bauprozessen*, VDI-Bautechnik, Jahresausgabe 2016/17, Bauingenieur, Springer, 2017, pp. 129–135 (ISSN: 0005-6650).
- [56] R.J. Scherer, S. Schapke, A distributed multi-model-based management information system for simulation and decision-making on construction projects, *Adv. Eng. Inform.* 25 (2011) 582–599, <https://doi.org/10.1016/j.aei.2011.08.007>.
- [57] Model View Definition, MVD, buildingSMART, <http://www.buildingsmart-tech.org/implementation/ifc4-implementation/model-view-definition-mvd>, Accessed date: 8 October 2018.
- [58] Level of Development, LOD, BIMForum, <http://bimforum.org/lof/>, Accessed date: 8 October 2018.
- [59] R. Schach, J. Otto, *Baustelleneinrichtung*, Vieweg + Teubner Verlag, Germany, 978-3834813992, 2011.
- [60] J. Teizer, A. Green, T. Hilfert, M. Perschewski, M. König, Mobile point cloud assessment for trench safety audits, 34th International Symposium on Automation and Robotics in Construction, 2017 <https://doi.org/10.22260/ISARC2017/0021>.
- [61] S. Kim, D. Lee, M. Yang, Offset triangular mesh using the multiple normal vectors of a vertex, *Comput.-Aided Des. Applic.* 1 (2004) 285–291, <https://doi.org/10.1080/16864360.2004.10738269>.
- [62] M. Malosio, N. Pedrocchi, L. Tosatti, Algorithm to offset and smooth tessellated surfaces, *Comput.-Aided Des. Applic.* 6 (2009) 351–363, <https://doi.org/10.3722/cadaps.2009.351-363>.
- [63] S. Liu, C. Wang, Fast intersection-free offset surface generation from freeform models with triangular meshes, *IEEE Trans. Autom. Sci. Eng.* 8 (2011) 347–360, <https://doi.org/10.1109/TASE.2010.2066563>.
- [64] Y. Chen, H. Wang, D. Rosen, J. Rossignac, A Point-Based Offsetting Method of Polygonal Meshes, Technical Report, Georgia Institute of Technology, 2005, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.4092>, Accessed date: 8 October 2018.
- [65] C. Kim, C.T. Haas, K.A. Liapi, J. McLaughlin, J. Teizer, F. Bosche, Rapid human-assisted, obstacle avoidance system using sparse range point, *Proceedings of the 9th Biennial ASCE Aerospace Division International Conference on Engineering, Construction, and Operations in Challenging Environments*, League City/Houston, Texas, 2004, pp. 115–122 [https://doi.org/10.1061/40722\(153\)17](https://doi.org/10.1061/40722(153)17).
- [66] IFC Tools Project, <http://www.ifctoolsproject.com>, Accessed date: 8 October 2018.
- [67] Oracle, Java3D, [https://docs.oracle.com/cd/E17802\\_01/j2se/javase/technologies/desktop/java3d/forDevelopers/j3dguide/j3dTOC.doc.html](https://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/desktop/java3d/forDevelopers/j3dguide/j3dTOC.doc.html), Accessed date: 8 October 2018.
- [68] Autodesk Revit, <https://www.autodesk.de/products/revit-family/overview>, Accessed date: 8 October 2018.
- [69] QuickHull3D, <https://www.cs.ubc.ca/~lloyd/java/quickhull3d.html>, Accessed date: 13 April 2018.
- [70] BooleanModeller, [http://www.openifctools.org/Open\\_IFC\\_Tools/bm\\_features.html](http://www.openifctools.org/Open_IFC_Tools/bm_features.html), Accessed date: 8 October 2018.
- [71] S. Zhang, J. Teizer, N. Pradhananga, C. Eastman, Workforce location tracking to model, visualize and analyze workspace requirements in building information models for construction safety planning, *Automation in Construction*, vol. 60, Elsevier, 2015, pp. 74–86 <https://doi.org/10.1016/j.autcon.2015.09.009>.
- [72] J. Teizer, J. Melzner, *Building information modeling – Technologische Grundlagen und industrielle Anwendungen: BIM im präventiven Arbeits – und Gesundheitsschutz*, in: A. Borrmann, M. König, J. Beetz, C. Koch (Eds.), *Reihe VDI-Buch*, Springer Verlag, 2015, [https://doi.org/10.1007/978-3-658-05606-3\\_19](https://doi.org/10.1007/978-3-658-05606-3_19).